Prilagodljive vježbe u sustavu za automatsko ocjenjivanje programskog kôda Edgar

Ćurić, Luka

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 399

PRILAGODLJIVE VJEŽBE U SUSTAVU ZA AUTOMATSKO OCJENJIVANJE PROGRAMSKOG KÔDA EDGAR

Luka Ćurić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 399

PRILAGODLJIVE VJEŽBE U SUSTAVU ZA AUTOMATSKO OCJENJIVANJE PROGRAMSKOG KÔDA EDGAR

Luka Ćurić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 399

Pristupnik:	**Luka Ćurić (0036523417)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	prof. dr. sc. Igor Mekterović

Zadatak: **Prilagodljive vježbe u sustavu za automatsko ocjenjivanje programskog kôda Edgar**

Opis zadatka:

Edgar je informacijski sustav za testiranje studenata i učenje razvijen na FER-u. Osnovna funkcionalnost mu je mogućnost strojnog ocjenjivanja programskog koda u programskim jezicima SQL, C, Java, itd. Osim toga, podržava pitanja s više ponuđenih odgovora, pitanja s unosom slobodnog teksta, te još nekoliko vrsta pitanja. Na temelju uspjeha studenata u Edgaru moguće je razlučiti pitanja po težini. Upoznati se s Teorijom odgovora na zadatke (engl. Item Response Theory), područjem primjene i ograničenjima te razlikama u odnosu na Klasičnu teoriju testova. Ocijeniti primjenjivost teorije na zadatke u Edgaru s obzirom na njihov tip i uzorak. Istražiti i ostale modele i pokazatelje težine zadatka te predložiti prikladne pokazatelje za različite vrste zadataka. Izraditi samostojeći modul koji će kontinuirano pratiti i računati statističke pokazatelje zadataka. Za računanje statističkih pokazatelja zadataka razmotriti jezike R i Python te integraciju s postojećim sustavom "CodeRunner" za sigurno obavljanje proizvoljnog kôda u Edgaru. Omogućiti pregled izračunatih pokazatelja te odgovarajuće vizualizacije. Omogućiti definiciju prilagodljivih vježbi za studente na temelju skupa pitanja s izračunatim svojstvima. Napraviti jednostraničnu web-aplikaciju putem koje će studenti moći vježbati zadatke. Težina pitanja bi se prilagođavala s obzirom na studentovu uspješnost tijekom obavljanja vježbe, a potencijalno i na temelju ostalih poznatih podataka (npr. uspješnost prethodnih vježbi). Razmotriti i ostale moguće primjene izračunatih pokazatelja, npr. primjenu Teorije odgovora na zadatke u oblikovanju testa. Donijeti ocjenu ostvarenog pristupa.

Rok za predaju rada: 28. lipnja 2024.

Zahvala

Zahvalio bih se svom mentoru, prof. dr. sc. Igoru Mekterović, na odličnoj suradnji i svoj potpori, strpljenju, korektnosti i razumijevanju za vrijeme pisanja diplomskog rada i za vrijeme svih suradnji na preddiplomskom i diplomskom studiju.

Zahvalio bih se svima koji su me podupirali za vrijeme trajanja cijelog studija, a najviše svojoj obitelji, roditeljima i bliskim prijateljima.

Zahvalio bih se fakultetu i profesorima za sve pružene prilike stjecanja znanja, novih sposobnosti i poznanstava te ugodnog okruženja za rad, učenje, vježbanje i razvoj znanja.

Sadržaj

Uvod 1

1. Ispitivanje znanja i sposobnosti 2

1.1. Problemi ispitivanja znanja 2

1.2. Ispitivanje znanja u kontekstu vježbi 3

1.3. Automatizacija i prilagodljivost vježbi 3

1.4. Sustavi za ispitivanje znanja i sustav *Edgar* 4

1.5. Problemi automatizacije ispitivanja i sustav *Judge0* 4

1.6. Korištenje sustava za ispitivanje u svrhu vježbanja 5

1.7. Implementacije sustava za prilagodljive vježbe 5

2. Metode i teorije vezane uz ispitivanje znanja 6

2.1. Klasična teorija testova 6

2.2. Teorija odgovora na zadatke 7

2.2.1. Ograničenja primjene teorije odgovora na zadatke 14

2.3. Ostali pokazatelji 16

2.4. Mjere konzistencije i pouzdanosti testa 16

2.4.1. Kuder-Richardsonova metoda 17

2.4.2. Cronbachova alfa 18

2.4.3. Indeks težine zadatka 18

2.4.4. Point-biserijalni korelacijski koeficijent 19

2.4.5. Indeks diskriminacije zadataka 19

2.5. Prikladnost korištenja modela 20

3. Primjena IRT-a u formiranju vježbi koristeći podatke iz sustava *Edgar* 21

3.1. Obrazloženje IRT-a – primjer 22

3.2. Način računanja parametara logističke krivulje 24

3.3. Normalizacija parametara logističke krivulje 28

4. Mogućnost primjene IRT-a za oblikovanje testova 29
5. Model sustava za provedbu prilagodljivih vježbi 32
5.1. Funkcionalni zahtjevi sustava 32
5.1.1. Zahtjevi vezani uz prilagodljive vježbe – nastavnik 33
5.1.2. Zahtjevi vezani uz prilagodljive vježbe – student 37
5.1.3. Opis toka definicije prilagodljivih vježbi 38
5.1.4. Opis toka provođenja prilagodljivih vježbi 41
5.2. Nefunkcionalni zahtjevi 45
6. Razvijeni sustav za provođenje prilagodljivih vježbi 46
6.1. Modul za izvođenje poslova 48
6.1.1. Modeliranje modula za izvođenje poslova 50
6.1.2. Relacijski model baze podataka modula 55
6.1.3. Aplikacijska logika izvršavanja poslova 59
6.1.4. Podrška za proširenja 62
6.2. Servis za računanje statističkih pokazatelja 63
6.2.1. Modeliranje servisa za izračun statističkih pokazatelja 64
6.2.2. Relacijski model baze podataka servisa 67
6.3. Web aplikacija za vježbanje i demonstraciju implementiranih funkcionalnosti 73
6.3.1. Relacijski model baze podataka web aplikacije 74
7. Rad s razvijenim sustavom 81
7.1. Pokretanje servisa za izračun statističkih pokazatelja 82
7.2. Pokretanje poslužitelja aplikacijske logike web aplikacije 84
7.3. Pokretanje poslužitelja korisničkog sučelja web aplikacije 85
7.4. Rad s web aplikacijom prilagodljivih vježbi 86
7.4.1. Pokretanje izračuna statističkih pokazatelja 86

7.4.2. Pregled prethodno pokrenutih poslova 88
7.4.3. Pregled statističkih pokazatelja zadataka 89
7.4.4. Rad s prilagodljivim vježbama – nastavnik 94
7.4.5. Rad s prilagodljivim vježbama – student 103
8. Pregled korištenih tehnologija 108
9. Ocjena ostvarenog pristupa 110
10. Smjernice za budući razvoj 111
Zaključak 112
Literatura 113
Sažetak 114
Summary 115
Privitak 116

Uvod

U obrazovanju je osim učenja i stjecanja novih sposobnosti cilj i procijeniti uspješnost postupka obrazovanja. Postoji više načina provođenja ove procjene, ali niti jedan od njih nije u potpunosti točan ili dovoljno precizan. Različite metode procjene znanja i sposobnosti dobre su za procjenu sposobnosti kod određenih oblika ispitivanja znanja u određenim okolnostima. Osim za ocjenjivanje i kvantifikaciju sposobnosti, ove metode bilo bi poželjno iskoristiti i pri samom postupku učenja tako da se uz pomoć njih usmjeravaju vježbe koje studenti odrađuju. Razmatranjem karakteristika pojedinih metoda ispitivanja potrebno je doći do zaključka koja od metoda je najbolje primjenjiva za određene oblike ispitivanja znanja. U nastavku slijedi pregled više metoda ispitivanja znanja, odabir jedne metode te opis implementacije sustava koji omogućuje studentima prilagodljivo i personalizirano vježbanje zadataka i procjenu novo naučenih sposobnosti.

1. Ispitivanje znanja i sposobnosti

Ispitivanje znanja, sposobnosti i inteligencije osoba zadatak je psihometrije – grane psihologije. Cilj istraživanja ove grane je kvantificirati znanje u matematički prikladnom obliku. Na ovaj način želi se opisati veza između ljudskog znanja i uspješnosti primjene tog znanja u stvarnosti. Ova veza se modelira raznim matematičkim funkcijama, ali one ne mogu opisati pravu karakteristiku ljudskog znanja budući da veza između ljudskog znanja i primjene znanja u stvarnosti nije pravilna i nije konzistentna.

1.1. Problemi ispitivanja znanja

Kod ispitivanja znanja velika je vjerojatnost pojave određenih problema koji utječu na uspješnost stvarne procjene znanja. Na ljude u određenim trenutcima mogu utjecati razni pozitivni i negativni faktori koji određuju uspješnost njihove primjene stečenog znanja (npr. okolina u kojoj se provodi ispitivanje, emocionalno stanje ispitanika, zdravstveno stanje ispitanika, …).

Jedan od temeljnih problema koji se javlja je varanje. Za ispitanike koji varaju se ne može reći da rezultat njihovog ispitivanja prikazuje njihovu stvarnu sposobnost i znanje ispitanog područja. Ovakvi ispitanici su problematični jer ih je teško uočiti izvan perioda provedbe ispita ako ispiti nisu „personalizirani“, ali i tada nema potpune sigurnosti u to jesu li oni varali ili ne.

S druge strane, javlja se i problem nedovoljnog vremena za provođenje ispitivanja. Nekada ispitanici iako dovoljno znaju i kvalitetno raspolažu znanjem bivaju pogrešno procijenjeni jer im okruženje u kojem su predstavljali svoje sposobnosti nije dalo dovoljno vremena da one dođu do izražaja. Do manjka vremena može doći zbog stresa ispitanika, greške ispitivača koji je sastavljao test (ispit) i pogrešno procijenio vrijeme potrebno za rješavanje ispita i sl.

Većina metoda ove situacije prikazuje u obliku greške u mjerenju, temeljnom vjerojatnošću točnog odnosno netočnog odgovora ili na neki drugi prikladan način.

1.2. Ispitivanje znanja u kontekstu vježbi

Ispitivanje znanja i vježbanje usko su povezani. Budući da su vježbe neki oblik probnog ispitivanja znanja osobe, svi koncepti i metode koje su primjenjive na ispitivanje primjenjive su i u kontekstu vježbanja. No u kontekstu vježbi postoji bitna razlika od konteksta ispitivanja: mogućnost uklanjanja glavnih problema.

Kod vježbi se ne javlja niti jedan od navedenih glavnih problema s ispitima. Studentima koji vježbaju nije u interesu varati na vježbama jer shvaćaju da time aktivnost vježbanja gubi smisao. S druge strane, kada studenti vježbaju, tom aktivnošću mogu se baviti proizvoljno dugo. U ovom kontekstu su zapravo eliminirane greške mjerenja te se bolje mogu usporediti procjene različitih metoda i teorija mjerenja ljudskog znanja.

1.3. Automatizacija i prilagodljivost vježbi

Studenti za vrijeme vježbanja poboljšavaju svoje znanje o temi koju vježbaju. Kada krenu vježbati bilo bi dobro da počnu od jednostavnijih zadataka prema težima i kompleksnijima jer takvi zadatci zahtijevaju višu razinu znanja. Kod automatizacije postupka vježbanja potrebno je uzeti u obzir ovaj prirodni tok poboljšanja znanja za vrijeme vježbanja.

Automatizirane prilagodljive vježbe su vježbe koje se prilagođavaju razini znanja određene tematike i ocjenjuju uspjeh ispitanika bez intervencije autora vježbi. Kada ispitanik započne ovakve vježbe sustav koji ih implementira bi ispitaniku trebao dati procijenjeno lakše zadatke na početku vježbe. Ovakva implementacija također cijelo vrijeme mora pratiti uspjeh ispitanika te mu na temelju uspjeha ponuditi različite težine zadataka. Na kraju ovakvog procesa vježbanja ispitaniku se može prikazati ocjena ili broj postignutih bodova, ali čak je dovoljno prikazati samo težinu zadnjeg zadatka koju je ispitanik dostigao.

Postupak vježbanja se naravno može automatizirati, ali to zahtjeva da se vježbe sastoje od zadataka koji su pogodni za automatski način ocjenjivanja.

1.4. Sustavi za ispitivanje znanja i sustav *Edgar*

Kako je ispitivanje znanja postupak koji se može (djelomično) automatizirati, razvijeni su sustavi koji služe za automatiziranu provedbu ispitivanja. Jedan od tih sustava je i sustav *Edgar* koji se koristi na Fakultetu elektrotehnike i računarstva.

Sustav *Edgar* se koristi već dugi niz godina i pokazao se kao pouzdani alat za provođenje postupka ispitivanja studenata. *Edgar* i studentima i profesorima omogućuje jednostavniju i bržu provedbu ispita i laboratorijskih vježbi nego što bi to bilo moguće *„na papiru“*.

U sustavu *Edgar* implementirani su razni oblici zadataka. Neki od tih oblika su:

- Zadatci s ponuđenim odgovorima (ABCD pitalice)
- Zadatci s odgovorom slobodnog teksta (*esejska* pitanja)
- Zadatci s automatskom evaluacijom odgovora (zadatci vezani uz programski kod)

1.5. Problemi automatizacije ispitivanja i sustav *Judge0*

Problemi u automatizaciji ispitivanja javljaju se uglavnom vezane uz oblik postavljenih zadataka. Zadatci čije ocjenjivanje se najjednostavnije da automatizirati su zadatci s više ponuđenih odgovora od kojih je bilo koji podskup skupa odgovora na zadatak točan.

Nešto kompliciranije za izvedbu je automatizacija provjere odgovora na zadatke koji zahtijevaju implementaciju programa. Za provjeru odgovora na ovakve zadatke potreban je poseban sustav za izvođenje i evaluaciju rješenja. Jedan od tih sustava je sustav *Judge0* koji se koristi u sklopu sustava *Edgar* za potrebe automatizirane provjere programskih rješenja.

Judge0 je sustav koji omogućuje izvršavanje proizvoljnog koda napisanog u proizvoljnom programskom jeziku. Izvršavanje programskog koda u sustavu *Judge0* je izolirano (*eng. sandbox*) i programi ne mogu trajno utjecati na računalo i operacijski sustav na kojem se izvršavaju. *Judge0* izolaciju postiže korištenjem tehnologije kontejnera (*eng. containerisation*).

1.6. Korištenje sustava za ispitivanje u svrhu vježbanja

S obzirom na to da se vježbe mogu prikazati kao ispiti, prilagodba sustava za automatizirano provođenje ispitivanja ne bi trebala biti komplicirana. U sustavu kao što je *Edgar*, zadatke koji su evidentirani i „ispitani“ u nekom prethodnom trenutku moguće je iskoristiti kao zadatke koji će graditi vježbe. Osim toga, *Edgar* ima mogućnost definicije zadataka koji bi se javljali isključivo u vježbama.

Kako bi se prilagodljive vježbe mogle provoditi, u sustav *Edgar* potrebno je dodati niz funkcionalnosti:

1. Mogućnost definicije prilagodljive vježbe
2. Mogućnost klasifikacije zadataka prema težini
3. Mogućnost pokretanja prilagodljive vježbe
4. Praćenje odgovora na zadatke
5. Evidentiranje rezultata i praćenje uspjeha studenata

1.7. Implementacije sustava za prilagodljive vježbe

Jedno rješenje za provođenje prilagodljivih vježbi već je integrirano u sustavu *Edgar*, a implementacija je odrađena u sklopu diplomskog rada *„Prilagodljive vježbe i lekcije u sustavu Edgar“* [1]. Glavna mana ove implementacije je korištenje ovisnosti o modulu (R skripti) za izračunavanje karakteristike i procjenu težine zadataka te nemogućnost nadogradnje cjelokupnog sustava bez narušavanja stabilnosti implementacije. Trenutno rješenje je u potpunosti direktno implementirano u sustav *Edgar* što stvara dodatan problem održavanja cijelog sustava. Osim toga, trenutno implementirani sustav omogućuje računanje samo jednoparametarskog (*Rasch*) i dvoparametarskog (*Lord*) modela za zadatke (Tablica 2).

Budući da je *Edgar* već jako opširni sustav, javlja se potreba za razvojem budućih funkcionalnosti u izolaciji. Ideja je da bi ove funkcionalnosti radile samostalno bez čvrste koherencije sa sustavom *Edgar* te se nakon potpunog razvitka kasnije integrirale sa sustavom. Ovim načinom razvoja sustava poboljšava se njegova skalabilnost, ali i jednostavnost održavanja i implementacije novih funkcionalnosti.

2. Metode i teorije vezane uz ispitivanje znanja

U psihometriji su se razvila razna stajališta, teorije i metode koje opisuju vezu između ljudskog znanja i uspješnosti njegove primjene. Danas najviše primijenjena metoda za ispitivanje ljudskog znanja je *klasična teorija testova* (*eng. classical test theory, CTT*). Osim nje, postoje i druge teorije i metode koje kod procjene znanja uzimaju u obzir drugačije parametre i više ili manje parametara. Glavni konkurent klasičnoj teoriji testova je *teorija odgovora na zadatke* (*eng. item response theory, IRT*).

2.1. Klasična teorija testova

Klasična teorija testova temelji procjenu ljudskog znanja (KE – procjena znanja, *eng. knowledge estimate*) na zbroju stvarnog znanja (AK – stvarno znanje, *eng. actual knowledge*) i nasumične greške (E – greška, *eng. error*) koja se javlja za vrijeme ispitivanja [2]. Problem CTT-a je u tome što procjenjuje znanje pristupnika na temelju cijelog ispita – zadatci na ispitu u zbroju procjenjuju ukupnu sposobnost ispitanika. Preciznije, ovaj pristup mjeri uspjeh ispitanika na testu, ali ne i njegovo znanje konkretno ispitane tematike. Procjena znanja predložena ovom teorijom prikazana je formulom (1).

$$KE = AK + E \tag{1}$$

Uzimajući u obzir to da se ispitom želi ispitati znanje pristupnika različitih razina znanja i sposobnosti nije teško zaključiti da su autori ispita skloniji zadavati zadatke prosječne težine. Osim toga, s obzirom na to da ispitivač ne može zadati jako velik broj „kvalitetnih“ zadataka, također se javlja problem pokrivanja znanja cijelog područja kojim se kolegij bavi. Zbog opširnosti područja ispitivanja možda do izražaja neće doći znanje pojedinih pristupnika te će oni možda biti neispravno ocijenjeni.

2.2. Teorija odgovora na zadatke

Za razliku od klasične teorije testova kod koje se procjena znanja temelji na rezultatu ispita, procjena znanja koju predlaže teorija odgovora na zadatke se temelji na pojedinom zadatku. Tako je za teoriju odgovora na zadatke procjena znanja zapravo sadržana u pokazatelju sposobnosti i vjerojatnosti točnog odgovora na pojedini zadatak uz posjedovanje određene razine sposobnosti. Na taj način teorija odgovora na zadatke kompenzira nedostatke klasične teorije testova [3].

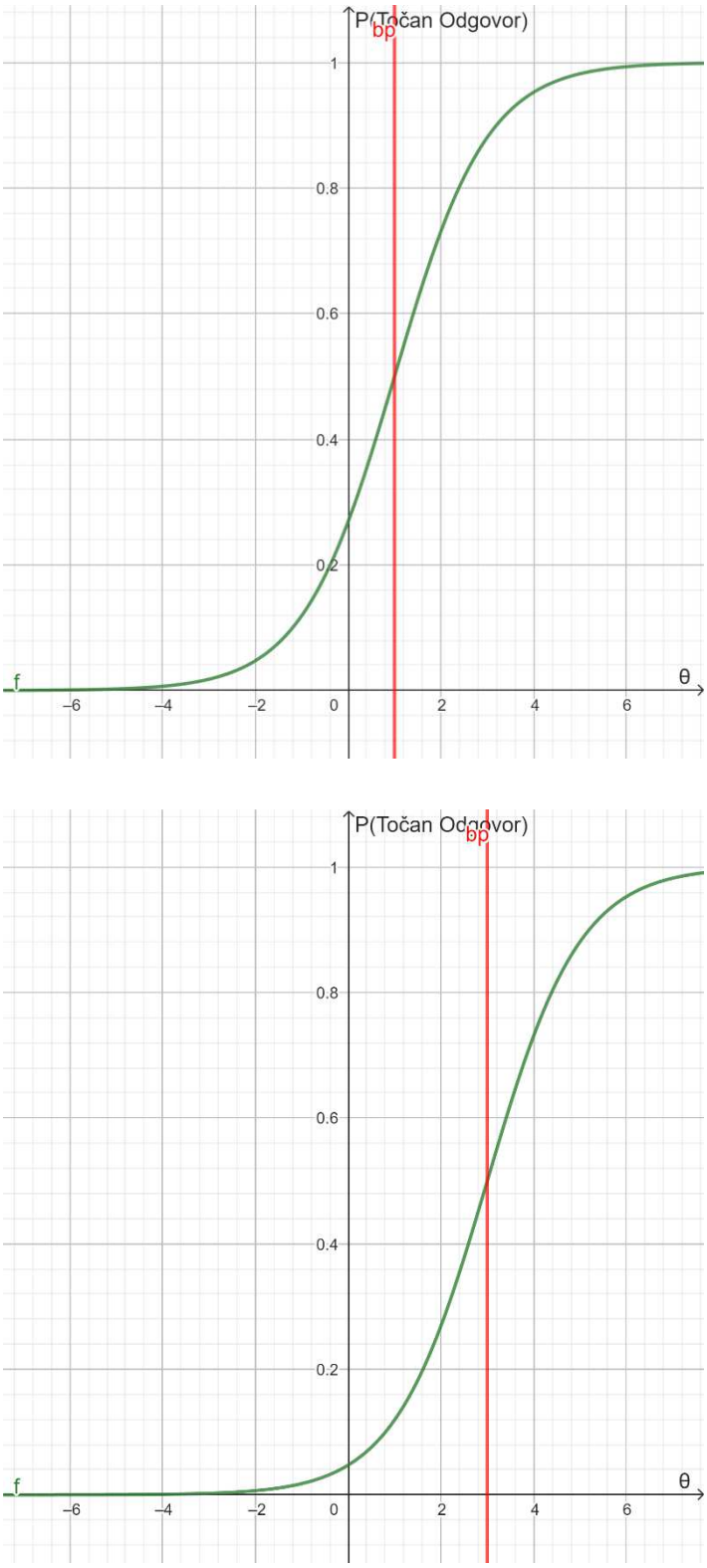
Teorija odgovora na zadatke temelji procjenu ljudskog znanja na matematičkom modelu logističke krivulje. Funkcija logističke krivulje još se naziva i *funkcija odgovora na zadatke* (*eng. item response function, IRF*) [4]. Ovom krivuljom upravljaju 4 parametra i jedna konstanta. Korištena logistička krivulja opisana je funkcijom (2). Ovako opisana logistička krivulja koristi se u *Barton-Lordovom*, odnosno 4-parametarskom modelu (Tablica 2).

$$f(\theta) = c_i + (d_i - c_i) \cdot \frac{1}{1 + e^{-a_i \cdot D_i \cdot (\theta - b_i)}} \qquad (2)$$

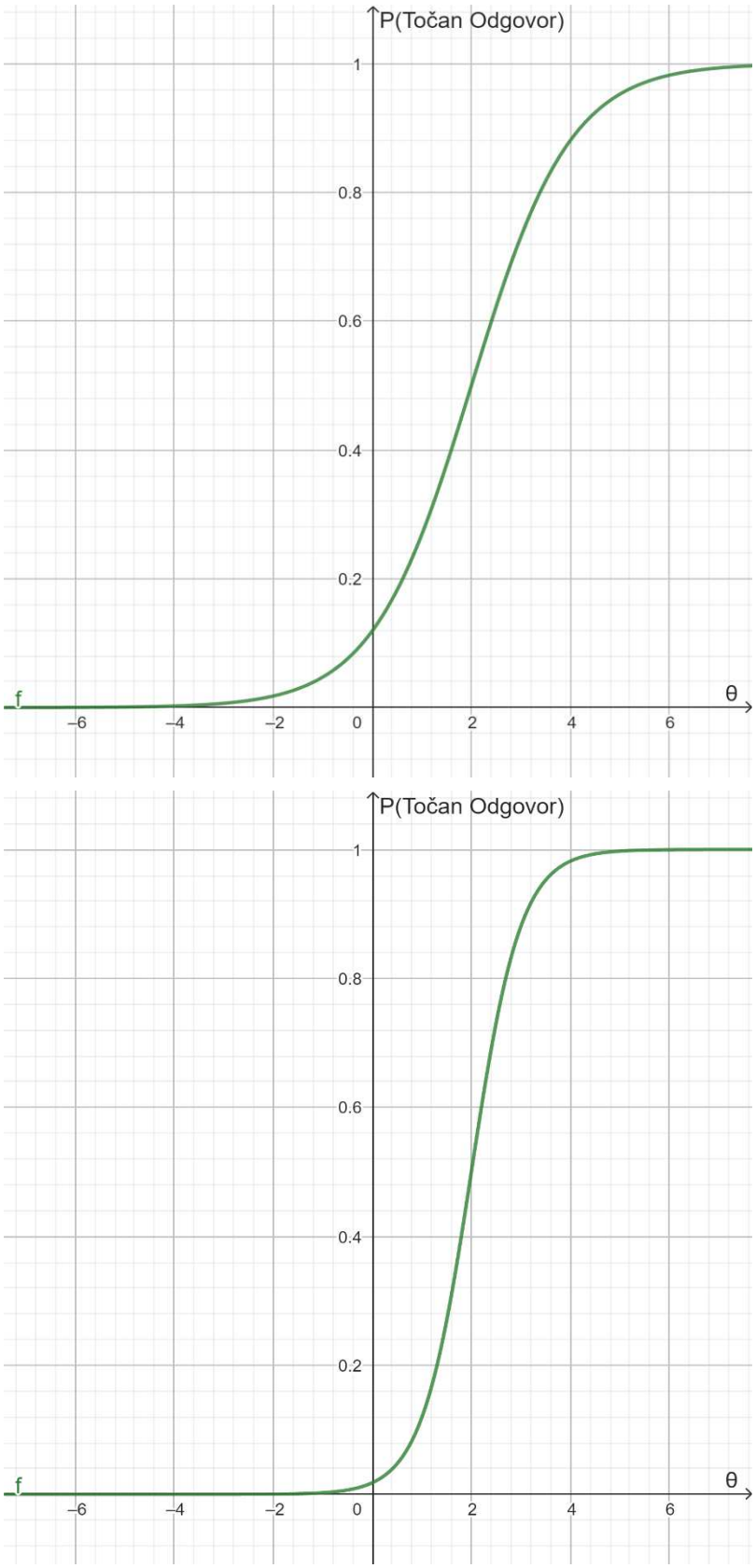
Vrijednost funkcije u određenoj točki predstavlja vjerojatnost točnog odgovora na zadatak koji funkcija modelira. Vrijednosti nad kojima se funkcija primjenjuje predstavljaju latentnu vrijednost sposobnosti ispitanika koju pojedini zadatak ispituje. Značenje parametara i njihov utjecaj na logističku krivulju opisani su u tablici (Tablica 1). Utjecaji promjene pojedinih parametara vidljivi su na slikama: za parametar *a* (Slika 2), za parametar *b* (Slika 1), za parametar *c* (Slika 3) i za parametar *d* (Slika 4).

Tablica 1 Značenje parametara i njihov utjecaj na logističku krivulju [4]

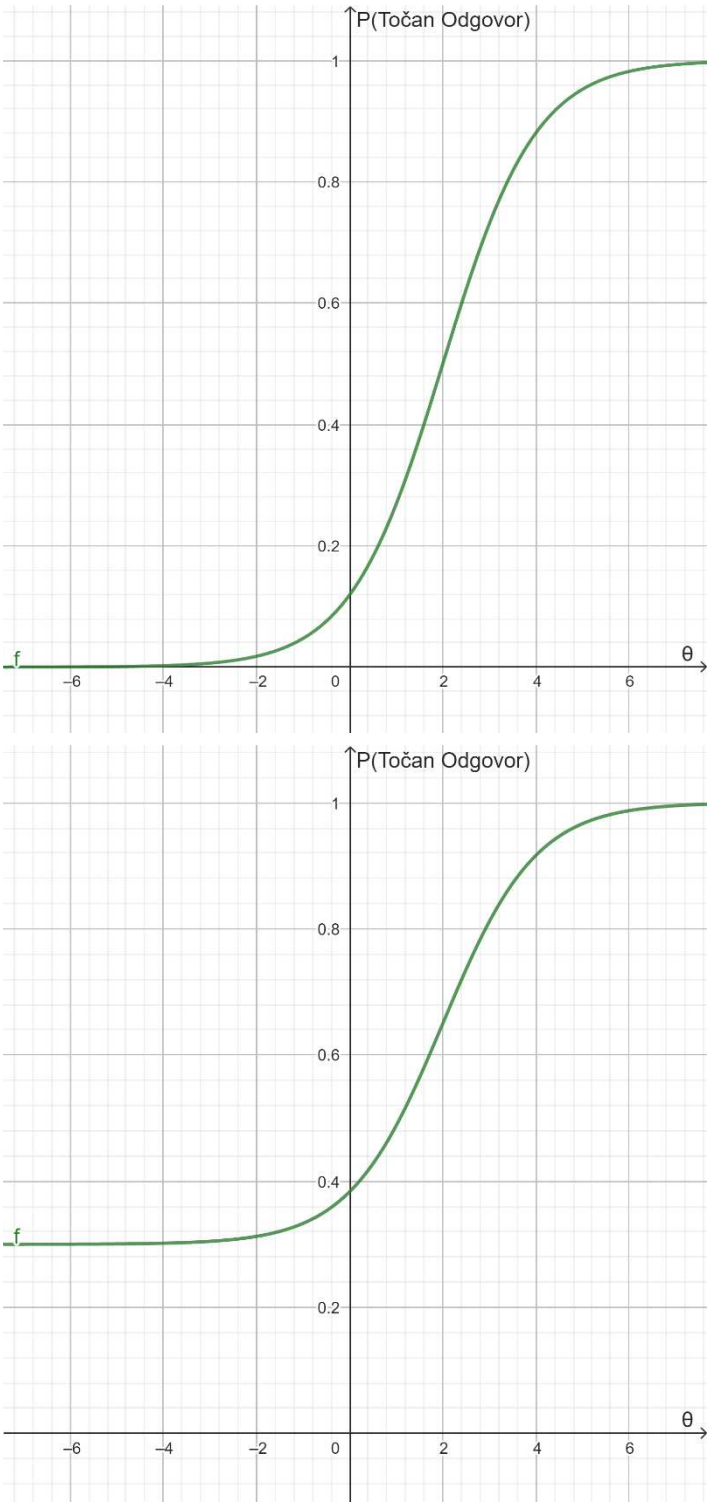
Parametar/ Konstanta	Značenje	Utjecaj
param. a_i	faktor diskriminacije (utjecaj znanja na uspješnost)	povećavanjem krivulja postaje strmija (matematički: razlika x vrijednosti točaka u kojima krivulja dolazi do asimptota je manja)
param. b_i	težina zadatka	povećanjem se krivulja pomiče udesno po x osi (matematički: točka infleksije)
param. c_i	vjerojatnost pogađanja točnog odgovora iako ispitanik ne posjeduje potrebno znanje	pomiče donju asimptotu krivulje
param. d_i	vjerojatnost točnog odgovora ako ispitanik posjeduje potrebno znanje (1 - vjerojatnost pogreške iako ispitanik posjeduje potrebno znanje)	pomiče gornju asimptotu krivulje
konst. D_i	empirijska konstanta za zadatak	utjecaj ekvivalentan promjeni parametra a_i



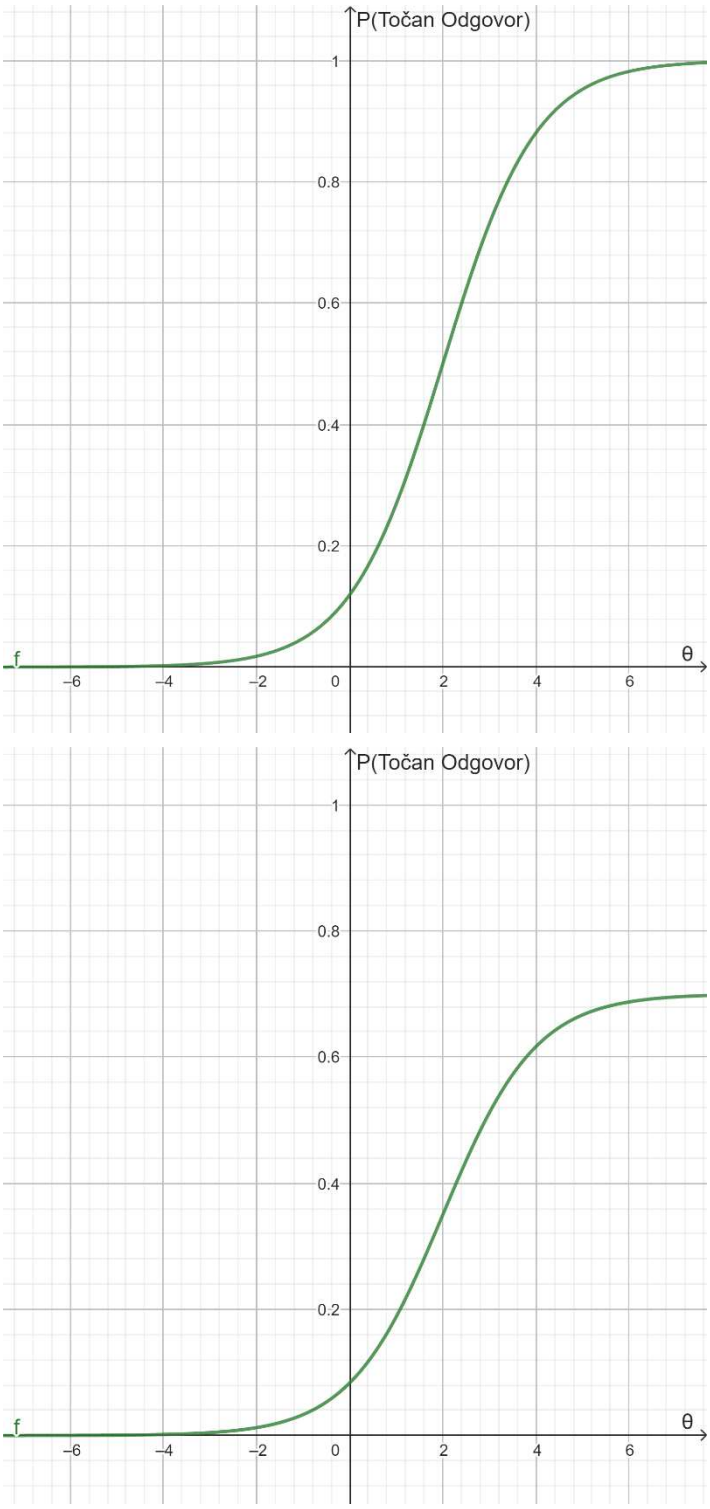
Slika 1 Utjecaj parametra *b* (težina zadatka) na logističku krivulju (a = 1, c = 0, d = 1): b = 1 [gore] i b = 3 [dolje]



Slika 2 Utjecaj parametra *a* (diskriminatorni faktor) na logističku krivulju (b = 2, c = 0, d = 1): a = 1 [gore] i a = 2 [dolje]



Slika 3 Utjecaj parametra *c* (vjerojatnost pogađanja točnog odgovora) na logističku krivulju (a = 1, b = 2, d = 1): c = 0 [gore] i c = 0.3 [dolje]



Slika 4 Utjecaj parametra *d* (vjerojatnost točnog odgovora dobrog ispitanika) na logističku krivulju (a = 1, b = 2, c = 0): d = 1 [gore] i d = 0.7 [dolje]

U teoriji odgovora na zadatke postoji više modela koji uzimaju u obzir samo neke parametre. Utjecaj promjene parametara i njihove pretpostavljene (*eng. default*) vrijednosti za pojedini model prikazane su tablicom (Tablica 2). Ako parametar modela ima pretpostavljenu vrijednost, funkcija logističke krivulje modela dobiva se tako da se pretpostavljena vrijednost parametra piše na njegovo mjesto u formuli (u tablici pretpostavljene vrijednosti parametara napisane su u zagradi). Primjer formule *Lordovog* 2-parametarskog modela prikazan je jednadžbom (3).

$$f(\theta) = 0 + (1 - 0) \cdot \frac{1}{1 + e^{-a_i \cdot D_i \cdot (\theta - b_i)}} = \frac{1}{1 + e^{-a_i \cdot D_i \cdot (\theta - b_i)}} \qquad (3)$$

Tablica 2 Utjecaj promjene i pretpostavljene vrijednosti parametara pojedinih modela

Parametar / Model	a_i	b_i	c_i	d_i
Rasch (1P) [5]	x (1)	✓	0	1
Lord (2P) [6]	✓	✓	0	1
Birnbaum (3P) [7]	✓	✓	✓	1
Barton-Lord (4P) [7]	✓	✓	✓	✓

2.2.1. Ograničenja primjene teorije odgovora na zadatke

Teorija odgovora na zadatke od ispitanika zahtijeva:

- racionalnost – niti jedan ispitanik neće namjerno dati krivi odgovor na zadatak
- ovisnost odgovora o ispitanoj sposobnosti

Teorija odgovora na zadatke od zadataka i ispita zahtijeva:

- međusobnu nezavisnost – točan odgovor na jedan zadatak ne povlači točan odgovor na drugom zadatku
- neograničeno vrijeme – ispitanik ima neograničeno (ili relativno dugo) vrijeme za koje rješava ovakav ispit

Jedan od problema koji se javlja s teorijom odgovora na zadatke je veličina uzorka. Ovisno o korištenom modelu (1P, 2P, 3P ili 4P) potreban je velik uzorak (između 500 i 1000) ispitanika kako bi se dobili pouzdani rezultati procjene parametara [3]. Ako uzorak nije dovoljno velik, modeli neće biti primjenjivi za analizu zadataka.

Zadatci koji najbolje odgovaraju ovim ograničenjima su zadatci višestrukog izbora. Kod ovakvog tipa zadataka ako ispitanik zna točan odgovor u velikoj većini slučajeva njega i zaokružuje kao odgovor. Također, ovakav oblik zadatka uglavnom ispituje znanje neke činjenice ili skupa povezanih činjenica što je u skladu s pretpostavkama oko ispitivanja jedne sposobnosti. Nadalje, postavljanje dva ili više zadatka višestrukog izbora koji međusobno impliciraju odgovore nema smisla čak niti u klasičnoj teoriji testova jer se to može smatrati „namjernim“ uvođenjem greške mjerenja znanja u ispit. Jedino ograničenje koje i dalje ostaje problematično je postojanje vremenskog ograničenja na ispitima.

