Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet
Matematički odsjek

Modeliranje dokumentske baze podataka za prijavu kandidata na fakultete

TIM ETA
Katarina Kozina
Mateo Martinjak
Mateja Pejić
Mihaela Zima

Profesor:
Ognjen Orel

Zagreb, lipanj 2021.

Sadržaj

1 Uvod

Zadatak: Modelirati dokumentsku bazu podataka za prijavu kandidata na fakultete i izgraditi aplikaciju nad njom.

Model treba sadržavati podatke o budućim studentima-kandidatima, njihovim ocjenama s mature i prijavama na pojedine fakultete i studije na tim fakultetima i slične detalje – po želji ga možete obogatiti. Odrediti što će biti agregati, nemojte se uplesti u relacijsko modeliranje. Napuniti bazu s podacima o najmanje 100 kandidata (generirati podatke). Koristiti mongoDB. Izgraditi aplikaciju koja će služiti za unos prijava novih kandidata i pregled upisanih podataka, u programskom jeziku/okruženju po želji. Glavna funkcija aplikacije je rad s agregatnim modelom, ali razmisliti i implementirati održavanje kvalitete pojedinih podataka. Napisati M/R upite za izradu rang lista kandidata po fakultetima i studijima.

U svrhu ovog projekta, izradili smo svoju bazu podataka čiji ključevi su OIB-i studenata i fakulteta. Usluge koje naš sustav nužno treba obavljati su: pružanje uvida u ocjene studentima, na koji je fakultet student upao, te popis studenata koji je upao na određeni fakultet.

2 Baza podataka

Kreirali smo dokumentacijsku bazu podataka project koja se sastoji od četiri kolekcije: studenti, fakulteti, global i admini.

project

DATABASE SIZE: 90.84KB INDEX SIZE: 120KB TOTAL COLLECTIONS: 5

CREATE COLLECTION

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
admini	2	324B	162B	1	20KB	20KB
fakulteti	16	4.38KB	281B	1	20KB	20KB
global	1	161B	161B	1	20KB	20KB
studenti	200	85.14KB	436B	1	24KB	24KB

Svaki agregat kolekcije studenti opisuje jednog studenta, a ukupno ih ima 200. U agregatu svakog studenta navedena su polja: _id, username, password, ime, prezime, datum_ rodenja, email, ocjene, drzavna_natjecanja i lista_fakulteta.

Lista ocjene sadrži informacije o studentovom prosjeku iz srednje škole, te ocjenama ostvarenim na maturi iz predmeta matematika, hrvatski, engleski i/ili izbornog predmeta. Pretpostavili smo da svaki student može polagati najviše jedan izborni predmet.

Lista drzavna_natjecanja sadrži informacije o nazivu predmeta iz kojeg je student sudjelovao na državnom natjecanju i mjestu koje je ostvario. Pretpostavili smo da je svaki student sudjelovao najviše na jednom natjecanju, te smo uzimali u obzir samo ako je ostvario prvo do treće mjesto.

U polju lista_fakulteta nalaze se informacije o oibima fakultetima koje student želi upisati, poslagane po prioritetima.

Detaljniji izgled kolekcije studenti prikazan je na slici.

Svaki agregat kolekcije fakulteti opisuje jedan fakultet, a ukupno ih ima 16. U agregatu svakog fakulteta navedena su polja: _id, oib, ime, prezime, admin_username, admin_password, kvota i uvjeti.

Lista uvjeti sadži informacije o koeficijentima koje ćemo koristiti pri računanju bodova pojedinog studenta, nazivu izbornog predmeta koji taj fakultet boduje, te nazivu predmeta iz kojeg se boduje natjecanje za taj fakultet. Koeficijenti za predmete matematiku, engleski i hrvatski se razlikuju za svaki fakultet. Koeficijent kojim množimo prosjek je 2. Ukoliko je student položio izborni koji fakultet boduje, koeficijent je 6. Prvo mjesto na državnom natjecanju iz odgovarajućeg predmeta donosi 1000 bodova i izravan upis na fakultet. Drugo i treće mjesto donose po 10 bodova.

Detaljniji izgled kolekcije fakulteti prikazan je na slici.

