Rješavanje zadataka iz predmeta Računalne mreže na računalu

Hrelja, Andrea

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:430153

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository

zir.nsk.hr

DIGITALNA KNJIŽNICA

Image not found or type unknown

Sveučilište u Rijeci – Odjel za informatiku

Jednopredmetni preddiplomski studij informatike

Andrea Hrelja

Rješavanje zadataka iz predmeta Računalne mreže na računalu

Završni rad

Mentor: v. pred. dr. sc. Vedran Miletić

Rijeka, 24. rujna 2020.

Sažetak

Kolegij "Računalne mreže" kao jednu od aktivnosti ima zadatke koji se mogu rješavati na papiru i na računalu. Ovaj rad opisuje rješavanje zadataka iz kolegija "Računalne mreže" koristeći računalo i programski jezik Python. Pored toga, govori se i o pripremi zadataka koji se rješavaju uz korištenje unaprijed zadanih ulaznih podataka i formula koje te podatke koriste. Na taj način pojednostavljuje se stvaranje ispitnih aktivnosti na sustavu za učenje Moodle pa specijalno i njegovoj inačici Merlin.

Ključne riječi

računalne mreže, Moodle, Merlin, Python, Django

Sadržaj

1. Uvod....1
2. Pretvaranje zadataka iz kolegija Računalne mreže u pitanja višestrukog izbora....2
   2.1. Pitanja višestrukog izbora....2
      2.1.1. Ograničenja kod pitanja višestrukog izbora....4
   2.2. Rješavanje zadataka i softverski alat rm1rm2kolokviji....7
      2.2.1. Daljnja obrada zadataka....10
3. Web aplikacija....13
   3.1. Struktura web aplikacije....14
   3.2. Dizajn web aplikacije....16
   3.3. Obrada zadataka i pretvorba u pitanja višestrukog izbora....19
      3.3.1. Validacija zadataka....20
      3.3.2. Transformacija zadataka....22
4. Korištenje u Merlinu....26
   4.1. Uvoz pitanja....26
   4.2. Uključivanje pitanja u Merlin aktivnosti....28
5. Zaključak....33

1. Uvod

Internet je neizostavni dio suvremenog umreženog svijeta gdje praktički ne postoji računalna infrastruktura koja ne koristi računalnu mrežu, bilo to za komunikaciju s jednog na drugi kraj sobe ili s jednog na drugi kraj svijeta.

Kako bi studenti dobili uvid u takve sustave, Odjel za informatiku ima u novom preddiplomskom studiju kolegij „Računalne mreže” koji daje pregled čitavog područja, odnosno u starom preddiplomskom studiju kolegije „Računalne mreže 1” i „Računalne mreže 2” koji zajedno tvore cjelinu. Prvi od dva kolegija izlaže arhitekturu mreža te strukturalne osobine računalnih i komunikacijskih sustava. Između ostalog, dobiva se prilika za usvajanjem osnovnih znanja o načinu rada računalnih mreža, na koji način su one međusobno povezane, kako komuniciraju, protokoli vezani uz takvu komunikaciju, a opisuje se i OSI model.

Naredni kolegij stavlja naglasak na zaštitu i sigurnost mreža. Kolegij objašnjava na koji način postići tajnost pojedinih sadržaja te kako ih zaštiti. Detaljnije se objašnjavaju preostali slojevi OSI modela (ponajviše transportni i aplikacijski), gdje se rješavaju i praktični zadatci.

Za zadatke koji se zadaju vrlo je naporno konstantno stvarati varijante i obrađivati te varijante ručno. Kako je proces stvaranja i obrade zadataka za zadatak određenog tipa uvijek isti, pretpostavka je da će računalo obaviti takav posao brže i točnije nego što će to čovjek koji ručno obrađuje. Ovim radom opisat ću proces kojim ćemo omogućiti računalu da obavlja proces zadavanja i rješavanja zadataka.

2. Pretvaranje zadataka iz kolegija Računalne mreže u pitanja višestrukog izbora

Zadatci koji se rješavaju u sklopu kolegija Računalne mreže daju uvid u stvarne metrike koje opisuju svojstva računalne mreže [4]. Takvi zadatci uključuju široki spektar izračuna i algoritama koje računala, kalkulatori, mrežni uređaji i ostali uređaji s računalnim predispozicijama po svojoj prirodi računaju vrlo efikasno i relativno precizno.

2.1. Pitanja višestrukog izbora

Prije nego su zadatci stvoreni, potrebno ih je standardizirati prema standardima koje podržava drugi sustav, a to će u ovom slučaju biti Moodle.

Kako bi standardi Moodle-a bili identificirani, potrebno je analizirati entitete koje želimo replicirati tako da se ustanovi način kojim će se njihovi standardi primijeniti na zadatke. Zadatci koji će kasnije biti stvoreni sastoje se od pitanja i odgovora. Sustav Moodle već ima definirane standarde za pitanja i odgovore koje je potrebno replicirati tako da stvoreni zadatci i odgovori budu identični Moodle pitanjima i odgovorima.

Za analizu entiteta „pitanje” i „odgovor” korišteno je Moodle korisničko sučelje i pitanja izvezena iz Moodle-a u XML formatu [5]. To je ujedno format koji će biti korišten za uvoz stvorenih zadataka u Moodle. Standardizirani entiteti „pitanje” i „odgovor” su u daljnjem tekstu označeni s Moodle Pitanje i Moodle Odgovor.

Tablica 1 prikazuje svojstva Moodle Pitanja i kratki opis svakog svojstva.

Naziv svojstva	Opis
Name	Naziv pitanja
Questiontext	Tekst pitanja
Defaultgrade	Bodovi za pitanje
Generalfeedback	Napomena
Idnumber	ID pitanja koji nije jedinstveni identifikator u Moodle bazi
Single	Pitanje ima jedan ili više točnih odgovora
Shuffleanswers	Miješanje odgovora kod ispisa
Answernumbering	Numeriranje odgovora
Penalty	Negativni bodovi kod novog pokušaja
Type	Tip pitanja
Correctfeedback	Napomena za točan odgovor
Partiallycorrectfeedback	Napomena za nepotpun odgovor
Incorrectfeedback	Napomena za netočan odgovor

Tablica 1: Svojstva Moodle Pitanja

Pitanje direktno ne sadrži informacije o Moodle Odgovorima, već su ta svojstva dostupna indirektno u entitetu Moodle Odgovor.

Naziv svojstva	Opis
Name	Naziv odgovora
Feedback	Napomena
Fractioncorrect	Postotak točnosti odgovora (od -100% do 100%)

Tablica 2: Svojstva Moodle Odgovora

Svako Moodle Pitanje u Moodle XML datoteci je reprezentirano kao XML element. Svaki Moodle Odgovor, točan ili netočan, je podelement Moodle Pitanja. Tako se prelaskom kroz svaki XML element Moodle Pitanje može dohvatiti njegove podelemente – Moodle Odgovore.

Uz navedene informacije, potrebno je definirati i „kolegij„ s „kategorijom” kojoj Moodle Pitanja i Moodle Odgovori pripadaju. Ti su entiteti označeni s Kolegij i Kategorija.

Naziv svojstva	Opis
Course name	Naziv kolegija
Category name	Naziv kategorije

Tablica 3: Svojstva Kolegija i Kategorije

Ovi entiteti su reprezentirani jednim XML elementom, a svojstva u tablicama su izvedena iz znakovnih nizova sadržanih unutar elementa.

2.1.1. Ograničenja kod pitanja višestrukog izbora

Nakon što su svojstva Moodle-ovih entiteta analizirana, potrebno je utvrditi eventualna ograničenja za svako svojstvo zajedno s njegovim tipom podatka. Tablica 4 prikazuje tip podatka u PostgreSQL notaciji i ograničenje za svako svojstvo Moodle Pitanja.

Naziv svojstva	Tip podatka	Ograničenje
Name	VARCHAR	Nema ograničenja
Questiontext	TEXT	Nema ograničenja
Defaultgrade	DOUBLE	Nema ograničenja
Generalfeedback	TEXT	Nema ograničenja
Idnumber	VARCHAR	Nema ograničenja
Single	BOOLEAN	Nema ograničenja
Shuffleanswers	BOOLEAN	Nema ograničenja
Answernumbering	VARCHAR	Može biti jedan od: • 'abc' • 'ABC' • '123' • 'iii' • 'III'
Penalty	DOUBLE	Može biti jedan od: • 0 • 100 • 50 • 33.33333 • 25 • 20 • 10
Type	VARCHAR	Može biti jedan od: • 'multichoice' • 'truefalse' • 'shortanswer' • 'matching' • 'cloze' • 'essay' • 'numerical' • 'description'
Correctfeedback	VARCHAR	Nema ograničenja
Partiallycorrectfeedback	VARCHAR	Nema ograničenja
Incorrectfeedback	VARCHAR	Nema ograničenja

Tablica 4: Ograničenja Moodle Pitanja

Tablica 5 prikazuje tip podatka u PostgreSQL notaciji i ograničenje za svako svojstvo Moodle Odgovora.

Naziv svojstva	Tip podatka	Ograničenje
Name	VARCHAR	Nema ograničenja
Feedback	TEXT	Nema ograničenja
Fractioncorrect	DOUBLE	Može biti jedan od: • 100 • 90 • 83.33333 • 80 • 75 • 70 • 66.66667 • 60 • 50 • 40 • 33.33333 • 30 • 25 • 20 • 16.66667 • 14.28571 • 12.5 • 11.11111 • 10 • 5 • None • -5 • -10 • -11.11111 • -12.5 • -14.28571 • -16.66667 • -20 • -25 • -30 • -33.33333 • -40 • -50 • -60 • -66.66667 • -70 • -75 • -80 • -83.33333 • -90 • -100

Tablica 5: Ograničenja Moodle Odgovora

Ovako definirana svojstva, njihove tipove podataka i ograničenja možemo nazvati Moodle shemom. Nadalje, potrebno je zadatke transformirati tako da podržavaju standarde Moodle sheme.

2.2. Rješavanje zadataka i softverski alat rm1rm2kolokviji

Kao početna točka stvaranja zadataka korišten je softverski alat rm1rm2kolokviji napravljen i podijeljen od strane mentora v. pred. dr. sc. Vedrana Miletića putem servisa Bitbucket u obliku repozitorija s Python [3] skriptama.

Alat je podijeljen u nekoliko modula koji međusobno komuniciraju kako bi zadane podatke transformirali u nove setove podataka. Podatci nad kojima alat vrši transformaciju su svojstva zadatka i nalaze se u zasebnom modulu alata.

Naziv svojstva	Opis
Tekst	Tekst zadatka s metrikama na temelju kojih se zadatak rješava
Varijable	Vrijednosti metrika
Rješenje	Funkcija koja uzima varijable i vraća točan rezultat
Odgovori	Tri identične funkcije koje uzimaju varijable i vraćaju netočan rezultat

Tablica 6: Svojstva nepotpuno opisanog zadatka

Navedena svojstva okvirno opisuju zadatak, a strukturirana su u obliku Python rječnika[1] (nadalje: rječnik). Ovaj modul tako sadrži više rječnika gdje svaki rječnik opisuje jedan zadatak.