Osim zadataka višestrukog izbora, i ostali se važniji tipovi zadataka veoma dobro ponašaju u zadanom okruženju ograničenja uz manje poteškoće. Tako se npr. zadatci slobodnog odgovora slažu sa svim ograničenjima, ali problem se javlja kod ocjenjivanja odgovora. Postupak ocjenjivanja je subjektivan i pri tome u samu procjenu unosi grešku u postupak procjene sposobnosti. Ako postoji više ispitivača koji su zaduženi za provođenje ispita, vrlo je vjerojatno da će dva ispitivača različito ocijeniti odgovor jednog ispitanika.

Nadalje, strojno ocijenjeni odgovori na zadatke koji zahtijevaju unos programskog koda kao rješenja ne nailaze na problem subjektivnosti u ocjenjivanju. Međutim, problem se može javiti kod zahtjeva međusobne nezavisnosti zadataka budući da se znaju pojaviti baš takvi zadatci. Kod ovih zadataka rješenje jednog zadatka se prenosi u rješenje drugog zadatka i tako krši jedno od postavljenih ograničenja (npr. jedan zadatak zahtjeva implementaciju neke funkcije, a drugi nadogradnju te iste funkcije da radi koristeći neki drugi algoritam). Treba paziti da se ovakvi zadatci ne zadaju u sklopu prilagodljivih vježbi.

Iz navedenog bi se dalo zaključiti da bi se zapravo svi oblici zadataka na neki način mogli prilagoditi korištenju teorije odgovora na zadatke. Pri postupku prilagodbe trebalo bi razmotriti olakšanje zahtjeva kako bi se što veći broj zadataka mogao iskoristiti za izradu testova temeljenih na teoriji odgovora na zadatke.

2.3. Ostali pokazatelji

Osim klasične teorije testova i teorije odgovora na zadatke postoji još metoda i pokazatelja koji se koriste za predočenje uspješnosti primjene znanja. Karakteristika ovih pokazatelja je u tome što se ne fokusiraju na analizu pojedine sposobnosti, već više međusobno (ne)zavisnih sposobnosti koristeći korelacijske matrice. Neke od ovih metoda su faktorska analiza (*eng. factor analysis*) i analiza grupiranih stavki (*eng. cluster analysis*).

S obzirom na to da je teorija odgovora na zadatke dovoljna za određivanje karakteristike pojedinog zadatka u izolaciji, a potrebno znanje i razumijevanje područja psihometrije i kompleksnost implementacije ovih metoda velika, one nisu razmotrene kao alat za sastavljanje prilagodljivih vježbi. Ali to ih ne bi trebalo izbaciti kao mogućnost za buduće implementacije sličnih ili nadogradnju postojećih sustava. Kod površinskog pregleda, ove metode mogle bi biti jako dobre za identifikaciju latentnih sposobnosti koje određeni zadatak ispituje.

2.4. Mjere konzistencije i pouzdanosti testa

Uz teorije vezane uz procjenu ljudskog znanja potrebno je donesti i ocjenu uspješnosti tih teorija u praksi. Budući da su i klasična teorija testova i teorija odgovora na zadatke alati uz pomoć kojih ispitivači mogu sastavljati testove, uspješnost njihove primjene može se mjeriti pokazateljima konzistencije i pouzdanosti testa. Nakon provođenja testa (ispita) iz rezultata se mogu donijeti zaključci o njegovoj pouzdanosti. Ako metoda smatra rezultate pouzdanima to znači da je test kvalitetno sastavljen.

Ovi pokazatelji konzistencije i pouzdanosti testa razvijeni su za klasičnu teoriju testova te na zadatcima koji se binarno ocjenjuju (tzv. dihotomni zadatci, *eng. dichotomous*). Ideja je neke od pokazatelja primijeniti kao vodilje pri klasifikaciji zadataka i izračuna parametara logističke funkcije kod izgradnje prilagodljivih vježbi.

2.4.1. Kuder-Richardsonova metoda

Kuder-Richardsonova metoda procjenjuje kvalitetu sastavljenog ispita na temelju broja zadataka u testu te točnosti odgovora na pojedine zadatke. Raspon koeficijenta je [0, 1]. Što je iznos koeficijenta veći to je prema metodi test pouzdaniji te zadatci na testu imaju viši stupanj korelacije [8]. Neki od problema ove metode procjene korelacije zadataka su:

- Trajanje testa – ako je test prekratak ili predug, prikupljeni podatci nisu vjerodostojni
- Velik broj zadataka – što je broj zadataka veći to je veća vjerojatnost da će koeficijent biti visok
- Težina zadataka – ako su svi zadatci na testu preteški ili prelagani, iznos zbroja umnožaka u brojniku drugog člana jednadžbe blizu je 0 što povećava iznos koeficijenta

Koeficijent (KR-20) koji određuje ova metoda prikazan je jednadžbom (4).

$$r = \frac{K}{K - 1}\left[1 - \frac{\sum_{i=1}^{K} p_i q_i}{\sigma_X^2}\right] \tag{4}$$

r – iznos koeficijenta

K – broj zadataka na testu

p_i – postotak točnih odgovora na zadatak i

q_i – postotak netočnih odgovora na zadatak i

σ_X^2 – varijanca ukupnih bodova svih pristupnika

Ako se pretpostavi da svi zadatci imaju istu težinu (vjerojatnost točnog odgovora je ista za svaki zadatak i iznosi *p*), dobiva se formula koeficijenta (KR-21) prikazana jednadžbom (5).

$$r = \frac{K}{K - 1}\left[1 - \frac{K \cdot p(1 - p)}{\sigma_X^2}\right] \tag{5}$$

2.4.2. Cronbachova alfa

Koeficijent koji predstavlja *metoda Cronbachove alfe* računa se na sličan način kao i koeficijent predstavljen *Kuder-Richardsonovom metodom*. Raspon *Cronbachove alfe* je [0, 1]. Za *Cronbachovu alfu* vrijedi da što je koeficijent veći to zadatci na testu imaju viši stupanj korelacije i test je pouzdaniji (kao i kod *K-R metode*). Točnije, *Cronbachova alfa* koeficijent je pouzdanosti promatrane mjere [9].

Razlika između *Cronbachove alfe* i *Kuder-Richardsonove metode* je u tome što se u drugom članu jednadžbe u brojniku ne računa suma umnožaka postotaka točnih i netočnih odgovora, već suma varijanci postignutih bodova na zadatku. Način računanja *Cronbachove alfe* prikazan je jednadžbom (6).

$$\alpha = \frac{k}{k\text{-}1}\left[1 - \frac{\sum_{i=1}^{k} \sigma_{y_i}^2}{\sigma_y^2}\right] \tag{6}$$

α – iznos koeficijenta

k – broj zadataka na testu

$\sigma_{y_i}^2$ – varijanca bodova na zadatku i

σ_y^2 – varijanca ukupnih bodova svih pristupnika

2.4.3. Indeks težine zadatka

Indeks težine zadatka koristi se za procjenu težine zadatka nakon provedenog testa. Raspon indeksa je [0, 1], a što je iznos indeksa veći to je promatrani zadatak teži. Ovaj indeks zapravo predstavlja postotak pristupnika koji su točno odgovorili na zadatak. Izračun ovog indeksa jednostavan je i prikazan jednadžbom (7).

$$P = \frac{R}{T} \tag{7}$$

P – indeks težine

R – broj točnih odgovora na zadatak

T – ukupan broj zadataka na testu

2.4.4. Point-biserijalni korelacijski koeficijent

Koeficijent koji se koristi u statistici kako bi se izračunala linearna korelacija vrijednosti dihotomne slučajne varijable (u ovom slučaju točnost odgovora na zadatak) [10]. Raspon koeficijenta je <-1, 1>, a predstavlja iznos linearne korelacije između broja ispitanika koji su točno odgovorili i broja ispitanika koji su netočno odgovorili na zadatak. Izračun ovog koeficijenta prikazan je jednadžbom (8).

$$r_{pb} = \frac{M_1 - M_0}{s_n} \sqrt{\frac{n_1 n_0}{n^2}} \tag{8}$$

r_{pb} – iznos koeficijenta

M_1 – srednja vrijednost broja točnih odgovora

M_0 – srednja vrijednost broja netočnih odgovora

s_n – standardna devijacija uspješnosti, tj. točnosti odgovora

n_1 – broj ispitanika koji su dali točan odgovor na zadatak

n_0 – broj ispitanika koji su dali netočan odgovor na zadatak

n – ukupni broj ispitanika

2.4.5. Indeks diskriminacije zadataka

Indeks diskriminacije zadatka pokazuje koliko dobro zadatak diskriminira dobre i loše pristupnike, odnosno dobre i loše rezultate [11]. Raspon vrijednosti indeksa je [-1, 1], a veći iznos indeksa znači da zadatak pogoduje skupini ispitanika koja je bolje pripremljena za test. Način izračuna indeksa prikazan je jednadžbom (9).

$$D = \frac{U - L}{N} \tag{9}$$

D – iznos indeksa

U – broj izvrsnih pristupnika koji su dali točan odgovor na zadatak

L – broj loših pristupnika koji su dali točan odgovor na zadatak

N – ukupni broj pristupnika

2.5. Prikladnost korištenja modela

Budući da se ostali modeli temelje na detaljnom proučavanju i analizi rezultata kako bi se donijela ocjena uspjeha ispitanika, CTT i IRT su jedine metode koje ima smisla ukomponirati u postupak automatiziranog ispitivanja i vježbanja. Ove dvije metode kvantificiraju znanje matematičkim rječnikom i zato su pogodne za strojnu obradu. S obzirom na to da je cilj izgraditi prilagodljive vježbe, najbolje je primjenjiva teorija odgovora na zadatke.

Osnovna ideja prilagodljivih vježbi je vođenje studenata od lakših prema težim zadatcima. Procjena težine pojedinog zadatka je teška ako ne i nemoguća korištenjem CTT-a jer procjena uspješnosti primjene znanja kod CTT-a zahtjeva u potpunosti riješen ispit, a time i unaprijed poznat skup zadataka na ispitu. To je nemoguće kod prilagodljivih vježbi zbog same prirode načina postavljanja zadataka i potrebe da se ispit *„uživo“* prilagođava pojedinom ispitaniku. Drugim riječima, potrebno je uspostaviti sustav koji će pratiti uspjeh studenta i za vrijeme vježbe mijenjati zadatke ovisno o težinskom razredu u kojem se student u tom trenutku nalazi.

Rezultat pristupa ovakvim vježbama na ovaj način je taj da iako dva studenta s identičnim razinama znanja pristupaju vježbanju u isto vrijeme, mogućnost da ta dva studenta dobiju isti skup zadataka neće biti vjerojatna. Budući da je svaka vježba nezavisna od prethodnih i da se u ovakvom sustavu studenti gledaju kao nezavisni pristupnici, svaki od njih dobiva personalizirane zadatke u trenutku vježbanja. Zato je odabir IRT-a kao metode kvantificiranja znanja i metode sastavljanja ispita idealan za sastavljanje prilagodljivih vježbi [12].

3. Primjena IRT-a u formiranju vježbi koristeći podatke iz sustava *Edgar*

S obzirom na to da se u *Edgaru* već dugi niz godina provode ispitivanja i bilježe rezultati rješavanja zadataka, moguće je te podatke iskoristiti za potrebe statističke analize. Korištenjem statističkih pokazatelja zadataka moguće je odrediti njihove karakteristike i parametre pripadajućih logističkih funkcija. Na ovaj način moguće je iskoristiti povijest pristupanja i odgovora na zadatke i uz pomoć tih informacija izgraditi temelj za korištenje zadataka u prilagodljivim vježbama.

Potrebno je naglasiti kako bi se svi zadatci trebali gledati jednako bez obzira na tip zadatka. Svi odgovori na zadatke se u nekom trenutku postupka prilagodbe za korištenje u IRT-u moraju svesti na točne ili netočne budući da je dihotomnost zadataka jedan od osnovnih zahtjeva IRT modela.

Korištenje statističkih pokazatelja za izračun parametara karakterističnih krivulja jedan je od načina njihovog računanja. Postoji biblioteka *„LTM“* razvijena za programski jezik R koja radi ovu procjenu i korištena je za potrebe trenutne implementacije sustava prilagodljivih vježbi u *Edgaru* [1]. Nova implementacija fokusirat će se na računanje parametara logističke funkcije koristeći statističke pokazatelje zadataka. Ovo omogućava novom sustavu da ne ovisi o određenim bibliotekama, ali i da izračunate pokazatelje prikaže uz konačni rezultat što će korisnicima (nastavnicima) omogućiti da lakše shvate zašto je sustav nekom zadatku dodijelio određenu težinu.

3.1. Obrazloženje IRT-a – primjer

Kod primjene IRT-a važna su dva faktora: znanje ispitanika predočeno latentnom sposobnošću (θ u jednadžbi logističke krivulje) i logistička krivulja zadatka. Glavna pretpostavka teorije je da studenti s većim iznosom sposobnosti imaju veću vjerojatnost točno odgovoriti na zadatak.

Ako se za primjer uzme da je mjerena latentna sposobnost *„poznavanje baza podataka“* te pretpostavi da postoje tri ispitanika:

- početnik (A)
- prosječni ispitanik (P)
- ispitanik koji se duži period bavi bazama podataka (M)

kojima se zadaju tri zadatka. Karakteristike tri zadatka prikazane su tablicom (Tablica 3), a karakteristike ispitanika tablicom (Tablica 4). Logističke krivulje zadataka s oznakama sposobnosti pojedinih ispitanika vidljive su na slici (Slika 5**Pogreška! Izvor reference nije pronađen.**).

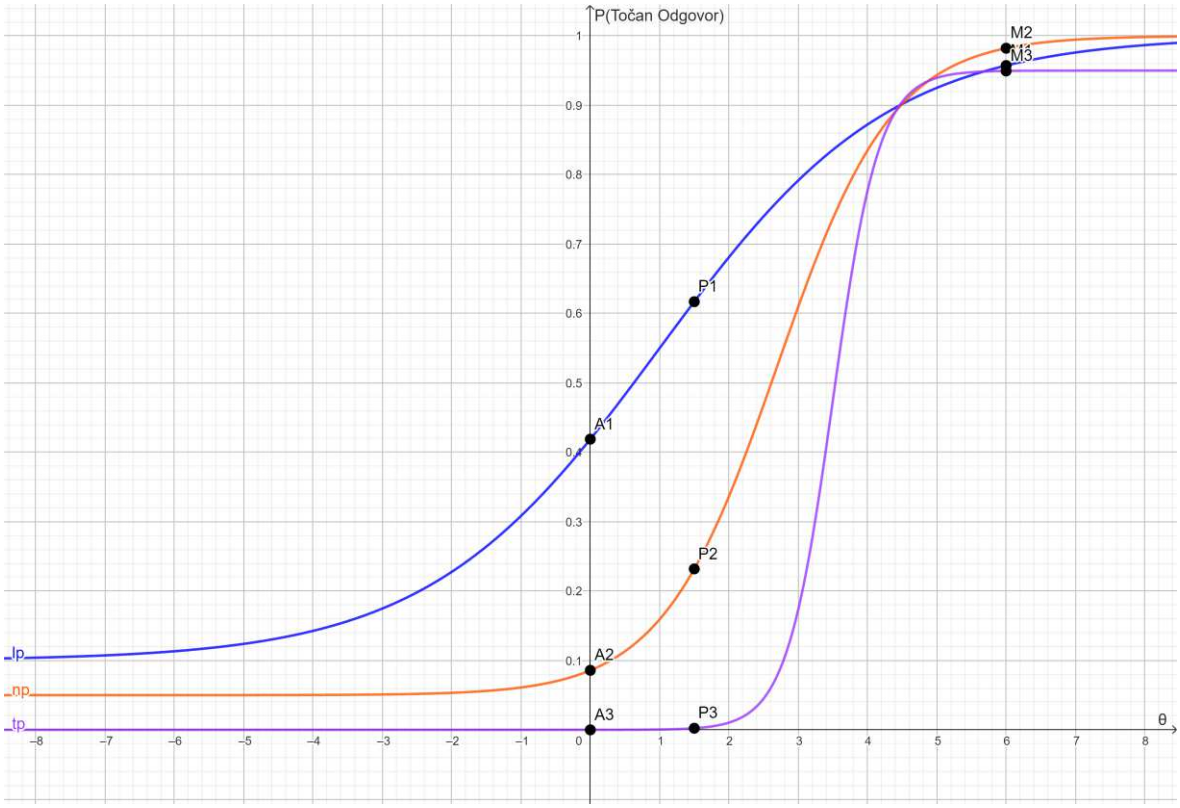
Tablica 3 Karakteristike zadataka u primjeru

Zadatak	disk. faktor (a)	težina (b)	P(pogađanje to[1]) (c)	P(no[2] dobrog stud.) (d)
Lagani zad. (1)	0.6	1.0	0.1	1.0
Normalni zadatak (2)	1.2	2.7	0.05	1.0
Teški zad. (3)	3.0	3.5	0.0	0.95

[1] to – točan odgovor
[2] no – netočan odgovor

Tablica 4 Karakteristike ispitanika i vjerojatnosti točnih odgovora u primjeru

Ispitanik	θ (poznavanje BP)	Lagani zadatak (1)	Normalni zadatak (2)	Teški zadatak (3)
A (početnik)	0.0	41.89%	8.58%	0.00%
P (prosječan)	1.5	61.70%	23.2%	0.23%
M (odličan)	6.0	95.73%	98.22%	94.95%



Slika 5 Prikaz logističkih krivulja zadataka iz primjera: lagani zadatak [plavo], normalni zadatak [narančasto] i teški zadatak [ljubičasto]

3.2. Način računanja parametara logističke krivulje

Za svaki zadatak mogu se izračunati statistički pokazatelji za stečene bodove na zadatku kao što su srednja vrijednost, medijan i standardna devijacija bodova na zadatku. Ovi statistički pokazatelji mogu se normalizirati s obzirom na ukupan broj bodova na zadatku i iskoristiti u koraku računanja parametara logističke krivulje.

Tako bi se npr. parametar *b* logističke krivulje (parametar *težine zadatka*) mogao izračunati formulom (10). Postotak riješenosti te postotci točnih i netočnih odgovora na zadatak u formuli računaju se na razini kolegija i akademskih godina. Točnim odgovorima smatraju se svi oni odgovori bodovani s 45% ili više bodova ostvarivih na pojedinom zadatku.

$$\mathrm{b(q)} = \frac{1.0 + 0.1}{\mathrm{AVG(PostotakRiješenosti(q))} + 0.1} \cdot \sqrt{N_{opservacija} + 1} \cdot \frac{\mathrm{PostotakNetočnih(q)} + 0.1}{\mathrm{PostotakTočnih(q)} + 0.1} \qquad (10)$$

Zato što parametar *b* određuje težinu zadatka, ako prosječan postotak riješenosti raste, iznos parametra treba padati (prvi član jednadžbe). Također, što je više ispitanika rješavalo zadatak to bi promjena prosječnog postotka riješenosti trebala biti utjecajnija (drugi član jednadžbe). Na posljetku, što zadatak ima više netočnih odgovora, parametar *b* bi trebao rasti, a što je više točnih odgovora na zadatak parametar *b* bi trebao padati (treći član jednadžbe). Dodavanje vrijednosti 0.1 u prvom i trećem članu neophodno je za izbjegavanje situacije dijeljenja s nulom. Ako su svi navedeni statistički pokazatelji zadatka jednaki 0, tada je iznos parametra *b* jednak 1.

Parametar *a* logističke krivulje mogao bi se računati formulom (11). Da bi se student smatrao dobrim mora imati više od 70%, a da bi se smatrao lošim manje od 40% postignutih bodova na testu u kojem se zadatak pojavio. Da bi se odgovor na zadatak smatrao točnim, student mora postignuti barem 45% bodova na zadatku. U protivnom, odgovor se smatra netočnim.

$$a(q) = \frac{N_{tod} + N_{nol}}{N_{nod} + N_{tol} + 1} \tag{11}$$

N_{tod} – broj točnih odgovora dobrih studenata

N_{nol} – broj netočnih odgovora loših studenata

N_{tol} – broj točnih odgovora loših studenata

N_{nod} – broj netočnih odgovora dobrih studenata

Jednadžba izračuna parametra je blago modificirani indeks diskriminacije zadatka (poglavlje 2.4.5) kako bi vrijednost parametra ostala pozitivna. Zato što parametar *a* predstavlja diskriminatornu moć zadatka, njegov iznos trebao bi rasti što je veći broj točnih odgovora dobrih studenata i što je veći broj netočnih odgovora loših studenata (brojnik). Nadalje, vrijednost parametra trebala bi padati što je veći broj netočnih odgovora dobrih studenata i što je veći broj točnih odgovora loših studenata (nazivnik). Nazivnik je takav zato što smanjuje vrijednost parametra u slučaju da loši studenti zadatak bolje rješavaju od dobrih studenata. U nazivniku je dodana vrijednost 1 kako bi se izbjeglo eventualno dijeljenje s nulom.

Primjer izračuna parametra *c* logističke krivulje dan formulom (12). Kao i kod izračuna parametra *a*, da bi se student smatrao lošim mora imati manje od 40% postignutih bodova na testu u kojem se zadatak pojavio. Da bi se odgovor na zadatak smatrao točnim, student mora postignuti barem 45% bodova na zadatku.

$$c(q) = \text{PostotakTočnih}(q) \cdot \frac{N_{tol}}{N_{pristupnika}} \tag{12}$$

N_{tol} – broj točnih odgovora loših studenata

$N_{pristupnika}$ – broj ispitanika koji su odgovorili na zadatak

Za zadatke višestrukog izbora, parametar *c* logističke krivulje mogao bi se izračunati na način prikazan formulom (13).

$$c(q) = \frac{1}{\text{broj\_ponuđenih\_odgovora}} \tag{13}$$

Budući da parametar *c* modelira vjerojatnost pogađanja točnog odgovora, njegov iznos raste s postotkom točnih odgovora na zadatak (prvi član jednadžbe), a smanjuje se što je veći udio loših studenata koji su dali točan odgovor u ukupnom broju studenata (drugi član).

Primjer izračuna parametra *d* logističke krivulje dan formulom (14). Kao i kod izračuna parametra *a*, da bi se student smatrao dobrim mora imati više od 70% postignutih bodova na testu u kojem se zadatak pojavio. Da bi se odgovor na zadatak smatrao netočnim, student mora postignuti manje od 45% bodova na zadatku.

$$d(q) = 1 - \text{PostotakNetočnih}(q) \cdot \frac{N_{nod}}{N_{\text{pristupnika}}} \tag{14}$$

N_{nod} – broj netočnih odgovora dobrih studenata

$N_{pristupnika}$ – broj ispitanika koji su odgovorili na zadatak

Budući da parametar *d* modelira vjerojatnost točnog odgovora dobrih ispitanika, njegov iznos raste s padom postotka netočnih odgovora na zadatak (prvi član jednadžbe), a smanjuje se što je veći udio dobrih studenata koji su dali netočan odgovor u ukupnom broju studenata (drugi član).

3.3. Normalizacija parametara logističke krivulje

Kako bi se bolje mogli tumačiti i koristiti, parametre logističke krivulje potrebno je normalizirati na željene raspone. Odabrani rasponi za implementaciju prilagodljivih vježbi prikazani su u tablicom (Tablica 5).

Parametri *a* i *b* mogu se koristiti u proizvoljnom rasponu, ali normalizirani su na navedene vrijednosti za potrebe klasifikacije zadataka po težini. Zbog normalizacije postupak određivanja težine zadatka bilo je lakše provesti i određene skupine težina bile su međusobno uniformne te je, budući da se logistička krivulja koristi u svrhu uspoređivanja zadataka, krivulje bilo lakše usporediti koristeći normalizirane parametre. Parametre *c* i *d* nije bilo potrebno normalizirati zato što je domena njihovih vrijednosti postotak pa su time ograničeni na raspon [0, 1]. Formula po kojoj su vrijednosti parametara *a* i *b* normalizirane opisana je jednadžbom (15).

$$x_{norm} = \frac{\log_{[min(x)\,+\,1.5]}(x + min(x)\ +\ 1.5)\ -\ 1.0}{\log_{[min(x)\,+\,1.5]}(max(x) + min(x)\ +\ 1.5)\ -\ 1.0} \quad (15)$$

Vrijednost *1.5* dodaje se u bazu logaritma kako ne bi došlo do situacije u kojoj je baza 1 (u slučaju da je min(x) = 1). Logaritam je korišten zato što smanjuje velike razlike u vrijednostima koje dolaze na ulaz pa se tako konačne vrijednosti bolje raspršuju u rasponu [0, 1].

Tablica 5 Normalizirane vrijednosti parametara u implementaciji

Parametar	Min.	Max.
a (diskriminatorni faktor)	0	1
b (težina zadatka)	0	1
c (temeljna vjerojatnost točnog odgovora)	0	1
d (temeljna vjerojatnost pogrešnog odgovora)	0	1

4. Mogućnost primjene IRT-a za oblikovanje testova

Kako bi IRT bila uspješna u oblikovanju pravih testova (ispita) prvo je potrebno stvoriti inicijalnu *„sliku“* znanja studenta. Ovo bi bilo moguće napraviti koristeći informacije o uspjehu studenta koje postoje u sustavu *Edgar* (kao npr. rezultati laboratorijskih vježbi, rezultati na povezanim kolegijima i sl.).

Druga mogućnost inicijalne procjene je da prije svakog ispita kojem pristupa, student riješi nekoliko klasifikacijskih zadataka. Uspješnost studenta na ovim zadatcima odredila bi početnu ocjenu s kojom bi student stupio u rješavanje pravog ispita. Tako implementirani sustav bi za svakog studenta prije svakog ispita mogao procijeniti početnu razinu latentne sposobnosti.

Ako informacije potrebne za inicijalnu procjenu svakog pojedinog studenta nisu dostupne, sustav bi za sposobnosti studenata trenutne generacije mogao pretpostaviti da su približno jednako distribuirane kao i kod prethodnih generacija. S ovakvim informacijama sustav bi se mogao koristiti kao pomoć pri stvaranju ispita tako da predlaže pitanja koja bi generaciju adekvatno ocijenila. Sustav bi tako za vrijeme izrade ispita nastavniku mogao prikazivati predikciju distribucije ocjena.

Sljedeći zahtjev ovakvog sustava bio bi izračun ili postavljanje funkcije odgovora na zadatke (logističke krivulje zadatka) za sve zadatke koji se mogu pojaviti u ispitu. Nastavnici bi trebali biti pažljivi pri izgradnji ovih krivulja kako bi pokrili cijeli raspon razina sposobnosti koje bi studenti mogli postići za vrijeme ispita. Dodatno bi trebalo paziti da se ne dogodi situacija u kojoj sustav nema više zadataka koje bi mogao ponuditi studentu (npr. nema više težih zadataka, a student je na prethodni dao točan odgovor). Ova situacija mogla bi se riješiti tako da se studentu zada najteži ili najlakši dostupni zadatak ovisno o točnosti odgovora na prethodno zadani zadatak.

U sljedećem koraku bitno je pronaći ili razviti prikladni algoritam koji će na temelju uspjeha studenata donositi odluku o sljedećem zadatku. Na ovaj način bi svaki ispit bio personaliziran studentu koji ga piše. Za vrijeme ispita sustav bi na temelju ispravnosti odgovora studentu trebao zadati teži zadatak ako je na prethodni dao točan odgovor te lakši zadatak ako je na prethodni dao pogrešan odgovor.

Nadalje sustav bi u pozadini cijelo vrijeme trebao mijenjati pokazatelj uspješnosti studenta. Iznos promjene trebao bi ovisiti o točnosti odgovora na zadatak te težini zadanog zadatka i trenutnoj vrijednosti pokazatelja uspješnosti. Tako bi se npr. studentu koji ima nizak pokazatelj uspješnosti, a točno odgovori na jako težak zadatak pokazatelj trebao više povećati nego kada bi s istim pokazateljem riješio lagani zadatak.

Problem koji se javlja i nije spomenut do sada je problem zadataka koji se ne mogu automatski ocijeniti. Jedno rješenje za takve zadatke (npr. esejska pitanja) bilo bi da se studentima zadaju na samom kraju ispita ili u pravilnim razmacima unutar ispita. Na težinu zadataka takvog tipa koje sustav odabere utjecao bi opći uspjeh studenta do trenutka odabiranja zadatka.

Na primjer, ako je plan svakom studentu zadati 3 zadatka koja se ne daju automatski ocijeniti, sustav bi studentu čija je promjena ocjene bila velika mogao zadati više težih zadataka. S druge strane, studentu za kojeg je najveća težina zadatka na koji je dao točan odgovor bila srednja, sustav bi mogao dati maksimalno tu težinu zadatka.

Konačno, način ocjenjivanja studenata sveo bi se na tri mogućnosti:

1. Ocijeniti studenta na temelju konačnog iznosa pokazatelja uspješnosti kojeg je stekao nakon pisanja ispita
2. Ocijeniti studenta na temelju broja lakših i težih zadataka na koje je točno odgovorio
3. Ocijeniti studenta na temelju razlike početnog i konačnog iznosa pokazatelja uspješnosti

Svi ovi načini ocjenjivanja bi u obzir trebali uzeti i točnost odgovora na zadatke koji se ne daju automatski ocijeniti.

U slučaju da se studenti ocjenjuju prema konačnom iznosu pokazatelja uspješnosti, raspon u kojem se pokazatelj uspješnosti nalazi na kraju testa bi odredio konačnu ocjenu studen4ta na ispitu. Teorijski primjer prikazan je tablicom (Tablica 6).

Tablica 6 Primjer mogućeg ocjenjivanja studenta ovisno o pokazatelju uspješnosti

Min. θ (uključivo)	Max. θ (isključivo)	Ocjena
-∞	1.5	1
1.5	3.0	2
3.0	4.0	3
4.0	5.5	4
5.5	+∞	5

5. Model sustava za provedbu prilagodljivih vježbi

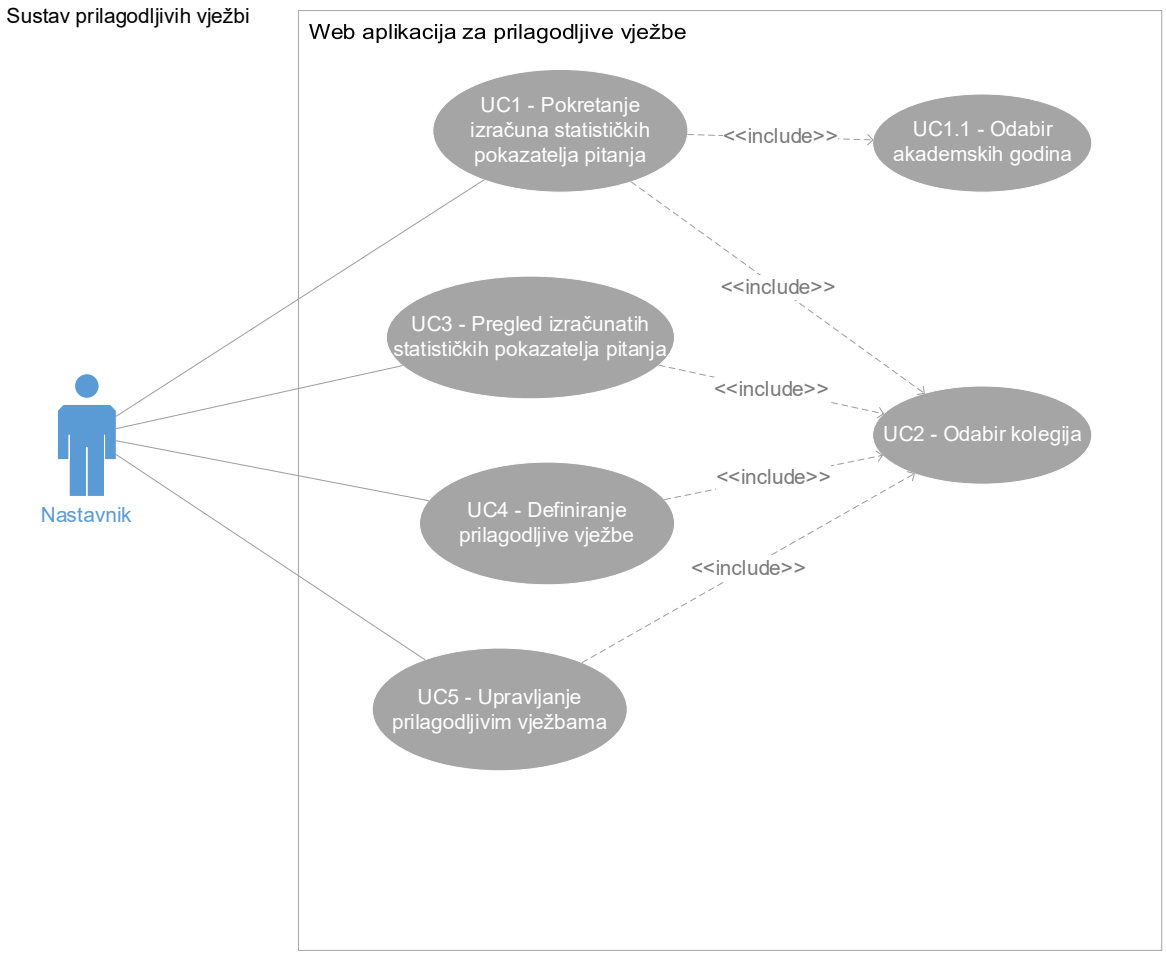
Sustav za provedbu prilagodljivih vježbi napravljen je prema osnovama teorije odgovora na stavke. Za prilagodljive vježbe zato je potrebno izračunati statističke pokazatelje za zadatke i omogućiti njihov pregled i korištenje u svrhu provođenja prilagodljivih vježbi. Izračun pokazatelja ključna je funkcionalnost ovakvog sustava jer stvara temelj za izgradnju karakterističnih logističkih krivulja za zadatke.

5.1. Funkcionalni zahtjevi sustava

Kako bi zadovoljio potrebe provođenja prilagodljivih vježbi, sustav mora omogućiti računanje i praćenje statističkih pokazatelja. Za izračunate vrijednosti pokazatelja pojedinih zadataka potrebno je omogućiti pregled odgovarajućim vizualizacijama. Na temelju izračunatih statističkih pokazatelja zadataka, sustav treba omogućiti izgradnju prilagodljivih vježbi i studentima dati mogućnost pristupa prilagodljivim vježbama kako bi vježbali zadatke. Težina zadataka koja se nudi studentima za vrijeme vježbe trebala bi ovisiti o uspješnosti studenta tijekom vježbe.

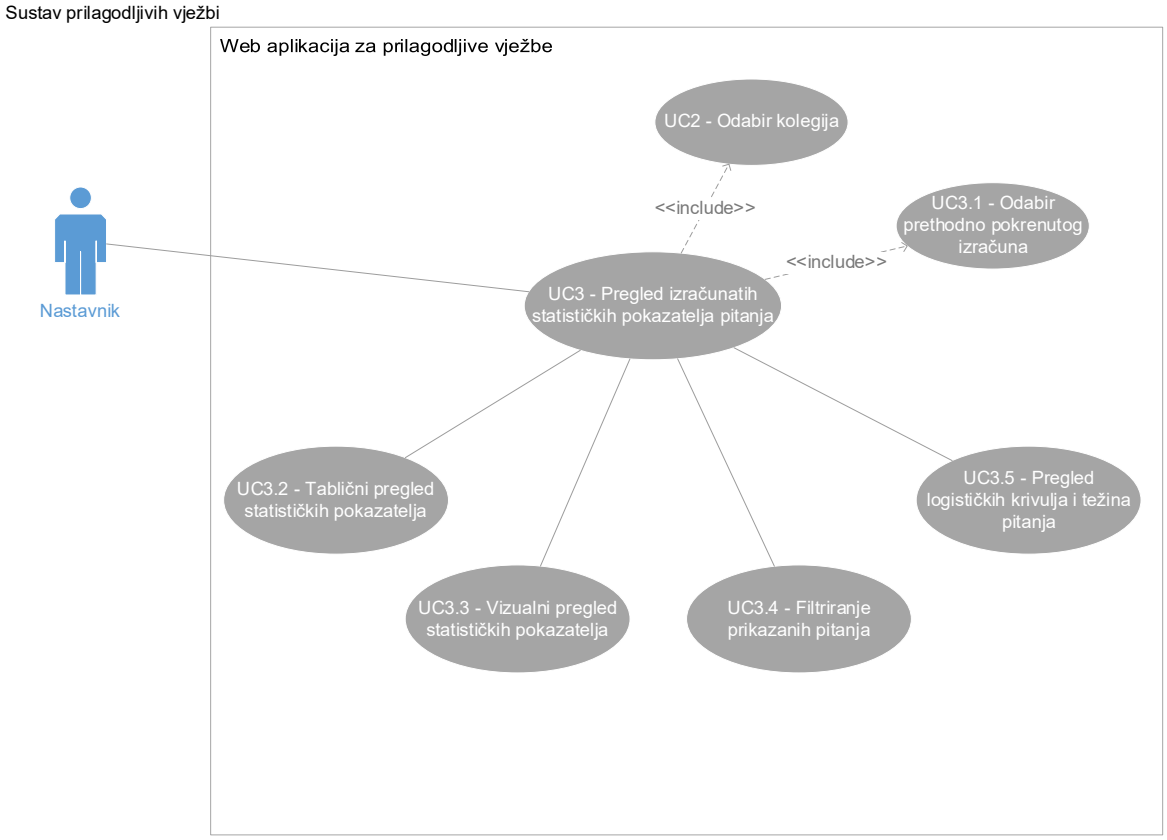
5.1.1. Zahtjevi vezani uz prilagodljive vježbe – nastavnik

Funkcionalni zahtjevi koji su vezani uz nastavnike prikazani su UML (*eng. unified modeling language*) dijagramom obrazaca uporabe (*eng. use case diagram*) na slici (Slika 6). Nastavnik bi u sustavu trebao moći pokrenuti računanje statističkih pokazatelja za zadatke. Ovo radi na razini odabranog kolegija i akademskih godina za koje želi napraviti izračun statističkih pokazatelja (Slika 6, UC1).