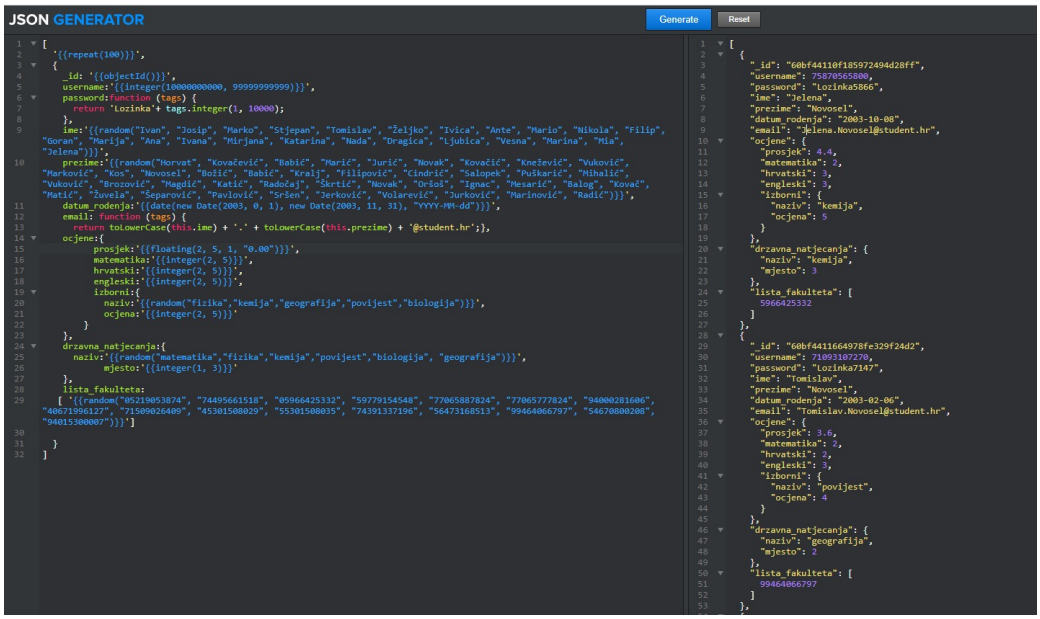
Agregat kolekcije global sadrži informacije o datumu zaključavanja rang listi i datumu objave rezultata. U agregatu su navedena polja: _id, PLANIRANI_DATUM_ZAKLJUCAVANJA, PLANIRANI_DATUM_OBJAVE_REZULTATA, UPISI_ZAKLJUCANI, AGREGACIJA_GOTOVA i REZULTATI_GOTOVI. UPISI_ZAKLJUCANI, AGREGACIJA_GOTOVA i REZULTATI_GOTOVI mogu poprimiti boolean vrijednosti.

Svaki agregat kolekcije admini sadrži informacije o adminu aplikacije. Postoje dva admina. U agregatu su navedena polja: _id, oib, ime, prezime, admin_username i admin_password. Uloga admina je osvježavanje podataka u bazi.


```
_id: "60b6c4bf9a2cf67564e43104"
oib: "66800066209"
ime: "Josip"
prezime: "Jakara"
admin_username: "admin.jakara"
admin_password: "QgJegvBFRDX"
```


2.1 Kreiranje baze podataka

Kolekcije su kreirane i napunjene korištenjem JSON Generatora, te spremljene u MongoDB Atlas. Tako su svi članovi grupe mogli jednsotavno pristupiti bazi podataka, te raditi upite na njoj.



3 Aggregation Pipeline Framework (APF)

Budući MongoDB Atlas (free verzija) ne podržava map reduce funkciju, umjesto M/R upita za rad s agregatnim modelom koristili smo Aggregation Pipeline Framework (APF).

Kako nam se podaci potrebni za stvaranje rang listi nalaze unutar dvije kolekcije, da bi ih mogli koristiti prvo smo te kolekcije trebali povezati.

```
[{$lookup: {
  from: 'studenti',
  localField: 'oib',
  foreignField: 'lista_fakulteta',
  as: 'lista'
}},
```

Funkcija lookup povezuje dve kolekcije iz iste baze, fakulteti i studenti. LocalField i foreingFiled povezuju kolekcije prema oibu fakulteta. Oni osiguravaju da u novostvorenoj kolekciji lista nemamo redundatne podatke, u našem slučaju npr. studente koji nisu prijavili niti jedan fakultet.

Zatim smo pomoću unwind funkcije dohvaćali potrebne podatake o pojedinom fakultetu i pojedinom studentu.

```
{$unwind: {
  path: "$lista"
}}, {$unwind: {
  path: "$uvjeti"
}},
```

U project funkciji računamo ostvarene bodove pojedinog studenta iz obaveznih i izbornih predmeta te natjecanja. Bodove računamo prema pravilima bodovanja pojedinog fakulteta. Na osnovu tih bodova ćemo kasnije stvarati rang liste.

Funkcijom set smo zbrojili bodove studenta ostvarene u pojedinom području (glavni predmet, izborni i natjecanja).

Sljedeći korak bio je sortiranje podataka po ukupnom broju bodova silazno u svrhu podjednostavljenja rangiranja.

Group funkcija nam grupira podatke po odredenom ključu, u našem slučaju oibu fakulteta, to znači da se po svakom fakultetu stvara točno jedan

```
{$set: {
 zbroj: {
   $sum: [
     '$zbrojG',
     '$zbrojN',
     '$zbrojI'
   ]
}
}},
```

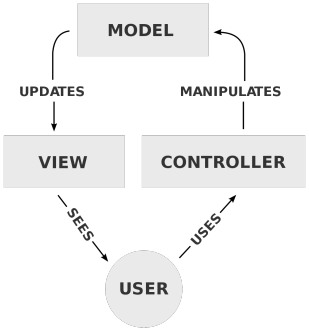
objekt koji sadrži informacije definirane u funkciji group. Kod nas su to svi studenti koji su stavili taj faks u svoj izbor. Uz ostvaren broj bodova te infomacija koji izbor je to bio 1., 2. ili 3. Također vraćamo i ostvaren broj bodova tih studenta.

```
{$group: {
  _id: '$fakultet',
  kvota: {$first: "$kvota"},

    lista_bodova: {
    $push: '$zbroj'
  },
  oibovi: {
    $push: '$oib'
  },
  izbor: {
    $push: '$izbor'
  }
}
```

Na samom kraju out nam dodaje željene podatke i nove informacije, defnirane u goup-u u željenu kolekciju. U našem slučaju stvaramo novu kolekciju koju smo nazvali lista.