1 Struktura u kojoj ključ podatka (najčešće znakovni niz) identificira podatak

```
{
    'tekst': """Domaćin H1 se spaja korištenjem TCP-a na drugi domaćin H2. Koliko TCP veza po sekundi
                može domaćin H1 napraviti prije nego što u TIME_WAIT stanju zauzme sva dostupna vrata?
                Pretpostavimo da su kratkoživuća vrata u rasponu {0}-{1} i da stanje TIME_WAIT traje {2} s.""",
    'varijable': [[46080, 48128, 49152], [57856, 58880, 60928], [60, 120, 240]],
    'rjesenje': lambda var0, var1, var2: si(round((var1 - var0) / var2, 2), 'se-01'),
    'odgovori': [
        lambda var0, var1, var2: si(round(var2 / (var1 - var0), 2), 'se-01'),
        lambda var0, var1, var2: si(round((var1 - var2) / var2, 2), 'se-01'),
        lambda var0, var1, var2: si(round((var0 - var2) / var2, 2), 'se-01')
    ]
},
```

Slika 1: Zadatak (preuzet iz [1, 2]) u obliku Python rječnika

Drugi će modul zatim evaluirati svaki zadatak tako što će nasumično odabrati jednu kombinaciju varijabli te ju provesti kroz funkcije spremljene u svojstvima „rjesenje" i „odgovori". Taj će modul stvoriti onoliko verzija pitanja koliko korisnik odabere.

Ako bismo željeli ručno izračunati rješenje ovog zadatka na način koji to drugi modul radi, morali bi odabrati jednu kombinaciju varijabli (npr. [46080, 58880, 240]) i uvrstiti te brojeve u formulu:

$$\frac{raspon_{max} - raspon_{min}}{\text{TIME_WAIT}}$$

Dobili bi izraz oblika

$$\frac{58880 - 46080}{240}$$

kojega možemo jednostavno izračunati uz pomoć računala ili kalkulatora. Ovaj izraz je iskazan funkcijom:

```
f = lambda var0, var1, var2: ((var1 - var0) / var2)
```

gdje je $\text{var0} = raspon_{max}$, $\text{var1} = raspon_{min}$, $\text{var2} = \text{TIME_WAIT}$. Funkcija f se sada može pozvati s $f(\text{var0}, \text{var1}, \text{var2})$. Ovakav zapis „lambda" funkcije je ekvivalentan sljedećem zapisu obične Python funkcije:

```
def f(var0, var1, var2):
    return ((var1 - var0) / var2)
```

Isto tako, varijable $\text{var0}, \text{var1}, \text{var2}$ su korištene za dinamičko punjenje tekst pitanja. Predviđena mjesta za varijable u „tekstu" zadatka označena su vitičastim zagradama {} koje sadrže redni broj

varijable u odabranoj kombinaciji, a varijable će u „tekst” poredati nativna Python funkcija *format*() [2].

Kada je riječ o trivijalnim zadatcima poput ovog na Slika 1, lambda funkcije koje rješavaju zadatak su poprilično jednostavne i nije potrebno koristiti dodatne funkcije da bi dobili traženi rezultat.

Ako je zadatak složeniji ili je potrebna dodatna transformacija podataka, korištene su funkcije poput si(arg1, arg2). Ova će funkcija kao prvi argument primiti numeričku vrijednost, a kao drugi argument mjernu jedinicu u obliku znakovnog niza. Povratna vrijednost te funkcije je znakovni niz koji reprezentira numeričku vrijednost i njezinu mjernu jedinicu. Na isti način će poslužiti funkcije koje rješavaju složenije probleme – one će dobiti vrijednosti nad kojima moraju raditi izračun, a povratna vrijednost će biti rezultat tog izračuna. Isti se postupak odvija za netočne odgovore. Razlika između točnog i netočnog odgovora je proces dolaska do rješenja, odnosno funkcija koja ih računa.

Primjer složenog zadatka je dan na slici 2.

```
{
    'tekst': """Pretpostavimo da se TCP koristi na vezi širine frekventnog pojasa {0} Gbit/s. Uz pretpostavku da TCP može koristiti kontinuirano cjelokupnu širinu frekventnog pojasa,
                koliko bi trebalo vremena da se sekventni broj počne brojati ispočetka? Prepostavimo da se polje vremenskog pečata duljine 32-bita inkrementira {1} puta
                za vrijeme za koje se sekventni broj počne brojati ispočetka. Koliko će trebati polju vremenskog pečata da se počne brojati ispočetka?""",
    'varijable': [[1, 2, 10], [800, 1000, 1200]],
    'rjesenje': lambda var0, var1: """Sekventni broj se počinje brojati ispočetka nakon {}.
                Vremenski pečat se počinje brojati ispočetka nakon {:.2e} s""".format(si(2**32 / convert.bits(var0, "Gbit", "B"), 's'), (2**32 * ((2**32 / convert.bits(var0, "Gbit", "B")) / var1))),
    'odgovori': [
        lambda var0, var1: """Sekventni broj se počinje brojati ispočetka nakon {}.
                Vremenski pečat se počinje brojati ispočetka nakon {:.2e} s""".format(si(convert.bits(var0, "Gbit", "B") / 2**32, 's'), (2**32 * ((convert.bits(var0, "Gbit", "B") / 2**32) / var1))),
        lambda var0, var1: """Sekventni broj se počinje brojati ispočetka nakon {}.
                Vremenski pečat se počinje brojati ispočetka nakon {:.2e} s""".format(si(2**30 / convert.bits(var0, "Gbit", "B"), 's'), (2**30 * ((2**30 / convert.bits(var0, "Gbit", "B")) / var1))),
        lambda var0, var1: """Sekventni broj se počinje brojati ispočetka nakon {}.
                Vremenski pečat se počinje brojati ispočetka nakon {:.2e} s""".format(si(convert.bits(var0, "Gbit", "B") / 2**30, 's'), (2**30 * ((convert.bits(var0, "Gbit", "B") / 2**30) / var1)))
    ]
},
```

Slika 2: Složeni zadatak u obliku Python rječnika

U ovom je slučaju korištena dodatna funkcija *bits*() iz modula *convert*. To je funkcija koja je ručno napravljena sa svrhom konverzije jedne mjerne jedinice u drugu. Funkcija prima tri argumenta, gdje je prvi argument numerička vrijednost, drugi argument mjerna jedinica u kojoj je numerička vrijednost izražena, a treći argument mjerna jedinica u koju želimo pretvoriti tu numeričku vrijednost. Povratna vrijednost funkcije je „float” decimalni broj. Tako će poziv *bits*(8,'bit','B') vratiti decimalni broj bajtova koji su sadržani u 8 bita, a to je broj 1.0.

Kada su vrijednosti odabrane iz jedne kombinacije varijabli konvertirane u traženu vrijednost, „lambda” funkcija će napraviti svoj izračun i vratiti rezultat tog izračuna. Može se primijetiti da u ovom slučaju vitičaste zagrade u tekstu i rješenju zadatka ne sadrže redne brojeve elemenata kombinacije varijabli. Isto tako, jedna od vitičastih zagrada u tekstu sadrži vrijednost ':.2e'.

2 https://docs.python.org/3.8/library/string.html#string.Formatter.format

U ovom slučaju vrijednost $':.2e'$ funkciji $format()$ napominje format u kojem će ona ispisati broj na rednom mjestu vitičaste zagrade koja sadrži tu napomenu. Razbijanjem te vrijednosti po njezinim dijelovima, može se lako interpretirati značenje svakog znaka. $:$ je oznaka za početak posebnog formatiranja, $.$ nalaže ispis decimalnog broja, 2 nalaže broj mjesta iza decimalne točke, a e nalaže ispis znanstvenog zapisa broja. Tako će poziv $\{\}$ ispisati $1.00e+02$, što je ekvivalentno zapisu $1.00 * 10^2$.

Nakon što su izračunati točni i netočni odgovori te je tekst pitanja dinamički popunjen s vrijednostima kombinacije varijabli, svako se pitanje ispisuje kao znakovni niz u Python interpreteru[3]. Takav znakovni niz potpuno opisuje zadatak, ali kao struktura nije prilagođena za obradu koja je potrebna. Potrebna je struktura koja omogućava lako dohvaćanje svojstava objekata kao što je rječnik.

2.2.1. Daljnja obrada zadataka

Alat rm1rm2kolokviji pri kraju stvaranja zadataka generira znakovni niz koristeći svojstva zadatka u obliku zasebnih Python varijabli[4]. U tom je koraku vrlo jednostavno svojstva spremljena u Python varijablama preusmjeriti u rječnik koji će potpuno opisati zadatak. Kako je svaki zadatak, tj. skup njegovih svojstava, zaseban rječnik, potrebno je uvesti strukturu Python liste[5] koja će sadržavati sve zadatke.

Kada su svi zadatci strukturirani, vrlo je jednostavno manipulirati njima. U ovom koraku promatra se prethodno definirana struktura Moodle Pitanja i Moodle Odgovora i uspoređuje se sa strukturom zadatka. Kako se zadatak još ne nalazi u bazi podataka i ne postoje ograničenja nad njima, jedina dodatna informacija koja je dostupna o svojstvima zadatka su tipovi podataka u Python notaciji.

Naziv svojstva	Tip podatka	Opis
Tekst	str	Tekst zadatka
Rješenje	str	Točno rješenje zadatka
Odgovori	list	Lista netočnih rješenja

Tablica 7: Svojstva potpuno opisanog zadatka

3 Program koji pokreće instrukcije (kod) pisane u programskom jeziku visoke razine poput Pythona, prevodi jezik visoke razine u jezik niže razine poput C/C++ koji se zatim kompilira

4 Spremište za neku proizvoljnu vrijednost. Python varijabla nije svojstvo „varijable” zadatka, već je to spremište proizvoljne vrijednosti

5 Struktura u kojoj redni broj podatka (indeks) opisuje podatak

Ovakvu strukturu zadataka je sada jednostavno promijeniti tako da podržava standarde Pitanja i Odgovora – potrebno je strukturu zadatak transformirati u entitete Zadatak i Odgovor koji će biti identični entitetima Moodle Pitanje i Moodle Odgovor.

Ograničenja postavljena kod Moodle Pitanja i Moodle Odgovora moraju biti ispoštovana kod entiteta Zadatak i Odgovor, što olakšava transformaciju zadatka u Zadatak i Odgovor. Kako su svi stvoreni zadatci osmišljeni kao pitanja višestrukog izbora, svojstvo „Type” Zadatka imati će vrijednost 'multiple'.

Slijedi odabir vrijednosti za svojstvo „Answernumbering”. Zbog održavanja konzistencije korišteno je isto numeriranje odgovora za svako pitanje. U ovom slučaju je izabrana vrijednost 'abc'.