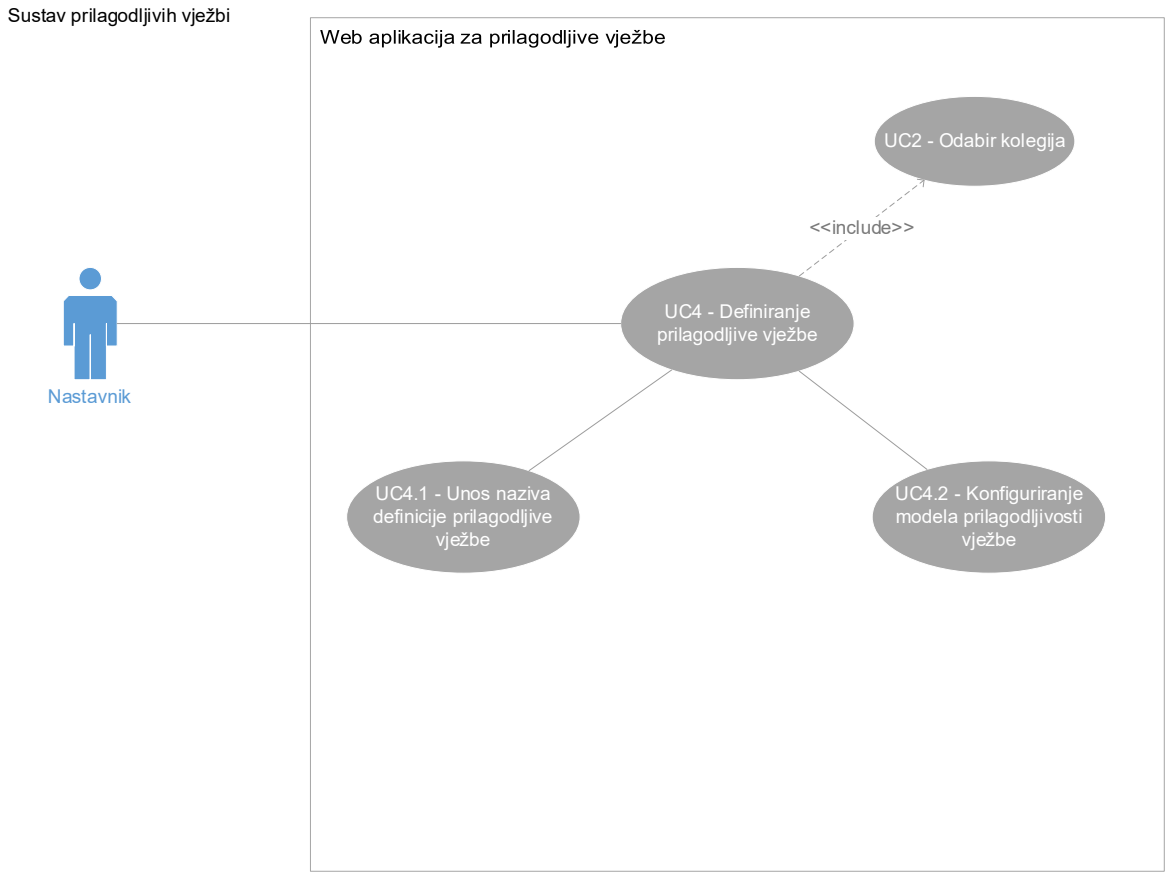
Slika 6 Prikaz UML dijagrama obrazaca uporabe funkcionalnih zahtjeva nastavnika

Nakon što u sustavu postoje izračuni statističkih pokazatelja za zadatke, nastavniku je potrebno omogućiti pregled izračunatih pokazatelja. Ovaj slučaj prikazan je na slici (Slika 7).



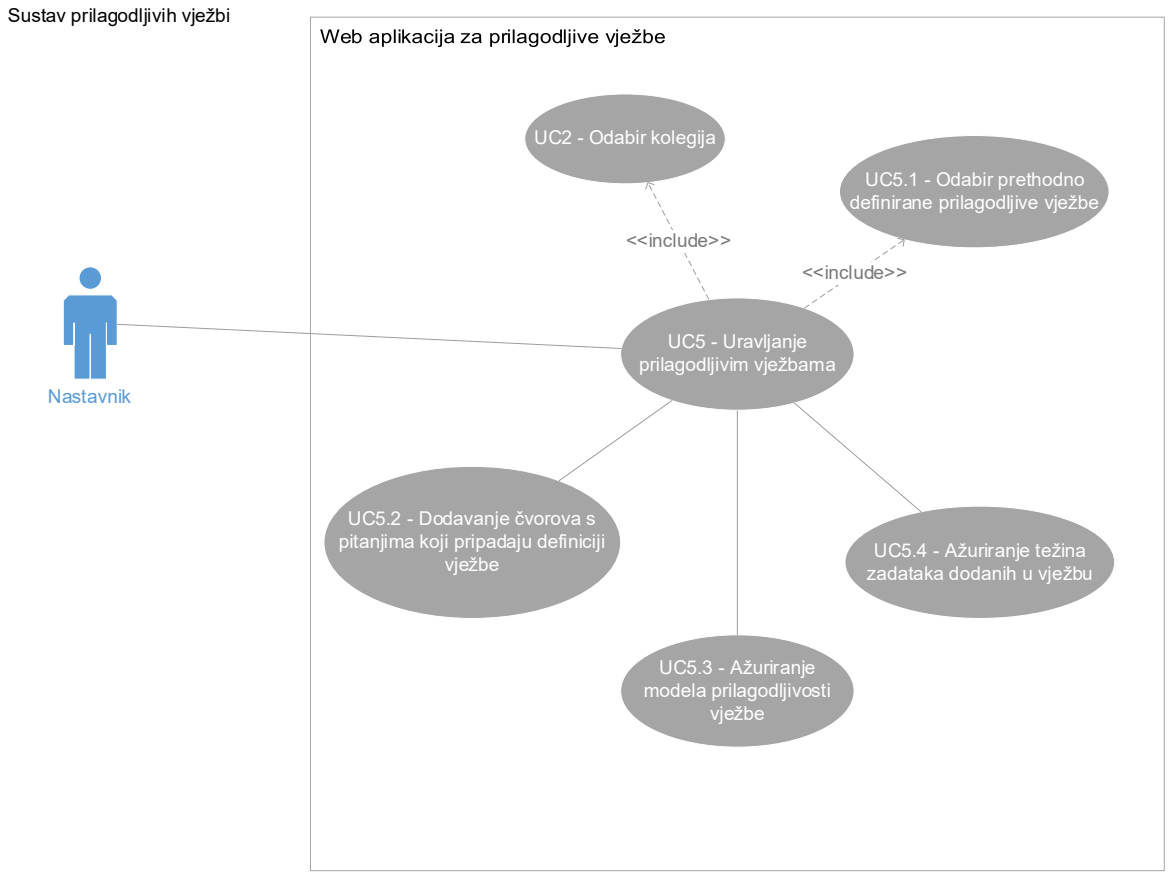
Slika 7 Prikaz obrasca uporabe pregleda statističkih pokazatelja

Nastavniku bi sustav trebao omogućiti stvaranje definicija prilagodljivih vježbi. Tako bi sustav omogućio jednostavno grupiranje zadataka koji su vezani uz neku tematiku (npr. programiranje u programskom jeziku C) i omogućio vježbanje konkretne sposobnosti. Ovaj slučaj korištenja prikazan je slikom (Slika 8).



Slika 8 Prikaz obrasca korištenja definiranja prilagodljivih vježbi

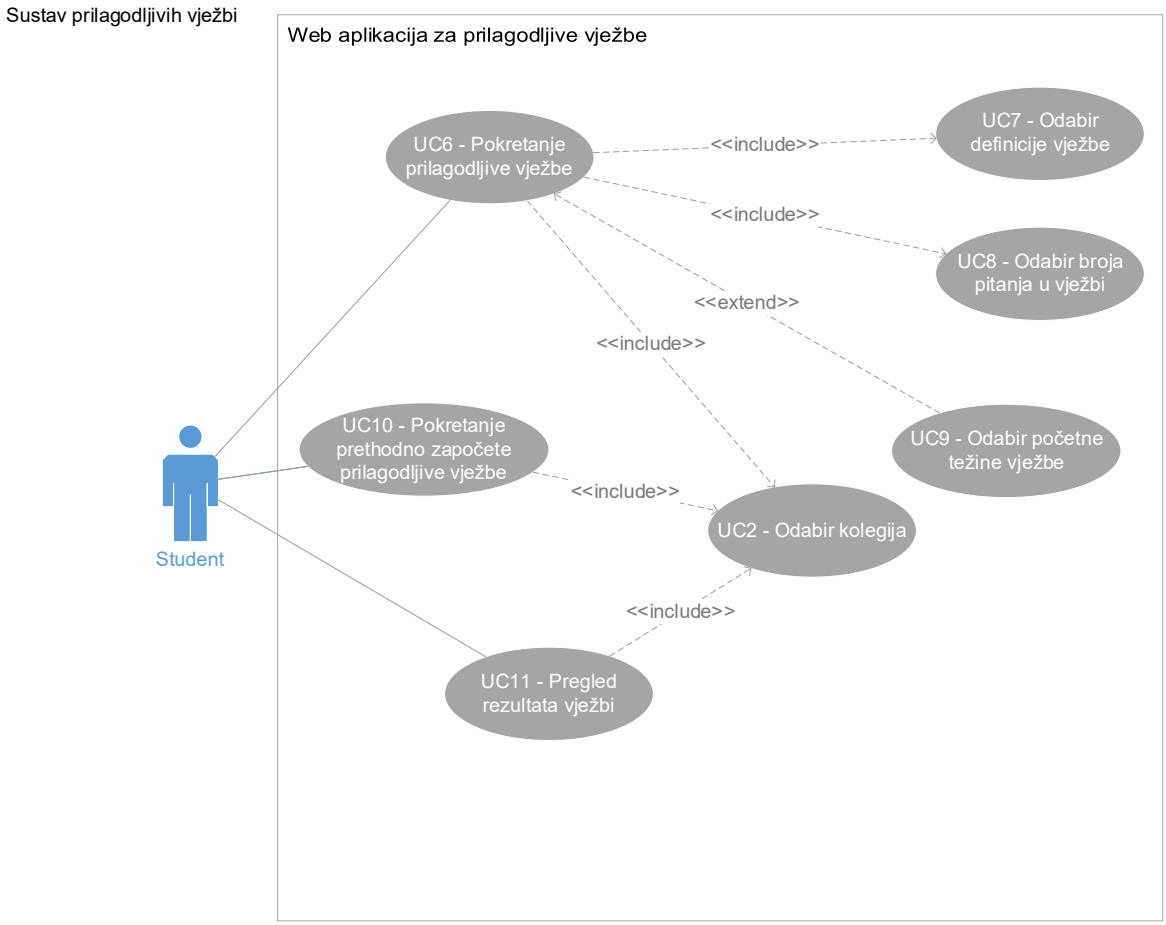
Nakon stvaranja definicije prilagodljive vježbe, nastavnik bi ju trebao moći uređivati. Uređivanje vježbe može uključivati dodavanje novih zadataka u skup zadataka (*eng. question pool*) iz kojeg se studentu zadaju zadatci za vrijeme vježbe. Također, nastavnik bi trebao moći ažurirati težine pojedinih zadataka ako smatra da sustav nije dobro procijenio težinu zadatka. Pregled ovih funkcionalnosti prikazan je slikom (Slika 9).



Slika 9 Prikaz obrasca uporabe upravljanja prilagodljivim vježbama

5.1.2. Zahtjevi vezani uz prilagodljive vježbe – student

Funkcionalni zahtjevi povezani uz studente prikazani su UML dijagramom obrazaca uporabe na slici (Slika 10).



Slika 10 Prikaz obrazaca uporabe sustava prilagodljivih vježbi za studente

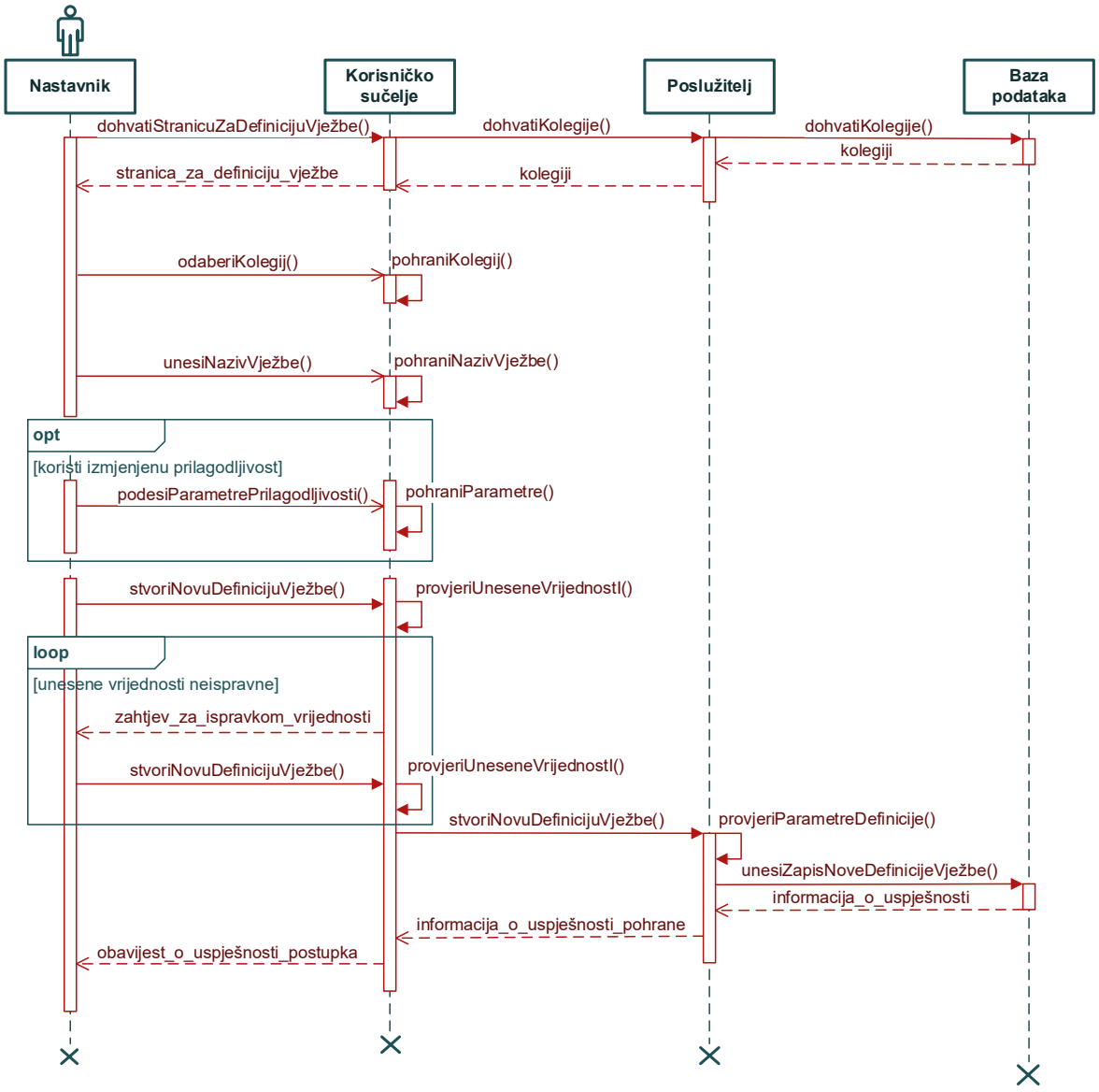
Nakon što odabere kolegij (na slici UC2), student bi pri pokretanju prilagodljivih vježbi trebao moći odabrati koje zadatke želi vježbati (na slici UC7). To radi odabirom jedne od definicija vježbi koje su prethodno za kolegij pripremili nastavnici. Student također mora moći odabrati kroz koliko zadataka želi da ga sustav provede (na slici UC8). Student može odabrati i težinu zadatka od koje bi htio započeti prilagodljivu vježbu (na slici UC9).

Sustav bi studentu koji je u nekom trenutku prekinuo vježbu trebao omogućiti da se vrati na zadatak na kojem je stao (na slici UC10).

Nakon završetka vježbe, student bi trebao moći pregledati postignuti rezultat na toj vježbi i na svim prethodnim vježbama koje je pokrenuo (na slici UC11).

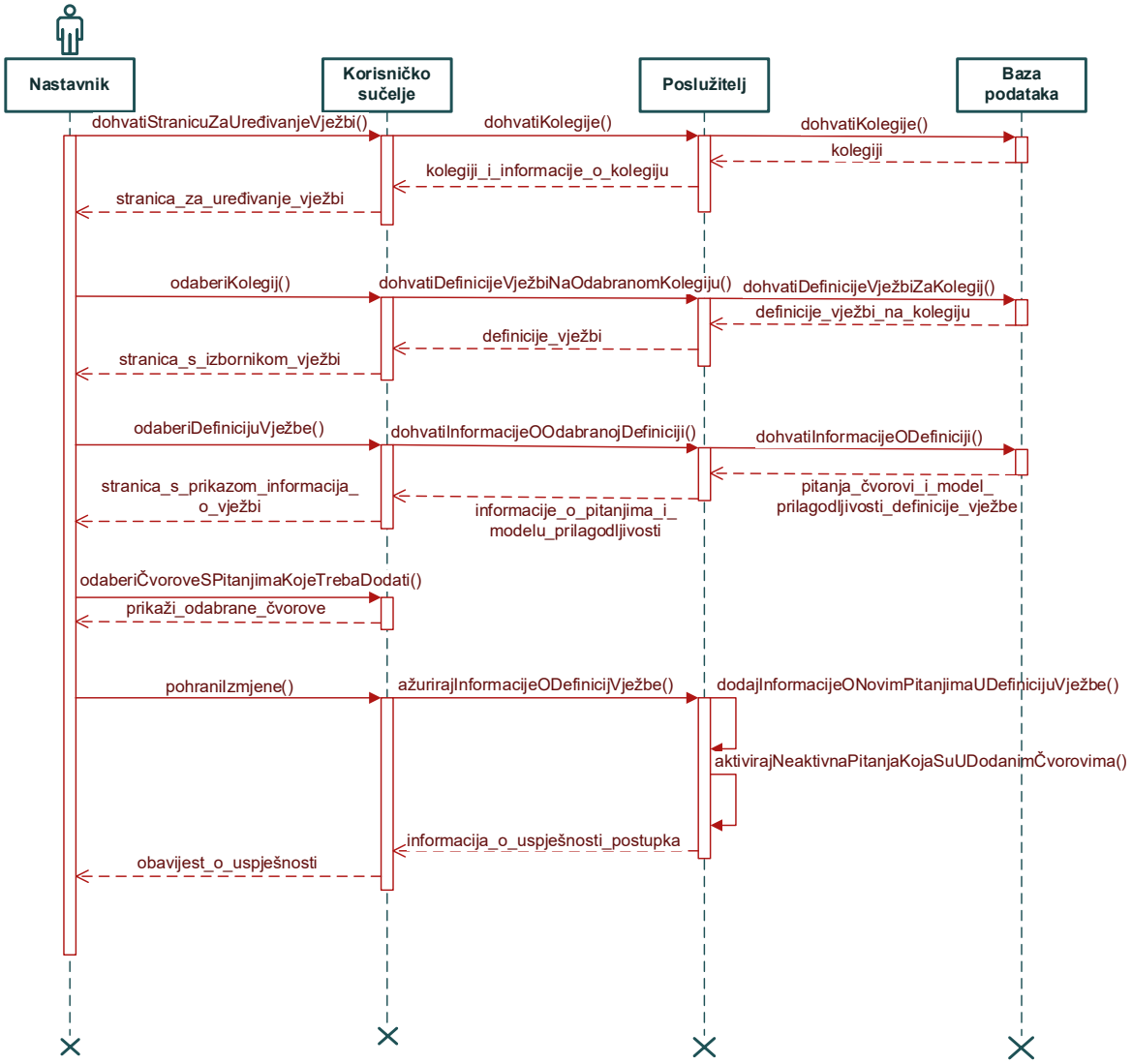
5.1.3. Opis toka definicije prilagodljivih vježbi

Kako bi prilagodljiva vježba mogla postojati u sustavu prvo bi trebalo stvoriti njezinu definiciju. Ovo radi nastavnik, a proces stvaranja prilagodljive vježbe prikazan je UML sekvencijskim dijagramom (*eng. UML sequence diagram*) na slici (Slika 11).



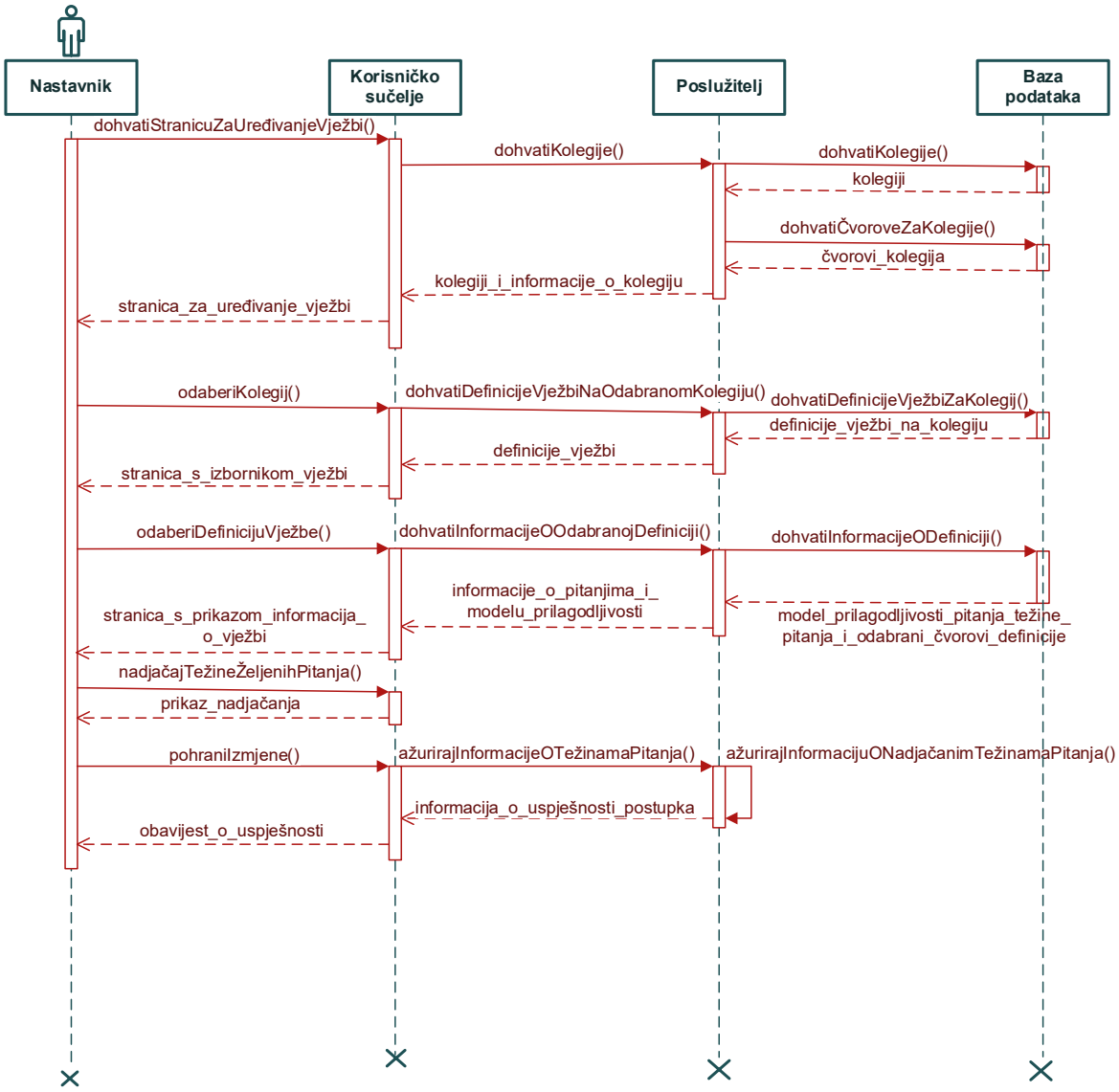
Slika 11 Sekvencijski dijagram stvaranja nove definicije vježbe na kolegiju

Nakon definiranja vježbe, nastavnik u vježbu mora dodati zadatke. Kako nastavnik ne bi dodavao zadatke pojedinačno, može dodati skup zadataka koji se nalaze u istom čvoru. Proces dodavanja zadataka u prilagodljivu vježbu prikazan je sekvencijskim dijagramom na slici (Slika 12).



Slika 12 Sekvencijski dijagram dodavanja novih zadataka u prilagodljivu vježbu

Ako nastavnik nije zadovoljan procijenjenim težinama zadataka, mora moći težinu zadatka nadjačati na razini vježbe. Ovaj proces prikazan je sekvencijskim dijagramom na slici (Slika 13).

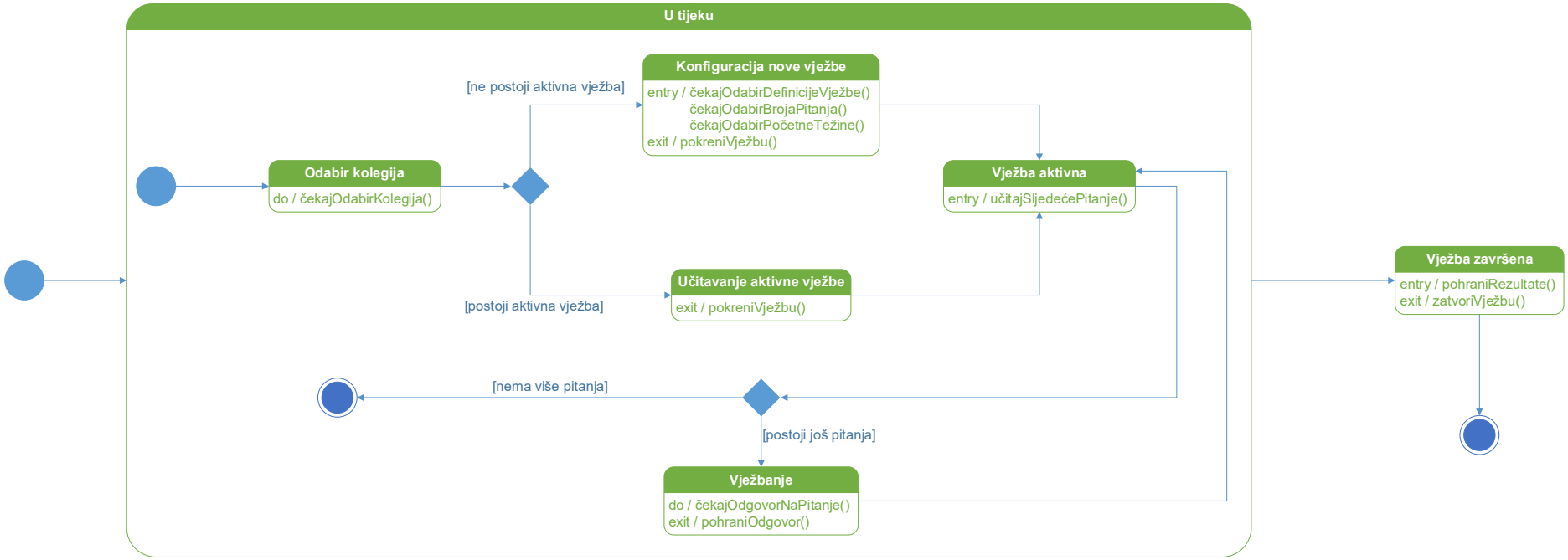


Slika 13 Sekvencijski dijagram nadjačavanja težine zadatka

5.1.4. Opis toka provođenja prilagodljivih vježbi

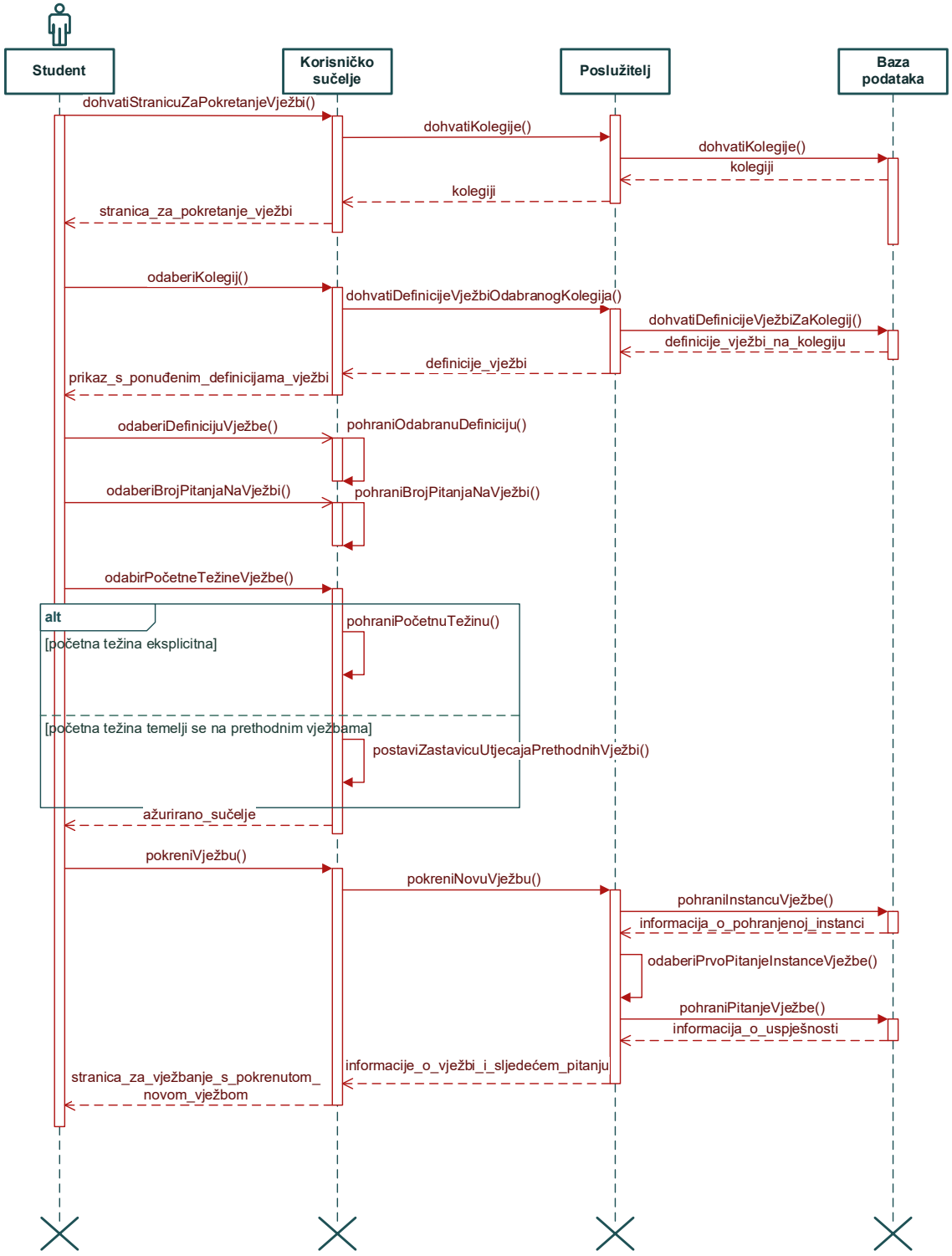
Za vrijeme provođenja prilagodljive vježbe sustav mora znati na kojem se zadatku student trenutno nalazi. Osim toga, sustav mora moći na temelju točnosti odgovora na prethodni zadatak odlučiti koji zadatak će sljedeći zadati studentu. Proces kroz koji sustav provodi prilagodljive vježbe moguće je opisati automatom.

Vježba se nalazi u jednom od dva stanja: stanje *„u toku“* ili *„završena“*. Dok je vježba u stanju *„u toku“* studentu zadaje zadatke i čeka na odgovor. U stanje *„završena“* vježba prelazi kada student da odgovor na posljednji zadatak. Ovaj proces prikazan je UML dijagramom stanja (*eng. UML state machine diagram*) na slici (Slika 14).



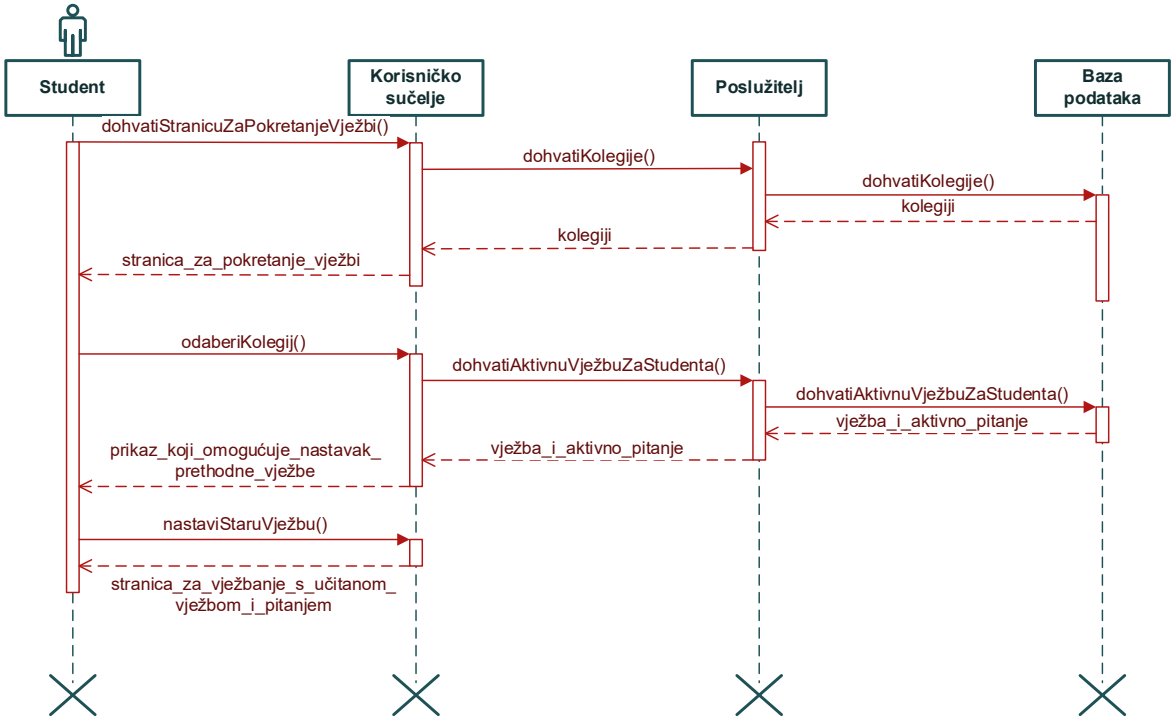
Slika 14 Dijagram stanja prilagodljive vježbe za vrijeme vježbanja

Nakon što je nastavnik definirao novu prilagodljivu vježbu, studenti mogu započeti vježbati zadatke te definicije. Nakon odabira kolegija na kojem želi pokrenuti prilagodljivu vježbu student može odabrati jednu od ponuđenih vježbi koje su sastavili nastavnici. Proces pokretanja nove vježbe prikazan je sekvencijskim dijagramom na slici (Slika 15).



Slika 15 Sekvencijski dijagram koji prikazuje proces pokretanja prilagodljive vježbe

Student također može pokrenuti prethodno pokrenutu vježbu na kolegiju. Budući da student može odabrati da prethodne vježbe utječu na njegovu početnu točku u novoj vježbi, nema smisla omogućiti studentu više aktivnih vježbi na jednom kolegiju. Proces nastavka prethodno pokrenute vježbe prikazan je sekvencijskim dijagramom na slici (Slika 16).



Slika 16 Sekvencijski dijagram nastavka prethodno pokrenute vježbe

5.2. Nefunkcionalni zahtjevi

Budući da sustav mora moći ispravno komunicirati i integrirati se sa sustavom *Edgar*, potrebno je zadovoljiti neke osnovne nefunkcionalne zahtjeve.

Sustav bi trebao biti implementiran kao modul kako se u kod sustava *Edgar* ne bi dodavale dodatne značajke. Na ovakav način sustav je moguće zasebno pustiti u pogon, a sustav *Edgar* bi ovaj sustav u pozadini koristio za provođenje prilagodljivih vježbi. Ovakav način razvoja sustava za provođenje prilagodljivih vježbi kao modula također omogućuje i kasnije umnažanje kako bi se postiglo balansiranje poslova (*eng. load balancing*) ako za to bude postojala potreba.

Za potrebe računanja statističkih pokazatelja trebalo bi iskoristiti sustav *Judge0* kojeg sustav *Edgar* već koristi. Ovo je potrebno kako bi se smanjile ovisnosti o konkretnim implementacijama procesa izračunavanja statističkih parametara. Trenutno je moguće ovaj proces provesti uz pomoć programskog jezika *R* koji je razvijen upravo za potrebe statističkih izračuna. Međutim, ako bi se u budućnosti javila potreba za implementacijom računanja u nekom drugom programskom jeziku, integracija sa sustavom *Judge0* uvelike će olakšati postupak prilagodbe.

Nadalje, izračun statističkih parametara zadataka trebao bi biti automatiziran i kontinuiran. Ovo je potrebno kako bi podatci o statistikama zadataka bili ažurni u slučaju da se zadatci, za koje su parametri prethodno izračunati, pojave u nekom novom testu (ispitu).

Aplikacija koja će se razviti za potrebu provođenja prilagodljivih vježbi mora biti jednostranična. Budući da je sučelje sustava *Edgar* razvijeno u radnom okviru za razvoj jednostraničnih aplikacija, u budućnosti, kada se javi potreba, rješenje se jednostavno može integrirati u postojeće sučelje.