4 Logička struktura projekta

Pri izradi projekta koristili smo Model–view–controller pattern (link) jer je posao tada puno jednostavniji i da se rastaviti na manje logičke jedinice.



4.1 Controller

Accepts input and converts it to commands for the model or view.

Kontrolere poziva korisnik kroz razne akcije, kao što su forme, akcije buttona, ili kroz sam browser. Oni mogu i nemoraju primati neke dodatne informacije na ulazu. Kontroleri koriste model i daju mu upite da bi napravio neku akciju. Kontorler također manipulira viewom i zadaje koji view se mora prikazati u kojem trenutku.

Primjer (Login). Nalazimo se na početnoj stranici s Login formom, te pritiskom za Submit button, akcija se pokrene koja poziva funkciju processLogin() unutar kontrollera AuthController. Taj kontroller također ima sve funkcije koje su logički vezane za posao ulogiravanja, kao npr. processRegistration(), checkIfUserExists() i slično.

4.2 Model

The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

Model je komponenta koju kontroller koristi za upravljanje podacima. Model ima direktnu kontrolu nad bazom podataka i svim ostalim podacima na sustavu koje kontroler treba. U modelu je također dobro pirpremiti podatke u dobrom formatu prije slanja tog controlleru. Općenito je napravi više modela od kojih svaki radi nešto logički različito.

Primjer. Imamo model SQLService kojeg naš kontroller koristi. Kontorler neće zadavat upite bazi ali ćemo to pitati model i on će složiti podatke dovoljno čitljive da samo imamo minimalan posao pirpremanja prije prikaza podataka. Naš SQLService bi mogao imati funkcije kao na primjer SQLService.getAllUsers() ili SQLService.getUserWithId(id), dosta je očito što oni rade.

4.3 View

Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

View je skup različitih pogleda (konkretno php ili html datoteka) koje su „skoro“ spremne za prikaz korisniku. Svaka datoteka u viewu zahtjeva neke podatke prije nego se može prikazati.
Kontroller ,jednom kada pripremi sve podatke ,na kraju pozove neki specifični view da pokaže podatke koje je pripremio.

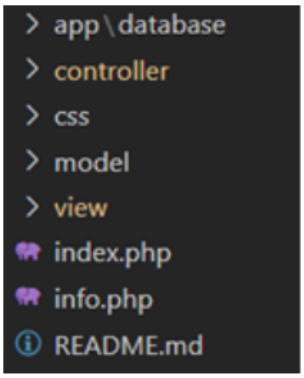
Primjer. Možemo imati view koji prikazuje login page, view koji pokazuje register page, te view za prikaz korisniku login success ili login reject. Jedan view mozemo imati kao main page jednom kad se korisnik ulogira u stranicu.

4.4 Database(dodatno)

Koristi ga model za pristup bazi. Ponekad može samo spajanje na bazu problematično pa je dobro to odvojiti u posebnu strukturu.Pogotovo ako imamo više baza iz više izbora, onda se model može spojit na koju hoće,kad hoće da bi obradio neki upit.

5 Struktura projekta

Projekt je dosta stukturno odvojen na cjeline kao što je objašnjeno u prošlom poglavlju. Dodane su još neke komponente za lakše korištenje MVC patterna.



5.1 Instalacija

5.1.1 Lokalna instalacija

Koristi se paket WAMP ,koju u sebi sadrži Apache server + PHP + MySql. PHP ne dolazi s podrškom za mongodb pa se moraju instalirati potrebni driveri za to. Pogledati više o tom ovdje.

Mora se skinuti dll datoteka mongodb.dll u instalaciju PHPa te promijeniti php.ini file da bi PHP prepoznao da je dodan mongodb (VAŽNO: ako se koristi WAMP,on ima svoj vlastiti php.ini koji se zove drugačije ,ali u istom folderu se nalazi gdje je i običan php.ini)

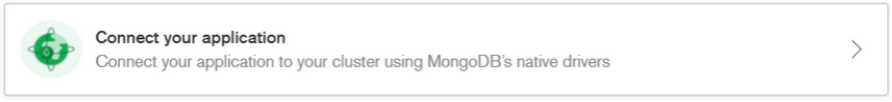
5.1.2 Instalacija na server

Koristili smo server hosta https://cyclades.okeanos-global.grnet.gr/. Na njemu se jednostavno složi Ubuntu server image i na njega se instaliraju svi potrebni paketi i Apache,te se kroz sučelje od hosta dobije IP adresa koju je potrebno povezat s našim VM.