Moodle Odgovor razlikuje točan od netočnog odgovora po svojstvu „Fractioncorrect”. Kada je odgovor točan, vrijednost „Fractioncorrect” je 100. Kada je odgovor netočan i netočno rješavanje studenta ne stvara negativne bodove, vrijednost „Fractioncorrect” je 'None', tj. 0. Kada je odgovor netočan i netočno rješavanje studenta stvara negativne bodove, može se odabrati jedna od vrijednosti u rasponu od -5 do -100. Vrijednosti u rasponu od -100 do 100 predstavljaju postotak boda za jedan Moodle Odgovor u odnosu svojstvo „Defaultgrade” Moodle Pitanja.

U slučaju Odgovora Zadatka, korištena je vrijednost 100 za točan odgovor i vrijednost 'None' za netočan odgovor, što znači da neće postojati negativni bodovi za Zadatak. Po strukturi Moodle sheme, svako pitanje ima posebne napomene za točne, nepotpuno točne i netočne odgovore („Correctfeedback”, „Partiallycorrectfeedback” i „Incorrectfeedback”). Kako ne postoji potreba za dokumentiranje ovih svojstava za svako posebno pitanje, oni se izdvajaju u poseban konfiguracijski rječnik. Tu se javlja potreba za još minimalno jednim, „nadrječnikom”, koji će obuhvaćati sve potrebne informacije o Zadacima, Odgovorima i „konfiguraciji” zadataka.

Da bi ovako opisani entiteti podržavali standarde Moodle sheme, potrebno je uvesti nove atribute „Course name” i „Category name”. Obzirom da je riječ o atributima koji jedinstveno predstavljaju skup pitanja koji se odjednom generira, dovoljno je te atribute uvesti u strukturu na razini iznad svih zadataka.

Kako bi daljnja obrada bila neovisna o alatu koji je stvorio podatke, oni se izvode u neki drugi format takav da ga drugi sustav može jednostavno uvesti. Jedan od više portabilnih formata je JSON[6], a Python nativno podržava uvoz/izvoz takvog formata uz pomoć modula „json”. Struktura JSON datoteke identična je kombinaciji struktura rječnika i liste u Pythonu, stoga je to idealan format za ovakav slučaj.

6 Čovjeku lako čitljiv (engl. *human readable*) format podataka (engl. akronim *JavaScript Object Notation*)

Slika 3 prikazuje strukturu Zadataka u ovom koraku.


```
{
    "kolegij": {
        "course_name": "RM1",
        "category": "1. kolokvij"
    },
    "zadaci_config": {
        "correctfeedback": "",
        "incorrectfeedback": "",
        "partiallycorrectfeedback": ""
    },
    "zadaci": [
        {
            "naziv": "Window size; Sequence number 1",
            "tekst": "Dizajnirate pouzdani protokol za protok podataka koji koristi klizni prozor (kao TCP). Taj protokol će raditi na mreži širine frekventnog pojasa 1 Gbit/s, RTT mreže je 100 ms, a MSL je 240.
            "type": "multichoice",
            "penalty": 0,
            "single": true,
            "rjesenje": "window size: 24 bit, sequence number: 35 bit",
            "odgovori": [
                "window size: 23 bit, sequence number: 34 bit",
                "window size: 35 bit, sequence number: 24 bit",
                "window size: 34 bit, sequence number: 23 bit"
            ],
            "defaultgrade": 3.0,
            "idnumber": "",
            "fraction_correct": 100,
            "fraction_incorrect": 0,
            "generalfeedback": "",
            "shuffleanswers": true,
            "answernumbering": "abc"
        },
        {
            "naziv": "Window size; Sequence number 2",
            "tekst": "Dizajnirate pouzdani protokol za protok podataka koji koristi klizni prozor (kao TCP). Taj protokol će raditi na mreži širine frekventnog pojasa 10 Gbit/s, RTT mreže je 40 ms, a MSL je 60.
            "type": "multichoice",
            "penalty": 0,
            "single": true,
            "rjesenje": "window size: 26 bit, sequence number: 37 bit",
            "odgovori": [
                "window size: 25 bit, sequence number: 36 bit",
                "window size: 37 bit, sequence number: 26 bit",
                "window size: 36 bit, sequence number: 25 bit"
            ],
            "defaultgrade": 3.0,
            "idnumber": "",
            "fraction_correct": 100,
            "fraction_incorrect": 0,
            "generalfeedback": "",
            "shuffleanswers": true,
            "answernumbering": "abc"
        },
```


Slika 3: Struktura Kolegija, Kategorije, Zadatka i Odgovora u JSON formatu

Cilj svake daljnje obrade je izvoz pitanja u Moodle XML formatu zbog jednostavnog uvoza u implementaciju sustava za e-učenje Moodle nazvanu Merlin.

3. Web aplikacija

Pretvaranje zadataka u Zadatke potpuno je omogućeno alatom rm1rm2kolokviji. Neposredno nakon takve transformacije, Zadatci su izvezeni u JSON format i tada su dostupni za korištenje s bilo kojim sustavom koji će biti prilagođen za daljnju transformaciju zadane strukture podataka. Postoji mnogo sustava koji mogu obaviti željene transformacije, a u sklopu ovog rada stvorena je web aplikacija koristeći okvir Django [6].

Django je web okvir otvorenog koda pisan u Pythonu koji programeru olakšava probleme koji se javljaju pri izradi web aplikacija. Neki problemi kod izrade web aplikacija su CRUD[7] operacije nad tablicama (entitetima) u bazi podataka. Ideja ovog okvira je da se procesi koji rješavaju taj problem uvijek izvode na isti način. Kod unosa (engl. *Create*) se ispisuju HTML elementi koji od korisnika prikupljaju podatke o kolumnama tablice (svojstvima entiteta), a kod izmjene (engl. *Update*) se ispisuju isti takvi HTML elementi čije su vrijednosti popunjene vrijednostima kolumne u tablici. Postoje dva načina ispisa (engl. *Read*) – prvi način je ispis liste koji ispisuje kolumne više redaka iz tablice u obliku zasebnog retka (instance entiteta), dok drugi način ispisuje kolumne samo jednog retka. Kada je riječ o brisanju (engl. *Delete*), ispisuje se HTML obrazac koji od korisnika traži potvrdu brisanja određenog retka pa se zatim taj redak može i ne mora izbrisati iz tablice u bazi.

Django u tu svrhu koristi generičke klase (CreateView, UpdateView, ListView, DetailView, DeleteView), a postavljanje CRUD operacija nad tablicom je s ovim okvirom vrlo jednostavno i brzo.

Django je vrlo robustan, skalabilan sustav visoke razine sigurnosti, koji je zbog svojih karakteristika ubrzao proces transformacije podataka korišten za izradu ovog rada.

Zadaća aplikacije je da obradi podatke o Kolegijima, Kategorijama, Zadatcima i Odgovorima te da obrađene podatke prikaže korisniku prije nego ih on odluči spremiti u bazu. Nakon što ih korisnik spremi u bazu, omogućen je izvoz tih podataka u JSON ili Moodle XML formatu.

7 engl. akronim za *Create*, *Read*, *Update*, *Delete* operacije nad podatcima

3.1. Struktura web aplikacije

Stvaranje novog Django projekta zahtjeva da je taj paket instaliran na računalu koje će ga pokretati. Takva instalacija se izvršava vrlo jednostavno, naredbom *pip install django* . Nakon što je paket uspješno instaliran, potrebno je stvoriti projekt naredbom *django−admin startproject <naziv−projekta>* . Ova naredba kreira direktorij s nazivom <naziv-projekta> datotekom *manage.py* koja je zapravo alias naredbe *django−admin* te pod direktorijem s konfiguracijskim datotekama. Slika 4 prikazuje stablo direktorija novog Django projekta.

```
ahrelja@AHRELJA-PC:/mnt/c/Users/AndreaHrelja/Documents/Django/naziv_projekta$ tree
.
├── manage.py
└── naziv_projekta
    ├── __init__.py
    ├── asgi.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

1 directory, 6 files
```

Slika 4: Stablo direktorija Django projekta

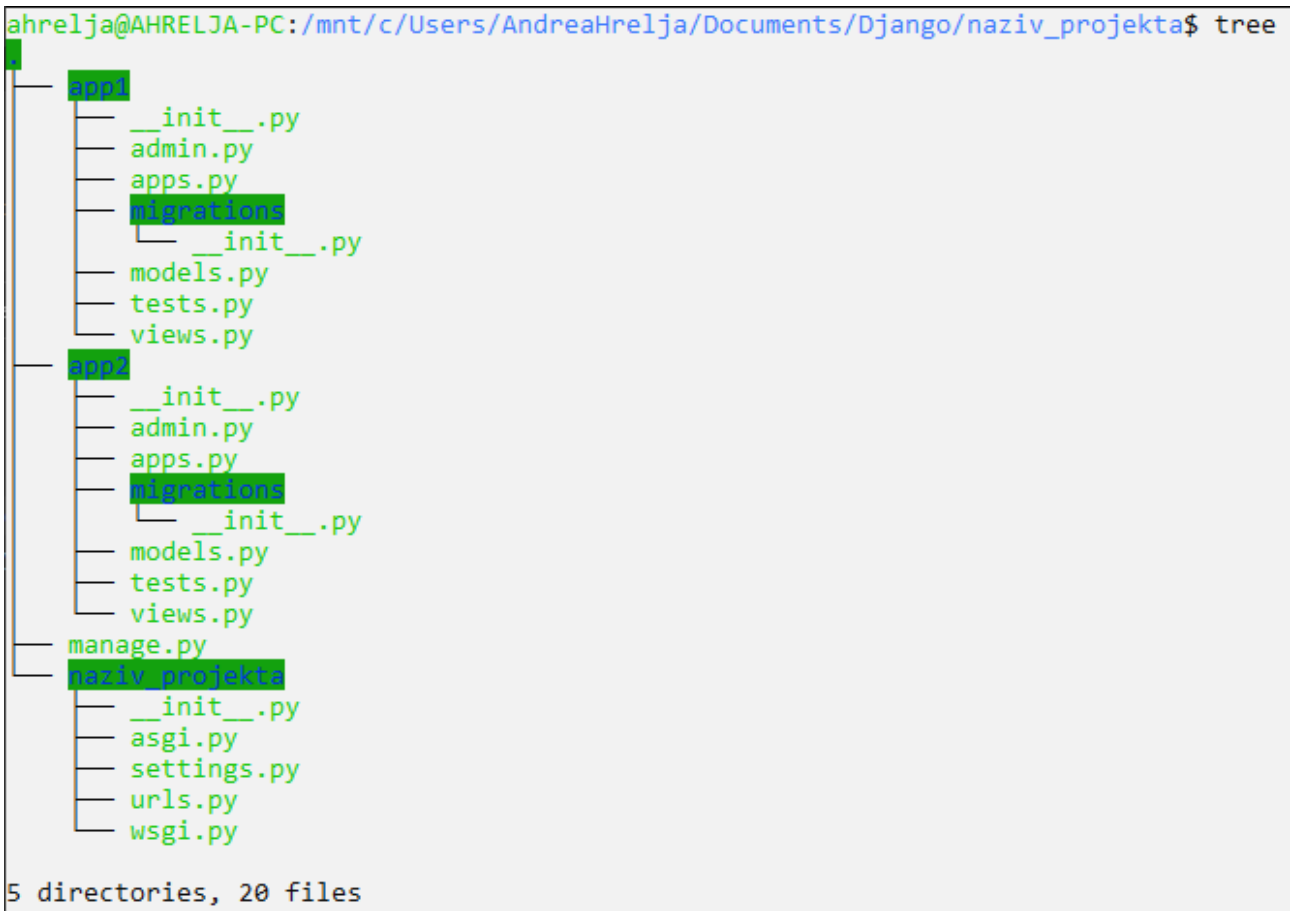
Trenutno nije moguće dodavati CRUD operacije nad nekom tablicom zato što konekcija na bazu ne postoji, kao ni Django model tablice koji se kasnije spominje. Za konfiguraciju konekcije na bazu, potrebno je urediti datoteku *settings.py* tako da su podatci o bazi dostupni unutar projekta. Moduli koji se povezuju na bazu su skriveni od programera, ali su dostupni u direktoriju na operativnom sustavu koji sadrži instalirane Python pakete.

```
# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'rm1rm2kolokviji',
        'USER': 'postgres',
        'PASSWORD': 'user'
    }
}
```

Slika 5: Primjer konfiguracije s informacijama o konekciji na bazu

Konfiguracija konekcije na bazu nije neophodna za daljnji razvoj, ali u ovom slučaju će konekcija biti potrebna. Kako Django koristi arhitekturu mikro servisa, ova će se web aplikacija sastojati od više manjih servisa (aplikacija), gdje će svaki servis obavljati svoju zadaću. Naredba *python* manage.py *startapp* <naziv-servisa> stvara takav servis (aplikaciju). Nakon što su servisi koji će tvoriti web aplikaciju stvoreni, stablo direktorija projekta se povećava.