6. Razvijeni sustav za provođenje prilagodljivih vježbi

Prema funkcionalnim i nefunkcionalnim zahtjevima razvijen je sustav za provođenje prilagodljivih vježbi. Razvijeni sustav sastoji se od tri dijela:

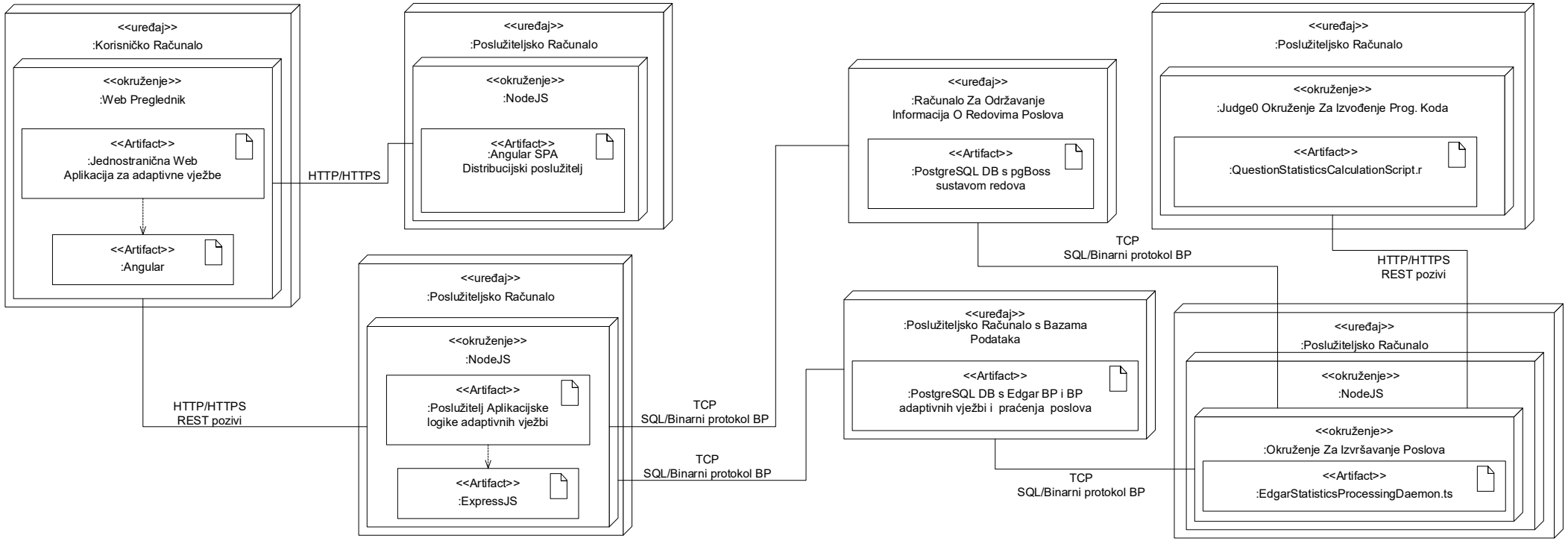
1. Modul za izvođenje poslova
2. Servis za računanje statističkih pokazatelja
3. Web aplikacija za vježbanje i demonstraciju implementiranih funkcionalnosti

Modul za izvođenje poslova osnova je za izgradnju cjelokupnog sustava za provođenje prilagodljivih vježbi. Modul je zadužen za izvođenje poslova, a servis za računanje statističkih pokazatelja implementiran je kao posao koji se izvršava unutar modula za izvođenje poslova. Modul za izvođenje poslova je općenit i može se koristiti i za druge potrebe čak i izvan *Edgara*.

Servis za računanje statističkih pokazatelja oslanja se na modul za izvršavanje poslova kako bi definirao i pokrenuo posao izračunavanja. Servis je zadužen za zaprimanje zahtjeva za izračun statističkih pokazatelja i prosljeđivanje tih zahtjeva u formatu prikladnom za obradu u modulu za izvođenje poslova. Osim toga, servis implementira određene korake posla koji su potrebni kako bi se posao izračuna statističkih pokazatelja ispravno proveo.

Web aplikacija za vježbanje i demonstraciju implementiranih funkcionalnosti omogućuje interakciju korisnika s implementiranim sustavom. Ova aplikacija zadužena je za definiciju i provođenje prilagodljivih vježbi.

UML dijagram ugradnje (*eng. UML deployment diagram*) implementiranog sustava za provođenje prilagodljivih vježbi prikazan je na slici (Slika 17).



Slika 17 Dijagram ugradnje razvijenog sustava

6.1. Modul za izvođenje poslova

Modul za izvođenje poslova osnovni je dio implementiranog sustava. Ovaj modul omogućuje definiciju i izvođenje proizvoljnog posla. Zadaća modula je da apstrahira posao i omogući jednostavnu implementaciju apstrahiranog posla.

Osnovna ideja iza modula je razbijanje poslova na određeni niz koraka. Svaki posao treba implementirati sljedeće komponente i strukturu kako bi se mogao izvoditi unutar modula:

1. Inicijalni dohvat radnog skupa podataka (1) – korak u kojem se za posao stvara inicijalni ulaz, odnosno dohvaćaju se podatci nad kojima će se posao izvršavati (*eng. initial input*)
2. Izvršitelj posla (1) – definicija implementacije izvršitelja posla koja omogućuje stvaranje konteksta za korake posla
3. Koraci posla (n) – proizvoljan niz koraka koji se moraju izvršiti određenim redoslijedom
4. Korak pohrane rezultata (1) – korak u kojem se konačni rezultat izvršavanja niza koraka odbacuje ili pohranjuje na proizvoljan način

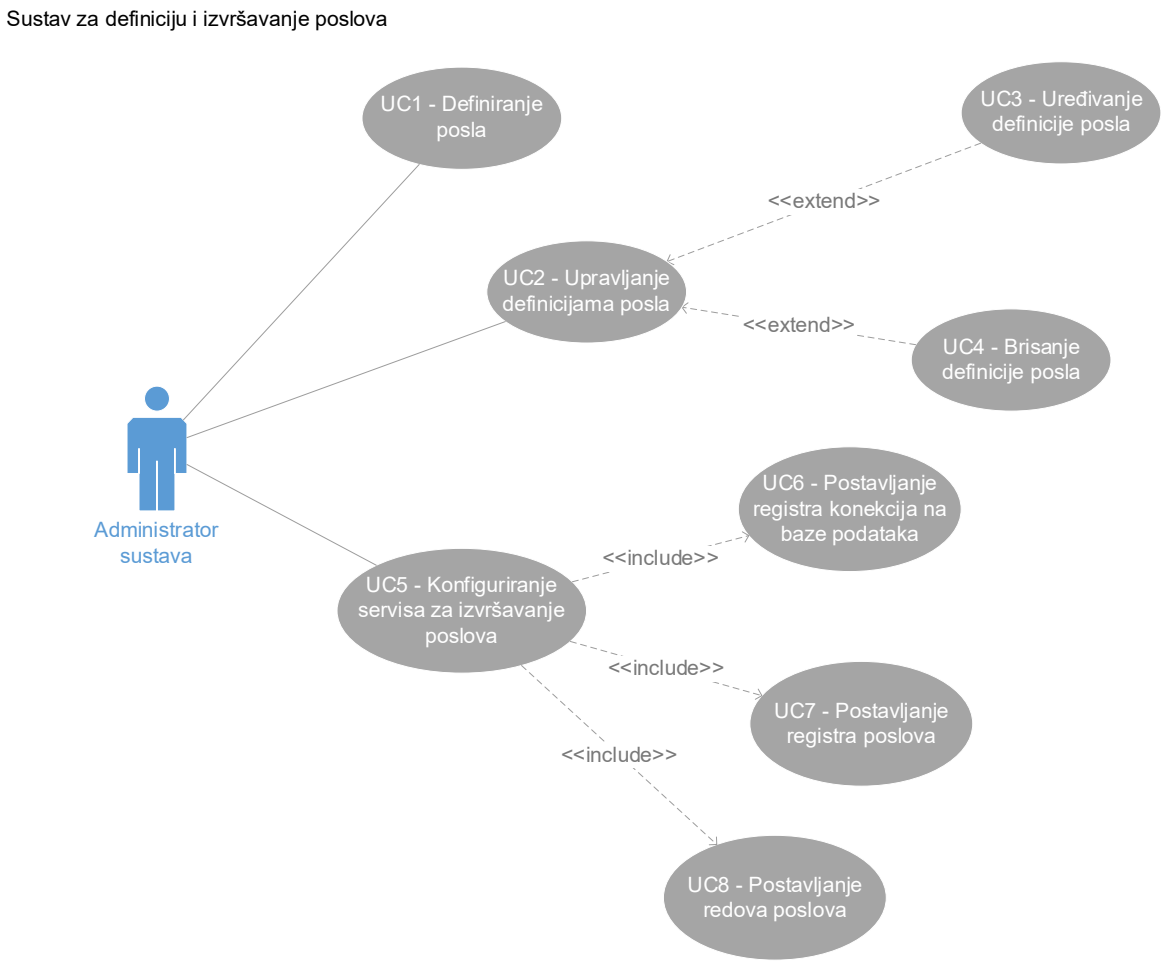
Korištenjem ovakve definicije poslova, moguće je opisati bilo kakav posao koji se treba izvršiti. Na primjer, korisnik bi tako mogao opisati posao slanja poruka. U prvom koraku dohvatile bi se sve poruke koje treba poslati. Nakon toga dohvaćene poruke poslale bi se u drugi korak (izvršitelj posla) gdje bi se obradile u nekoliko koraka (priprema, dohvat priloga, slanje i brisanje poslanih poruka iz reda čekanja). Na kraju bi se mogle pohraniti informacije o vremenu slanja pojedine poruke te stvorili zapisi o porukama koje se nisu uspjele poslati. Detaljniji opis primjera nalazi se u poglavlju *6.1.3 – Aplikacijska logika izvršavanja poslova*.

Također, svakom definiranom poslu moguće je na jednostavan način promijeniti način provođenja nekog od koraka bez da se mijenja implementacija ostalih dijelova posla. Ovaj dio sustava prati SOLID[3] principe objektno orijentiranog razvoja te time omogućava dobru interoperabilnost i jednostavniju nadogradnju i zamjenu dijelova sustava.

[3] SOLID – engleska skraćenica za 5 osnovnih OOP načela – *Načelo jedinstvene odgovornosti (eng. Single responsibility principle), Načelo otvoren-zatvoren (eng. Open-closed principle), Načelo Liskovine supstitucije (eng. Lisk's substitution principle), Načelo segregacije sučelja (eng. Interface segregation principle) i Načelo inverzije ovisnosty (eng. Dependency inversion principle)*

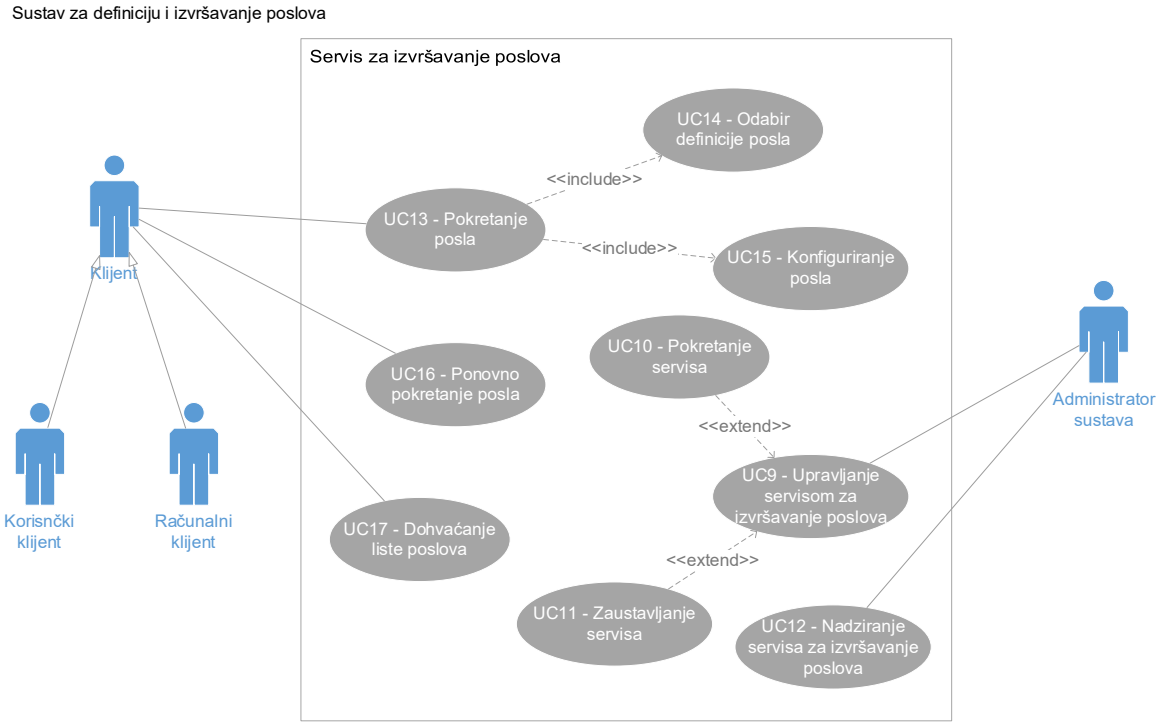
6.1.1. Modeliranje modula za izvođenje poslova

Za proces izvođenja poslova korisniku je potrebno omogućiti da sustavu zada posao i u nekom trenutku pokrene njegovo izvršavanje. Općeniti zahtjevi za ovakav sustav prikazani su UML dijagramom obrazaca uporabe (Slika 18).



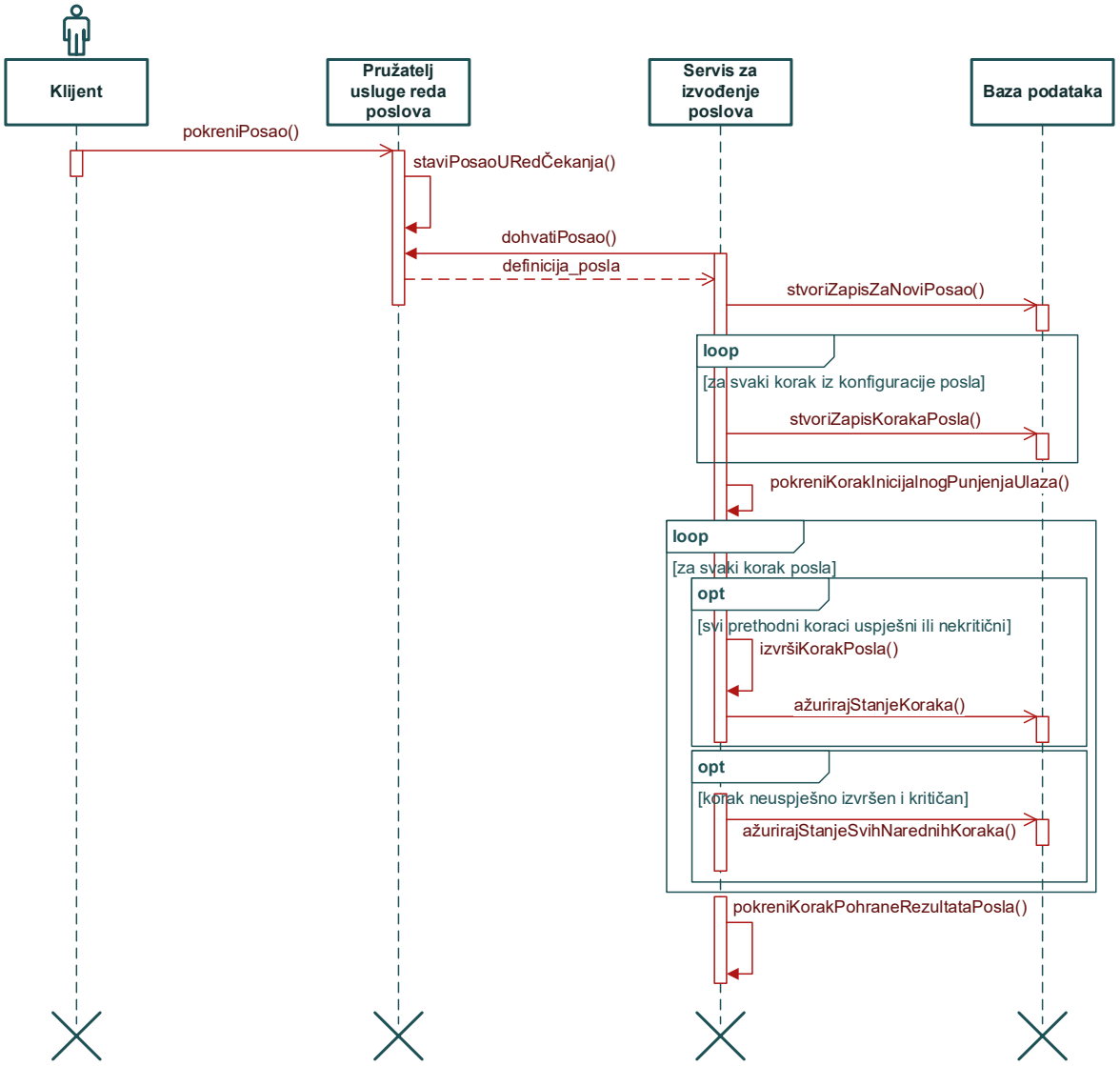
Slika 18 Dijagram općenitih obrazaca uporabe sustava za izvršavanje poslova

U sklopu ovog sustava moguće je pokrenuti više servisa za izvođenje poslova. Ovi servisi su zaduženi za izvršavanje posla kojeg je zadao neki od korisnika. U teoriji jedan servis unutar modula za izvođenje poslova bi izvodio poslove određenog tipa. Zahtjevi servisa sustava za izvršavanje poslova prikazani su dijagramom obrazaca uporabe (Slika 19).



Slika 19 Dijagram obrazaca uporabe servisa za izvršavanje poslova

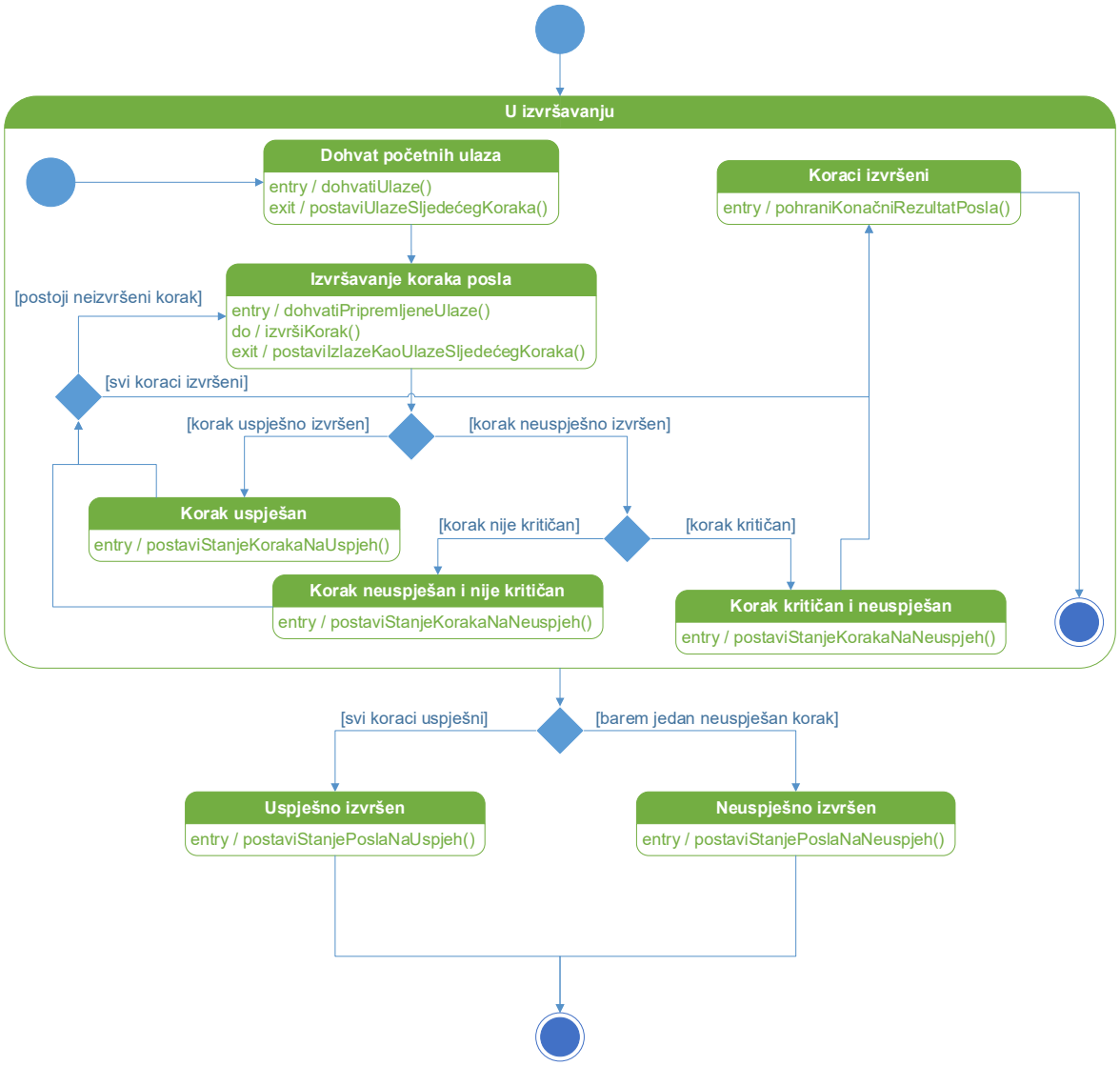
Proces zadavanja i pokretanja posla opisan je sekvencijskim dijagramom (Slika 20).



Slika 20 Sekvencijski dijagram koji prikazuje proces izvršavanja posla

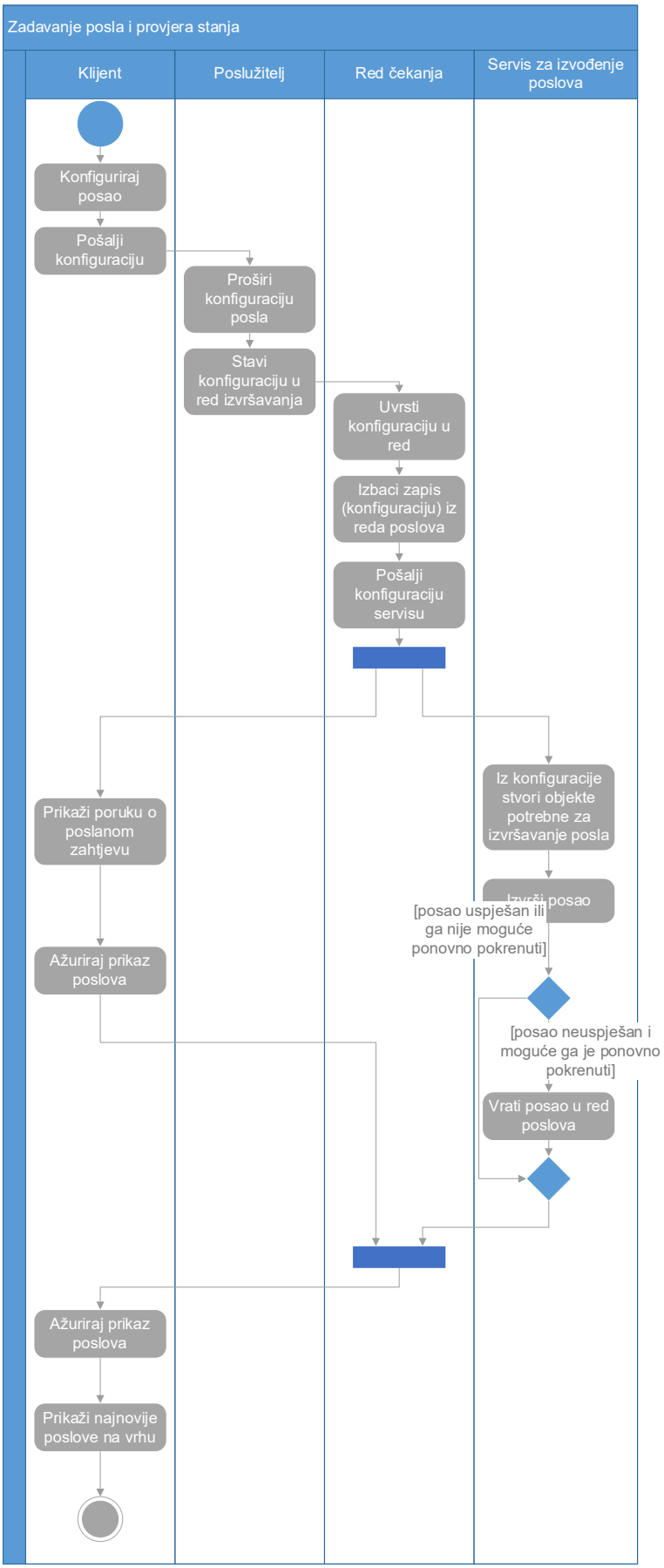
Klijent sustavu (servisu za izvođenje poslova) šalje (asinkroni) zahtjev za izvršavanjem posla. Klijentski zahtjev pohranjuje se u red čekanja zahtjeva (*eng. request queue*) te u nekom trenutku dospije do servisa za izvođenje poslova. Nakon dolaska u servis posao je došao na red i počinje se izvršavati.

Posao se može naći u stanjima *„u izvršavanju“*, *„neuspješno izvršen“* i *„uspješno izvršen“*. Načini na koje posao ulazi u pojedino stanje opisani su dijagramom stanja (Slika 21).



Slika 21 Dijagram stanja posla

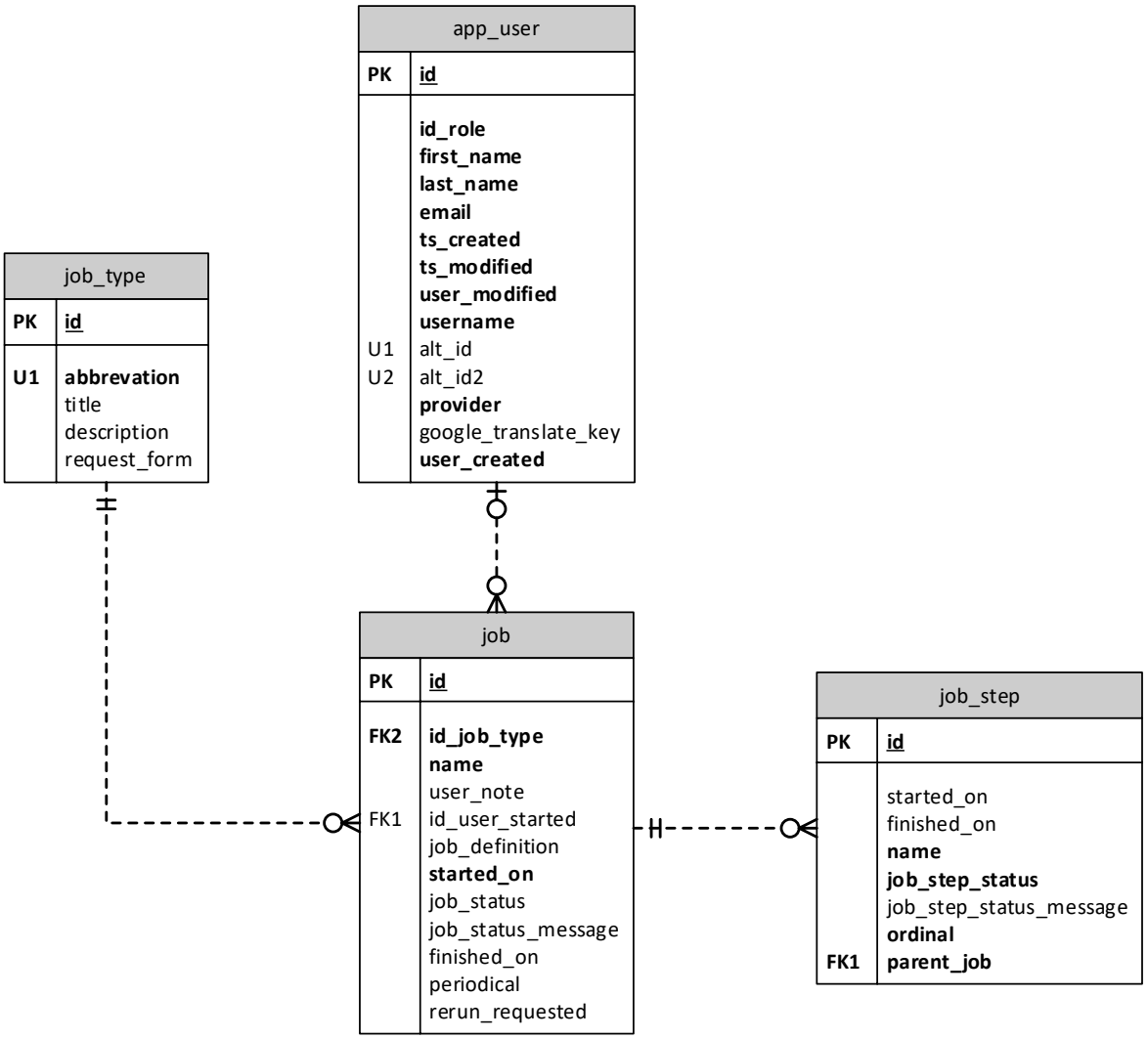
U bilo kojem trenutku klijent može poslati zahtjev aplikaciji koja implementira pregled poslova koje sustav obavlja i tako dobiti informaciju o stanju pojedinog posla. Ovaj proces u kombinaciji s procesom zadavanja i pokretanja posla prikazan je UML dijagramom aktivnosti (*eng. UML activity diagram*) na slici (Slika 22).



Slika 22 Dijagram aktivnosti zadavanja i pokretanja te provjere stanja poslova

6.1.2. Relacijski model baze podataka modula

Modul za izvođenje poslova zahtjeva praćenje izvođenja poslova i njihovih koraka. Za tu potrebu koristi objektno-relacijsku bazu podataka budući da je potrebno pohranjivati definicije poslova koje su u JSON[4] formatu (atribut *job_definition* relacije *job* i atribut *request_form* relacije *job_type*). Relacijski model prikazan je na slici (Slika 23).



Slika 23 Relacijski model dijela baze podataka (shema *job_tracking_schema* i povezane relacije iz sheme *public*) vezanog za modul izvršavanja poslova; atributi tipa JSON su atribut *job_definition* relacije *job* te atribut *request_form* relacije *job_type*

[4] JSON – *JavaScript Object Notation*, način serijalizacije objekata koji se koriste u programskom jeziku *JavaScript*

Relacija *job_type* služi za pohranu informacija o određenim tipovima posla. Ova relacija omogućuje definiciju novih oblika poslova. Detaljan opis relacije nalazi se u tablici (Tablica 7).

Tablica 7 Opis atributa relacije *job_type*

Atribut	PK[5]	UQ[6]	NN[7]	FK[8]	Tip podatka	Opis
id	✓	-	✓	x	INT	-
abbrevation	x	✓	✓	x	VARCHAR(10)	Jedinstvena skraćenica za tip posla
title	x	x	x	x	VARCHAR(512)	Naziv tipa posla
description	x	x	x	x	VARCHAR(1024)	Opis tipa posla
request_form	x	x	x	x	JSON	Forma koja služi za stvaranje zahtjeva izvršavanja posla (za buduće potrebe)

Iz primjera sa slanjem poruka, u ovu relaciju bi se dodao zapis:

abbrevation	title	description	request_form
MSGSND	Posao za slanje poruka.	Posao koji uzima poruke iz reda čekanja, priprema ih, dohvaća pripadajuće priloge poruka te ih šalje na odredište. Nakon uspješnog slanja briše poruku, a ako poruka nije uspješno poslana to zapisuje u bazu podataka.	NULL

Na temelju ovog tipa posla implementirao bi se postupak izvođenja i postavljanja zahtjeva za izvođenjem posla slanja poruka.

[5] PK – primarni ključ (*eng. primary key*)
[6] UQ – jedinstvena vrijednost (*eng. unique*)
[7] NN – unos vrijednosti je obavezan (*eng. not null*)
[8] FK – integritet stranog ključa (*eng. foreign key*)

Relacija *job* služi za pohranu informacija o izvršavanjima poslova. Svaki posao koji započne s izvršavanjem upisuje se u ovu relaciju i kasnije se zapis ažurira ovisno o napretku posla i stanju njegovih koraka. Detaljan opis relacije nalazi se u tablici (Tablica 8).

Tablica 8 Opis atributa relacije *job*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id	✓	-	✓	x	VARCHAR(512)	-
id_job_type	x	x	✓	✓ (job_type[id])	INT	Strani ključ na tip kojem posao pripada
name	x	x	✓	x	VARCHAR(2048)	Naziv posla
user_note	x	x	x	x	TEXT	Dodatne informacije o poslu zanimljive korisnicima
id_user_started	x	x	x	✓ (app_user[id])	INT	Identifikator korisnika koji je započeo posao
job_definition	x	x	x	x	JSON	JSON serijalizirana definicija posla
started_on	x	x	✓	x	TIMESTAMP	Vrijeme početka
job_status	x	x	x	x	ENUM('RUNNING', 'FINISHED', 'FAILED')	Stanje u kojem se posao trenutno nalazi
job_status_message	x	x	x	x	TEXT	Poruka statusa u kojem se posao nalazi
finished_on	x	x	x	x	TIMESTAMP	Vrijeme završetka
periodical	x	x	x	x	BOOLEAN	Zastavica periodičnog izvršavanja posla
rerun_requested	x	x	x	x	BOOLEAN	Zastavica potrebe za ponovnim izvršavanjem posla

Iz primjera sa slanjem poruka, svaki put kada bi neki korisnik postavio zahtjev za slanjem poruka, u ovoj relaciji bi se pojavio novi zapis.

Relacija *job_step* služi za pohranjivanje informacija o koracima posla koji se izvršava. Svaki posao može se sastojati od više koraka. Detaljan opis relacije nalazi se u tablici (Tablica 9).

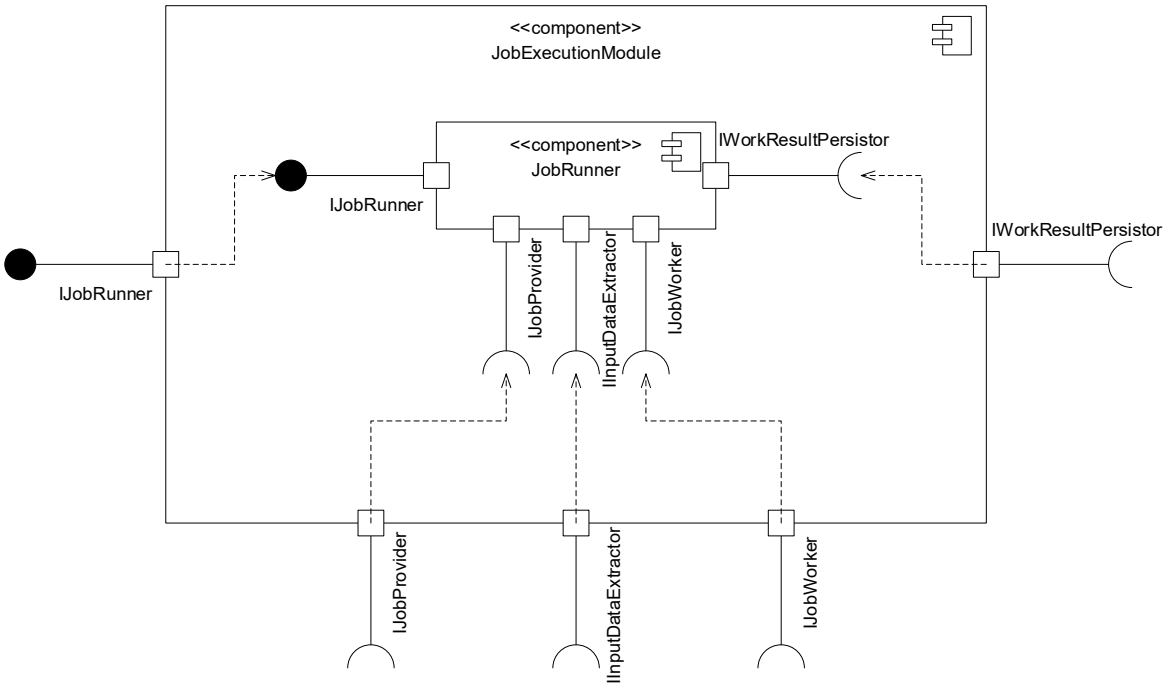
Tablica 9 Opis atributa relacije *job_step*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id	✓	-	✓	x	VARCHAR(512)	-
started_on	x	x	x	x	TIMESTAMP	Vrijeme početka izvršavanja koraka
finished_on	x	x	x	x	TIMESTAMP	Vrijeme završetka izvršavanja koraka
name	x	x	✓	x	VARCHAR(2048)	Naziv koraka
job_step_status	x	x	✓	x	ENUM('NOT_STARTED', 'RUNNING', 'SUCCESS', 'FAILURE', 'SKIP_CHAIN', 'CRITICALLY_ERRORED')	Stanje u kojem se posao trenutno nalazi
job_step_status_message	x	x	x	x	TEXT	Poruka statusa u kojem se posao nalazi
ordinal	x	x	x	x	INT	Dodatne informacije o poslu zanimljive korisnicima
parent_job	x	x	✓	✓ (job[id])	VARCHAR(512)	Identifikator korisnika koji je započeo posao

Iz primjera sa slanjem poruka, svaki put kada bi neki korisnik postavio zahtjev za slanjem poruka, u ovoj relaciji bi se pojavila 4 nova zapisa (po jedan za svaki od koraka posla slanja poruka).

6.1.3. Aplikacijska logika izvršavanja poslova

Modul poslove izvršava unutar komponente *JobRunner*. Ova komponenta dijelove posla izvodi točno određenim redoslijedom. Budući da izvođenje bilo kojeg dijela posla može završiti greškom, *JobRunner* mora paziti da te greške uhvati i postupa na način na koji je to propisao posao. Komponenta *JobRunner* koji je osnova modula za izvršavanje poslova prikazana je UML dijagramom komponenti (*eng. UML component diagram*) na slici (Slika 24).



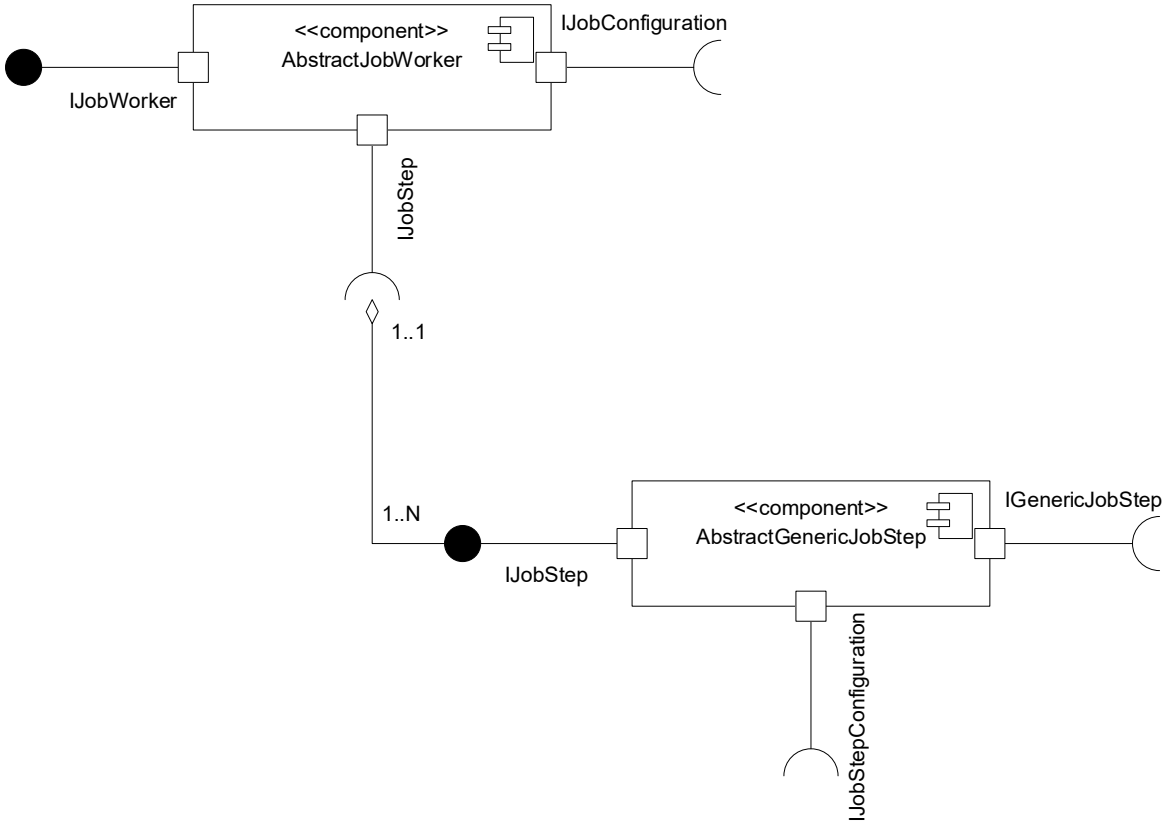
Slika 24 Dijagram komponenti koji opisuje osnovnu građu modula za izvršavanje poslova

Kako bi se poslovi mogli izvršavati, komponenti *JobRunner* potrebno je dati implementacije određenih sučelja. Ova sučelja predstavljaju mjesta na kojima se mogu ubaciti različite implementacije dijelova poslova (*eng. hooks*).