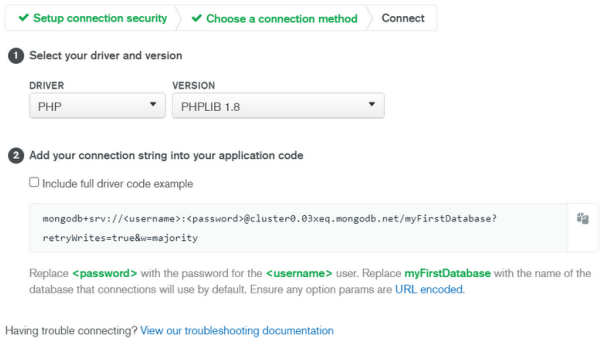
Jednog kada Apache radi, koristi se git da bi se updejtala verzija (sudo git pull) aplikacije na tom serveru, koje prvo napravimo na lokalnoj verziji.

5.1.3 Korištenje MongoDB u PHPu

Mi smo koristili Mongo Atlas bazu, o tome više u sectionu o samoj bazi. Atlas nudi način da se poveže s njim direktno iz PHPa.Kad se napravi cluster, i odabere Connect:



Atlas nudi string koji se mora upisati prilikom svakog spajanja:



Unutar PHP-a, veza se uspostavlja pomoću naredbe.

```php
$db = new MongoDB\Driver\Manager("STRING");
```

Jednom kada imamo vezu na bazu, zadaju se upiti.

Mongo upit za čitanje

```php
$filter  = [ 'ime' => 'Marko '];
$options = [];

$query = new \MongoDB\Driver\Query($filter, $options);

$result = $db->executeQuery("project.ucenici",$query);
```

U fiter polje se zadaje po kojem atrubutu tražimo dokumente u bazi. ExecuteQuery je glavna funkcija koja vraća rezultate upita.

(VAŽNO: upiti se vraćaju u obliku Mongo Cursor Objekta pa je poželjno prije return rezultata pretvoriti ga u array s $result-¿toArray())

Kako dobiti sve dokumente u kolekciji? Tako da ćemo u Query proslijediti prazno polje.

Mongo upit za edit

```php
function changeUcenikWithId($userId,$atribut, $vrijednost){
    $db = DB2::getConnection();
    $bulk = new MongoDB\Driver\BulkWrite;
    $bulk->update(
        ['_id' => $userId],
        ['$set' => [$atribut => $vrijednost  ]],
        ['multi' => false, 'upsert' => false]
    );

    $result = $db->executeBulkWrite("project.studenti",$bulk);
}
```

Izmjene u kolekciju se rade malo drugačije,preko BulkWrite objekta. Kroz njegovu metodu update zadajemo po kojem kluču će naći odgovarajuće dokumente koje treba izmjeniti, te ćemo kroz $set proslijediti atribute koje želimo izmjeniti. Vrijednost atributa može biti lista, varijabla, null...

Mongo agregacije

```php
public function getStudentsList($student){

  $db = DB2::getConnection();

  $command = new MongoDB\Driver\Command([
      'aggregate' => 'studenti',
      'pipeline' => [
          ['$match' => ['_id' => $student->_id]],

          ['$sort' => ['redniBroj'=> 1]],

      ],
      'cursor' => new stdClass,
  ]);

  $result=$db->executeCommand('project', $command)->toArray();

  return  $result;
}
```

Agregacije se rade kroz Command objekt. Aggregate atribut kaže nad kojom se kolekcijom agregira, pipelineu dajemo pipeline kod isto kao i kod konzolnog upita, i cursoru moramo proslijedit tip izlaza.

VAŽNO: Agregacije za konzolni upis se mogu relativno jednostavno konvertirati u PHP sintaksu za pipeline kod ako se poštuju ova pravila: Zamijeni se svaka pojava { } s [ ], te svaka pojava : s => , također se svako ime atributa stavi u navodnike kao u primjeru.

5.2 index.php

Startna točka aplikacije. Svaki upit dolazi u tu datoteku kroz GET atribut i ta datoteka zove odgovarajuće kontrollere.

Svaki upit koji aplikacija koristi je formata

index.php?rt=controler/funkcija/broj

gdje

- Controller: ime kontrolera kojeg zovemo - obavezno
- Funkcija: funkcija u kontrolleru koju zovemo - nije obavezno, ako nije navedeno zove se index funkcija unutar kontrolera
- Broj: neki dodatni parametar koji neka funkcija prima - nije obavezno, tada funkcija prima null

Kako konkretno index.php radi?

Njegov rad možemo podijeliti u četiri koraka:

- Učita polje rt dobiveno GET metodom.
- Rt string se podijeli na odgovarajuće dijelove.
- Provjerava ako postoji ta funkcija, ako ne postoji koristi se 'index'.
- Zove se odgovarajući kontroller, tj. njegova funkcija.

Primjer

Upisali smo login podatke u login formu. Forma šalje POST metodom sve podatke na adresu index.php?rt=auth/login. Konkretno, tu je auth kontroller a login() njegova funkcija. Ona će obraditi login podatke i pozvati neki drugi kontroller i/ili pozvati neki view da pokaze success/reject.

Important note

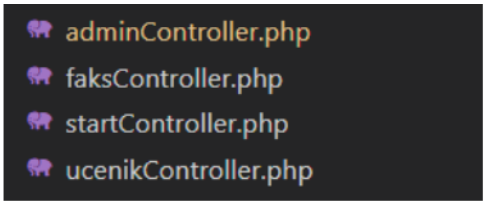
Potrebno je razlikovati index.php u root direktoriju koja obavlja posao routinga i index() funkcije koju svaki kontroler ima.