Slika 6: Stablo direktorija Django projekta sa stvorenim servisima

Svaki servis se sastoji od datoteka koje će se koristiti za stvaranje modela i manipulaciju njima, i direktorija migrations/ koji će sadržavati informacije o promjenama modela i reflektiranju tih promjena u bazi.

3.2. Dizajn web aplikacije

Kao reprezentacija tablice u bazi, stvaraju se Django modeli. Svaki servis sadrži datoteku *models.py* koja će spremati meta podatke o tablicama. Projekt u sklopu ovog rada sadrži modele u servisu import_json .

Da bi izvoz Zadataka i Odgovora sadržavao sve informacije potrebne sustavu Merlin za kategorizaciju Moodle Pitanja, potrebno je uključiti i informacije o kolegiju te o kategoriji kolegija za koju su Zadatci stvoreni. To znači da će replika Moodle baze sadržavati entitete (modele) Kolegij i Kategorija.

```python
class MoodleZadatak(models.Model):

    name                = models.CharField("Naziv pitanja", max_length=255)
    questiontext        = models.TextField("Tekst pitanja")
    defaultgrade        = models.FloatField("Ocjena za pitanje")
    generalfeedback     = models.TextField("Napomena", null=True, blank=True,
                                           help_text="Napomena o pitanju namijenjena studentu nakon prikaza ocjene")
    idnumber            = models.CharField("Moodle ID", null=True, blank=True,
                                           help_text="Moodleova oznaka za ID pitanja", max_length=50)
    single              = models.BooleanField("Jedan ili više točnih odgovora", choices=SINGLE_CHOICES, default=1)
    shuffleanswers      = models.BooleanField("Miješanje poretka odgovora", default=1)
    answernumbering     = models.CharField("Oznake odgovora", default=ANSWERNUMBERING_CHOICES[0][0], choices=ANSWERNUMBERING_CHOICES,
                                           help_text="Oznaka za ispis odgovora u sustavu Moodle", max_length=3)
    penalty             = models.FloatField("Penal za netočni pokušaj", default=PENALTY_CHOICES[0][0], choices=PENALTY_CHOICES)

    type                = models.IntegerField("Tip Moodle zadataka", default=TYPE_CHOICES[0][0], choices=TYPE_CHOICES)
    moodle_kategorija   = models.ForeignKey("import_json.MoodleKategorija", verbose_name="moodle_kategorija", on_delete=models.CASCADE)

    correctfeedback     = models.CharField("Napomena - Točan odgovor", null=True, blank=True, max_length=50)
    incorrectfeedback   = models.CharField("Napomena - Netočan odgovor", null=True, blank=True, max_length=50)
    partiallycorrectfeedback = models.CharField("Napomena - Nepotpun odgovor", null=True, blank=True, max_length=50)
```

Slika 7: Django model - Pitanje

```python
class MoodleOdgovor(models.Model):

    name    = models.CharField("Odgovor", max_length=255)
    correct = models.BooleanField("Točan odgovor", default=False)

    feedback            = models.TextField("Napomena", null=True, blank=True,
                                           help_text="Napomena o individualnom odgovoru namijenjena studentu nakon prikaza ocjene")
    fraction_correct    = models.FloatField("Udio točnog odgovora u bodovanju", choices=FRACTION_CHOICES)

    moodle_zadatak      = models.ForeignKey("import_json.MoodleZadatak", verbose_name="moodle_zadatak", on_delete=models.CASCADE)


class MoodleKolegij(models.Model):

    course_name = models.CharField("Naziv kolegija", choices=COURSE_CHOICES, help_text="Naziv kolegija u sustavu Moodle", max_length=3)


class MoodleKategorija(models.Model):

    category_name  = models.CharField("Naziv kategorije", max_length=64)
    moodle_kolegij = models.ForeignKey("import_json.MoodleKolegij", verbose_name="moodle_kolegij", on_delete=models.CASCADE)
```

Slika 8: Django modeli - Odgovor, Kolegij, Kategorija

Na slici 7 prikazan je vanjski ključ modela *MoodleZadatak* (Pitanje) kao atribut moodle_kategorija. Na slici 8 prikazani su vanjski ključ modela *MoodleOdgovor* (Odgovor) kao atribut moodle_zadatak te vanjski ključ modela *MoodleKategorija* (Odgovor) kao atribut moodle_kolegij. Ti su ključevi objekti klase $models.ForeignKey()$ koja je u bazi reprezentirana kao kolumna tipa podatka INTEGER i ograničenjem FOREIGN_KEY na vanjsku tablicu.

Korištene su i druge klase za reprezentaciju SQL kolumni opisane u Tablica 8.

Naziv klase	SQL reprezentacija
CharField	VARCHAR
TextField	TEXT
FloatField	DOUBLE
IntegerField	INTEGER
BooleanField	BOOLEAN

Tablica 8: Django tipovi podataka prevedeni u PostgreSQL tipove podataka

Zbog ograničenja postavljenih na Moodle Pitanje i Moodle Odgovor, potrebno je ista ta ograničenja primijeniti na Pitanja i Odgovore u replici baze. Django to olakšava tako što dozvoljava unos samo predefiniranih vrijednosti za određena svojstva. Takva se ograničenja definiraju uz model u datoteci *models.py* .

```
TYPE_CHOICES = (
    (0, 'multichoice'),
    (1, 'truefalse'),
    (2, 'shortanswer'),
    (3, 'matching'),
    (4, 'cloze'),
    (5, 'essay'),
    (6, 'numerical'),
    (7, 'description')
)
```

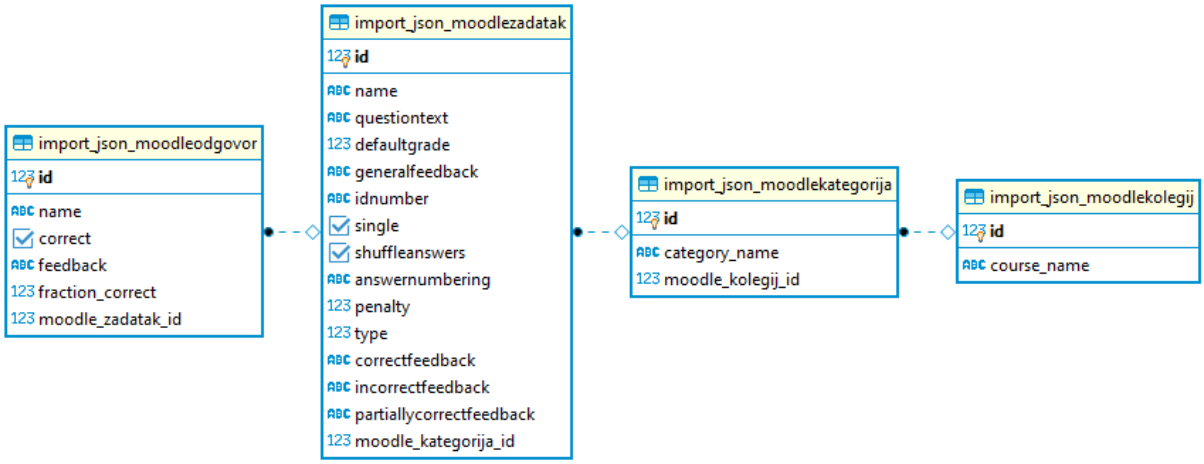
Slika 9: Ograničenja za "Type"

Ova se ograničenja konstruiraju unutar klase za reprezentaciju SQL kolumni koristeći TYPE_CHOICES kao vrijednost argumenta choices pri inicijalizaciji klase, odnosno deklaracije modela.

Migriranje modela u bazu postiže se naredbama dvjema naredbama. Naredba *python* manage.py *makemigrations* stvara SQL naredbe te informacije o promjenama modela i reflektiranju tih promjena na bazi. Naredba *python* manage.py *migrate* zatim izvršava sav SQL koji je prethodna naredba generirala.

Prije obrade podataka, stvorena je replika Moodle baze koja se sastoji od modela potrebnih za uvoz Zadataka i Odgovora u web aplikaciju te izvoz Zadataka i Odgovora u Moodle XML i konačno u sustav Merlin.

Slika 10 prikazuje EV dijagram[8] generiran koristeći DBeaver[9],



Slika 10: EV dijagram Moodle sheme

Može se primijetiti kako pri kreiranju SQL tablica, Django modelu dodaje prefiks s imenom servisa i pretvara bilo koje veliko slovo u malo slovo. Veze između tablica su prikazane isprekidanim crtama gdje ispunjena crna točka predstavlja brojnost „Many”, a romb bez ispune predstavlja brojnost „1”.