Implementacija sučelja *IJobProvider* treba implementirati funkcionalnost *„stvaranja“* odnosno *„zadavanja“* poslova. U nizu poziva, *JobRunner* prvo poziva metodu za dohvat posla koju upravo implementira sučelje *IJobProvider*.

Nakon što je dobila posao, komponenta *JobRunner* poziva metodu za dohvat inicijalnog ulaza u posao sučelja *IInputDataExtractor*. Rezultat ovog poziva metode predat će se prvom koraku posla.

Implementacija sučelja *IJobWorker* predstavlja kontekst izvođenja posla te sadrži listu koraka posla. Komponenta poziva metode koraka posla redom kako su definirane u listi sadržanoj u implementaciji ovog sučelja. Argument poziva ovih metoda je rezultat prethodnih koraka posla. Agregat koraka posla koji je predstavljen ovim sučeljem prikazan je dijagramom komponenti na slici (Slika 25).



Slika 25 Dijagram komponente za implementaciju sučelja *IJobWorker* preko komponente *AbstractJobworker*

Svaki korak posla konstruira se iz konfiguracije posla. Pri konstrukciji svakog koraka iz konfiguracije se čitaju konfiguracijski parametri koraka koji su modelirani sučeljem *IJobStepConfiguration*. Komponenta *AbstractGenericJobStep* je samo *adapter*[9] prema sučelju *IGenericJobStep* koji kao generički parametar prima konkretni tip konfiguracije posla kako bi se omogućio statički polimorfizam.

[9] adapter – oblikovni obrazac

Na posljetku, ako je svaki korak posla uspješno bio izvršen, komponenta *JobRunner* poziva metodu za pohranu konačnog rezultata koju treba ponuditi implementacija sučelja *IWorkResultPersistor*. Ova interakcija modelira pohranu rezultata posla, ali ovisno o zahtjevima posla rezultati se mogu i ignorirati.

Na primjer, korisnik bi proces periodičkog slanja poruka uz pomoć modula mogao napraviti tako da ponudi implementaciju sučelja *IJobProvider* koja bi nakon određenog vremena (npr. svakih 10 dana) stvorila jednu konfiguraciju posla. Ova konfiguracija bi se referencirala na implementaciju sučelja *IInputDataExtractor* (implementacija bi se mogla zvati *MessageInputDataExtractor*) koju bi korisnik također morao ponuditi, a koja bi pokupila sve poruke iz baze podataka (ili nekog drugog izvora) koje je u međuvremenu stvorio neki transakcijski sustav, web ili neka druga aplikacija.

Ovom poslu nije potreban kontekst pa korisnik može iskoristiti pretpostavljenu implementaciju sučelja *IJobWorker* te njoj zadavati potrebne korake. Konkretno za ovaj primjer, koraci koje bi korisnik mogao implementirati (implementacije sučelja *IJobStep*) i na njih se referencirati u konfiguraciji posla bili bi:

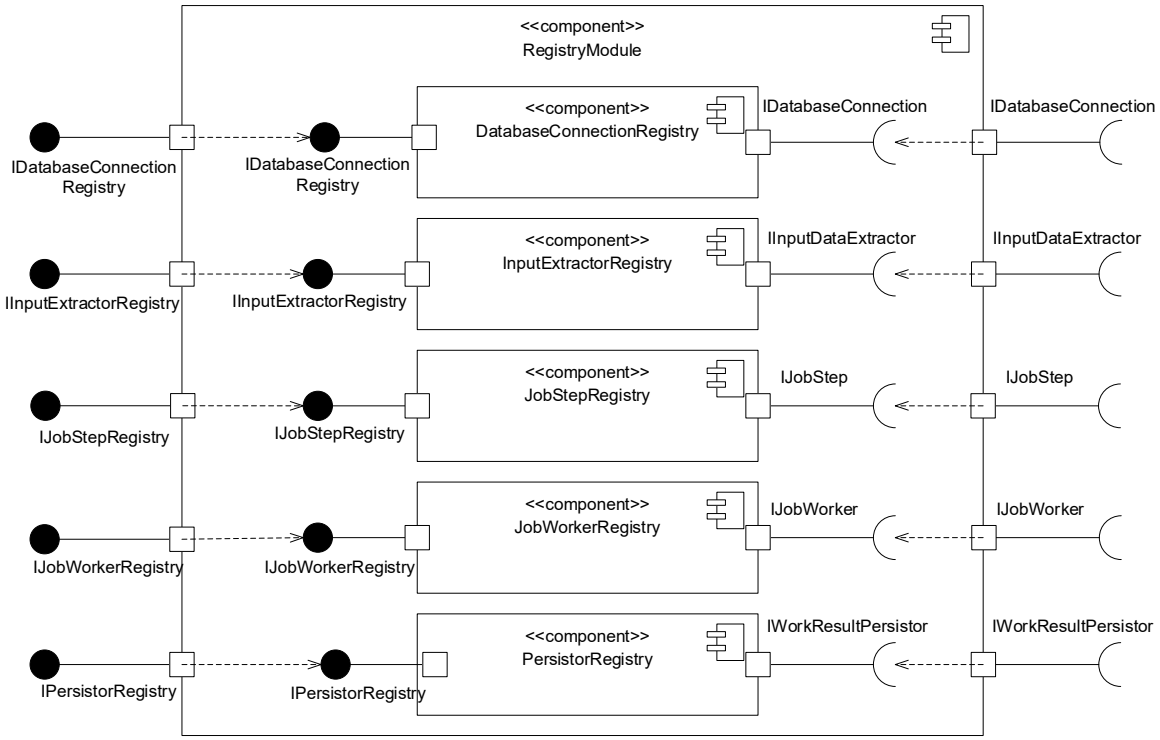
1. priprema poruka za slanje (implementacija bi se mogla zvati *PrepareMessagesJobStep*)
2. dohvat referenciranih privitaka (implementacija bi se mogla zvati *AttatchmentFetchJobStep*)
3. slanje poruka (implementacija bi se mogla zvati *SendMessagesJobStep*)
4. brisanje uspješno poslanih poruka iz baze podataka (implementacija bi se mogla zvati *CleanSentMessagesJobStep*)

Na kraju, ovaj postupak može, ali ne mora nužno producirati rezultat. Ako korisnika rezultat ne zanima, on može koristiti u modulu definiranu implementaciju sučelja *IWorkResultPersistor* pod nazivom *NullDataPersistor* koja ignorira konačni rezultat izvršavanja posla.

Ako bi korisnik ipak htio zapisati neke informacije (npr. vrijeme trajanja slanja pojedine poruke, koje poruke se nisu uspješno poslale i sl.) mogao bi ponuditi implementaciju sučelja *IWorkResultPersistor* koja bi uzela rezultat zadnjeg koraka (korak *CleanSentMessagesJobStep*) i zapisala ga u bazu podataka (implementacija bi se mogla zvati *SaveMessageSendingInfoResultPersistor*).

6.1.4. Podrška za proširenja

Modul za izvođenje poslova zamišljen je kao jednostavno proširivi modul. Budući da razne implementacije zahtijevaju različita okruženja izvršavanja, ovaj modul sadrži komponente registara pojedinih dijelova posla. Dijelovi posla u ove registre ulaze pod određenim identifikatorom (nazivom), a njihovu implementaciju prati oblikovni obrazac *generičkih tvornica*. Modul registara koji modul za izvođenje poslova koristi prikazan je slikom (Slika 26).



Slika 26 Prikaz registara koje nudi modul za registre

Uz pomoć ovog modula vanjske nadogradnje modula za izvršavanje poslova moguće je dodati u obliku proširenja (*eng. plugin*).

6.2. Servis za računanje statističkih pokazatelja

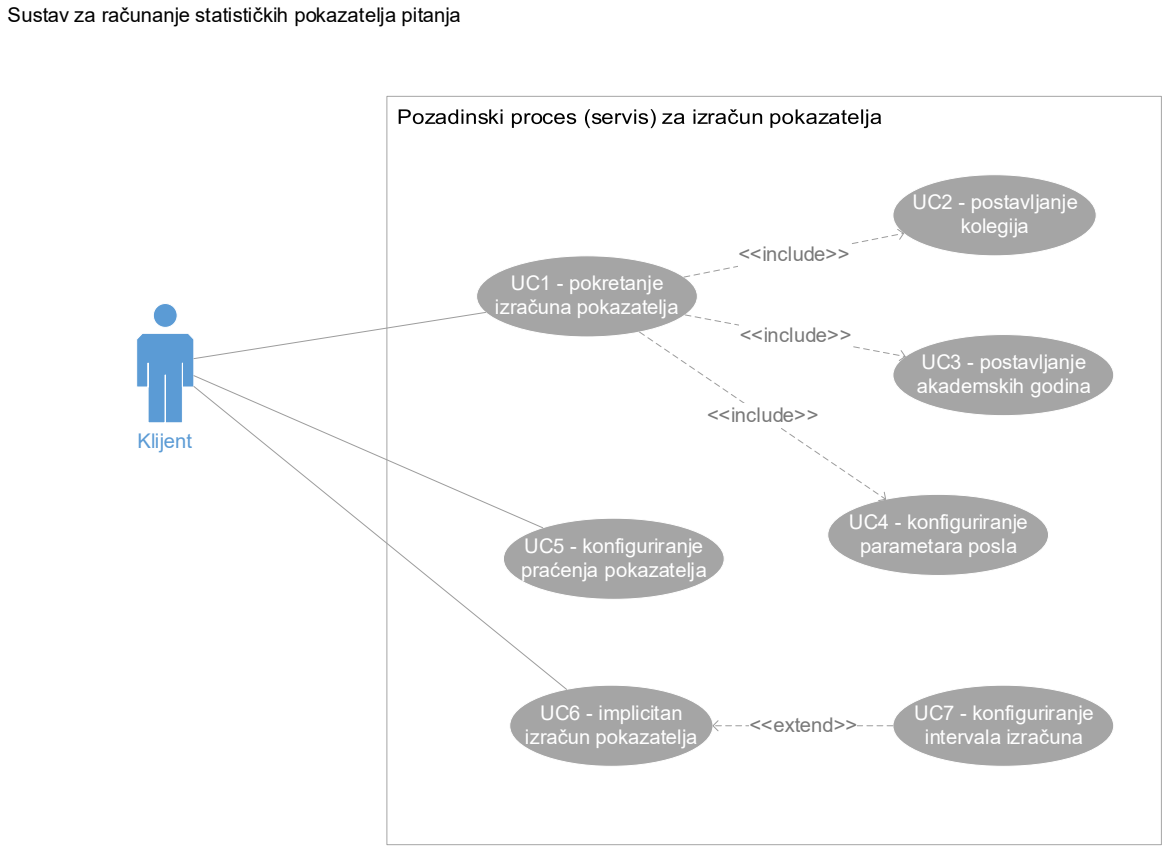
Servis za računanje statističkih pokazatelja oslanja se na razvijeni modul za izvođenje poslova. Izračun statističkih pokazatelja reprezentiran je kao posao koji se izvodi unutar modula za izvršavanje poslova. Postupak izračuna uključuje sljedeće korake:

1. Prikupljanje informacija o zadatcima iz baze sustava *Edgar*
2. Priprema podataka za statističku obradu
3. Računanje statističkih pokazatelja koristeći sustav *Judge0* i skriptu pisanu u programskom jeziku R
4. Računanje parametara logističkih krivulja izračunom obuhvaćenih zadataka
5. Klasifikacija zadataka prema težini
6. Pohrana statističkih pokazatelja, parametara logističkih funkcija i težina zadataka u bazu podataka

Servis za računanje statističkih pokazatelja zadataka izveden je kao pozadinski proces (*eng. daemon*). Osim izračuna, servis je zadužen za praćenje i osvježavanje statističkih pokazatelja te periodičko pokretanje izračuna bez eksplicitnog zahtjeva korisnika.

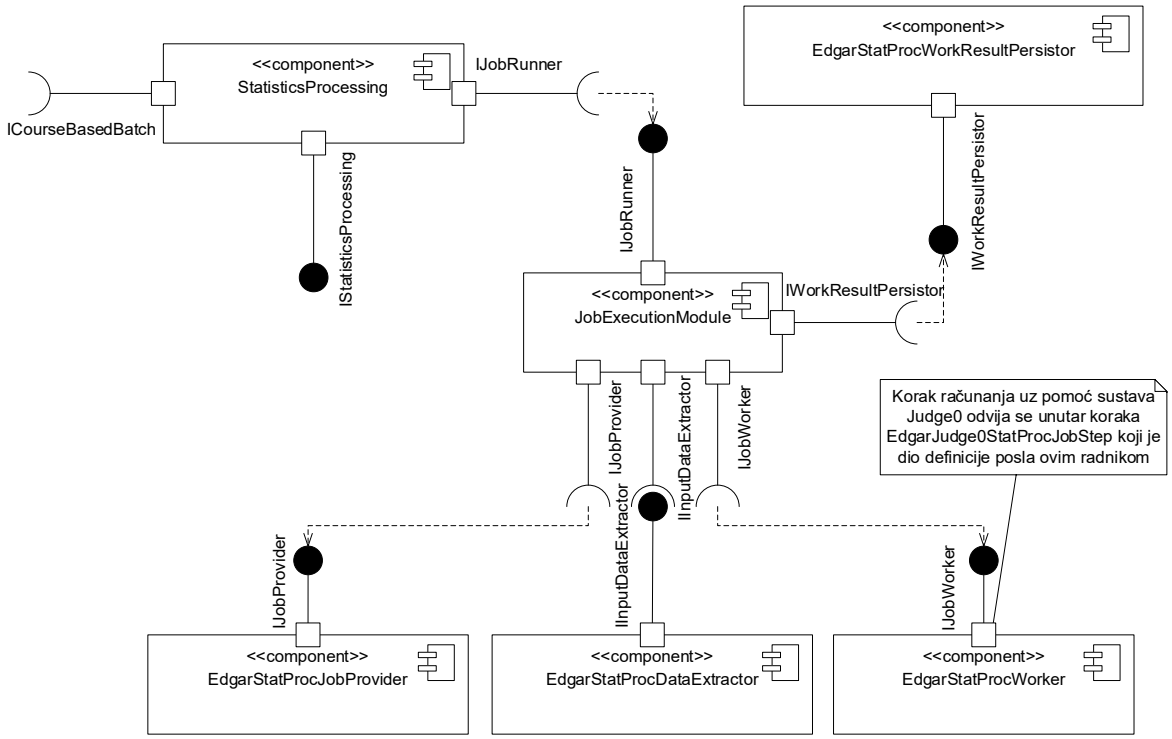
6.2.1. Modeliranje servisa za izračun statističkih pokazatelja

Servis bi svakom klijentu koji mu pristupa trebao omogućiti konfiguraciju i pokretanje posla izračuna statističkih pokazatelja. Kako bi se izračun mogao pokrenuti, klijent mora ispravno konfigurirati posao tako što mu zada kolegij, raspon akademskih godina izračuna te općenite informacije potrebne za izvršavanje posla. Osim toga, budući da servis sam može provoditi određene aktivnosti vezane uz izračun pokazatelja (npr. implicitan izračun pokazatelja, praćenje i osvježavanje pokazatelja …) potrebno je klijentu omogućiti konfiguraciju tih aktivnosti. Funkcionalni zahtjevi servisa prikazani su dijagramom obrazaca uporabe (Slika 27).



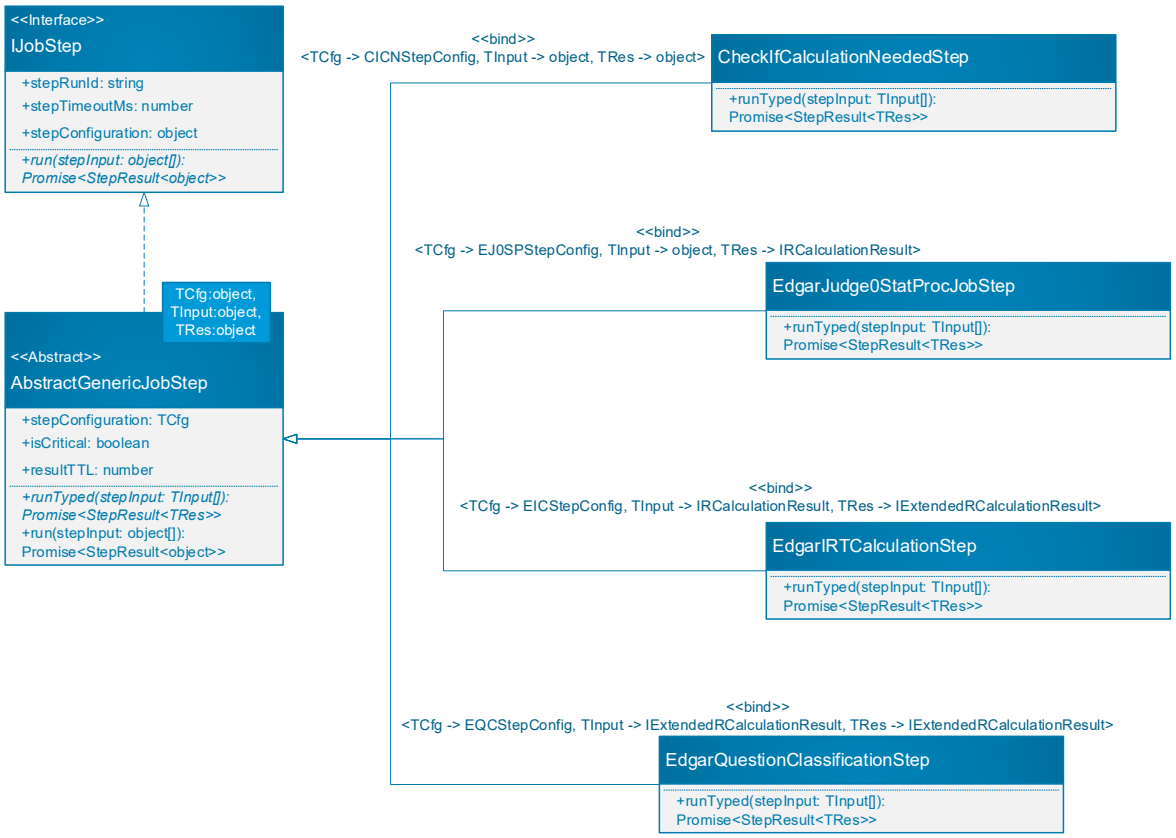
Slika 27 Dijagram obrazaca uporabe servisa za izračun statističkih pokazatelja

Mora implementirati sučelja koja propisuje modul za izvršavanje poslova. Svaka komponenta koju servis za izračun pokazatelja koristi implementira sučelja potrebna za integraciju s implementiranim sustavom za izvršavanje poslova. Dijagramom komponenti (Slika 28) prikazane su veze implementiranih komponenti s glavnim dijelom modula za izvođenje poslova.



Slika 28 Dijagram komponenti konkretne implementacije posla izračuna statističkih pokazatelja zadataka

Radi bolje preglednosti implementiranog rješenja, konkretni koraci posla izračuna statističkih pokazatelja sadržani u komponenti *EdgarStatProcWorker* prikazani su UML dijagramom razreda (*eng. UML class diagram*) na slici (Slika 29).

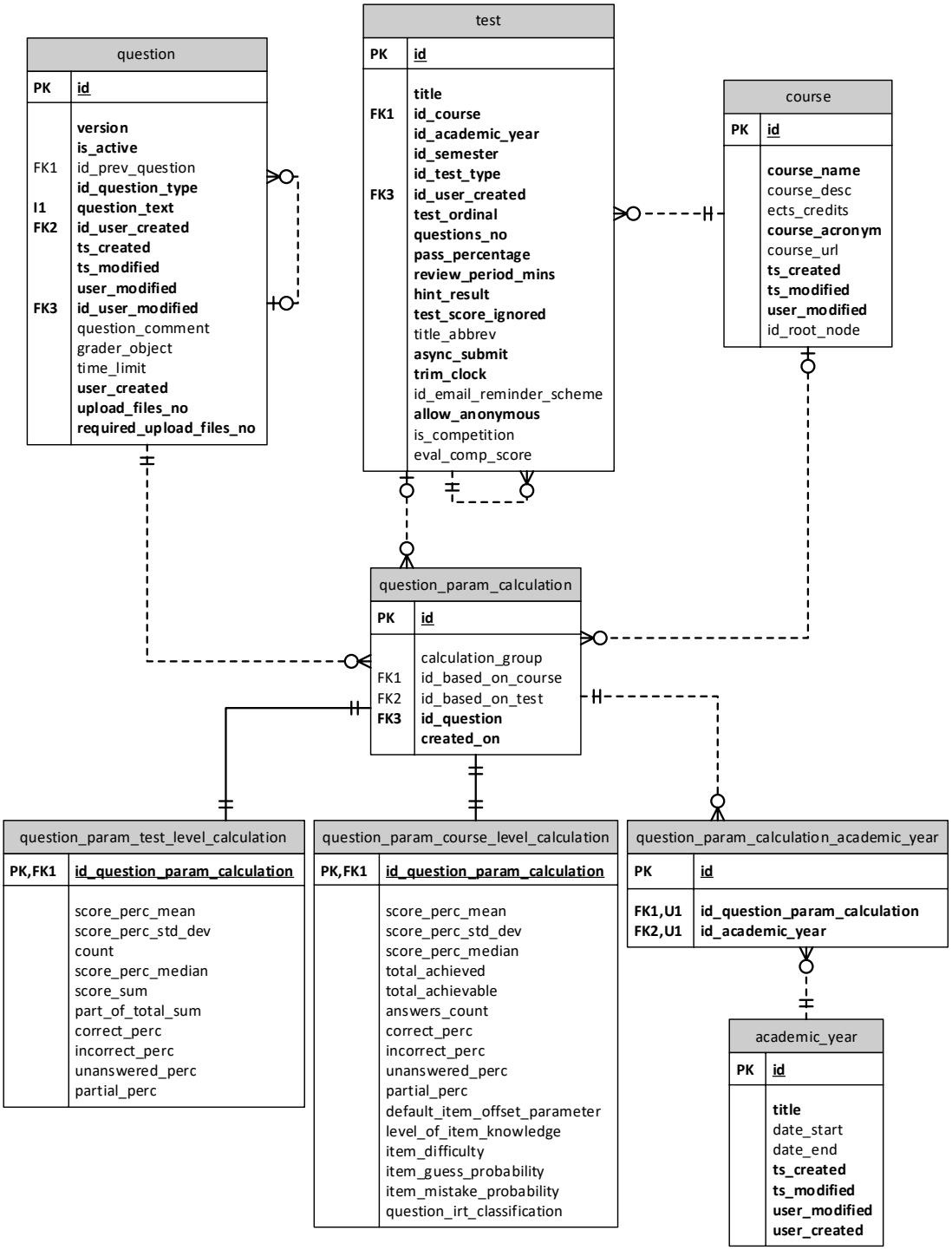


Slika 29 Dijagram razreda implementacije koraka poslova sadržanih u komponenti *EdgarStatProcWorker*

Zadatak implementacije razreda *CheckIfCalculationNeededStep* brine se o tome da se izračun ne izvršava ako nije potreban. Razred *EdgarJudge0StatProcJobStep* implementira računanje statističkih pokazatelja zadataka korištenjem R skripte i sustava za izvođenje proizvoljnog koda *Judge0*. Implementacija koraka *EdgarIRTCalculationStep* zadužena je za izračun parametara logističke krivulje za zadatke obuhvaćene zahtjevom. Na posljetku korak implementiran razredom *EdgarQuestionClassificationStep* određuje težine zadatcima.

6.2.2. Relacijski model baze podataka servisa

Nakon što se posao izvrši, servis u bazu podataka upisuje sve izračunate statističke pokazatelje, parametre logističkih krivulja i procijenjene težine zadataka. Model korištene baze podataka opisan je relacijskim dijagramom (Slika 30).



Slika 30 Relacijski dijagram dijela baze podataka (shema *statistics_schema* i povezane relacije iz sheme *public*) vezanog uz servis izračuna statističkih pokazatelja

Relacija *question_param_calculation* sadrži osnovne informacije o izračunu. Budući da se izračuni provode na razini cijelog kolegija i na razini pojedinog testa, oni su modelirani kao *„instance“* kalkulacije parametara zadataka. Detaljni pregled atributa relacije prikazan je tablicom (Tablica 10).

Tablica 10 Opis atributa relacije *question_param_calculation*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id	✓	-	✓	x	INT	-
calculation_group	x	x	x	x	VARCHAR(512)	Identifikator kalkulacije kojoj zapis pripada
id_based_on_course	x	x	x	✓ (course[id])	INT	Identifikator kolegija za koji je zapis izračunat
id_based_on_test	x	x	x	✓ (test[id])	INT	Identifikator testa za koji je zapis izračunat
id_question	x	x	✓	✓ (question[id])	INT	Identifikator zadatka za koji su izračunati parametri
created_on	x	x	✓	x	TIMESTAMP	Vremenska oznaka stvaranja zapisa

Relacija *question_param_calculation_academic_year* sadrži po jedan zapis za svaku akademsku godinu obuhvaćenu pokrenutim izračunom. Opis atributa relacije prikazan je u tablici (Tablica 11).

Tablica 11 Opis atributa relacije *question_param_calculation_academic_year*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id	✓	-	✓	x	INT	-
id_qpc[10]	x	(1)	✓	✓ (qpc[11][id])	INT	Identifikator izračunatih parametara
id_academic_year	x	(1)	✓	✓ (academic_year[id])	INT	Identifikator ak. godine

[10] id_qpc – skraćeno od *id_question_param_calculation*
[11] qpc – skraćeno za *question_param_calculation*

Relacija *question_param_course_level_calculation* služi za pohranu statističkih pokazatelja zadataka na razini kolegija. Svaki zapis sadrži informacije kao što su standardna devijacija, srednja vrijednost i medijan postotka ostvarenih bodova i ostale pokazatelje. Atributi relacije opisani su tablicom (Tablica 12).

Tablica 12 Opis atributa relacije *question_param_course_level_calculation*[12]

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id_qpc	✓	-	✓	✓ (qpc[id])	INT	Identifikator izračunatih parametara
score_perc_mean	x	x	x	x	DOUBLE PRECISION	Srednja vrijednost postotaka postignutih bodova na razini kolegija
score_perc_std_dev	x	x	x	x	DOUBLE PRECISION	Standardna devijacija postotaka postignutih bodova na razini kolegija
score_perc_median	x	x	x	x	DOUBLE PRECISION	Medijan postotaka postignutih bodova na razini kolegija
total_achieved	x	x	x	x	DOUBLE PRECISION	Ukupan zbroj postignutih bodova na zadatku
total_achievable	x	x	x	x	DOUBLE PRECISION	Maksimalni broj ostvarivih bodova
answers_count	x	x	x	x	INT	Broj pristupnika (na razini kolegija)
correct_perc	x	x	x	x	DOUBLE PRECISION	% točnih odgovora (na razini kolegija)
incorrect_perc	x	x	x	x	DOUBLE PRECISION	% netočnih odgovora (na razini kolegija)
unanswered_perc	x	x	x	x	DOUBLE PRECISION	% neodgovorenih (na razini kolegija)
partial_perc	x	x	x	x	DOUBLE PRECISION	% parcijalno ocijenjenih (na razini kolegija)
default_item_offset_param	x	x	x	x	DOUBLE PRECISION	Empirijska konstanta zadatka (IRT konstanta *D*)
level_of_item_knowledge	x	x	x	x	DOUBLE PRECISION	Diskriminatorni faktor zadatka (IRT parametar *a*)

[12] Svi atributi vezani uz teoriju odgovora na zadatke (atributi označeni s IRT) objašnjeni u (Tablica 1)

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
item_difficulty	x	x	x	x	DOUBLE PRECISION	Težina zadatka (IRT parametar *b*)
item_guess_probability	x	x	x	x	DOUBLE PRECISION	Vjerojatnost pogađanja točnog odgovora (IRT parametar *c*)
item_mistake_probability	x	x	x	x	DOUBLE PRECISION	Vjerojatnost točnog odgovora dobrog ispitanika (IRT parametar *d*)
question_irt_classification	x	x	x	x	ENUM('very_easy', 'easy', 'normal', 'hard', 'very_hard')	Procijenjena težina zadatka

Relacija *question_param_test_level_calculation* služi za pohranu statističkih pokazatelja zadataka na razini testa (ispita). Svaki zapis sadrži informacije kao što su standardna devijacija, srednja vrijednost i medijan postotka ostvarenih bodova i ostale pokazatelje. Atributi relacije opisani su tablicom (Tablica 13).

Tablica 13 Opis atributa relacije *question_param_test_level_calculation*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id_qpc	✓	-	✓	✓ (qpc[id])	INT	Identifikator izračunatih parametara
score_perc_mean	x	x	x	x	DOUBLE PRECISION	Srednja vrijednost postotaka postignutih bodova na razini testa
score_perc_std_dev	x	x	x	x	DOUBLE PRECISION	Standardna devijacija postotaka postignutih bodova na razini testa
count	x	x	x	x	INT	Broj pristupnika
score_perc_median	x	x	x	x	DOUBLE PRECISION	Medijan postotaka postignutih bodova na razini testa
score_sum	x	x	x	x	DOUBLE PRECISION	Ukupan zbroj postignutih bodova na testu za određeni zadatak
part_of_total_sum	x	x	x	x	DOUBLE PRECISION	Udio koji zadatak ima u svim bodovima ostvarenima na testu
correct_perc	x	x	x	x	DOUBLE PRECISION	% točnih odgovora (na razini testa)
incorrect_perc	x	x	x	x	DOUBLE PRECISION	% netočnih odgovora (na razini testa)
unanswered_perc	x	x	x	x	DOUBLE PRECISION	% neodgovorenih (na razini testa)
partial_perc	x	x	x	x	DOUBLE PRECISION	% parcijalno ocijenjenih (na razini testa)

6.3. Web aplikacija za vježbanje i demonstraciju implementiranih funkcionalnosti

Interakcija s korisnikom (nastavnikom ili studentom) odvija se u ovom dijelu cjelokupnog sustava. Funkcionalni zahtjevi web aplikacije opisani su i modelirani u poglavlju *Model sustava za provedbu prilagodljivih vježbi*. Koristeći se web aplikacijom nastavnici pokreću izračun statističkih pokazatelja zadataka i definiraju prilagodljive vježbe, a studenti mogu pokretati prilagodljive vježbe koje su napravili nastavnici.

Nastavnik u web aplikaciji može pokrenuti posao izračuna statističkih pokazatelja i pregledati njegove rezultate. Nakon uspješno izvršenog posla izračuna pokazatelja, može pregledati statističke pokazatelje zadataka, parametre logističkih krivulja (prema IRT) zadataka i procijenjene težine zadataka uz odgovarajuće vizualne reprezentacije (npr. histogram standardnih devijacija postotaka postignutih bodova na zadatku, stupčasti dijagram procijenjenih težina zadataka …).

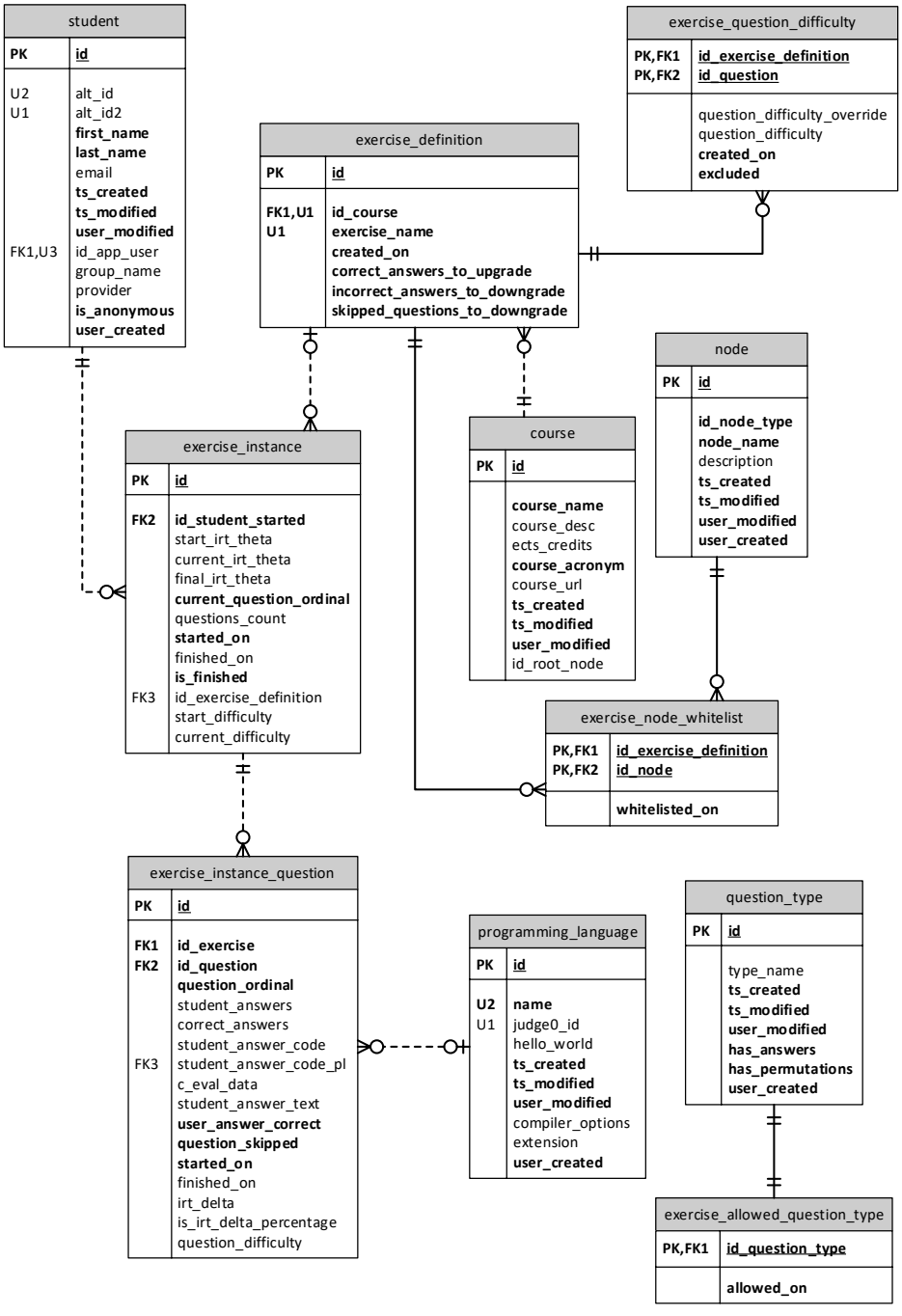
Ako u sustavu postoji važeća kalkulacija parametara za neki od kolegija definiranih u *Edgarovoj* bazi podataka te ako za taj kolegij postoje definirane prilagodljive vježbe, student može započeti prilagodljivu vježbu.

Web aplikacija razvijena je kao samostalni dio cjelokupnog sustava, ali se oslanja na razvijeni *servis za izračun statističkih pokazatelja* kako bi pružila uslugu računanja statističkih pokazatelja zadataka. Budući da je aplikacija razvijena prateći CSR[13] (*eng. controller-service-repository*) arhitekturu, implementacija logike računanja statističkih pokazatelja zadataka (servis) na poslužitelju aplikacijske logike (*eng. backend*) jednostavno se može zamijeniti drugim servisom koji ne koristi razvijeni *servis za izračun statističkih pokazatelja* kako bi pružio tu uslugu.

[13] CSR – arhitektura aplikacija koja aplikacijsku logiku raščlanjuje u 3 razine: razina upravljača (*eng. controller layer*), razina servisa (*eng. service layer*) i razina repozitorija (*eng. repository layer*)

6.3.1. Relacijski model baze podataka web aplikacije

Razvijena web aplikacija bazu podataka koristi za pohranu informacija o definicijama, sastavu i modelima prilagodljivosti prilagodljivih vježbi. Pohranjuje i podatke o pokretanju prilagodljivih vježbi te podatke o uspješnosti studenata na svakoj pokrenutoj vježbi. Model baze podataka prikazan je relacijskim dijagramom (Slika 31).



Slika 31 Relacijski model dijela baze podataka (shema *adaptive_exercise* i povezane relacije iz sheme *public*) vezanog za web aplikaciju prilagodljivih vježbi

Relacija *exercise_allowed_question_type* služi za pohranu tipova zadataka koji se smiju pojaviti u prilagodljivim vježbama. Ako se neki tip zadatka ne nalazi u ovoj relaciji, niti jedna definicija prilagodljive vježbe neće zadavati zadatke s tim tipom studentima. Atributi relacije opisani su u tablici (Tablica 14).

Tablica 14 Opis atributa relacije *exercise_allowed_question_type*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id_question_type	✓	-	✓	✓ (question_type[id])	INT	Tip zadatka na koji se zapis odnosi
allowed_on	x	x	✓	x	TIMESTAMP	Vremenska oznaka dopuštanja tipa zadatka u vježbe

Relacija *exercise_definition* predstavlja entitet vježbe. Unutar ove relacije pohranjuju se osnovne informacije o vježbama. Atributi relacije opisani su tablicom (Tablica 15).