Important note 2

Svejedno je hoće li user ili funkcija pozvati index.php?rt=nekiKontroler/index ili index.php?rt=nekiKontroler. U oba ova slučaja se zove funkcija index() unutar nekiKontroler.

5.3 Kontrolleri

Mi koristimo 4 kontrollera. Po jedan za svaku vrstu korisnika, te jedan koji će se brinuti samo o autentikaciji korisnika i Loginu.



5.3.1 Kako kontrolleri točno rade u što sve pozivaju

Svaka funkcija u kontrollerima ima kod nalik ovome:

1. Inicijaliziraj session - session_start(). U ovom koraku inicijalizira se session međuspremnik koji daje PHPu do znanja da može koristiti session varijable u kodu.

2. checkPrivilege() – opis ove privatne funkcije je dolje naveden

3. $m= new MongoService(). Inicijalizira se Objekt koji je zadužen za komunikaciju s bazom. Kroz njega ćemo slat upite na bazu i primat gotov array ili dokument koji ćemo direkno onda slati u view.

4. $activeInd=0; Ova varijabla je samo tu da da do znanja side Baru gdje se korisnik nalazi zbog izgleda linkova u sideBaru, nema veze s kodom.Imaju ga samo one fje. Koje se mogu pozvat iz side bara.

5. *** pozivaj $m i zatraži podatke ***

6. require_once __DIR__.'/../view/'.$USERTYPE.'/******.php Ovo je finalni korak. Tu se poziva view koji će čitat one varijable koje smo definirali u 5.koraku,te ih prikazati u obliku HTMLa.View koristi te varijable ,pa se mora paziti koja ćemo imena dodijeliti tim podacima koje dobimo u 5.koraku.

Svi kontroleri imaju jednu posebnu privatnu funkciju koja se zove checkPrivilege(). Ta funkcija uglavnom provjerava dvije stvari: dali je korisnik

uopće autentificiran, i druga koji tip (vrsta) = student,faks,admin mu je dodijeljen pri Loginu Tu funkciju pozivaju sve public funkcije unutar svakog Controllera.

Svaka public funkcija u kontrolerima (dakle sve one koje su dostupne bilokome preko browsera) na samom početku pozivaju checkPrivilege().

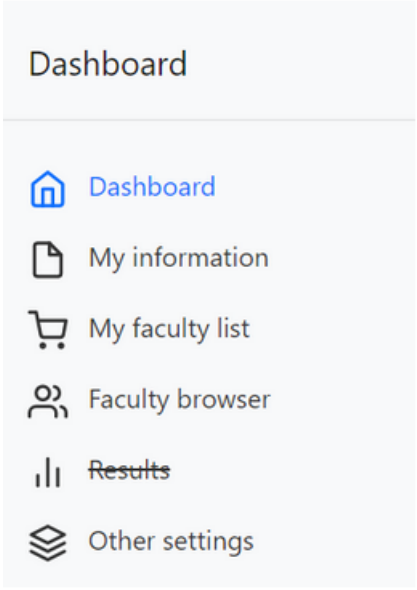
Npr. ako je učenik autentificiran (u Login fazi bit će mu proslijeđena session varijabla account_type=ucenik), tada će ga apliacija redirectati na Main page na adresi ucenik/index. Ali ako učenik slučajno ode na adresu na koju nebi smio doći, npr. na faks/index, funkcija index() u Controlleru Faks poziva svoju funkciju checkPrivilege() te provjerava je li account_type==faks. Očito je to false te se korisnik redirecta na svoj main page.

5.3.2 Start Controller

On je zadužen za login i autentifikaciju korisnika. Lista funkcija koju kontroler sadrži je sljedeća:

1. PRIVATE checkPrivilege() – pogledaj section prije
2. index() - Ova funckija inicijalizira glavni login page. Tu korisnici upisuju svoje podatke ,te će biti proslijeđeni na jedan od ***Check***() funkcija kada ispune jednu od formi.
3. loginCheckFaks()
4. loginCheckAdmin()
5. loginCheckUcenik() - Funkcije dobivaju iz baze učenika s tim usernameom koji je korisnik upisao, KAO I korisnikove podatke iz forme preko POST metode, te ih uspoređuje. Ako je sve u redu, korisniku se dodjeljuje session varijabla account_type kao i par drugih session varijabli da uvijek imamo uvid u to tko je trenutno ulogiran korisnik. Može se dogoditi da taj korisnik uopće nije u bazi, tada MongoService model vraća null te se korisnik redirecta na login page.
6. logout() - Funkcija briše kompletan session međuspremnik, te redirecta korisnika na index() stranicu gdje se nalazi Login page.

5.3.3 Učenik Controller



Kontroler koji se brine o korisniku = učenik.

Linkovi na funkcije 2-7 se nalaze na side Baru. Te fukcije nemaju nekog drugog značaja osim da dohvate iz baze tražene tablice i ispisuju ih korisniku.

Lista funkcija koju kontroler sadrži je sljedeća:

1. PRIVATE checkPrivilege() - pogledaj section prije.

2. index() - Prikazuje main dashboard korisniku, glavnu stranicu na koju dođe pri loginu.