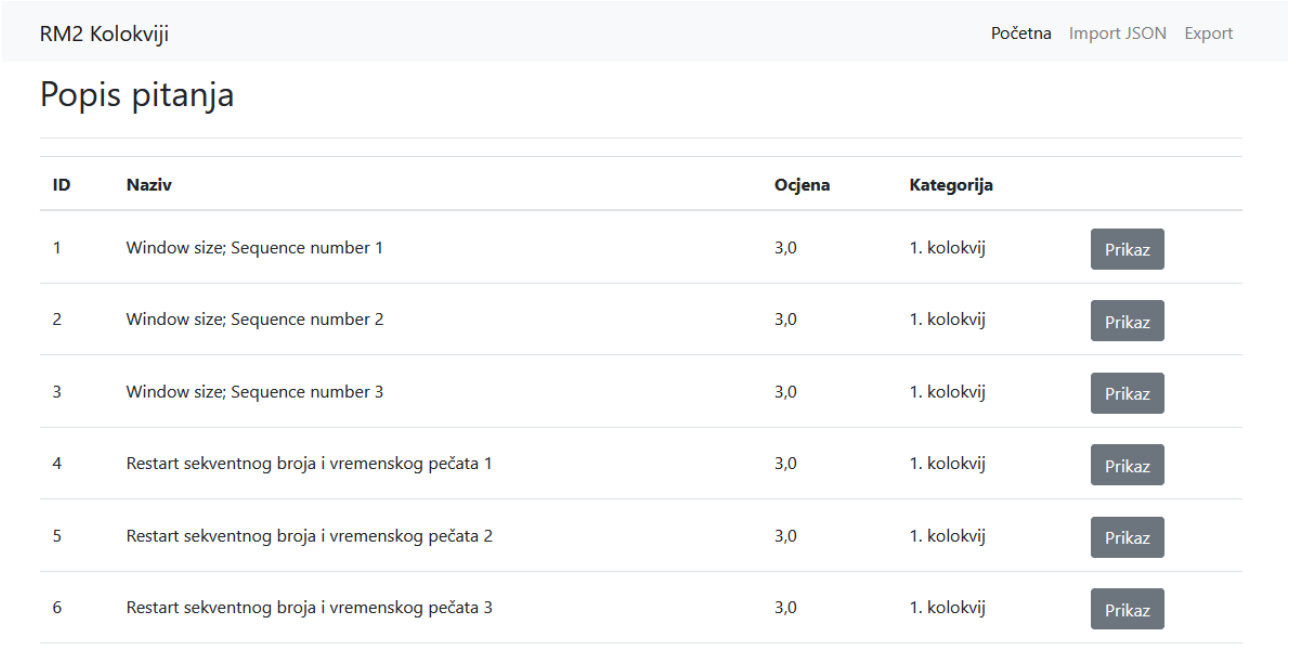
Sada kad je baza s modelima stvorena, aplikacija je spremna za obradu podataka.

3.3. Obrada zadataka i pretvorba u pitanja višestrukog izbora

Kako bi aplikacija sigurno prikupila i spremila sve podatke o Zadatcima, potrebno je te podatke validirati te dovesti u format koji poštuje ograničenja zadana u *models.py* .

Da bi se podaci mogli obraditi, potrebno ih je uvesti u aplikaciju putem korisničkog sučelja odabirom gumba „Import JSON.

8 Dijagram entiteta i veza
9 DBMS (engl. *Database Managment System*), alat otvorenog koda za rukovođenje baza podataka



Slika 11: Korisničko sučelje aplikacije

U kontekstu ove web aplikacije, validacija će obuhvaćati provjere tipova podataka za svojstva Zadataka te provjeru o dopuštenim vrijednostima za svojstva kod kojih postoji neko ograničenje. Za provjeru dopuštenih vrijednosti kod svojstava Zadataka koristit će se zadana ograničenja, dok će validacija tipova podatka koristiti novu, praznu strukturu koja je replika strukture Zadatka te sadrži mapu svojstava s tipovima podataka.

3.3.1. Validacija zadataka

Aplikacija validira Zadatke tako što iterira kroz sve ključeve nadrječnika ('kolegij', 'zadaci_config') i rječnika sadržanih u listi pod ključem 'zadaci'. Zatim provjerava tipove vrijednosti Zadataka tako što ih uspoređuje s tipovima definiranih u mapi svojstva i tipa podatka prikazanih na slici 12. Ako postoji svojstvo Zadatka čiji se tip podatka ne podudara s onime u mapi, aplikacija će javiti grešku i neće dozvoliti daljnju obradu podataka.

```
VALID_TYPES = {
    'kolegij': {
        'course_name' : str,
        'category'    : str
    },
    'zadaci_config': {
        'correctfeedback'   : str,
        'incorrectfeedback' : str,
        'partiallycorrectfeedback': str
    },
    'zadaci': {
        'naziv'     : str,
        'tekst'     : str,
        'rjesenje'  : str,
        'odgovori'  : list,
        'type'      : str,
        'single'    : bool,
        'penalty'   : float,
        'idnumber'  : str,
        'defaultgrade'      : float,
        'fraction_correct'  : float,
        'fraction_incorrect': float,
        'generalfeedback'   : str,
        'shuffleanswers'    : bool,
        'answernumbering'   : str
    }
}
```

Slika 12: Mapa svojstva i tipa podatka

Kada su svi tipovi podataka validirani, aplikacija će te podatke dovesti u format koji poštuje zadana ograničenja. Slično mapi svojstva i tipa podatka, stvoren je rječnik koji sadrži samo dozvoljene vrijednosti za svojstva koja su ograničena.

```
VALID_DATA = {
    'zadaci': {
        'type'      : [tup[1] for tup in choices.TYPE_CHOICES],
        'single'    : [tup[0] for tup in choices.SINGLE_CHOICES],
        'penalty'   : [float(tup[1]) for tup in choices.PENALTY_CHOICES],
        'fraction_correct'  : [float(tup[1]) if tup[1] != 'None' else 0 for tup in choices.FRACTION_CHOICES],
        'fraction_incorrect': [float(tup[1]) if tup[1] != 'None' else 0 for tup in choices.FRACTION_CHOICES],
        'answernumbering'   : [tup[0] for tup in choices.ANSWERNUMBERING_CHOICES]
    }
}
```

Slika 13: Dopuštene vrijednosti za ograničena svojstva

Ova se provjera izvodi iteracijom kroz svaki Zadatak i uspoređivanjem vrijednosti svojstva s dopuštenim vrijednostima. Ako vrijednost svojstva nije prisutna u zadanim granicama, aplikacija će javiti grešku i neće dozvoliti daljnju obradu podataka.

Kada su sve vrijednosti svojstva unutar zadanih granica, aplikacija će te podatke transformirati tako da podržavaju standarde definirane u *models.py*.

3.3.2. Transformacija zadataka

Ograničenja zadana u *models.py* nalažu da vrijednost svojstva mora biti sadržano u strukturi koja postavlja ta ograničenja, poput one na slici 9. Kada je to slučaj, Django će uzeti vrijednost u „tuple”-u[10] koja opisuje ograničenje koristeći „lambda” funkciju get_choice. Primjerice, kada je riječ o tipu Zadatka 'multiple', get_choice će kao „Type” uzeti vrijednost '0' jer ona opisuje granicu 'multiple' u strukturi TYPE_CHOICES.

```
get_choice = lambda choice, value: [tup[0] for tup in choice if tup[1] == value][0]

def transform_dictionary(dictionary):
    dictionary['kolegij']['category_name'] = dictionary['kolegij'].pop('category')
    for i, zadatak in enumerate(dictionary['zadaci']):
        dictionary['zadaci'][i]['name'] = dictionary['zadaci'][i].pop('naziv')
        dictionary['zadaci'][i]['questiontext'] = dictionary['zadaci'][i].pop('tekst')
        dictionary['zadaci'][i]['type'] = get_choice(choices.TYPE_CHOICES, dictionary['zadaci'][i]['type'])
        dictionary['zadaci'][i]['single'] = int(dictionary['zadaci'][i]['single'])
        dictionary['zadaci'][i]['shuffleanswers'] = int(dictionary['zadaci'][i]['shuffleanswers'])
        dictionary['zadaci'][i]['fraction_correct'] = get_choice(choices.FRACTION_CHOICES, str(dictionary['zadaci'][i]['fraction_correct']))
        dictionary['zadaci'][i]['fraction_incorrect'] = get_choice(choices.FRACTION_CHOICES, str(dictionary['zadaci'][i]['fraction_incorrect']).replace("0", "None"))
    return assign_ids(dictionary)
```

Slika 14: Transformacija Zadataka

Time su Moodle sheme ograničenja potpuno ispoštovane.

Nadalje, prije nego se obrađeni podatci prikažu korisniku za potvrdu o spremanju u bazu, svakom se Zadatku stvara vrijednost za svojstvo „Idnumber”. To će svojstvo biti korišteno u ispisu podataka za korisnika sa svrhom povezivanja Zadatka i Odgovora.

10 Nepromjenjiva Python struktura slična listi

```python
def assign_ids(dictionary):
    zadatak = MoodleZadatak.objects.last()

    try:
        min_id = zadatak.id
    except AttributeError:
        min_id = 0

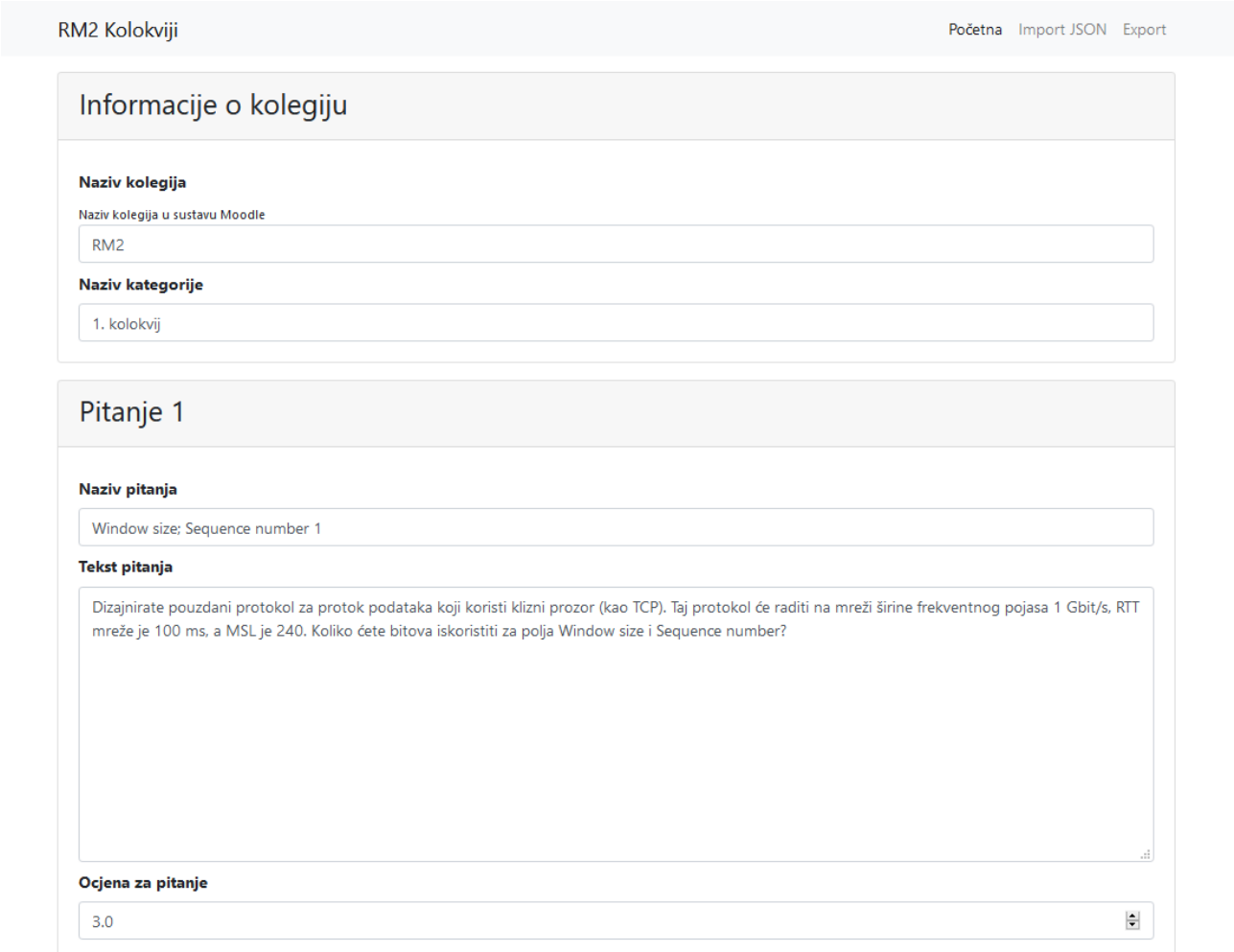
    for i, zadatak in enumerate(dictionary['zadaci']):
        dictionary['zadaci'][i]['idnumber'] = str(min_id + i + 1)
    return dictionary
```

Slika 15: Stvaranje svojstva "Idnumber" za svaki Zadatak

Funkcija assign_ids(dictionary) će pokušati dohvatiti najveći „Idnumber” za Zadatak iz baze. Ako ne postoji zapis o Zadatcima u bazi, za najveći „Idnumber” uzet će se vrijednost '0'.