Tablica 15 Opis atributa relacije *exercise_definition*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id	✓	-	✓	x	INT	-
id_course	x	(1)	✓	✓ (course[id])	INT	Identifikator kolegija uz koji je definicija vježbe vezana
exercise_name	x	(1)	✓	x	VARCHAR(256)	Naziv vježbe koji će se prikazati studentima
correct_answers_to_upgrade	x	x	✓	x	INT	Broj točnih odgovora zaredom za povećanje težine
incorrect_answers_to_downgrade	x	x	✓	x	INT	Broj netočnih odgovora zaredom za smanjenje težine
skipped_questions_to_downgrade	x	x	✓	x	INT	Broj preskočenih zadataka zaredom za smanjenje težine
created_on	x	x	✓	x	TIMESTAMP	Vremenska oznaka stvaranja definicije

Relacija *exercise_node_whitelist* pohranjuje informacije o čvorovima zadataka koji su sadržani u definiciji vježbe. U vježbi se studentu mogu pojaviti zadatci samo iz skupa zadataka čiji roditelji su čvorovi definirani u ovoj relaciji, a čvorovi se određuju na razini definicije vježbe. Opis atributa nalazi se u tablici (Tablica 16).

Tablica 16 Opis atributa relacije *exercise_node_whitelist*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id_exercise_definition	✓	-	✓	✓ (exercise_definition[id])	INT	Identifikator definicije vježbe na koju se zapis odnosi
id_node	✓	x	✓	✓ (node[id])	INT	Identifikator čvora iz kojeg je dozvoljeno dohvaćati zadatke vježbe
whitelisted_on	x	x	✓	x	TIMESTAMP	Vremenska oznaka stvaranja zapisa

Relacija *exercise_question_difficulty* služi za pohranu informacija o težinama zadataka. U ovu relaciju spremaju se svi zadatci na razini definicije vježbe koja pripadaju čvoru iz kojeg se zadatci mogu pojaviti. Uz pomoć ove relacije omogućeno je nadjačavanje težina zadatka te pamćenje stanja težina zadataka u trenutku kada je čvor dodan definiciji vježbe. Pojašnjenje atributa relacije nalazi se u tablici (Tablica 17).

Tablica 17 Opis atributa relacije *exercise_question_difficulty*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id_exercise_definition	✓	-	✓	✓ (exercise_definition[id])	INT	Identifikator definicije vježbe na koju se zapis odnosi
id_question	✓	x	✓	✓ (question[id])	INT	Identifikator čvora iz kojeg je dozvoljeno dohvaćati zadatke vježbe
question_difficulty	x	x	x	x	ENUM('very_easy', 'easy', 'normal', 'hard', 'very_hard')	Težina zadatka na razini vježbe
question_difficulty_override	x	x	x	x	ENUM('very_easy', 'easy', 'normal', 'hard', 'very_hard')	Nadjačana težina zadatka
created_on	x	x	✓	x	TIMESTAMP	Vremenska oznaka stvaranja zapisa
excluded	x	x	✓	x	BOOLEAN	Zastavica uključenja zadatka u bazen zadataka koja se mogu pojaviti u vježbama

Relacija *exercise_instance* predstavlja instancu definirane prilagodljive vježbe. Svaka instanca pripada studentu koji ju je pokrenuo te sadrži informaciju o definiciji vježbe kojoj pripada. Svi atributi relacije prikazani su tablicom (Tablica 18).

Tablica 18 Opis atributa relacije *exercise_instance*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id	✓	-	✓	x	INT	-
id_student_started	x	x	✓	✓ (student[id])	INT	Identifikator studenta koji je započeo vježbu
id_exercise_definition	x	-	✓	✓ (exercise_definition[id])	INT	Identifikator definicije vježbe na koju se zapis odnosi
start_difficulty	x	x	x	x	ENUM('very_easy', 'easy', 'normal', 'hard', 'very_hard')	Težina početnog zadatka na vježbi
current_difficulty	x	x	x	x	ENUM('very_easy', 'easy', 'normal', 'hard', 'very_hard')	Težina trenutnog zadatka na vježbi
start_irt_theta	x	x	x	x	DOUBLE PRECISION	Početni pozadinski pokazatelj latentne sposobnosti
current_irt_theta	x	x	x	x	DOUBLE PRECISION	Trenutni pozadinski pokazatelj latentne sposobnosti
final_irt_theta	x	x	x	x	DOUBLE PRECISION	Završni pozadinski pokazatelj latentne sposobnosti
current_question_ordinal	x	x	✓	x	INT	Redni broj trenutnog zadatka vježbe
questions_count	x	x	x	x	INT	Ukupan broj zadataka

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
finished_on	x	x	x	x	TIMESTAMP	Vremenska oznaka kraja
is_finished	x	x	✓	x	BOOLEAN	Zastavica završetka vježbe

Relacija *exercise_instance_question* pohranjuje informacije o zadatcima koji su se pojavili u sklopu prilagodljive vježbe. Za svaki zadatak je, zbog potrebe za pohranom stanja koje je bilo za vrijeme izvršavanja vježbe, potrebno zapisati njegovu težinu. Opis atributa nalazi se u tablici (Tablica 19).

Tablica 19 Opis atributa relacije *exercise_instance_question*

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
id	✓	-	✓	x	INT	-
id_exercise	x	x	✓	✓ (exercise_instance[id])	INT	Identifikator instance vježbe kojoj zadatak pripada
id_question	x	x	✓	✓ (question[id])	INT	Identifikator zadatka
question_difficulty	x	x	x	x	ENUM('very_easy', 'easy', 'normal', 'hard', 'very_hard')	Težina zadatka na instanci vježbe
question_ordinal	x	x	x	x	INT	Redni broj zadatka
student_answers	x	x	x	x	INT[]	Redni brojevi odgovora koje je student označio
correct_answers	x	x	x	x	INT[]	Redni brojevi točnih odgovora na zadatak
student_answer_code	x	x	x	x	TEXT	Programski kod koji je student unio kao odgovor na zadatak

Atribut	PK	UQ	NN	FK	Tip podatka	Opis
c_eval_data	x	x	x	x	TEXT	Evaluacijski podatci koda
student_answer_text	x	x	x	x	TEXT	Tekstualni odgovor studenta na zadatak
user_answer_correct	x	x	✓	x	BOOLEAN	Zastavica točnosti odgovora na zadatak
question_skipped	x	x	✓	x	BOOLEAN	Zastavica koja prikazuje je li zadatak preskočen
started_on	x	x	✓	x	TIMESTAMP	Vremenska oznaka zadavanja zadatka
finished_on	x	x	x	x	TIMESTAMP	Vremenska oznaka kraja rješavanja zadatka
irt_delta	x	x	x	x	DOUBLE PRECISION	Promjena pokazatelja latentne sposobnosti
is_irt_delta_percentage	x	x	x	x	BOOLEAN	Zastavica koja označava je li vrijednost atributa 'irt_delta' postotna ili apsolutna promjena

Na ovu relaciju je postavljen okidač koji za svaki novi zadatak automatski upiše sljedeći broj zadatka na vježbi.

7. Rad s razvijenim sustavom

Kako bi sustav ispravno radio potrebno je pokrenuti:

- bazu podataka
- instancu sustava *Judge0* za proizvoljno izvršavanje koda
- servis za izračun statističkih pokazatelja zadataka
- web aplikaciju za rad s prilagodljivim vježbama

Demonstracija rada obavljena je na operacijskom sustavu *Microsoft Windows*.

Bazu podataka *PostgreSQL* potrebno je instalirati i pokrenuti prema uputama za određeni operacijski sustav[14]. Nakon toga potrebno je uspostaviti bazu podataka prema uputama iz *GitLab* repozitorija[15] sustava *Edgar*. Nakon toga je u uspostavljenoj bazi podataka potrebno pokrenuti SQL (*eng. structured query language*) skriptu koja će stvoriti sve relacije potrebne za rad aplikacije za rad s prilagodljivim vježbama.

Sustav *Judge0* potrebno je preuzeti sa službene stranice[16] i pokrenuti u proizvoljnom okruženju. Potrebno je samo osigurati da preuzeto okruženje *Judge0* distribucije podržava izvođenje koda pisanog u programskom jeziku R.

Nakon toga potrebno je pokrenuti servis za izračun statističkih pokazatelja zadataka. Potrebno je osigurati ispravnu konfiguraciju servisa, ali najvažnije je ispravno postaviti identifikator programskog jezika R koji mu je instanca sustava *Judge0* dodijelila.

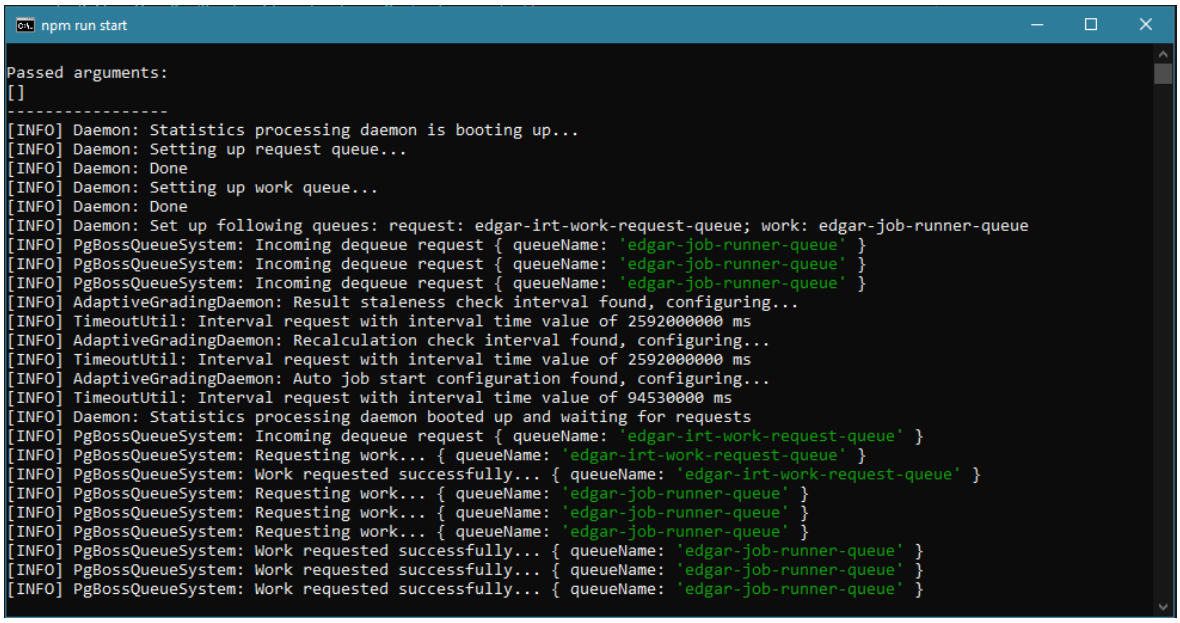
Na posljetku potrebno je pokrenuti poslužitelj aplikacijske logike i poslužitelj korisničkog sučelja web aplikacije. Nakon uspješnog pokretanja web aplikaciji moguće je pristupiti kroz web preglednik.

---

[14] Sve potrebne upute moguće je pronaći na službenoj stranici *PostgreSQL-a*: https://www.postgresql.org/
[15] https://gitlab.com/edgar-group/edgar
[16] https://judge0.com/

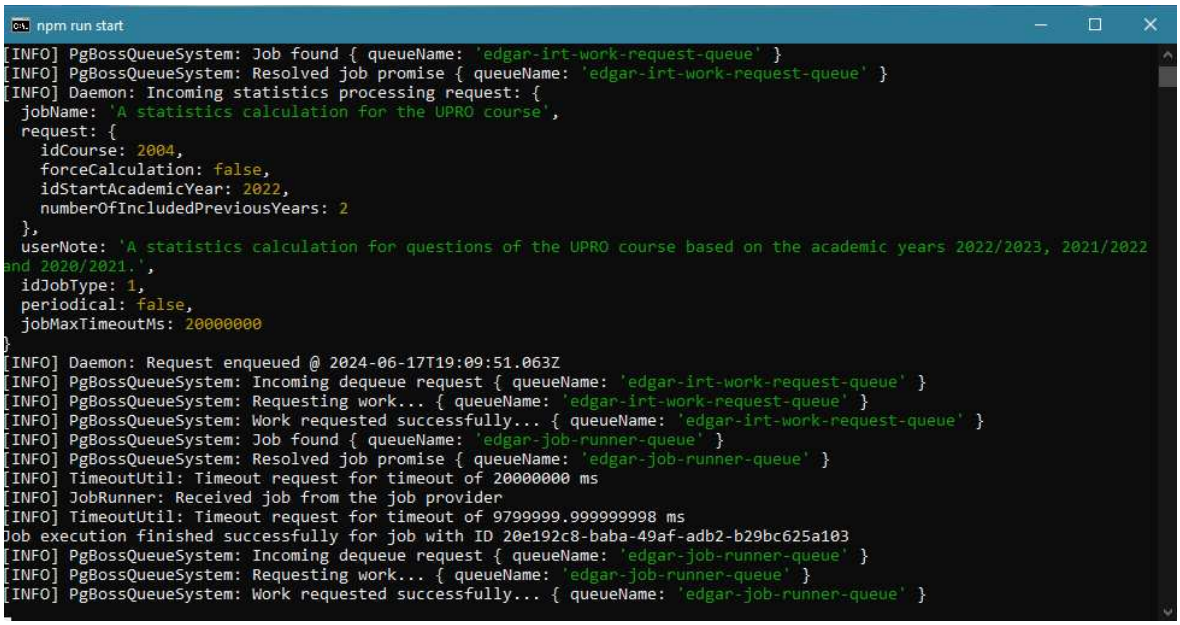
7.1. Pokretanje servisa za izračun statističkih pokazatelja

Servis za izračun statističkih pokazatelja pokreće se iz naredbenog retka (*eng. command line, shell*) ljuske operacijskog sustava. Nakon pokretanja otvara se prozor prikazan na slici (Slika 32).



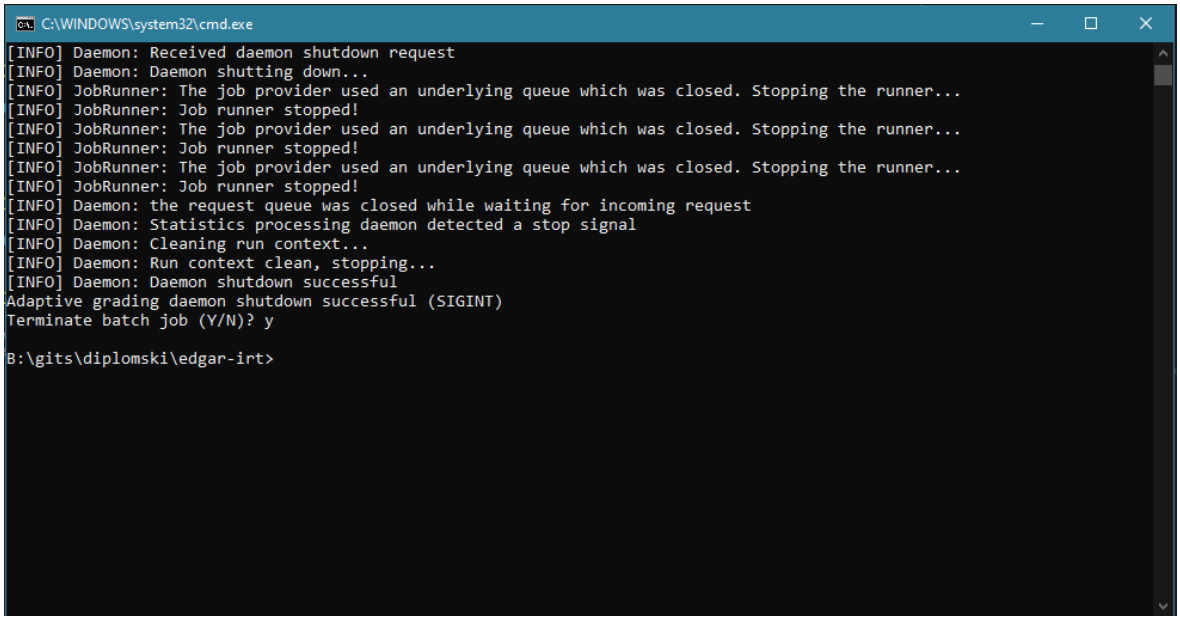
Slika 32 Prikaz konzole nakon pokretanja servisa

Servis će za svaki zaprimljeni posao ispisati neke osnovne informacije (kao naziv posla iz zahtjeva, identifikator kolegija, identifikator početne akademske godine …) te izvještavati o statusu izvođenja posla (Slika 33).



Slika 33 Prikaz konzole servisa nakon zaprimljenog posla

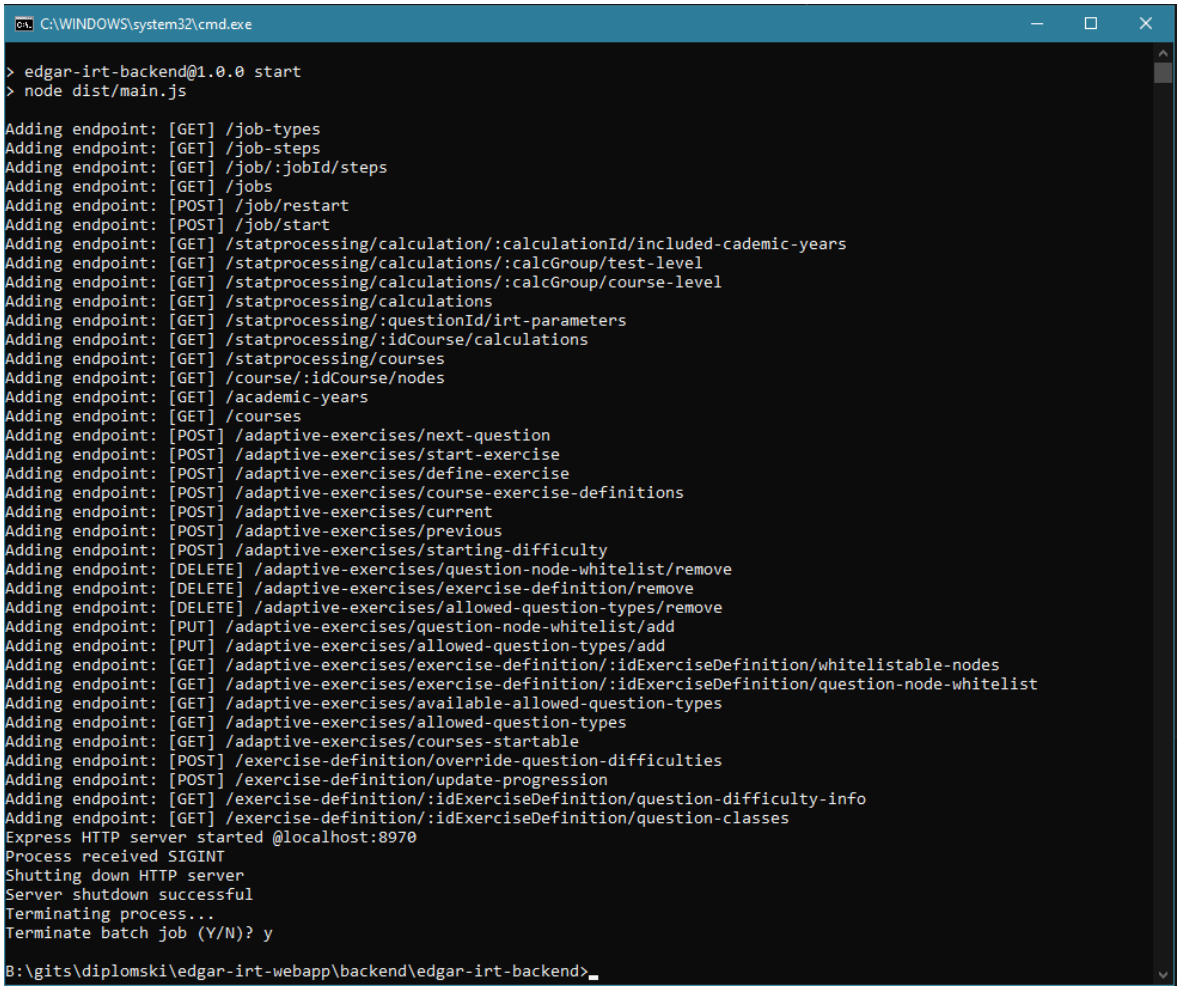
Servis će za vrijeme rada u konzolu ispisivati sve poruke koje su vezane uz izvođenje poslova koji dolaze. Gašenje servisa moguće je kombinacijom tipki Ctrl + C, a rezultat gašenja prikazan je na slici (Slika 34).



Slika 34 Prikaz konzole nakon pritiska kombinacije tipki Ctrl + C

7.2. Pokretanje poslužitelja aplikacijske logike web aplikacije

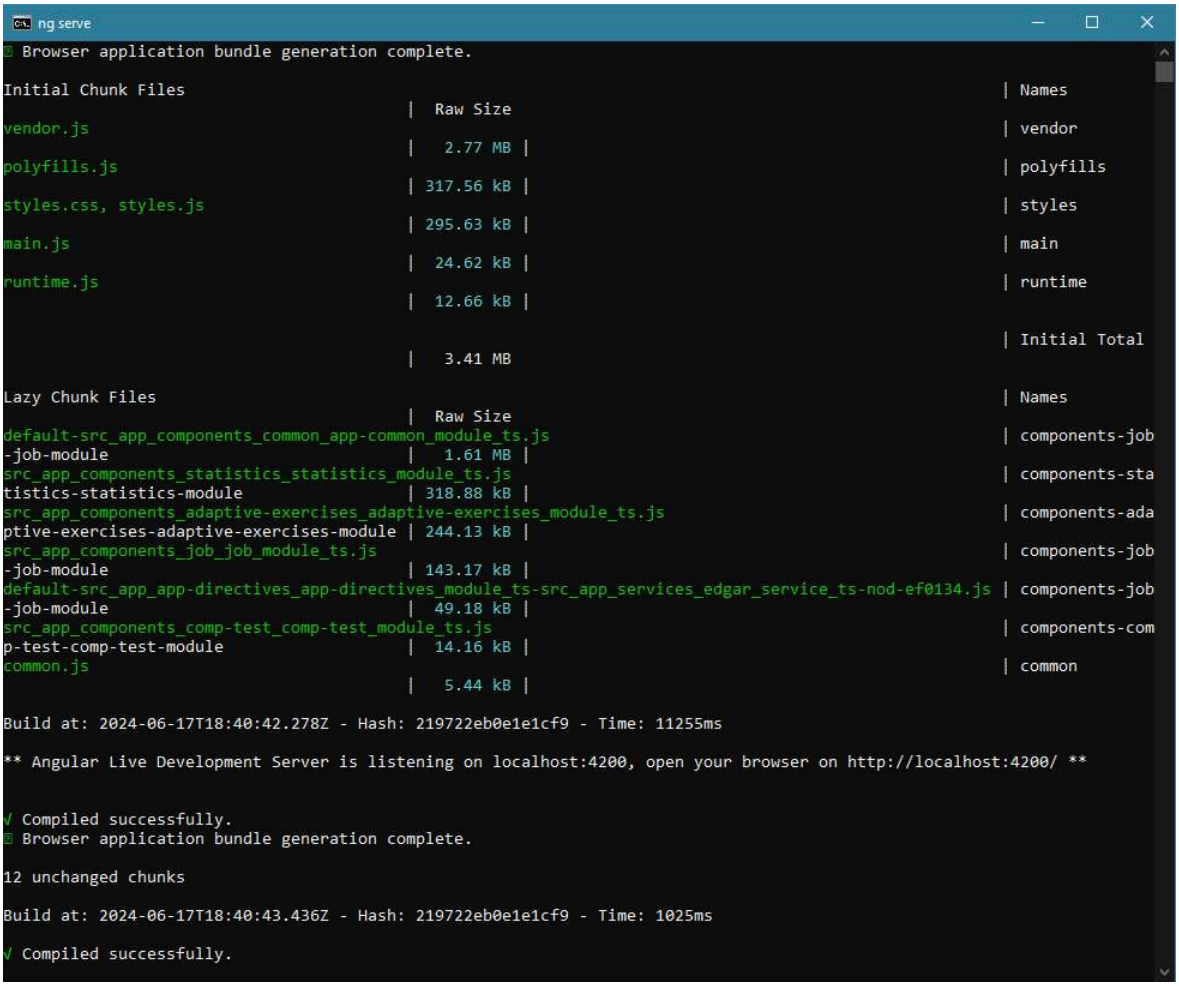
Poslužitelj aplikacijske logike također se pokreće iz naredbene linije ljuske operacijskog sustava. Nakon pokretanja, HTTP (*eng. hypertext transfer protocol*) poslužitelj počinje slušati na konfiguriranim vratima (pretpostavljena vrijednost je TCP:8970). Poslužitelj nakon pokretanja ispisuje sve puteve na kojima poslužuje zahtjeve. Ispis postupka pokretanja i gašenja HTTP poslužitelja prikazan je slikom (Slika 35).



Slika 35 Ispis pokretanja i gašenja poslužitelja aplikacijske logike web aplikacije

7.3. Pokretanje poslužitelja korisničkog sučelja web aplikacije

Korisničko sučelje razvijeno u *Angularu* poslužuje zaseban poslužitelj. Kao i ostali servisi, ovaj poslužitelj pokreće se u naredbenom retku ljuske operacijskog sustava. Proces pokretanja prikazan je slikom (Slika 36).



Slika 36 Prikaz ispisa pokretanja poslužitelja korisničkog sučelja

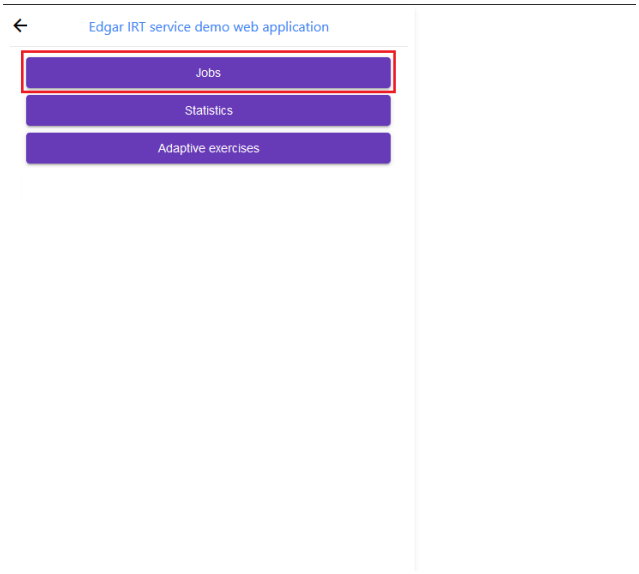
7.4. Rad s web aplikacijom prilagodljivih vježbi

Web aplikacija prilagodljivih vježbi razvijena je za dvije korisničke uloge: nastavnik i student. Nastavnici mogu pokretati izračune statističkih pokazatelja i definirati prilagodljive vježbe dok studenti mogu pokretati prilagodljive vježbe.

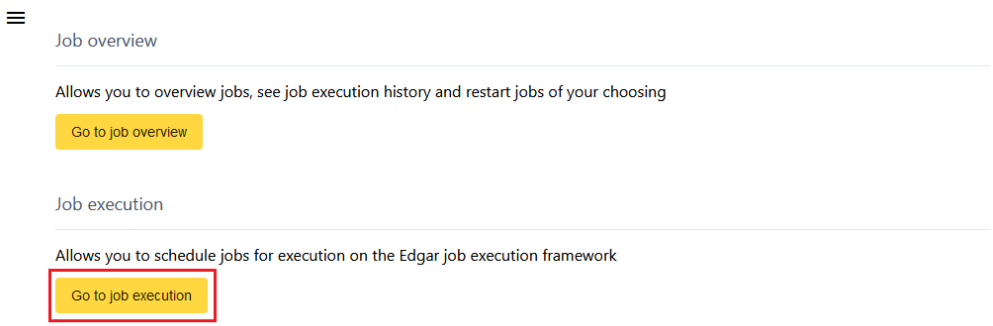
Budući da je izolacija uloga nije cilj zadatka, u razvijenoj aplikaciji postoji samo jedan korisnik koji može biti u ulozi nastavnika ili studenta.

7.4.1. Pokretanje izračuna statističkih pokazatelja

Za pokretanje izračuna prvo je iz izbornika potrebno kliknuti na gumb *„Jobs“* (Slika 37) koji korisnika odvodi na stranicu mogućih izbora vezanih uz poslove na kojem treba odabrati gumb *„Go to job execution“* (Slika 38).



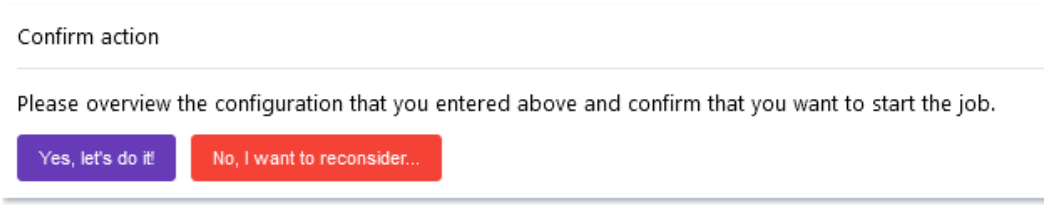
Slika 37 Prikaz izbornika aplikacije s naglaskom na gumb *„Jobs“*



Slika 38 Prikaz ekrana s opcijama vezanim uz poslove s naglaskom na gumb *„Go to job execution“*

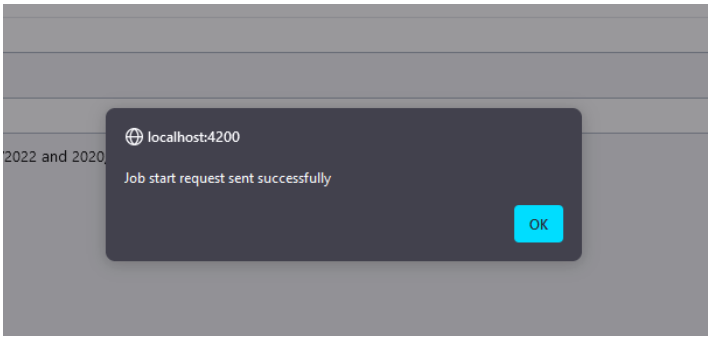
Nakon odabira gumba korisnik dolazi na stranicu (Slika 39) na kojoj treba ispuniti obrazac za pokretanje posla. Nakon što korisnik popuni obrazac treba pritisnuti gumb *„Next“* na dnu stranice pri čemu će ga aplikacija uputiti na provjeru i potvrdu upisanih vrijednosti (Slika 40). Klikom na gumb *„Yes, let's do it!“* pokreće se posao za izračun statističkih pokazatelja na konfiguriranom kolegiju.

Slika 39 Prikaz ispunjenog obrasca za izračun statističkih pokazatelja zadataka



Slika 40 Prikaz upita o potvrdi ili odbijanju pokretanja posla

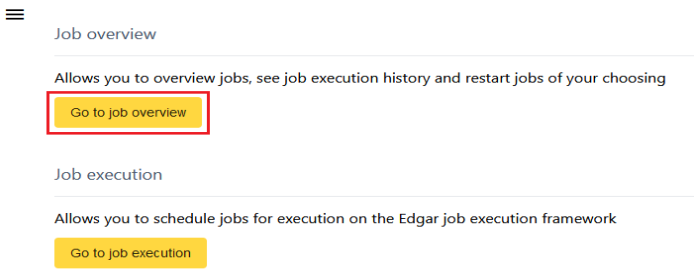
Nakon uspješnog pokretanja posla web aplikacija korisniku šalje obavijest o uspjehu (Slika 41).



Slika 41 Prikaz modala koji se pojavi nakon uspješnog zahtjeva za obavljanje posla

7.4.2. Pregled prethodno pokrenutih poslova

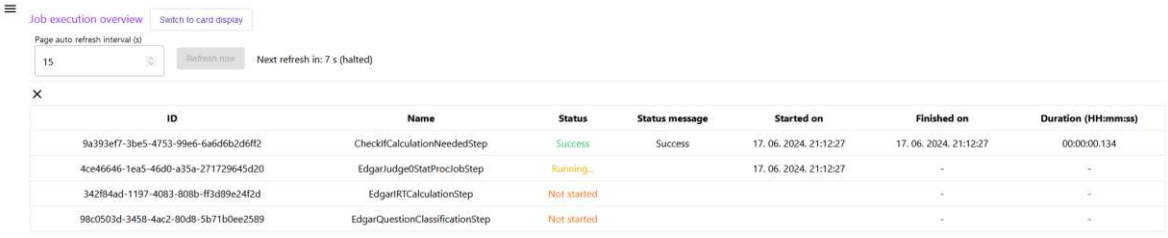
Ako korisnik na ekranu ponuđenih aktivnosti vezanih uz poslove (Slika 42) odabere gumb *„Go to job overview“*, odlazi na stranicu na kojoj su prikazane informacije o prethodno pokrenutim poslovima (Slika 43). Na ovom ekranu korisnik može pregledati osnovne informacije o poslu kao što su naziv, korisnička napomena, stanje, vrijeme početka i završetka te trajanje svakog posla.



Slika 42 Prikaz ekrana s opcijama vezanim uz poslove s naglaskom na gumb *„Go to job overview“*

Slika 43 Prikaz ekrana s informacijama o prethodno pokrenutim poslovima

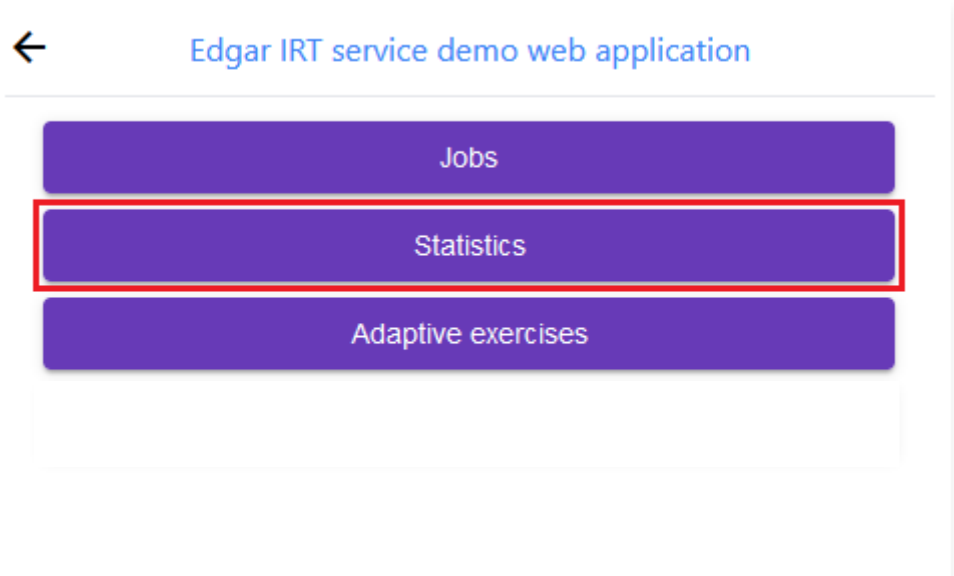
Na ovom ekranu (Slika 43) korisnik u gornjem dijelu može upravljati s automatskim osvježavanjem liste poslova. Klikom na zapis iz liste poslova otvara se nova stranica s detaljima odabranog posla (Slika 44). Ovdje korisnik može pregledati nazive, statuse, poruke statusa, vrijeme pokretanja, vrijeme završetka i trajanje pojedinog koraka posla.



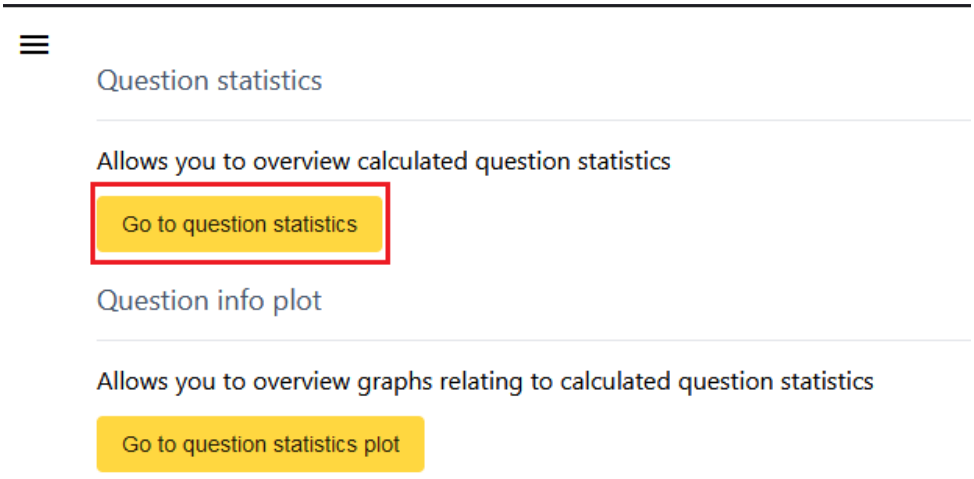
Slika 44 Prikaz ekrana s detaljima posla *„A statistics calculation for the UPRO course“*

7.4.3. Pregled statističkih pokazatelja zadataka

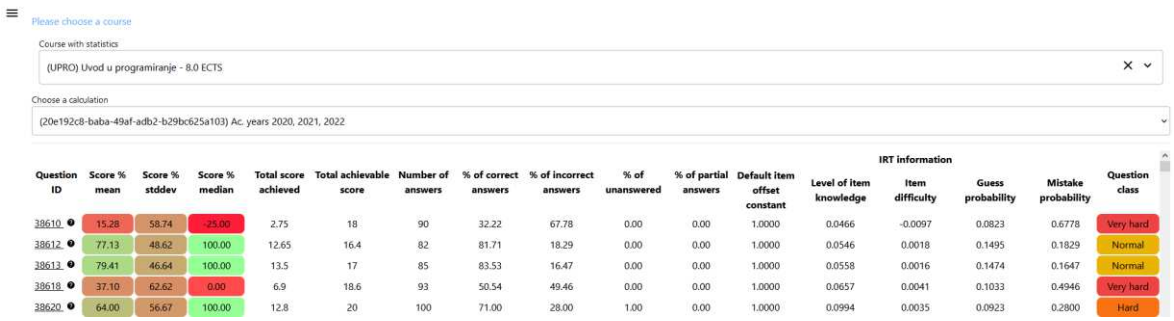
Nakon uspješno izvršenog izračuna statističkih pokazatelja, korisnik može pregledati statističke pokazatelje zadataka obuhvaćenih izračunom. Do stranice pregleda pokazatelja u tabličnom obliku (Slika 47) dolazi tako što u izborniku odabere gumb *„Statistics“* (Slika 45) te na otvorenoj stranici odabere gumb *„Go to question statistics“* (Slika 46).