3. myInfo() - Prikazuje korisniku osobne informacije, kao i podatke o ocjenama i natjecanjima na kojima je osvojio značajna mjesta.

4. myList() - Ova stranica prikazuje korisnikov odabir fakulteta na koje bi htio ići. Rangira ih od 1. nadalje ,te ih može naknadno sortirati po želji jednom kad ima svoj konačan popis.

5. browser() - Na ovoj stranici korisnik odabire fakultete na koje bi se želio prijaviti. Tu se također nalaze sve potrebne informacije za učenika o fakultetima, kao kvota fakulteta i koji se predmeti gledaju kod upisa, kao i faktori s kojima se množe pojednine ocjene iz predmeta.

6. results() - Ova stranica će postati aktivna jednom kada je upis gotov. Tu će biti prikaz rezultata učenika jednom kad upisi budu gotovi (prije tog je stranica onemogućena).

7. otherSettings() - Na ovoj stranici se samo daje mogućnost učeniku da

promijeni neke osobne podatke, kao ime, username, prezime, email...

8. otherSettingsCheck() - Tu se provjerava što je unešeno na formi s „otherSettings“ i mijenja se podaci u bazi po potrebi.

Sljedeće se funkcije bave učenikovim izborom fakulteta na stranici myList

9. myListPushUp($index) - Poziv funkcije s nekim indexom zadaje modelu upit da zamijeni poredak u popisu fakulteta na tom indexu, konkretno faks s indexom $index stavit će na jedno mjesto više na učenikovom popisu.

10. myListPushDown($index) - Radi isto kao i prethodna funkcija samo što stavlja faks na jedno mjesto niže.

11. myListInsert($faksOib) - Funkcija ubacuje fakultet s $faksOib u učenikovu listu za odabir.

12. myListDelete($index) - Funkcija briše fakultet s liste s indeksom $index.

5.3.4 Faks Controller

Faks kontroler se brine o korisniku account_type=faks. Služi da admini fakulteta imaju uvid u to tko ga je upisao.

1. checkPrivilege()
2. index()
3. results()
4. myInfo()

5.3.5 Admin Controller

1. checkPrivilege()
2. index()
3. otherSettingsCheck()
4. globalSettings()
5. browser()
6. reset()
7. start()

8. resultsSwitch()

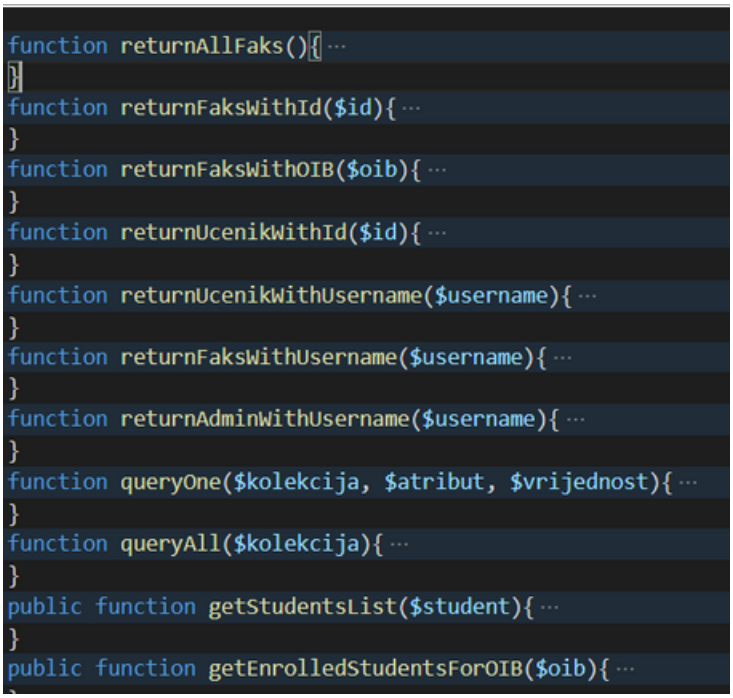
9. lockSwitch()

5.4 Modeli

Imamo 2 modela.

5.4.1 MongoService

MongoSevice je ogromna kolekcija funkcija koje rade upite nad našom bazom, ukratko ćemo opisati što svaki upit radi.



Sve osim dvije zadnje sastoje se od nekih jednostavnih upita koje vraćaju ili jedan ili sve dokumente iz kolekcija.

Sve te funkcije zapravo pozivaju QueryOne i QueryAll funkcije,zbog kompaktijeg koda.

getStudentsList vraca listu objekata fakulteti s OIBovima koji pisu u ucenikovoj listi izbora.

getEnrolledStudentsFromOIB vraća popis studenta koji su se upisali na fakultet s zadanim OIBom.

```
function changeUcenikWithId($userId,$atribut, $vrijednost){ ···
}
function pushNewListToStudentWithId($userId,$lista,$index,$action="UP",$newElement=null){ ···
```

Funkcija changeUcenikWithId za učenika s zadanim $userId mijenja neki atribut, mi ga konkretno koristimo da bi promijenili njegov username,ime, ili neke podatke koje su pogrešno unešeni.

pushNewListToStudentWithId manipulira studentovom listom fakulteta,tj arrayjem OIBova fakulteta.Kad god student želi promijeniti bilokako svoj poredak fakulteta ova funkcija će promijeniti poredak,dodati ili maknuti traženi fakultet s liste.