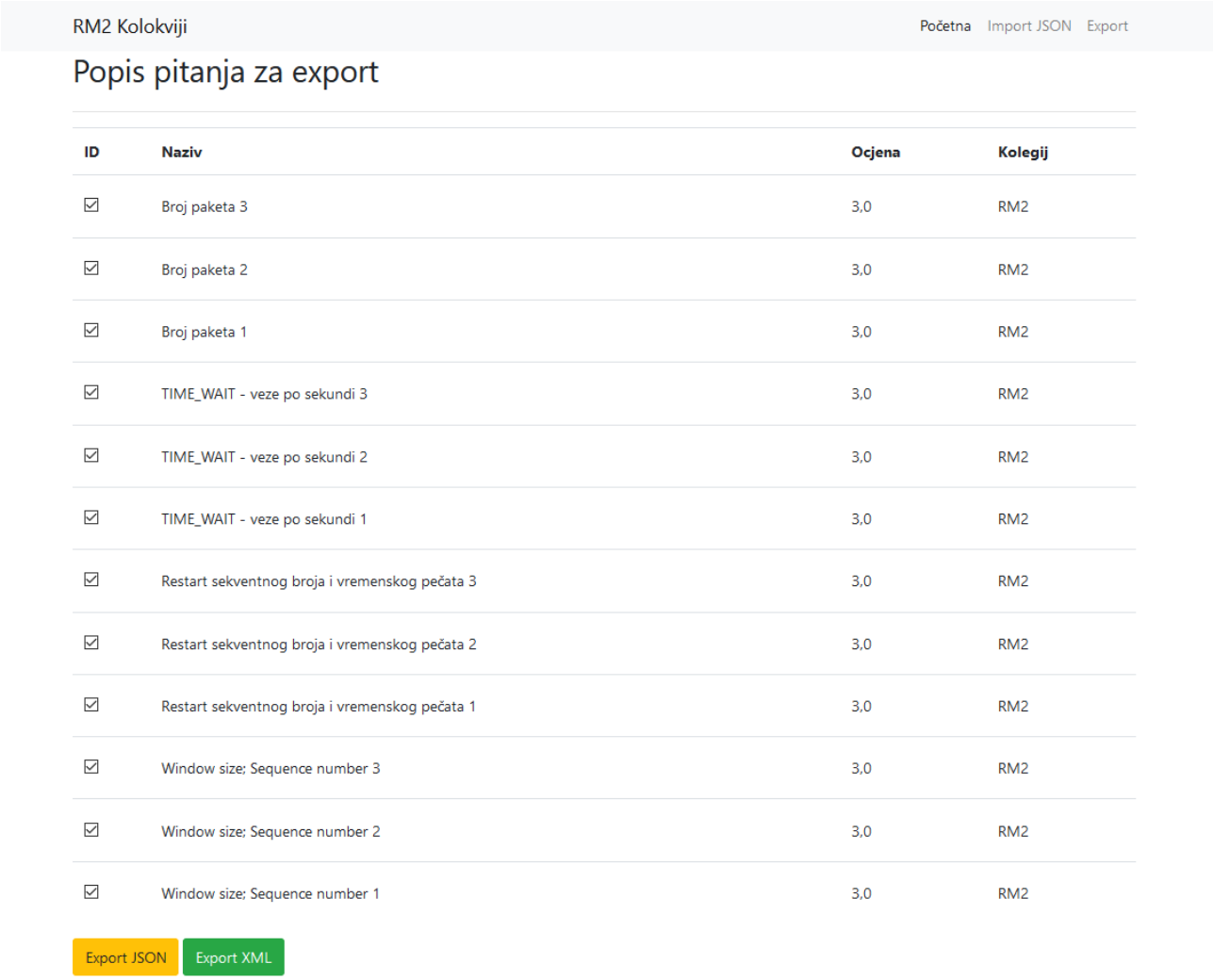
U ovom koraku se Moodle Pitanje može poistovjetiti s Zadatkom, a Moodle Odgovor sa Odgovorom. Nadalje, koristit će se termini Pitanje i Odgovor.

Kada su podatci transformirani, prikazuju se korisniku i omogućava se pojedinačna izmjena Kolegija, Kategorije, Zadataka i Odgovora. U ovom koraku korisnik može i ne mora izmijeniti ispisane podatke. Oni još nisu spremljeni već su učitani iz uvezene JSON datoteke.



Slika 16: Izmjena uvezenih podataka

Kada je izmjena podataka završena, aplikacija preusmjerava korisnika na početnu stranicu gdje je omogućena opcija za izvoz klikom na gumb „Export”.



Slika 17: Izvoz podataka

U ovom je koraku omogućen izvoz podataka u JSON ili XML formatu. XML format podržava Moodle XML standarde i spreman je za uvoz u Moodle sustav. Time je izvoz u Moodle XML uspješno obavljen.

4. Korištenje u Merlinu

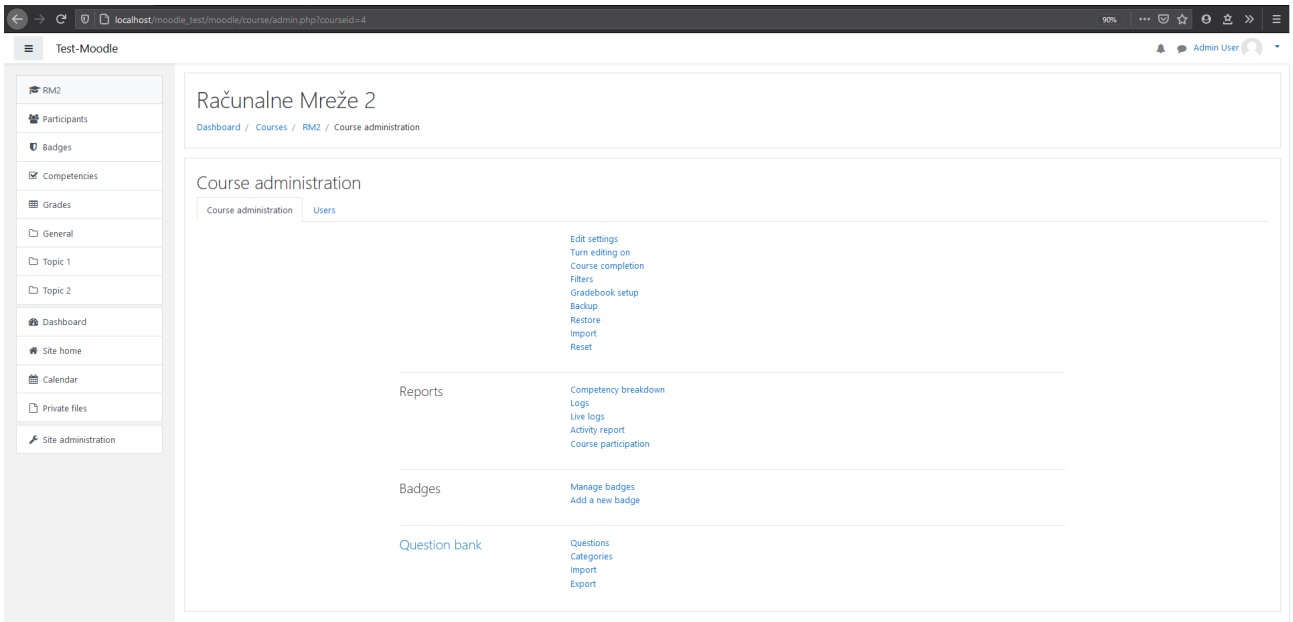
Moodle XML datoteke se koriste za uvoz jednog ili više Pitanja u sustav Moodle, ili njegove podsustave poput Merlina. Za testiranje u sklopu ovog rada korišten je Moodle 3.8.

4.1. Uvoz pitanja

Uvoz pitanja za željeni kolegij moguće je obaviti navigacijom do administracije kolegija.