Slika 45 Prikaz izbornika aplikacije s naglaskom na gumb *„Statistics“*

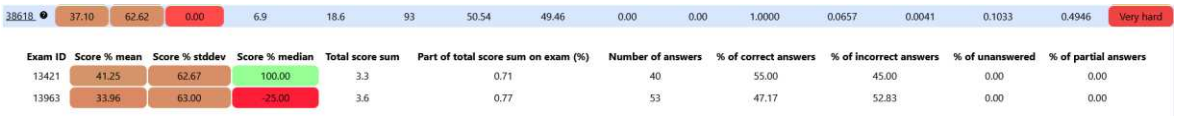


Slika 46 Prikaz ekrana s opcijama vezanim uz pregled statističkih pokazatelja s naglaskom na gumb *„Go to question statistics“*

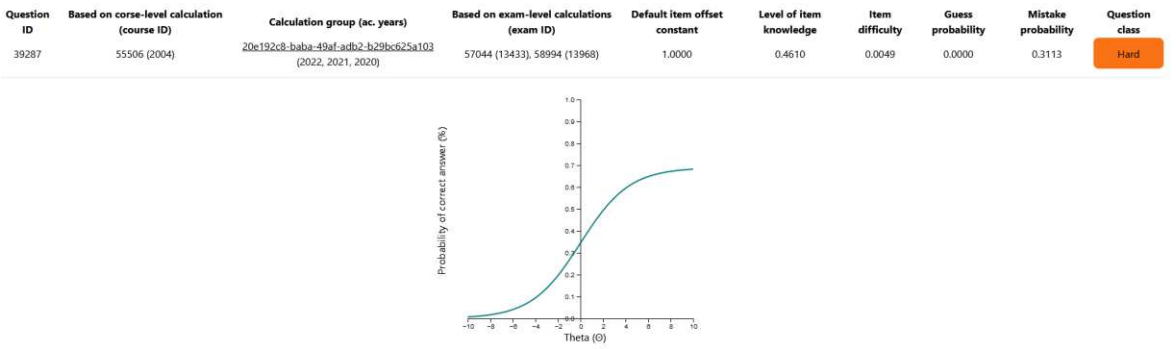


Slika 47 Prikaz ekrana na kojem su izračunati statistički pokazatelji zadataka

Klikom na redak tablice korisnik može dobiti dodatne statističke pokazatelje koji su vezani uz zadatak, ali su izračunati na razini testa (Slika 48). Korisnik također može kliknuti na identifikator zadatka pri čemu će ga aplikacija odvesti na stranicu detaljnog prikaza IRT parametara zadatka za svaki izračun u kojem se zadatak nalazio (Slika 49).

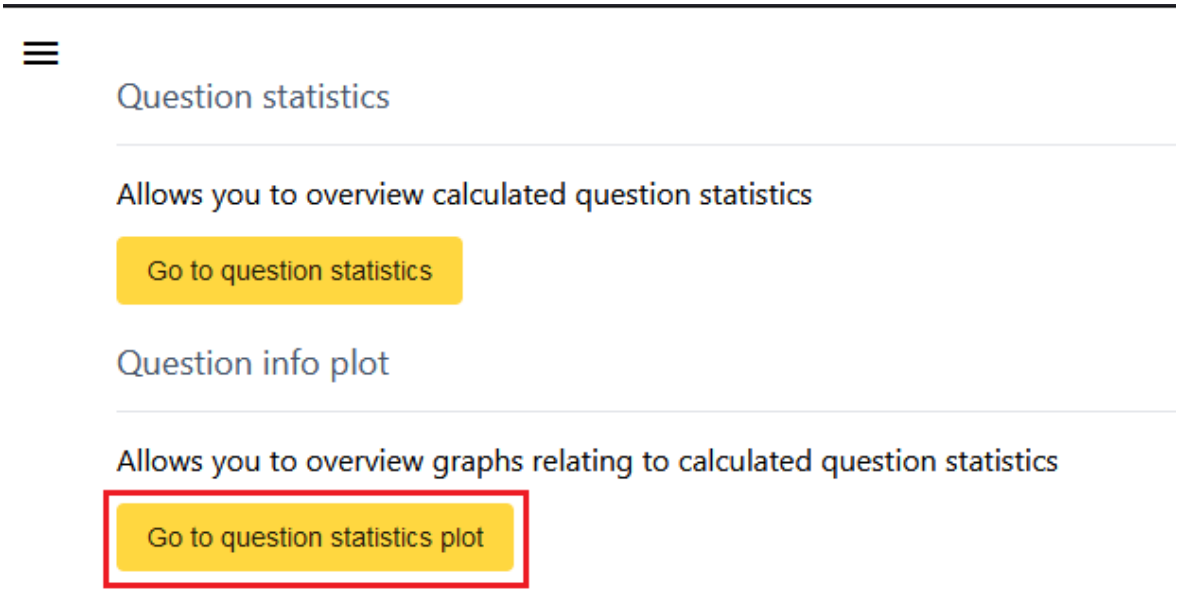


Slika 48 Prikaz proširenih informacija o statističkim pokazateljima zadatka

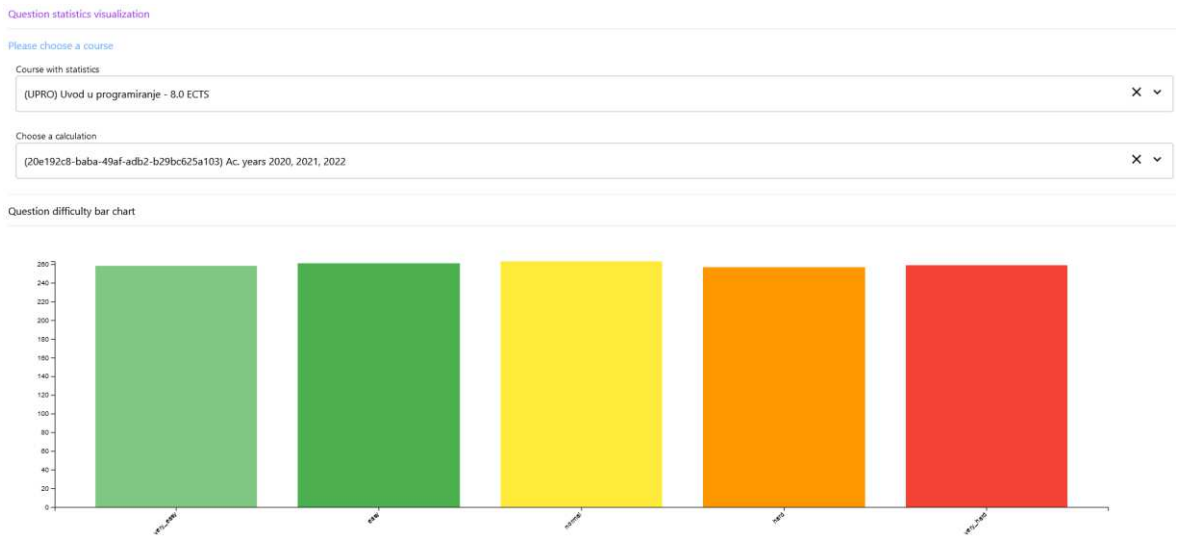


Slika 49 Prikaz IRT parametara i prikaz logističke krivulje za jedan od zadataka

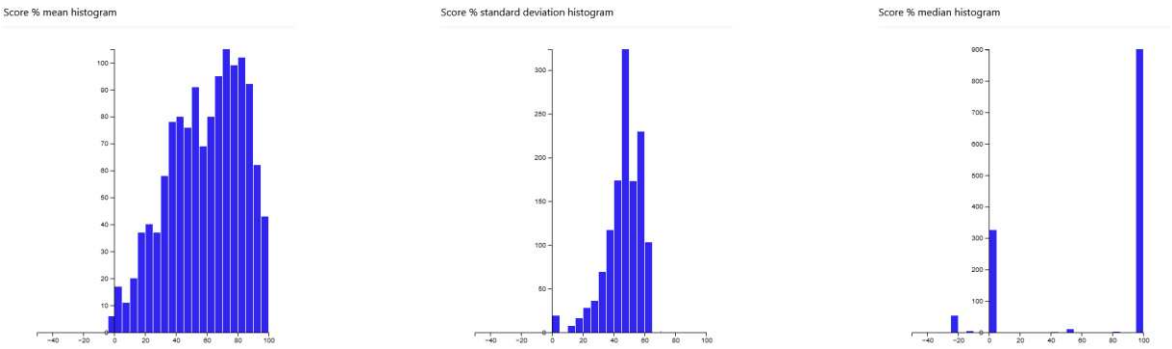
Osim tabličnog pregleda, web aplikacija omogućava i vizualni pregled statističkih pokazatelja (Slika 52) i procijenjenih težina zadataka (Slika 51). Do vizualizacija korisnik može doći ako slijedi poveznicu *„Go to question statistics plot“* (Slika 50).



Slika 50 Prikaz ekrana s opcijama vezanim uz pregled statističkih pokazatelja s naglaskom na gumb *„Go to question statistics plot“*

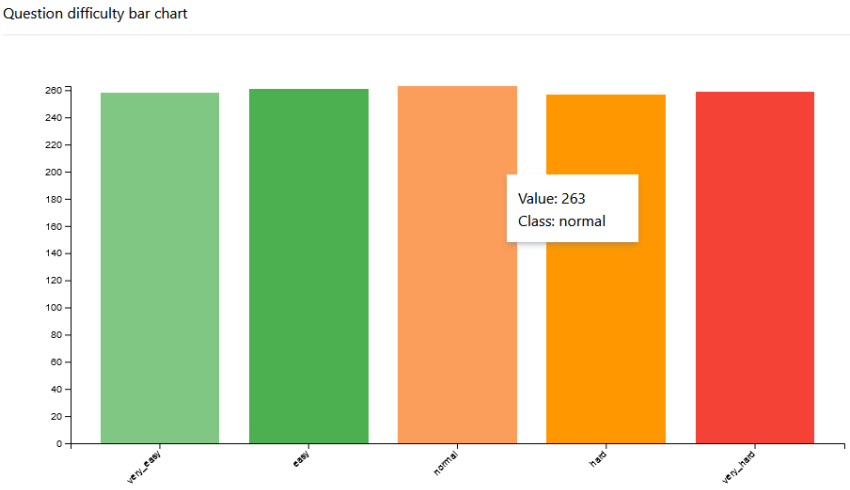


Slika 51 Prikaz stupčastog dijagrama procijenjenih težina izračuna statističkih pokazatelja



Slika 52 Vizualizacije (histogrami) statističkih pokazatelja zadataka obuhvaćenih izračunom: srednja vrijednost [lijevo], standardna devijacija [sredina] i medijan [desno] postotaka ostvarenih bodova

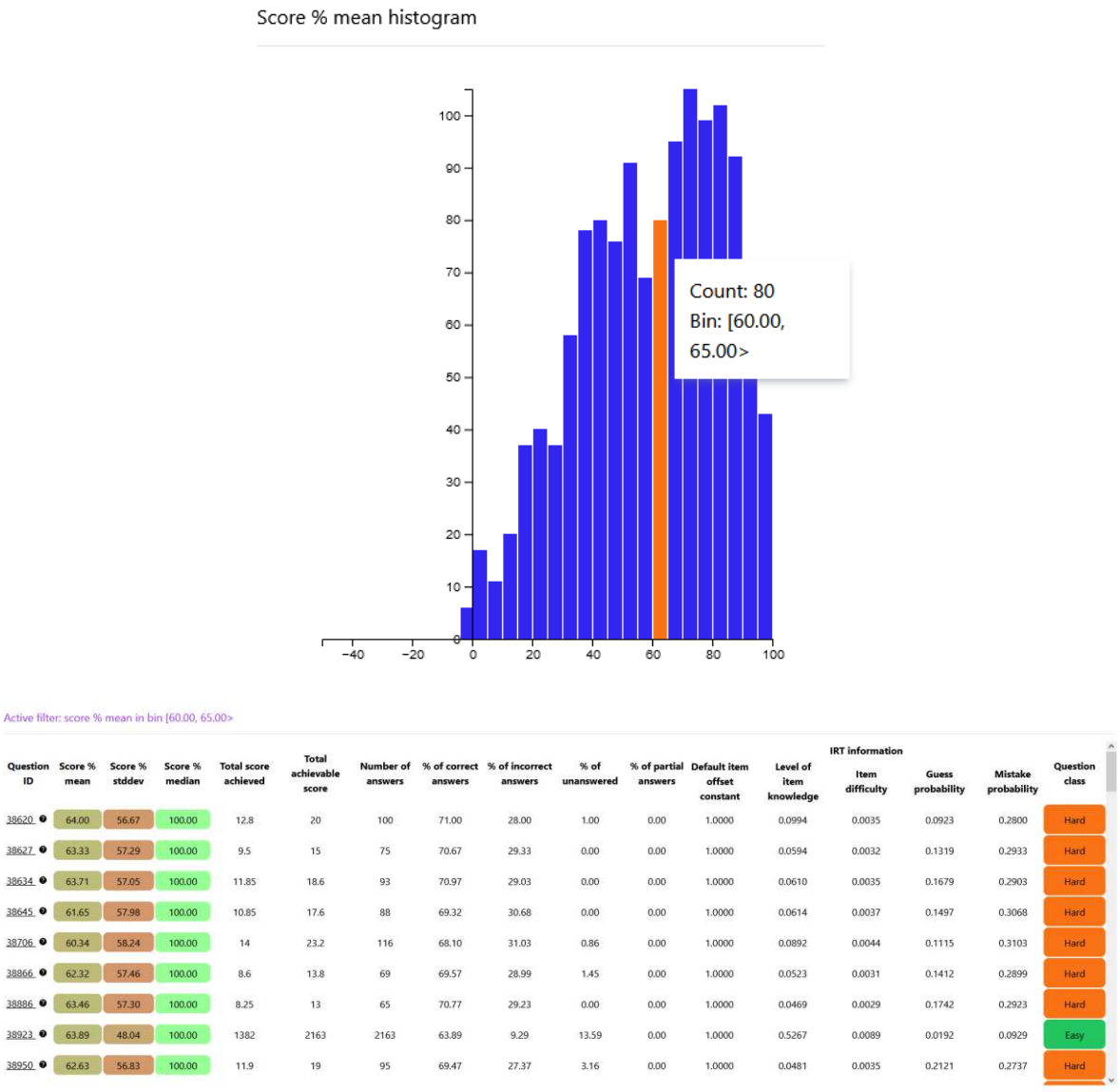
Vizualizacije su interaktivne te je odabirom vizualnih elemenata korisniku omogućen pregled zadataka koji pripadaju određenom odabiru (Slika 53 i Slika 54).



Active filter: questions of [Normal] difficulty

Question ID	Score % mean	Score % stddev	Score % median	Total score achieved	Total achievable score	Number of answers	% of correct answers	% of incorrect answers	% of unanswered	% of partial answers	Default item offset constant	Level of item knowledge	IRT information: Item difficulty	IRT information: Guess probability	IRT information: Mistake probability	Question class
38612	77.13	48.62	100.00	12.65	16.4	82	81.71	18.29	0.00	0.00	1.0000	0.0546	0.0018	0.1495	0.1829	Normal
38613	79.41	46.64	100.00	13.5	17	85	83.53	16.47	0.00	0.00	1.0000	0.0558	0.0016	0.1474	0.1647	Normal
38626	75.25	50.06	100.00	15.2	20.2	101	80.20	19.80	0.00	0.00	1.0000	0.0753	0.0021	0.1270	0.1980	Normal
38629	79.62	46.43	100.00	14.65	18.4	92	83.70	16.30	0.00	0.00	1.0000	0.0553	0.0017	0.1820	0.1630	Normal
38641	71.88	52.22	100.00	12.65	17.6	88	77.27	21.59	1.14	0.00	1.0000	0.0530	0.0023	0.1493	0.2159	Normal
38643	74.68	49.92	100.00	11.5	15.4	77	79.22	18.18	2.60	0.00	1.0000	0.1107	0.0018	0.0412	0.1818	Normal
38662	77.38	47.15	100.00	13	16.8	84	80.95	14.29	4.76	0.00	1.0000	0.0428	0.0015	0.1542	0.1429	Normal
38681	71.54	51.93	100.00	13.45	18.8	94	76.60	20.21	3.19	0.00	1.0000	0.0865	0.0023	0.0815	0.2000	Normal
38700	75.26	50.07	100.00	14.45	19.2	96	80.21	19.79	0.00	0.00	1.0000	0.0701	0.0021	0.1170	0.1979	Normal

Slika 53 Prikaz filtriranja po težini zadatka (odabrana težina je „*Normal*“)

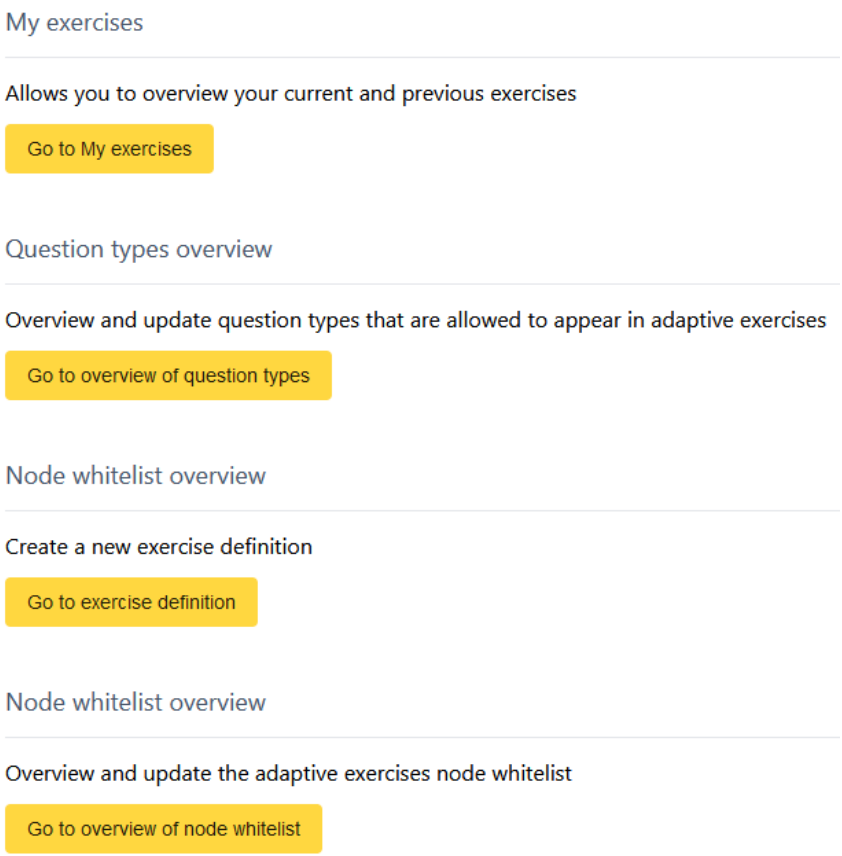


Question ID	Score % mean	Score % stddev	Score % median	Total score achieved	Total achievable score	Number of answers	% of correct answers	% of incorrect answers	% of unanswered	% of partial answers	Default item offset constant	Level of item knowledge	Item difficulty	Guess probability	Mistake probability	Question class
38620	64.00	56.67	100.00	12.8	20	100	71.00	28.00	1.00	0.00	1.0000	0.0994	0.0035	0.0923	0.2800	Hard
38627	63.33	57.29	100.00	9.5	15	75	70.67	29.33	0.00	0.00	1.0000	0.0594	0.0032	0.1319	0.2933	Hard
38634	63.71	57.05	100.00	11.85	18.6	93	70.97	29.03	0.00	0.00	1.0000	0.0610	0.0035	0.1679	0.2903	Hard
38645	61.65	57.98	100.00	10.85	17.6	88	69.32	30.68	0.00	0.00	1.0000	0.0614	0.0037	0.1497	0.3068	Hard
38706	60.34	58.24	100.00	14	23.2	116	68.10	31.03	0.86	0.00	1.0000	0.0892	0.0044	0.1115	0.3103	Hard
38866	62.32	57.46	100.00	8.6	13.8	69	69.57	28.99	1.45	0.00	1.0000	0.0523	0.0031	0.1412	0.2899	Hard
38886	63.46	57.30	100.00	8.25	13	65	70.77	29.23	0.00	0.00	1.0000	0.0469	0.0029	0.1742	0.2923	Hard
38923	63.89	48.04	100.00	1382	2163	2163	63.89	9.29	13.59	0.00	1.0000	0.5267	0.0089	0.0192	0.0929	Easy
38950	62.63	56.83	100.00	11.9	19	95	69.47	27.37	3.16	0.00	1.0000	0.0481	0.0035	0.2121	0.2737	Hard

Slika 54 Prikaz filtriranja po srednjoj vrijednosti postotaka osvojenih bodova (odabran raspon je [60, 65>)

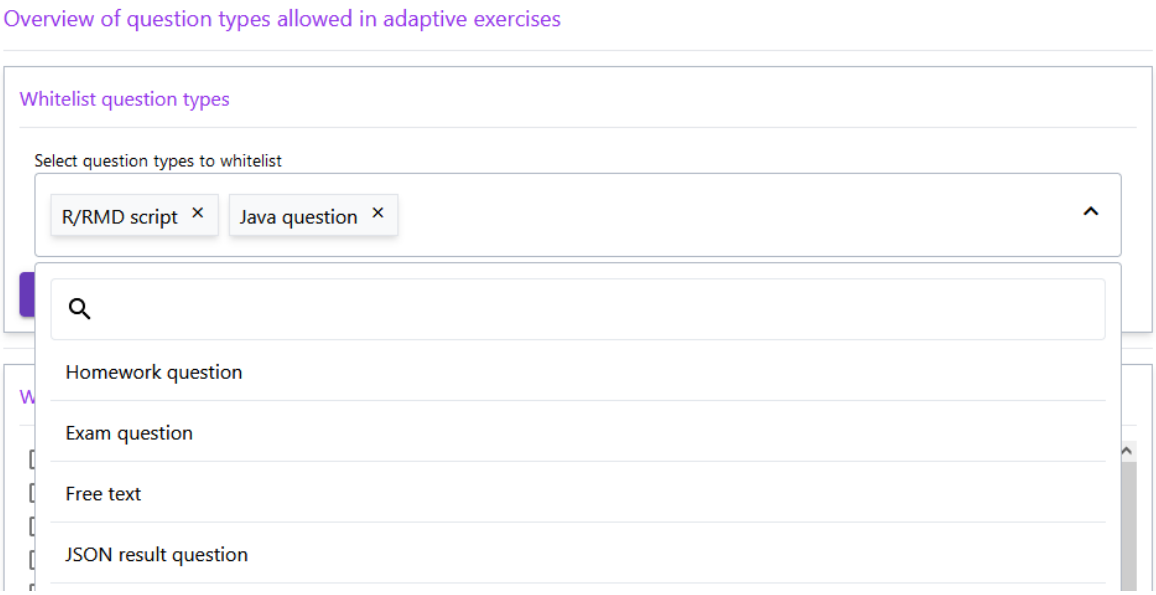
7.4.4. Rad s prilagodljivim vježbama – nastavnik

Ako korisnik klikne na gumb *„Adaptive exercises“* u izborniku, dobiva pregled dostupnih opcija (Slika 55).

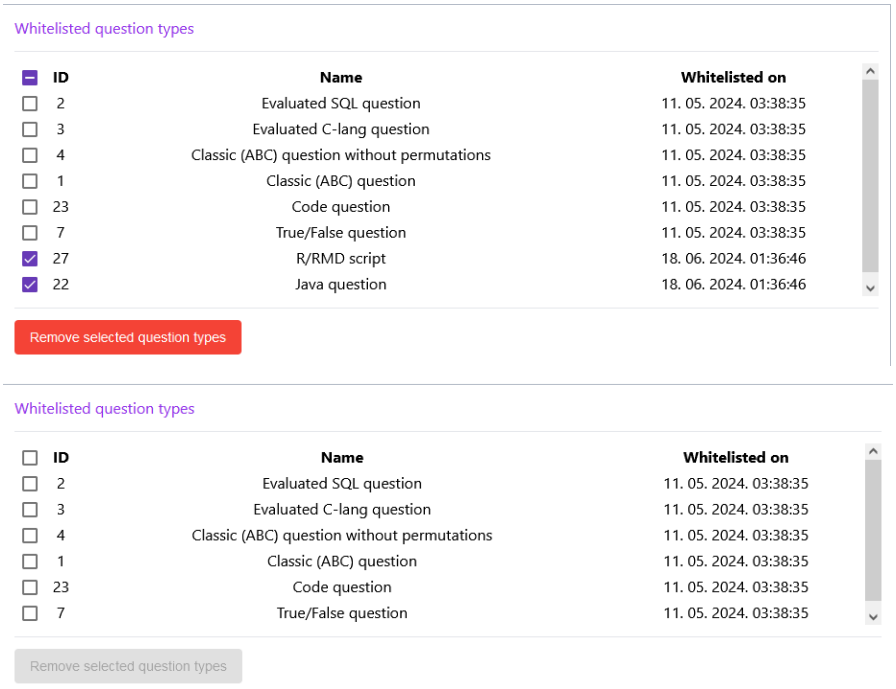


Slika 55 Opcije vezane uz prilagodljive vježbe

Odabirom opcije *„Go to overview of question types“* korisnik može otvoriti stranicu na kojoj će moći odabrati koji tipovi zadataka se smiju pojaviti u prilagodljivim vježbama (Slika 56). Korisnik također može pregledati i obrisati tipove zadataka koji su dozvoljeni na prilagodljivim vježbama (Slika 57).

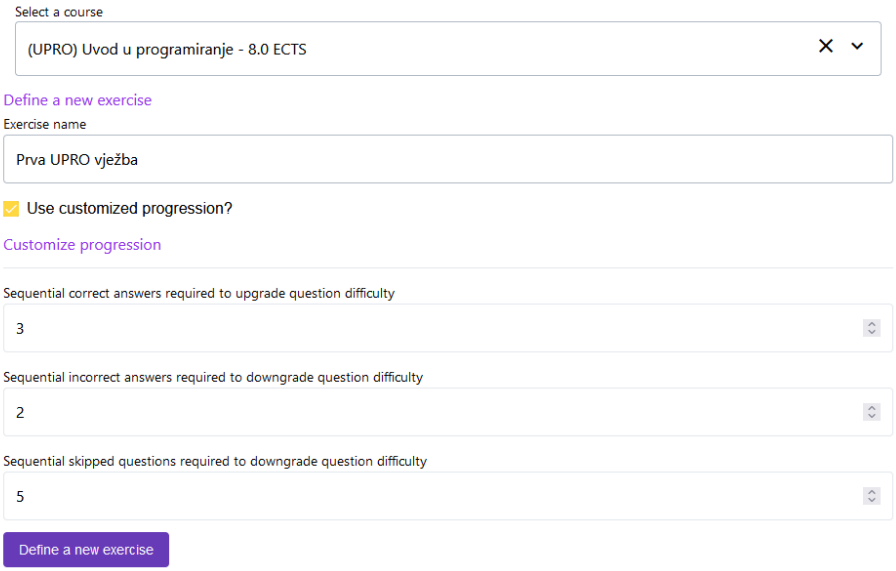


Slika 56 Prikaz funkcionalnosti dodavanja novih tipova zadataka



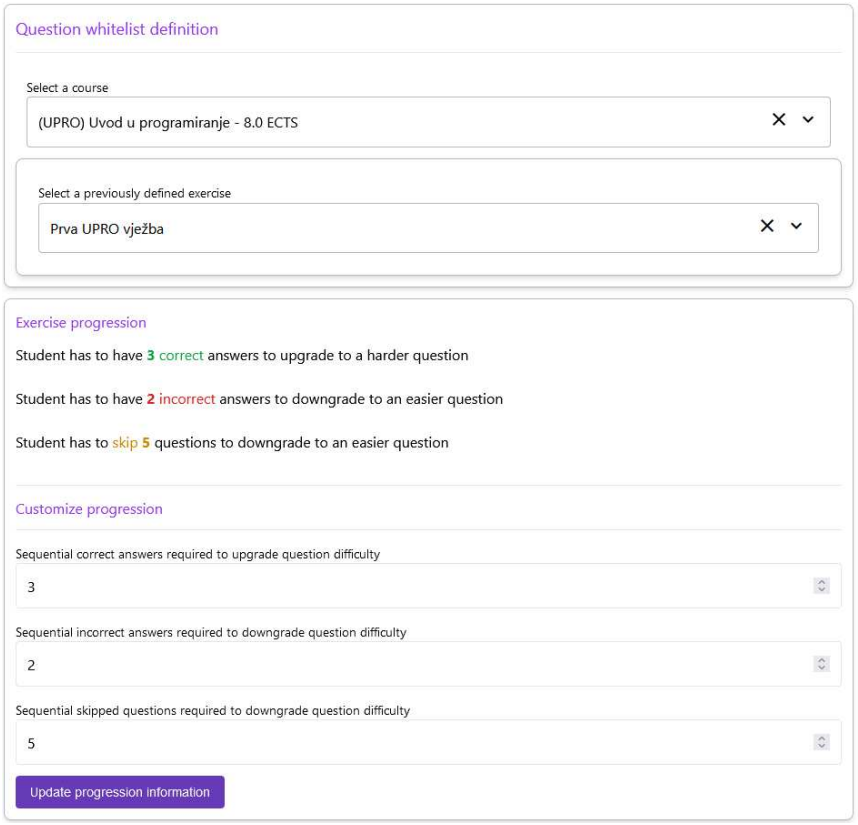
Slika 57 Prikaz funkcionalnosti uređivanja dozvoljenih tipova zadataka: stanje prije brisanja [gore] i stanje nakon brisanja [dolje]

Odabirom opcije *„Go to exercise definition“* korisnik može definirati novu prilagodljivu vježbu (Slika 58). Na ovom ekranu korisnik odabire kolegij i nakon toga unosi naziv vježbe koju želi definirati. Korisnik također može odlučiti nadjačati pretpostavljene parametre modela prilagodljivosti ako to želi.

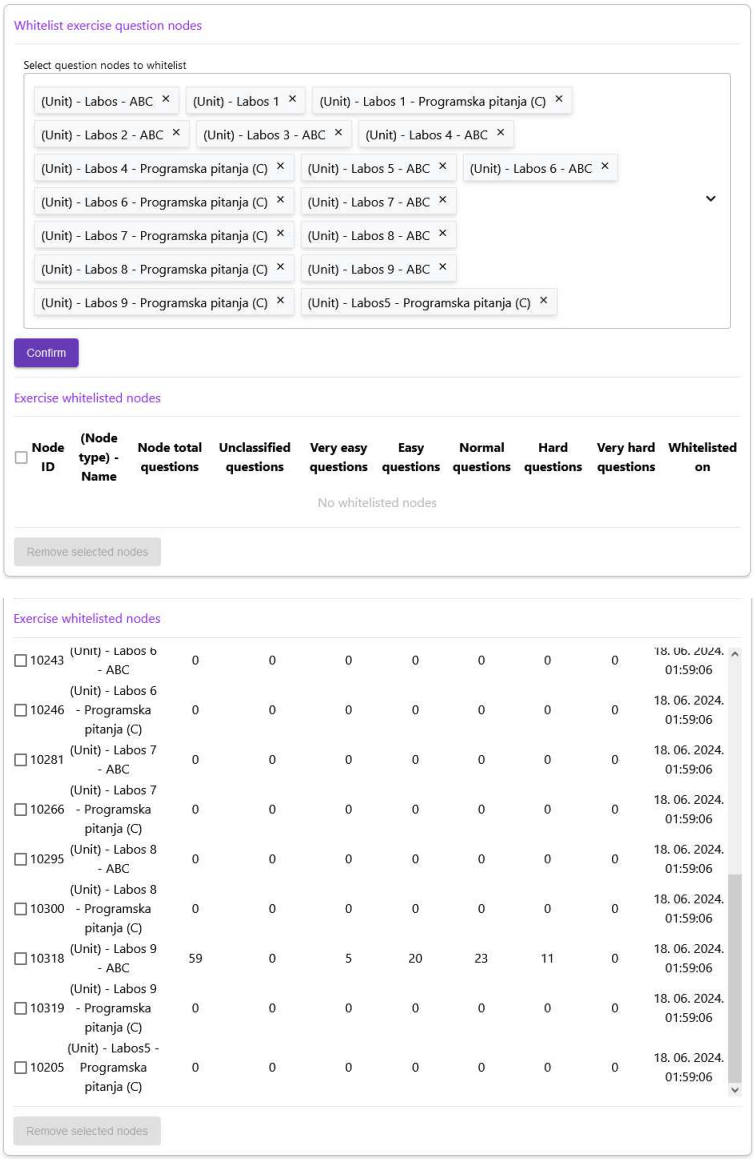


Slika 58 Funkcionalnosti definiranja nove prilagodljive vježbe (prikazane vrijednosti parametara modela prilagodljivosti su pretpostavljene vrijednosti)

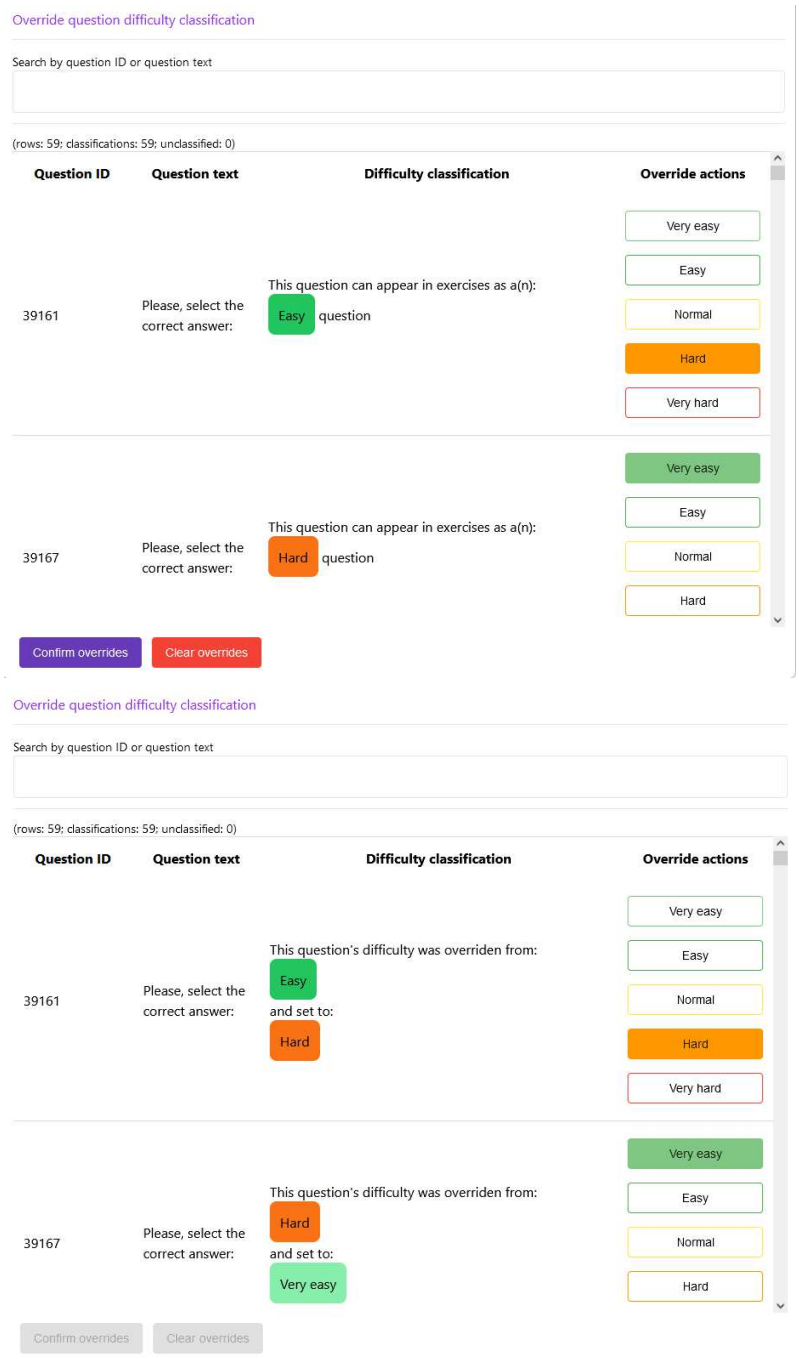
Nakon što je stvorio vježbu, korisnik može urediti sastav vježbe tako što će odabrati opciju *„Go to overview of node whitelist“* na glavnom ekranu. Aplikacija nakon odabira opcije otvara ekran na kojem korisnik prvo bira kolegij te zatim bira jednu od prethodno definiranih vježbi. Nakon toga može uređivati parametre prilagodljivosti (Slika 59), čvorove iz kojih se u vježbama smiju pojaviti zadatci (Slika 60) te ako postoje zadatci koji su dodani u definiciju vježbe, korisnik im može nadjačati procijenjene težine (Slika 61). Težine zadataka u vježbi također imaju i vizualni prikaz koji je interaktivan (Slika 62).