Iduće su na redu funkcije koje rade samo agregiranje i grupiranje studenata po fakultetima.

startAggregation() je funkcija koja inicira agregaciju. Funkcija redom stvara tablicu pomoću funkcije stvoriTablicuLista() ,koja stvara kolekciju fakulteta i rang liste SVIH studenata koji su zadovoljili uvjete od tog fakulteta.Jednom kad imamo tu tablicu, startAggregation će povući tu tablicu i tablicu studenti da bi napravila konačan popis studenata po fakultetima,te će također svakom studentu upisati u atribut „upisao=OIB_fakulteta“.

startAggregation() tada poziva fju. Run() koja obavlja kritičan posao grupiranja studenata po fakultetima.Algoritam koji koristi run() se nalazi u sectionu niže.

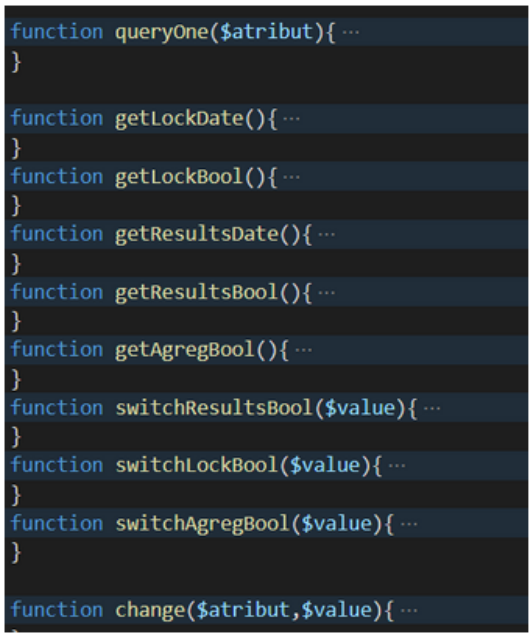
```
public function startAggreagtion(){
  echo "You initiated aggregation procedure.";

  $this->stvoriTablicuLista();

  $faksevi=$this->queryAll("lista");
  $faksevi=$this->urediFakseve($faksevi);

  $studenti=$this->queryAll("studenti");
  $studenti=$this->urediStudente($studenti);

  $this->run($faksevi,$studenti);
  $this->saveResultsAllFaks();
```

```php
public function resetAggreagtion(){
  $db = DB2::getConnection();
  echo "You initiated reset. DON'T CLOSE BROWSER.";

  $this->resetirajStudente();

  $db->executeCommand('project',
  new \MongoDB\Driver\Command(["drop" => "lista"])
  );
```

Posljednja funkcija u mongoService je resetAggregation(). Ako smo nešto napravili krivo ili se startAgreggation terminirao iz nekog razloga, resetAggregation će vratiti sve u prvobitno stanje,tj. postavit će atribute svih studenata „upisao=null“, i izbrisati kolekciju Lista s završnim bodovima studenata po fakultetima.

5.4.2 GlobalService

GlobalService se brine za globalne postavke naše aplikacije. On upravlja varijablama koje se brinu za to kad je vrijeme zaključavanja studentkih lista, kad je objava rezultata itd.



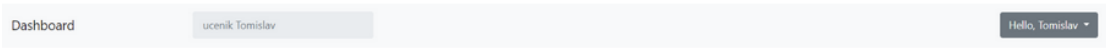
Postoji poseban dokument u našoj bazi koji čuva sve globalne postavke o aplikaciji, i mongoService manipulira njime i čita ga.

6 View

Viewova koje koristimo ima mnogo pa ih nećemo posebno navoditi. Svaki view zahtjeva od kontrolera sve varijable koje koristi i sam view. Potrebno je da, jednom kad su u viewu, varijable su relativno lako čitljive, i da su već zapakirane u stringovima.

Imamo par viewova koje koristimo u našoj aplikaciji, index.php na primjer je jako čest jer je on main page za svakog tipa korisnika, kao i main page za Login menu.

Imamo poseban view za sideBar i za navigation bar.



Pošto imamo 3 različita tipa sideBara, jedan za svaki tip korisnika, imamo 3 posebna viewa. Navigation bar je uvijek isti.

7 Algoritam rangiranja studenata

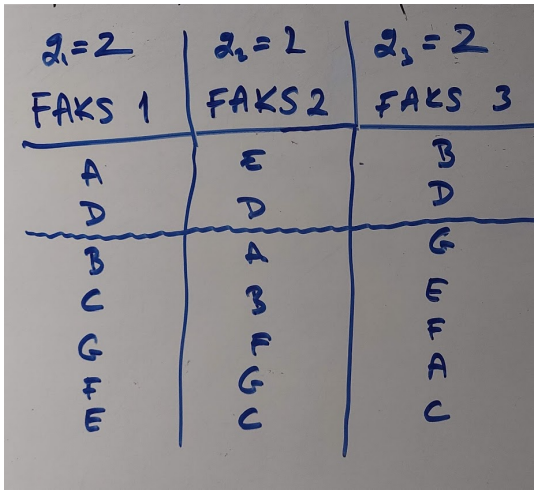
Algoritam radi na način da kroz cikluse upisuje u fakultete one studente koji su zadovoljili uvjete i nalaze se unutar kvote. Kada jedan ciklus završi, upisani studenti se zabilježe i brišu iz liste, te se gleda koliko mjesta na fakultetima je ostalo i onda se radi identičan odabir s ostalim studentima.