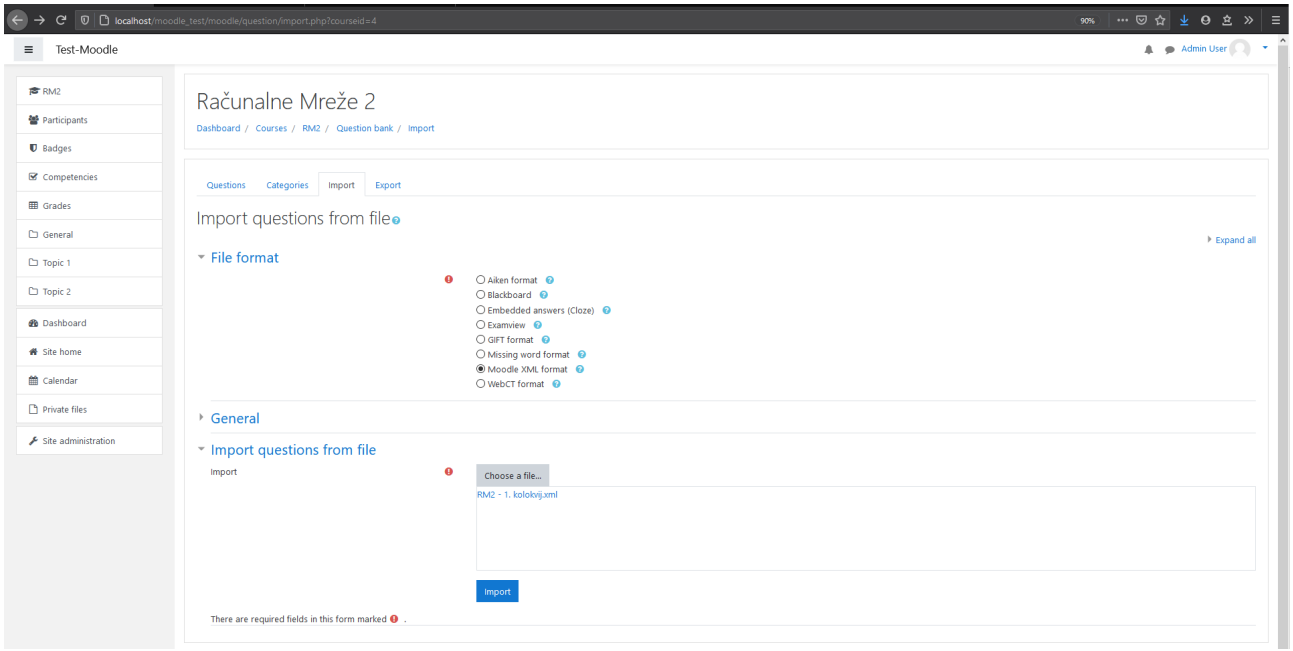


Slika 18: Navigacija prema Administraciji kolegija



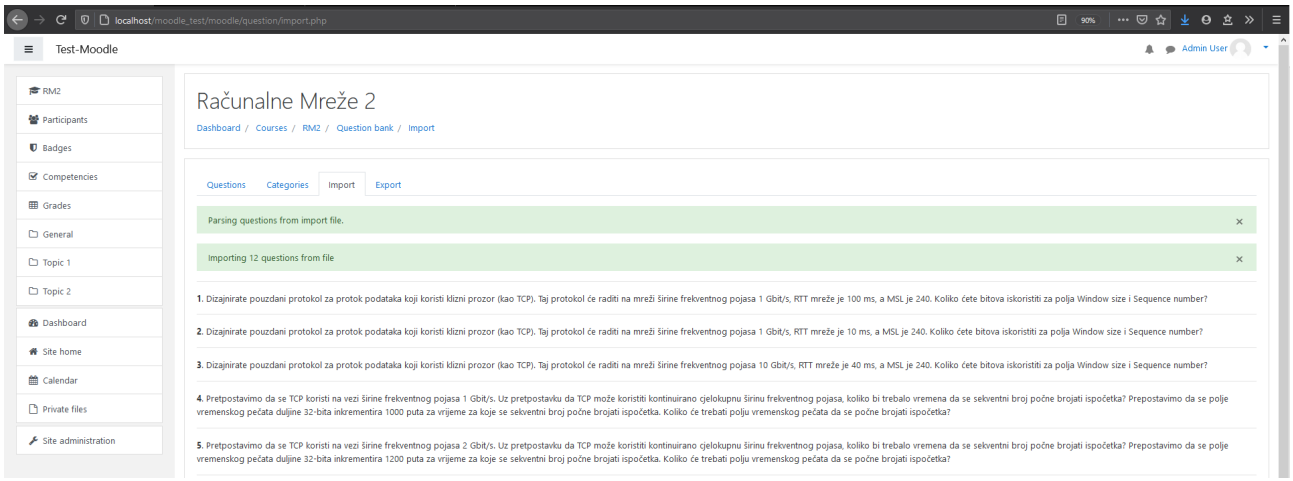
Slika 19: Administracija kolegija

U Administraciji kolegija potrebno je odabrati opciju „*Import*” kod sekcije „*Question bank*”. Klikom na opciju „*Import*” otvara se prozor koji od korisnika traži *upload* datoteke. Potrebno je označiti format datoteke, koji će u ovom slučaju biti „*Moodle XML format*” te odabrati datoteku za uvoz.



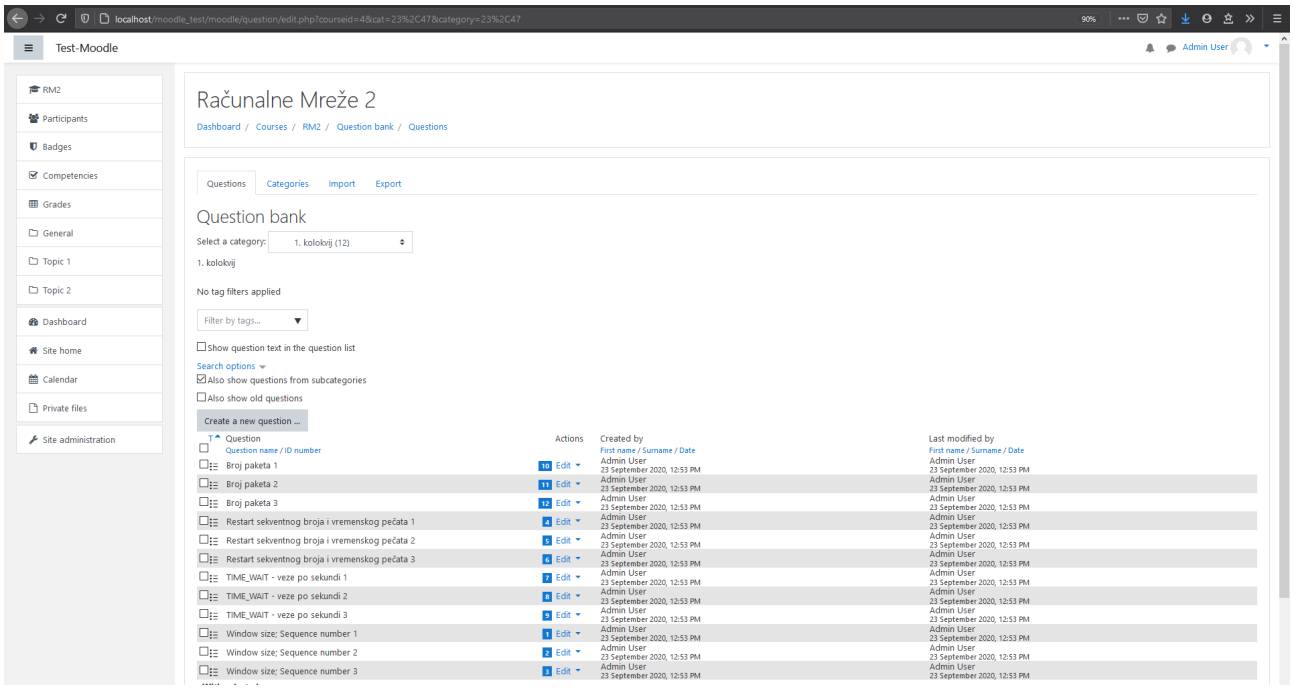
Slika 20: Uvoz Moodle XML datoteke

Nakon odabira formata datoteke i *uploada* datoteke, klikom na dugme „*Import*”, sustav će parsirati XML datoteku i ispisati parsirana pitanja.



Slika 21: Ispis parsiranih pitanja

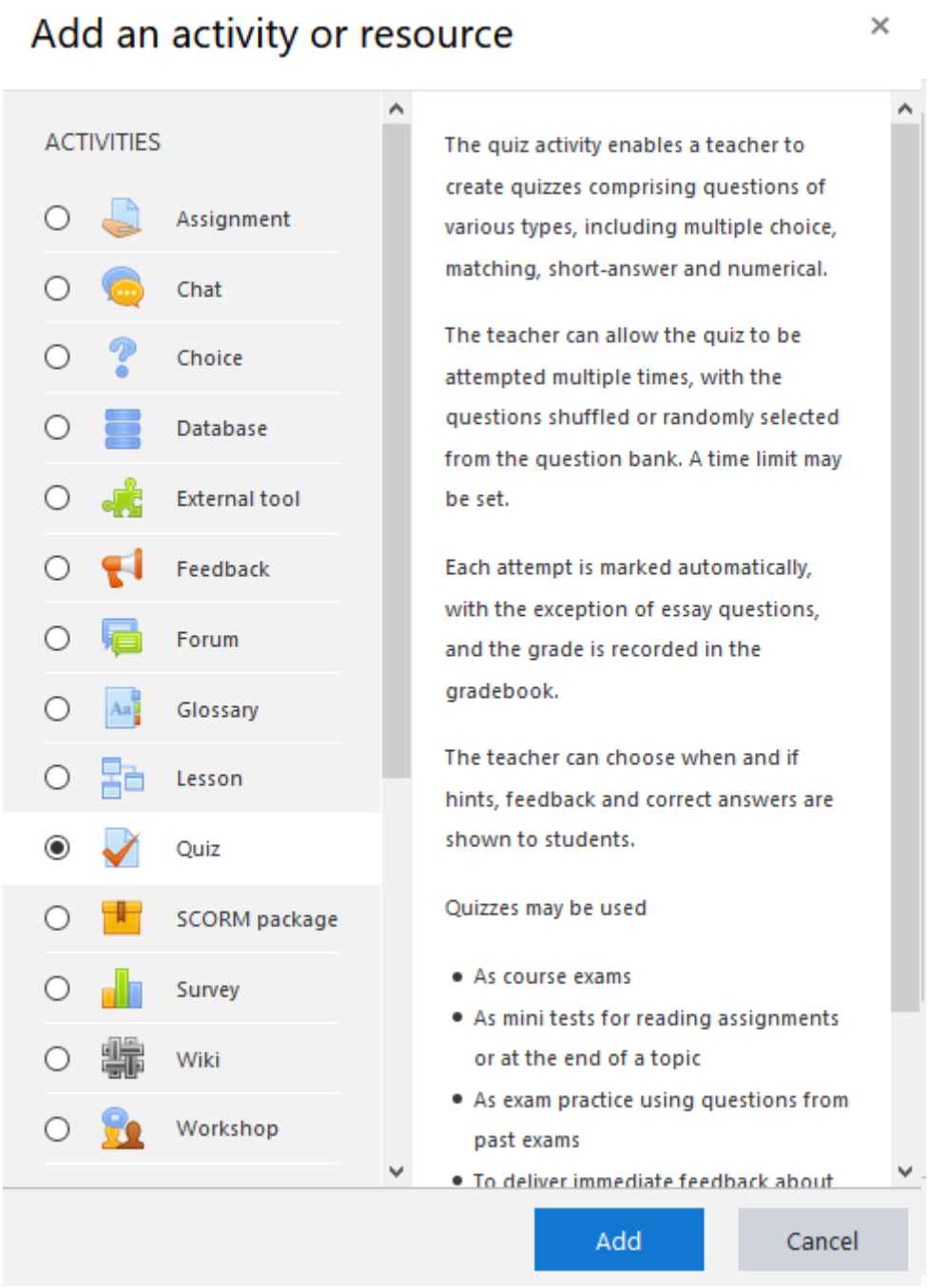
Sada je proces uvoza pitanja u potpunosti obavljen, a moguće ih je vidjeti ispod taba „*Questions*” unutar sekcije „*Question bank*”.