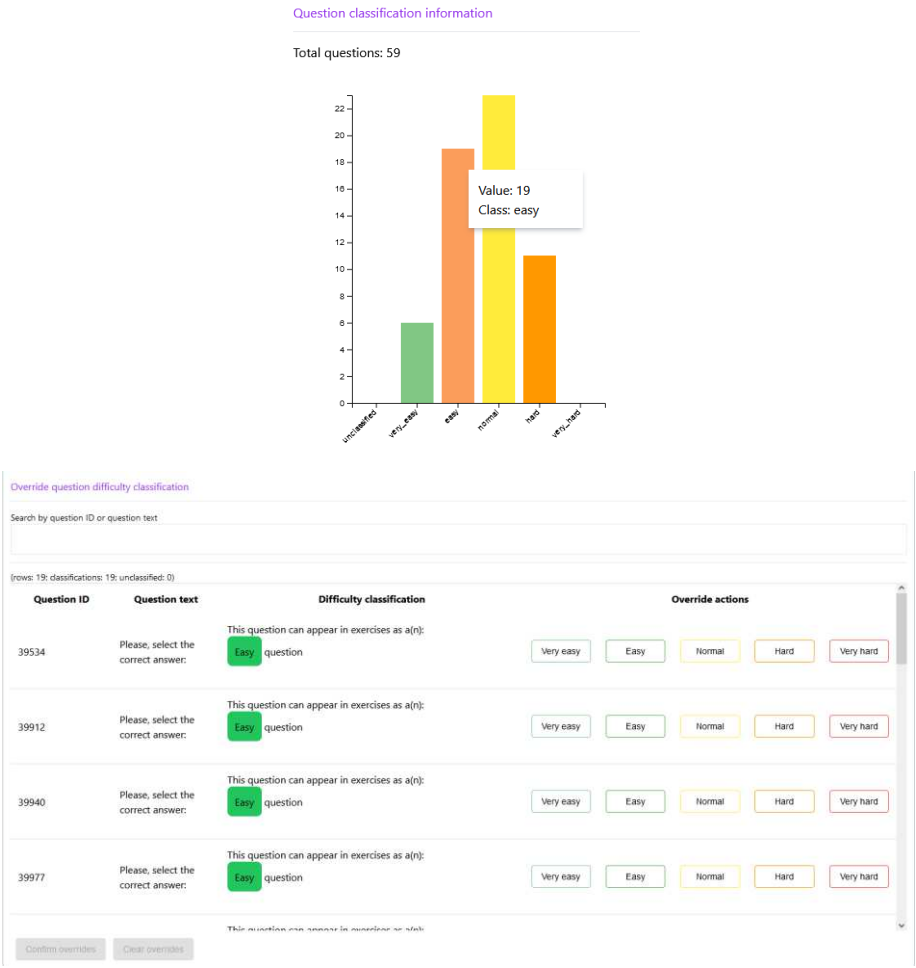
Slika 59 Prikaz dijela stranice za uređivanje modela prilagodljivosti nakon odabira definicije vježbe



Slika 60 Prikaz funkcionalnosti dodavanja čvorova sa zadatcima u definiciju vježbe prije [gore] i nakon [dolje]



Slika 61 Prikaz funkcionalnosti nadjačavanja težina zadataka: stanje za vrijeme nadjačavanja težine [gore] i nakon nadjačavanja težine [dolje] zadatka

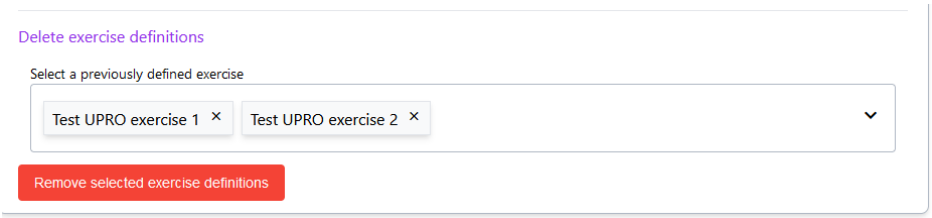


Slika 62 Težine zadataka u odabranoj definiciji vježbe i filtriranje prema težini (odabrana težina zadatka je *„Easy“*)

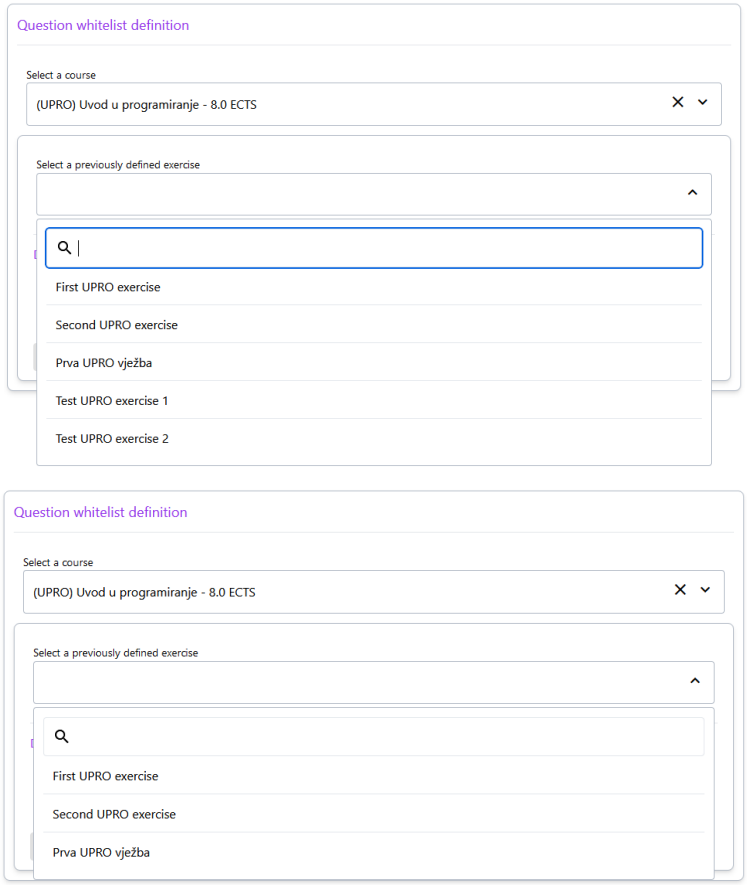
Aplikacija omogućuje i zadavanje težine zadatcima za koje nije bilo moguće izračunati statističke pokazatelje (Slika 63). Kako bi se ova situacija prikazala bilo je potrebno dodati nekoliko čvorova u definiciju vježbe kako bi se pojavili zadatci za koje nisu izračunati statistički pokazatelji. Budući da se zadatci bez procijenjene težine ne mogu pojaviti u vježbama, na ovaj način moguće je za zadatak koji nije obuhvaćen izračunom odrediti težinu i tako ga dodati u bazen zadataka vježbe.

Slika 63 Prikaz stanja težina zadataka definicije vježbe prije [lijevo] i nakon [desno] postavljanja težine *„Hard“* nekim od zadataka bez procijenjene težine

Na ovom ekranu korisnik može i obrisati odabrane definicije vježbi ako to želi (Slika 64 i Slika 65).



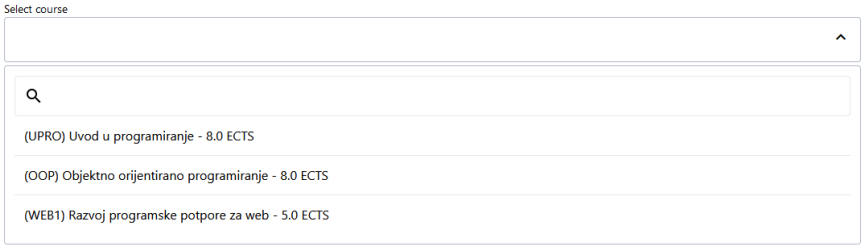
Slika 64 Odabir definicija vježbi koje treba obrisati



Slika 65 Prikaz funkcionalnosti brisanja: mogućnost izbora vježbi prije [gore] i nakon [dolje] brisanja definicija vježbi

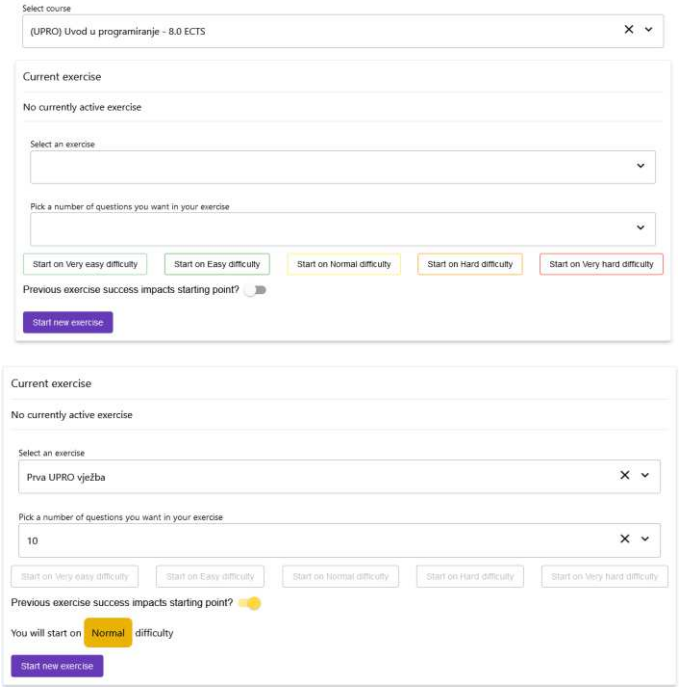
7.4.5. Rad s prilagodljivim vježbama – student

Odabirom opcije *„Go to my exercises“* korisnik pristupa ekranu na kojem je moguće pregledati prethodne vježbe, pokrenuti novu vježbu ili pokrenuti staru vježbu. Kako bi korisnik imao pristup ovim mogućnostima, prvo mora odabrati kolegij iz padajućeg izbornika (Slika 66). U padajućem izborniku moguće je odabrati samo one kolegije za koje postoji definirana barem jedna prilagodljiva vježba.



Slika 66 Prikaz padajućeg izbornika s izborom kolegija

Nakon odabira kolegija, korisnik može pokrenuti novu vježbu ako na odabranom kolegiju ne postoji prethodno pokrenuta vježba (Slika 67). Kako bi pokrenuo vježbu mora odabrati definiciju vježbe i broj zadataka na vježbi. Nakon što je ispunio obrazac, korisnik vježbu pokreće klikom na gumb *„Start new exercise“*.

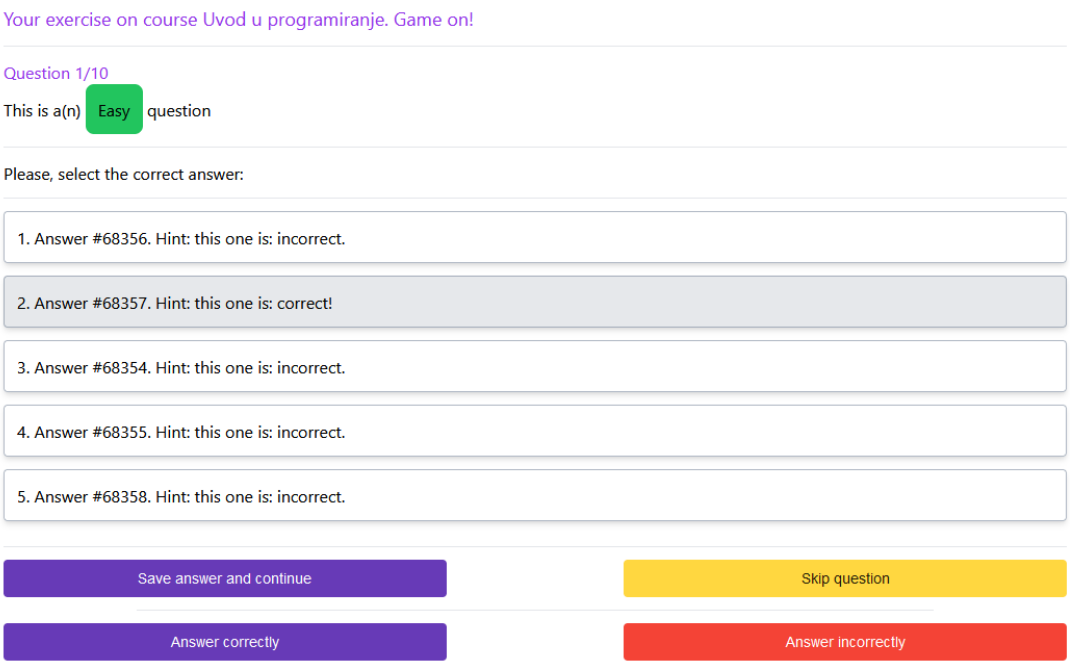


Slika 67 Prikaz praznog [gore] i popunjenog [dolje] obrasca za pokretanje prilagodljive vježbe

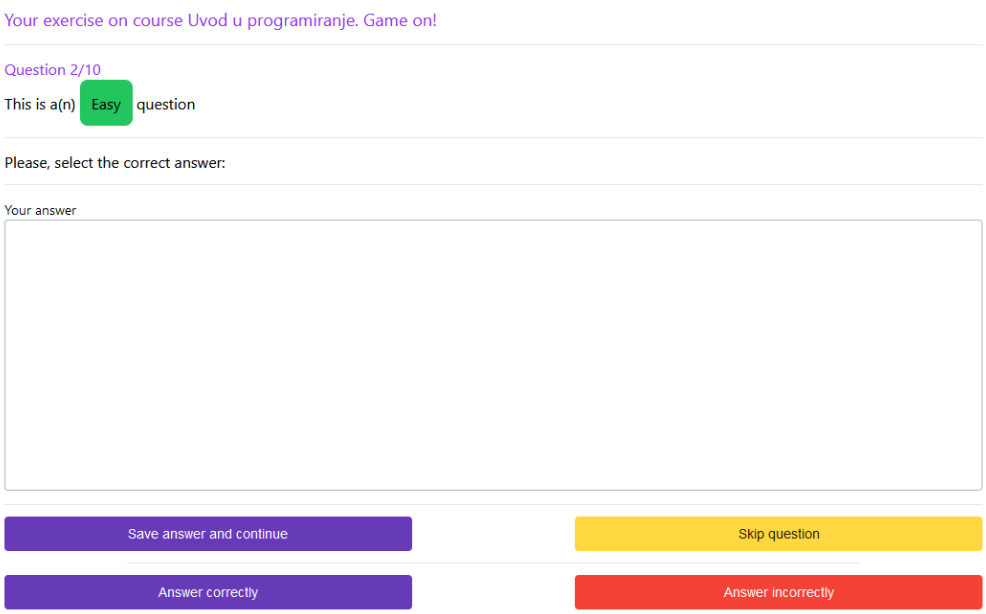
Korisnik može odabrati i jednu od težina početnog zadatka vježbe ili može zatražiti sustav da procijeni težinu početnog zadatka na temelju prethodnih vježbi unutar te definicije vježbe (opcija *„Previous exercise success impacts starting point?“*). Odabir težine nije obavezan te ako student ne unese početnu težinu sustav će mu zadati zadatak prosječne krajnje dostignute težine na prethodnim instancama odabrane definicije vježbe ili zadatak težine *„Easy“* ako je to prva instanca vježbe te definicije.

U primjeru će biti prikazana situacija pokretanja vježbe bez odabira težine ili procjene težine prvog zadatka. Aplikacija korisnika nakon pokretanja vježbe preusmjerava na stranicu za vježbanje (Slika 68 i Slika 69). U *„zaglavlju“* je prikazan naziv kolegija te broj i težina zadatka na kojem se student trenutno nalazi. U *„tijelu“* se nalaze tekst zadatka i ponuđeni odgovori ili polje za unos teksta ili koda (ovisno o tipu zadatka). U *„podnožju“* se nalaze moguće akcije:

- *„Save answer and continue“* – pohranjuje odgovor na trenutni zadatak i učitava sljedeći
- *„Skip question“* – preskače trenutni i učitava sljedeći zadatak
- *„Answer correctly“* – *„demonstracijska“* opcija za postavljanje točnog odgovora
- *„Answer incorrectly“* – *„demonstracijska“* opcija za postavljanje netočnog odgovora

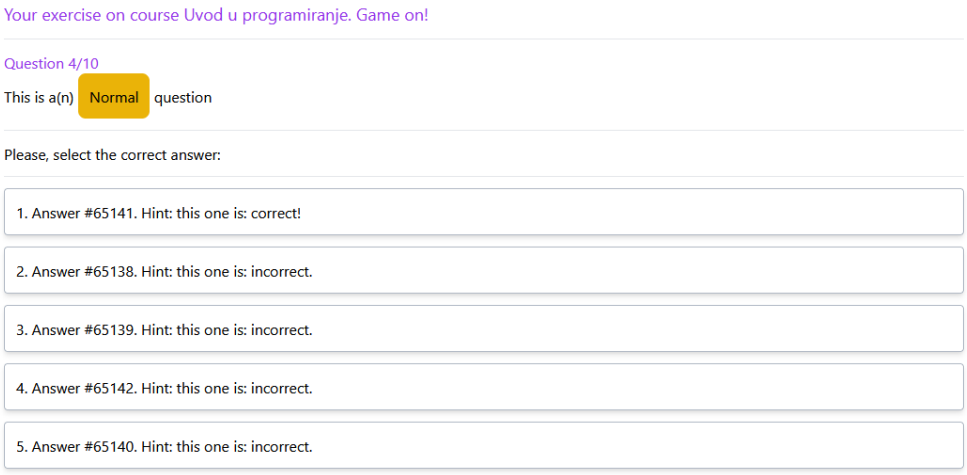


Slika 68 Prikaz ekrana prvog zadatka pokrenute vježbe s označenim odgovorom pod rednim brojem 2



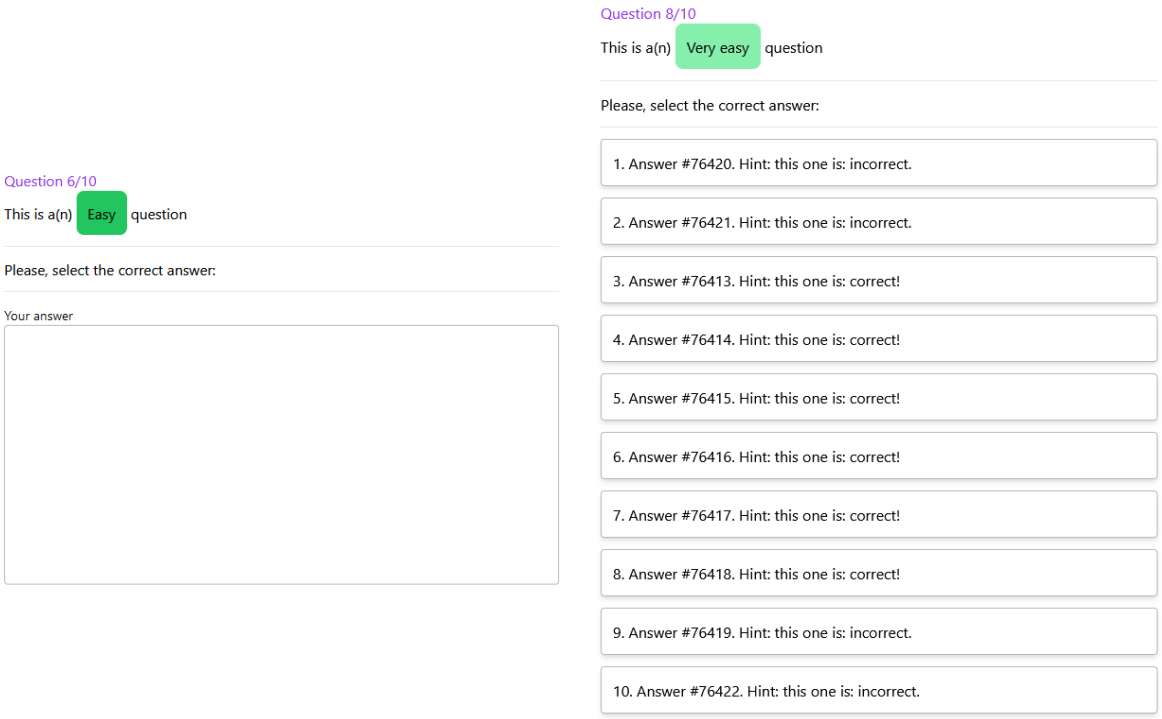
Slika 69 Prikaz zadatka s tekstualnim odgovorom

Model prilagodljivosti koji je definiran na razini definicije vježbe primjenjuje se u instanci definicije vježbe. Ako student na 3 uzastopna zadatka da točan odgovor dobit će zadatak sljedeće težine (u slučaju primjera s težine *„Easy“* prelazi na težinu *„Normal“* – Slika 70).



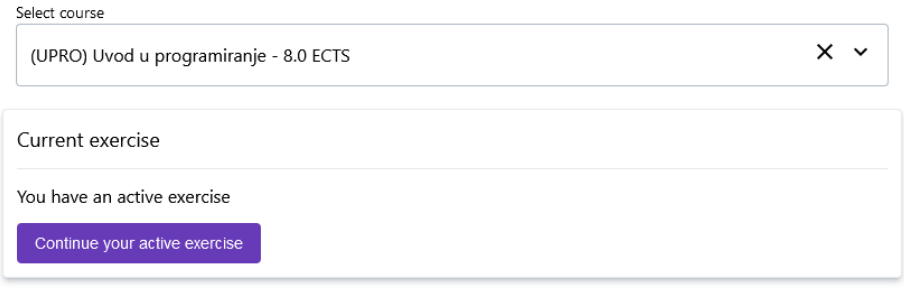
Slika 70 Zadatak zadan nakon 3 uzastopna točna odgovora

Ako zatim pogrešno odgovori na dva uzastopna zadatka vraća se na prethodnu težinu (u slučaju primjera s težine *„Normal“* vraća se na težinu *„Easy“*). S još dva uzastopna netočna odgovora prelazi u sljedeću nižu razinu (u slučaju primjera s težine *„Easy“* prelazi na težinu *„Very easy“* – Slika 71).



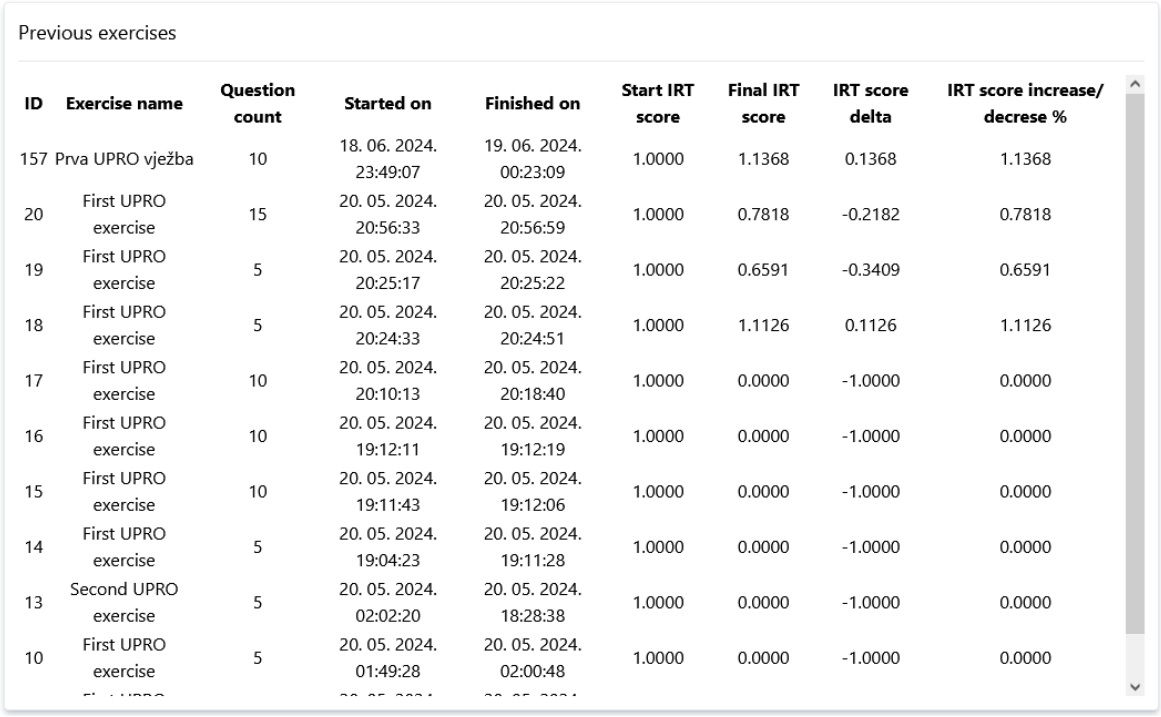
Slika 71 Zadatak zadan nakon dva [lijevo] i četiri [desno] uzastopna pogrešna odgovora

Ako korisnik u nekom trenutku izađe iz vježbe može ju nastaviti na stranici *„My exercises“* (Slika 72). Klikom na gumb *„Continue your active exercise“* aplikacija korisnika preusmjerava na stranicu za vježbanje.



Slika 72 Prikaz ekrana *„My exercises“* u slučaju kada postoji pokrenuta vježba

Na stranici prilagodljivih vježbi korisnik može pregledati rezultate prethodno odrađenih vježbi (Slika 73). U tablici se prikazuju naziv definicije vježbe uz koju je vezana instanca vježbe, broj zadataka, vrijeme početka, vrijeme završetka, početna i konačna vrijednost pokazatelja latentne sposobnosti te iznos apsolutne i relativne promjene pokazatelja latentne sposobnosti.



Previous exercises

ID	Exercise name	Question count	Started on	Finished on	Start IRT score	Final IRT score	IRT score delta	IRT score increase/ decrese %
157	Prva UPRO vježba	10	18. 06. 2024. 23:49:07	19. 06. 2024. 00:23:09	1.0000	1.1368	0.1368	1.1368
20	First UPRO exercise	15	20. 05. 2024. 20:56:33	20. 05. 2024. 20:56:59	1.0000	0.7818	-0.2182	0.7818
19	First UPRO exercise	5	20. 05. 2024. 20:25:17	20. 05. 2024. 20:25:22	1.0000	0.6591	-0.3409	0.6591
18	First UPRO exercise	5	20. 05. 2024. 20:24:33	20. 05. 2024. 20:24:51	1.0000	1.1126	0.1126	1.1126
17	First UPRO exercise	10	20. 05. 2024. 20:10:13	20. 05. 2024. 20:18:40	1.0000	0.0000	-1.0000	0.0000
16	First UPRO exercise	10	20. 05. 2024. 19:12:11	20. 05. 2024. 19:12:19	1.0000	0.0000	-1.0000	0.0000
15	First UPRO exercise	10	20. 05. 2024. 19:11:43	20. 05. 2024. 19:12:06	1.0000	0.0000	-1.0000	0.0000
14	First UPRO exercise	5	20. 05. 2024. 19:04:23	20. 05. 2024. 19:11:28	1.0000	0.0000	-1.0000	0.0000
13	Second UPRO exercise	5	20. 05. 2024. 02:02:20	20. 05. 2024. 18:28:38	1.0000	0.0000	-1.0000	0.0000
10	First UPRO exercise	5	20. 05. 2024. 01:49:28	20. 05. 2024. 02:00:48	1.0000	0.0000	-1.0000	0.0000

Slika 73 Prikaz dijela ekrana *„My exercises“* koji s informacijama o prethodnim vježbama

8. Pregled korištenih tehnologija

Za sustav upravljanja bazama podataka (SUBP, *eng. database management system, DBMS*) korišten je *PosgreSQL* DBMS. *PostgreSQL* je odabran zato što se već koristi u sustavu *Edgar*, jednostavan je za instalaciju i korištenje, posjeduje pogodnu organizaciju baze podataka u sheme, sustav je otvorenog koda (*eng. open source*) i najbliži je SQL standardu.

Aplikacijska logika modula za izvođenje poslova, servisa za izračun statističkih pokazatelja, poslužitelja web aplikacije te korisničkog sučelja web aplikacije razvijena je u *NodeJS* okruženju. *NodeJS* okruženje omogućuje razvoj *„klasičnih“* aplikacija u programskom jeziku *JavaScript*, iako je primarno izvršno okruženje koda pisanog u *JavaScriptu* internetski preglednik.

Za razvoj potrebnih funkcionalnosti korišteno je nekoliko ovisnosti (*eng. dependency*) iz repozitorija *npm-a* (*eng. node package manager*). Program *npm* služi za rad s paketima ovisnosti u okruženju *NodeJS-a*, a u njegovom repozitoriju se nalazi velik broj raznih rješenja otvorenog koda.

Za potrebe rada s redovima kod organizacije poslova korišten je modul *pg-boss*. Ovaj modul oslanja se na *PostgreSQL* bazu podataka kako bi pružio usluge zadavanja i preuzimanja poslova. Ova funkcionalnost bila je korisna za povezivanje web aplikacije i servisa za izračun statističkih pokazatelja. Zahtjev za izvršavanje posla iz web aplikacije se uz pomoć *pg-bossa* smješta u red (*eng. queue*) na čije promjene se, također uz pomoć *pg-bossa*, prethodno pretplatio servis za izračun statističkih pokazatelja.

Za potrebe razvoja HTTP poslužitelja aplikacijske logike web aplikacije korišten je programski okvir *expressJS* koji omogućuje brzi razvoj web aplikacija korištenjem jednostavnog sučelja. U ovom programskom okviru moguće je jednostavno definirati na kojim putanjama (*eng. path*) poslužitelj čeka zahtjeve.

Korisničko sučelje web aplikacije razvijeno je u radnom okviru (*eng. framework*) *Angular. Angular* je radni okvir otvorenog koda za razvoj jednostraničnih web aplikacija. Za potrebe iscrtavanja vizualizacija u korisničkom sučelju korišten je radni okvir D3.js. D3.js jedan je od najkorištenijih alata za izradu vizualizacija u programskom kodu.

Sav programski kod pisan je u programskom jeziku *TypeScript*. *TypeScript* je programski jezik koji dodaje statičke tipove (*eng. static type system*) u *JavaScript*. U konačnici sav kod pisan u *TypeScriptu* se prevodi u *JavaScript*. Odabran je zato što znatno olakšava razvoj jer smanjuje mogućnost pogrešaka uzrokovanih sustavom dinamičkih tipova i omogućuje njihovo ranije otkrivanje.

Kao alat za upravljanje verzijama koda korišten je alat *git*. Ovaj alat koristan je za praćenje promjena inkrementalnih procesa na tekstualnim datotekama.

Za vizualizacije matematičkih funkcija (preciznije logističke krivulje) korišten je alat *GeoGebra* dok je za izradu svih dijagrama korišten alat *Microsoft Visio*.

9. Ocjena ostvarenog pristupa

Razvijeni modul za izvođenje poslova servisu za izračun statističkih pokazatelja omogućava jednostavno nadograđivanje i prilagodljivost postupka izračuna statističkih pokazatelja i parametara IRT modela zadataka.

Razvijeni servis moguće je samostalno pokrenuti bez postojanja instance *Edgara* ili web aplikacije koja će servis koristiti. Iako ovo nema smisla u navedenom kontekstu, potrebno je naglasiti kako je cilj razvoja bio samostalni modul za izračun statističkih pokazatelja zadataka. Budući da je servis moguće prilagoditi bilo kojem sustavu, ovaj zahtjev se može smatrati ispunjenim.

Servis za izračun statističkih pokazatelja može koristiti sustav *Judge0* za potrebe izračuna, čime je ostvaren zahtjev podržavanja integracije s okruženjem sustava *Edgar*.

Korisnici u sklopu razvijene web aplikacije mogu pokrenuti izračun statističkih pokazatelja i pregledati ih koristeći odgovarajuće vizualizacije. Na temelju izračunatih pokazatelja sustav izrađuje IRT modele zadataka te omogućuje definiranje prilagodljivih vježbi na temelju tih modela. Definirane prilagodljive vježbe moguće je uređivati (na razini definicije vježbe dodavati nove zadatke i nadjačavati težine zadataka).

U sklopu razvijene web aplikacije korisnici mogu pokretati definirane prilagodljive vježbe. Za vrijeme trajanja vježbe korisniku sustav daje zadatke koji su temeljeni na njegovom uspjehu u određenom trenutku vježbe. Težinu početnog zadatka vježbe eksplicitno može odabrati korisnik ili ju sustav implicitno određuje na temelju informacija o prethodnim instancama vježbe koju korisnik pokreće.

Svi osnovni zahtjevi zadatka su ispunjeni te se može smatrati da su ostvareni pristup i razvijeni sustav zadovoljili potrebe zadatka.

10. Smjernice za budući razvoj

U budućnosti je potrebno sve funkcionalnosti sustava i web aplikacije integrirati sa sustavom *Edgar*. Budući da su razvojna okruženja i korištene tehnologije oba sustava uglavnom jednaka, dio koda razvijenog sustava može se preuzeti u sustav *Edgar* bez previše izmjena.

Na strani funkcionalnosti bilo bi dobro dodati mogućnost pregleda i analize pristupa vježbama. Tako bi se za svaku definiciju vježbe mogao na primjer dodati dijagram koji prikazuje tok težina zadataka vježbe. Osim toga mogle bi se dodati statistike uspješnosti rješavanja zadataka prema težinama i odgovarajuće vizualizacije. Postupak dodavanja zadataka definicijama vježbi mogao bi se obogatiti algoritmom balansiranja težina zadataka kako bi svaka težina bila jednako zastupljena na razini definicije vježbe.

Također, bilo bi poželjno izabrati i isprobati neki drugi način izračuna parametara logističke krivulje. Bilo bi dobro predložiti više različitih opcija izračuna te ih ispitati, usporediti i ocijeniti te izabrati najbolju od njih na temelju kvalitete procjene težina zadataka.

Korisničko sučelje procesa vježbanja moglo bi se obogatiti općenitim pregledom prethodnih zadataka kako bi korisnik imao uvid u napredak vježbe za vrijeme vježbanja. U prikaz povijesti vježbanja trebalo bi dodati i prikaz detalja vezanih uz zadatke (npr. broj točnih odgovora, broj prijelaza iz niže u višu težinu i obrnuto, broj zadataka koji su se pojavili …). Uz to bilo bi dobro dodati ekran koji bi omogućio usporedbu rezultata vježbanja s drugim korisnicima.

Zaključak

Analizom više različitih metoda ispitivanja znanja, za prilagodljive vježbe najbolje se pokazala Teorija odgovora na zadatke (IRT). Uz pomoć statističke podloge koju teorija pruža izrađen je sustav za provođenje prilagodljivih vježbi. Ovakav sustav omogućava studentima jednostavnije savladavanje materijala jer ih vodi od lakših prema težim zadatcima i to ovisno o procijenjenoj sposobnosti.

Implementirani sustav za provođenje prilagodljivih vježbi dosta je kompleksan. Potrebno je imati više dijelova koji međusobno razmjenjuju podatke te omogućavaju korisnicima pregled i izmjenu podataka. Bitno je ovako kompleksne sustave kvalitetno organizirati i projektirati kako ne bi došlo do problema *„zastarijevanja koda“* i problema teške ili nemoguće nadogradnje. Zato je potrebno pronaći prikladni način dekomponiranja sustava u nezavisne module.

Budući da je sustav *Edgar* već dosta razrađen i opširan, javlja se potreba za rasterećenjem glavnog repozitorija koda. Ovo rasterećenje omogućava razvoj funkcionalnosti kao nezavisnih modula gdje god je to moguće. Na takav način u sustavu *Edgar* samo je potrebno dodati komponente sučelja koje bi komunicirale s razvijenim komponentama te bi se tako olakšao postupak razvoja novih funkcionalnosti koje moraju u potpunosti biti razvijene u *Edgaru*.

Najveći problem kod razvoja sustava bio je organizirati programski kod za potrebe statističke analize podataka o zadatcima. U početku je ideja bila to obaviti direktno u kodu, ali ubrzo se javio problem potrebe za proširivanjem postojećeg procesa izračuna. Budući da je za ovakav oblik procesa to dosta čest slučaj, nakon nekog vremena razvila se ideja o razvoju *„podsustava“* za izvođenje poslova. Budući da se poslovni proces pripreme zadataka za prilagodljive vježbe može rastaviti na više kontekstno nezavisnih, ali slijedno zavisnih koraka, razvoj *„podsustava“* za izvođenje poslova riješio je glavni problem razvoja cjelokupnog sustava.

Literatura

[1] Ž. Baranek, *Prilagodljive vježbe i lekcije u sustavu Edgar,* Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, 2018.

[2] T. Albano, »CTT review - Introduction to Educational and Psychological Measurement Using R,« 23 Veljača 2020. [Mrežno]. Available: https://www.thetaminusb.com/intro-measurement-r/irt.html#ctt-review. [Pokušaj pristupa 19 Lipanj 2024].

[3] T. Albano, »IRT vs CTT - Introduction to Educational and Psychological Measurement Using R,« 23 Veljača 2020. [Mrežno]. Available: https://www.thetaminusb.com/intro-measurement-r/irt.html#comparing-with-irt. [Pokušaj pristupa 19 Lipanj 2024].

[4] T. Albano, »IRT models - Introduction to Educational and Psychological Measurement Using R,« 23 Veljača 2020. [Mrežno]. Available: https://www.thetaminusb.com/intro-measurement-r/irt.html#the-irt-models. [Pokušaj pristupa 19 Lipanj 2024].

[5] J. Wood, »Logistic IRT Models,« 12 Studeni 2017. [Mrežno]. Available: https://quantdev.ssri.psu.edu/sites/qdev/files/IRT_tutorial_FA17_2.html. [Pokušaj pristupa 23 Lipanj 2024].

[6] M.-I. Chang, »A Comparison of Two MCMC Algorithms for Estimating the 2PL IRT Models,« 8 Siječanj 2017. [Mrežno]. Available: http://opensiuc.lib.siu.edu/dissertations/1446. [Pokušaj pristupa 23 Lipanj 2024].

[7] J. Brzezińska, »Item response theory models in the measurement theory with the use of LTM package in R,« 19 Ožujak 2018. [Mrežno]. Available: https://www.dbc.wroc.pl/Content/40421/Brzezinska_Item_Response_Theory_Models_In_The_Measurement.pdf. [Pokušaj pristupa 23 Lipanj 2024].

[8] Z. Bobbitt, »Kuder-Richardson Formula 20 (Definition & Example),« 7 Siječanj 2022. [Mrežno]. Available: https://www.statology.org/kuder-richardson-20/. [Pokušaj pristupa 19 Lipanj 2024].

[9] C. Goforth, »Using and Interpreting Cronbach's Alpha,« 16 Studeni 2015. [Mrežno]. Available: https://library.virginia.edu/data/articles/using-and-interpreting-cronbachs-alpha. [Pokušaj pristupa 19 Lipanj 2024].

[10] K. Calkins, »More Correlation Coefficients,« 1 Kolovoz 2005. [Mrežno]. Available: https://www.andrews.edu/~calkins/math/edrm611/edrm13.htm#POINTB. [Pokušaj pristupa 11 Lipanj 2024].

[11] Više, »Discrimination Index Calculator,« 3 Listopad 2023. [Mrežno]. Available: https://calculator.academy/discrimination-index-calculator/. [Pokušaj pristupa 11 Lipanj 2024].

[12] T. Albano, »IRT - Introduction to Educational and Psychological Measurement Using R,« 23 Veljača 2020. [Mrežno]. Available: https://www.thetaminusb.com/intro-measurement-r/irt.html. [Pokušaj pristupa 10 Lipanj 2024].

Sažetak

Prilagodljive vježbe u sustavu za automatsko ocjenjivanje programskog kôda

Edgar

Rad daje pregled psihometrijskih metoda koje se koriste u izradi testova. Fokus rada je na teoriji odgovora na zadatke i njezinoj primjeni u razvoju sustava za provedbu prilagodljivih vježbi. Na temelju razrađene teorije opisuje modeliranje i implementaciju sustava.

Ključne riječi

IRT, Teorija odgovora na zadatke, Modul, Poslovi, TypeScript, NodeJS, Angular, D3.js, Servisi, Višekomponentna aplikacija, Web aplikacija

Summary

Adaptive exercises in Edgar automated programming assessment system

The thesis gives an overview of the psychometric methods used in test creation. The thesis focuses on the item response theory and its application in the development of an adaptive exercising system. Based on the theory, the thesis describes modelling and implementation of the system.

Keywords

IRT, Item response theory, Module, Jobs, TypeScript, NodeJS, Angular, D3.js, Daemons, Multi-component application, Web application

Privitak

Legenda tablica opisa relacija:

Ako se u stupcu UQ pojave vrijednosti oblika *(x)* gdje je x broj, to označava da je atribut opisan u retku dio ograničenja jedinstvenosti sa svim ostalim atributima s istom oznakom *(x)*.

Sav izvorni kod te popratni dijagrami, upute i dokumentacija dostupni su na *GitHub* repozitorijima:

https://github.com/lukacur/edgar-irt

https://github.com/lukacur/edgar-irt-webapp