Algoritam završava kada:

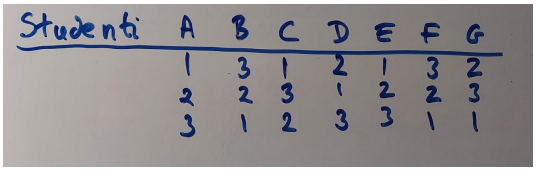
- nema više studenata koji traže fakultet
- sve su kvote popunjene

7.1 Primjer

Prvi korak je da zapišemo rangirani popis studenata koji su zadovoljili kriterije za te fakultete. Tada će svaki faks imati svoj vlastiti popis studenata sortiran od najboljeg koji oni žele primiti do onih lošijih, te također označujemo kvotu na listama.

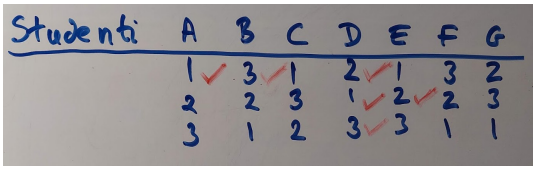


Isto napravimo i za studente, oni imaju rangiran popis fakulteta koje žele upisati.



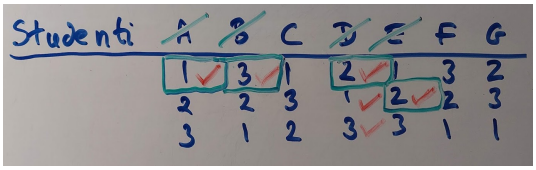
Ciklus 1

U popisu fakulteta kod studenata ćemo označiti sve one koji su unutra kvote za pojedini fakultet.



Jednom tako označeni, za svakog studenta biramo koji je prvi označen fakultet na njihovoj listi (na taj ima pravo upisa jer je unutar kvote i faks očekuje da on prihvati svoje mjesto).

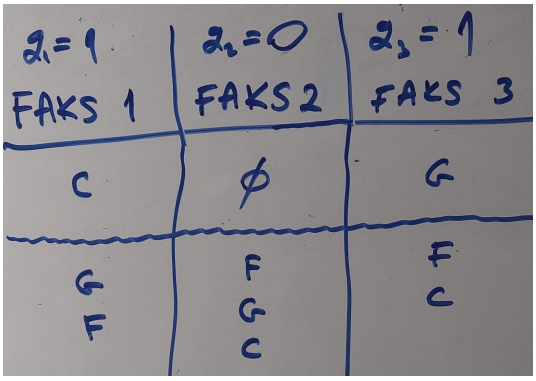
Nakon što odaberemo sve fakultete za učenike koje je to moguće (neki nisu nigdje prošli u ovoj rundi), spremamo ih kao „upisane“ i brišemo iz svih tablica.



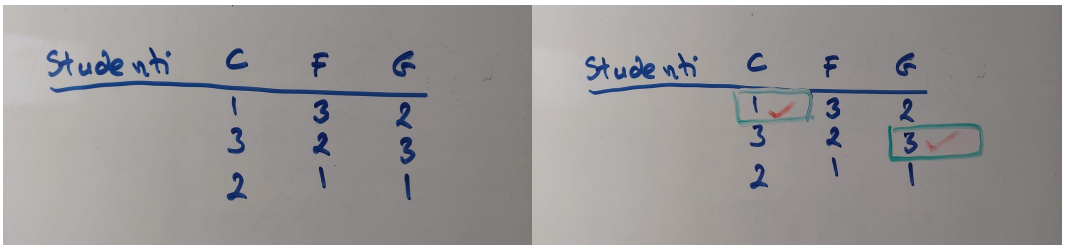
Također mijenjamo kvotu pojedinog fakulteta, ovisno kolko je mjesto još ostalo.

Ciklus 2

Pošto smo izbrisali one studente koji su se uspješno upisali, ostaje nam da popunimo ostala mjesta. Vidimo da je ostalo jedno mjesto na 1 i 3 fakultetu, a još ima 3 studenta koji zadovoljavaju njihove uvjete. Faks 2 više ne upisuje nikoga ,ali i dalje vidi popis studenata koji traže faks.



Pravo upisa imaju u ovom ciklusu jedino studenti C i G, i faks 1 i 3 im daju opciju upisa.Oni automatski prihvaćaju jer nema fakseva na njihovim listama koji imaju veći prioritet i koji im daju pravo upisa.



Algoritam tu završava jer su sva mjesta popunjena. Učenik G nije ništa upisao jer nije bilo vise mjesta, mada je zadovoljavao njihove uvjete.

Radi li algoritam ako je suma kvota fakulteta veca od broja učenika?

DA. Ako je studenata manje od sume kvota, tada će algoritam također završiti jer će lista učenika koji traže faks biti prazan. Tada će minimalno jedanom fakultetu faliti učenika.