Slika 22: Ispis pitanja u sekciji "Question bank"

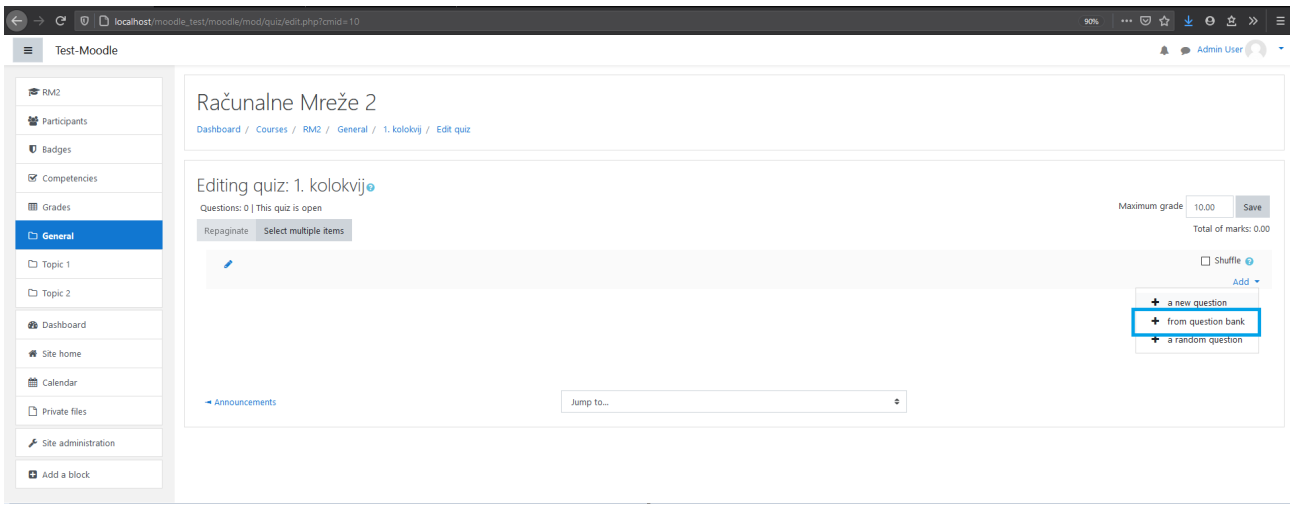
4.2. Uključivanje pitanja u Merlin aktivnosti

Kada su pitanja dostupna unutar „*Question bank*” kolegija, uključivanje u aktivnosti poput „*Quiz*”-a je poprilično jednostavno. Uključivanjem opcije uređivanja kod kolegija omogućuje se opcija stvaranja nove aktivnosti.



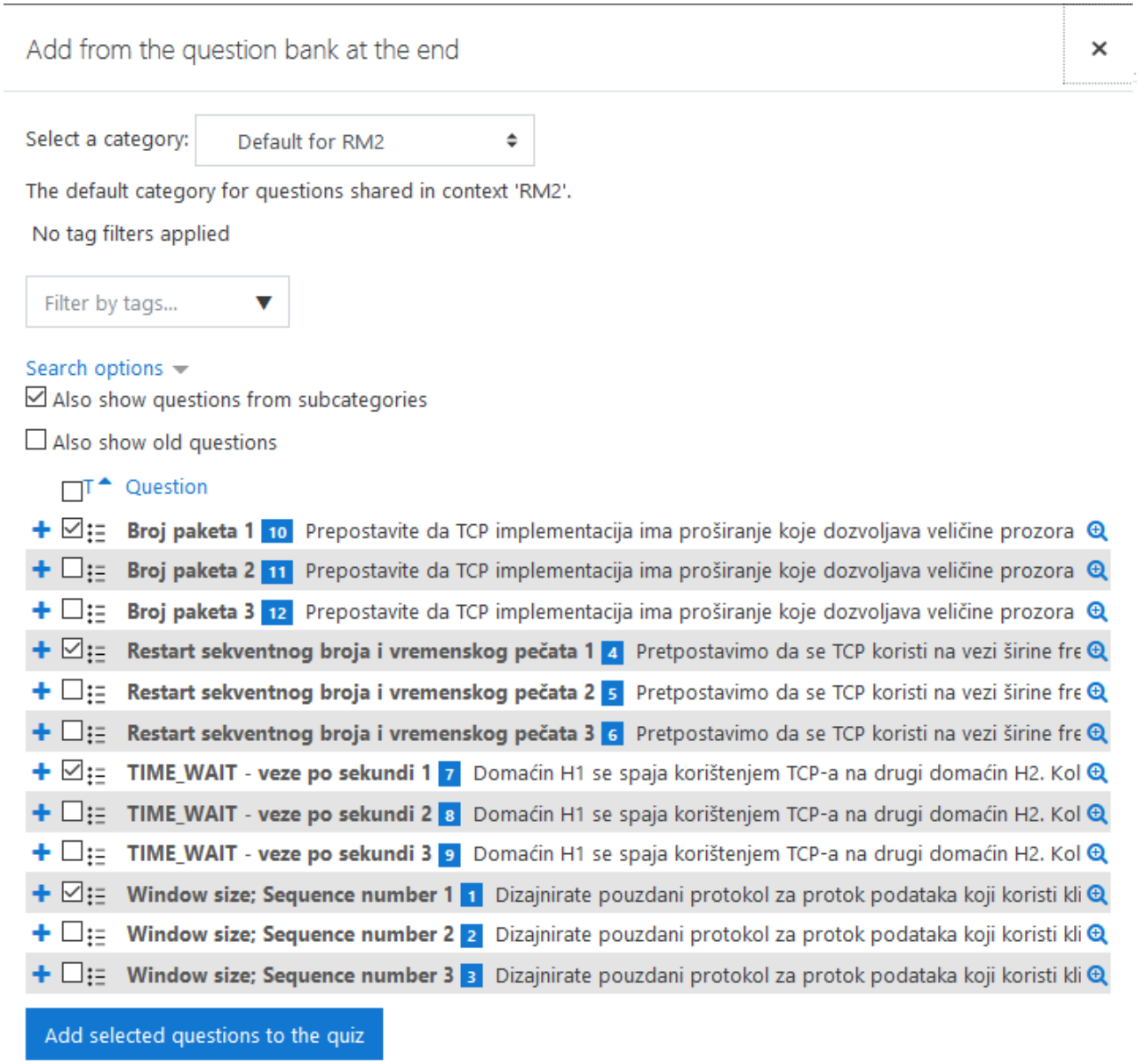
Slika 23: Stvaranje aktivnosti "Quiz"

Nakon unosa naziva i ostalih atributa, pokazuje se prozor koji omogućuje odabir opcije uređivanja „*Quiz*”-a. Klik na opciju „*Edit quiz*” otvara drugi prozor koji omogućuje dodavanje pitanja za „*Quiz*”.



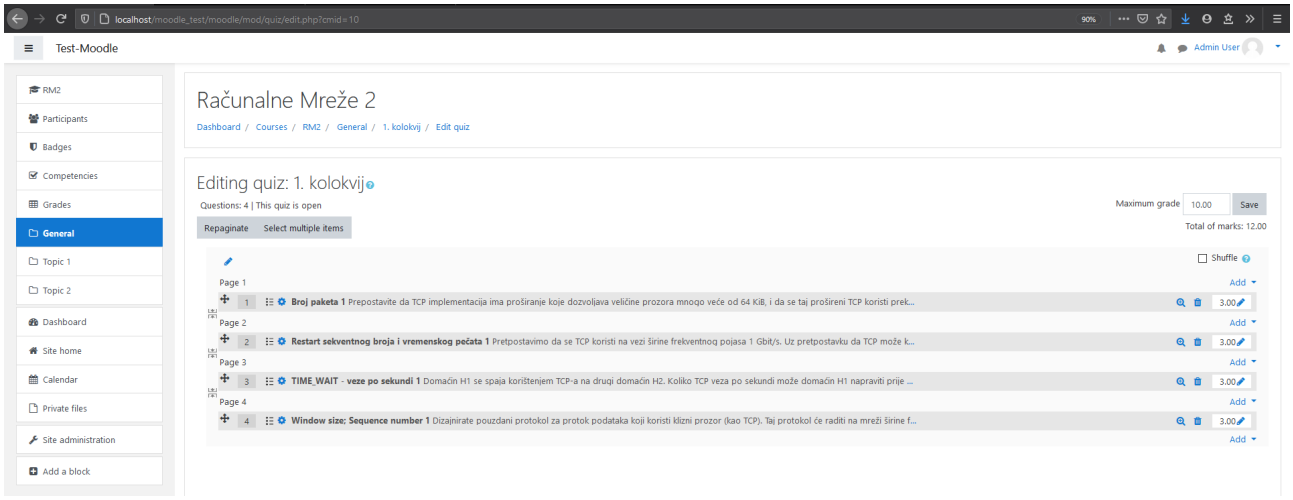
Slika 24: Uređivanje "Quiz"-a

Kako se pitanja nalaze u „*Question bank*”, njihovo dodavanje dostupno je klikom na opciju „*Add*” -> ”*from question bank*”. Sada se otvara novi prozor koji ispisuje dostupna pitanja i omogućuje dodavanje pojedinačnog pitanja.



Slika 25: Ispis dostupnih pitanja iz "Question bank"

Odabirom „*Add selected questions to the quiz*”, pitanja će biti povezana sa stvorenim „*Quiz*”-om i dostupna za rješavanje.



Slika 26: "Quiz" sa odabranim pitanjima

Time je uvoz pitanja u sustav Moodle potpuno obavljen.

5. Zaključak

Početna pretpostavka glasi da će računalo obaviti posao rješavanja zadataka brzo i točno. Ručnim rješavanjem zadataka koje su rješavali alati obrađeni u ovom radu dolazi se do istog rješenja. Ovisno o arhitekturi računala koje obavlja izračun nad zadanim podatcima, rezultat može varirati u decimalnim razmjerima, što je za praktičnu primjenu u nastavi nevažno.

Ručno rješavanje takvih zadataka i njihov unos u sustav Merlin zahtjeva veliku količinu truda i koncentracije od strane čovjeka. Računalo je u takvom procesu primjenjivije jer je jednako uspješno rješava i prvu i stotu instancu istog zadatka. Količina truda koju čovjek mora uložiti je složenošću postupka rješavanja, a za suvremena računala to su izračuni koji se vrlo kratko izvode. Dobra definicija struktura podataka i implementacija algoritama nad njima će izravno utjecati na složenost funkcija koje računalo obavlja u procesu obrade podataka i rješavanja zadataka.

Ovim je radom dana jedna od mogućih definicija struktura podataka i implementacija algoritama nad njima koja minimizira složenost funkcija potrebnih za rješavanja zadataka i njihovu pretvorbu u format koji se može uvesti u Moodle.

Literatura

[1] Peterson, Larry L., and Bruce S. Davie. *Computer Networks: A Systems Approach*. 3rd ed. Amsterdam ; Boston: Morgan Kaufmann Publishers, 2003.

[2] “Računalne mreže 2 — CNPSLab Homepage.” Accessed September 23, 2020. https://lab.miletic.net/hr/nastava/kolegiji/RM2/#kolegij-rm2.

[3] “Python 3.8.6rc1 Documentation.” Accessed September 23, 2020. https://docs.python.org/3/.

[4] “Computer Network.” In *Wikipedia*, September 21, 2020. https://en.wikipedia.org/w/index.php?title=Computer_network&oldid=979641440.

[5] “Moodle XML Format - MoodleDocs.” Accessed September 23, 2020. https://docs.moodle.org/39/en/Moodle_XML_format.

[6] “The Web Framework for Perfectionists with Deadlines | Django.” Accessed September 23, 2020. https://www.djangoproject.com/.