

Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet
Matematički odsjek

Napredne baze podataka

Aplikacija za administraciju ispita: *DeltaExam*

Projektni zadatak

Tim δ:

Bruno Fabulić
Mia Matijašević
Maja Piskač
Mia Tadić
Kristina Udovičić

Nastavnik:

Ognjen Orel

Zagreb, svibanj 2020.

Sadržaj

1	Tema projektnog zadatka	1
2	Korištena tehnologija	1
3	Modeliranje dokumentne baze podataka	2
3.1	Objašnjenje agregatnog modela	4
3.2	Kreiranje podataka	4
3.3	Modificiranje agregatnog modela	4
3.4	Testiranje agregatnog modela	5
4	Aplikacija DeltaExam	5
4.1	Konekcija MongoDB-a i Visual Studio-a	5
4.2	Zahtjevi i funkcionalnosti aplikacije	6
5	Map/Reduce upiti	12
5.1	Obični i težinski prosjek studenta	12
5.2	Prosjek ispita	13
5.3	Prosjek kolegija	14
6	Poteškoće i rješavanje	15
7	Ideje za poboljšanja	16
8	Zaključak	16
	Reference	16

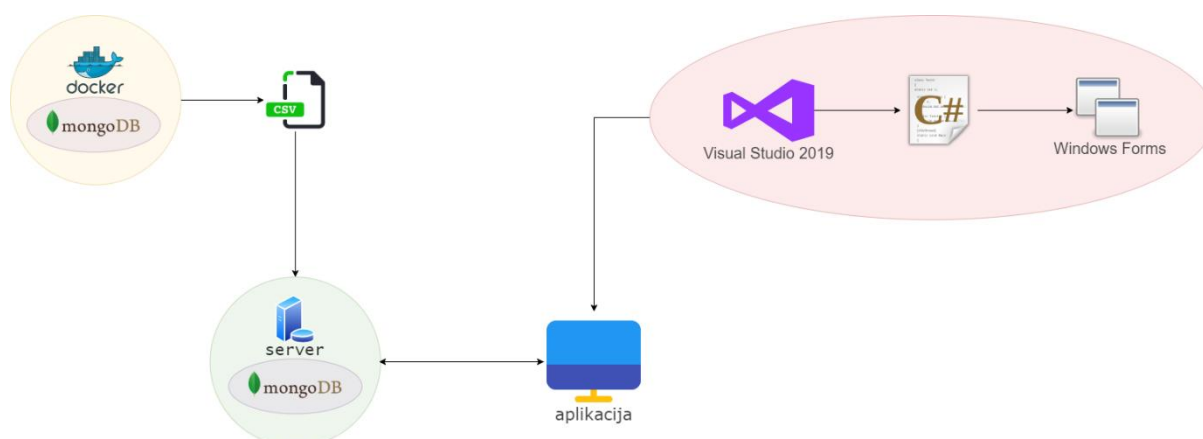
1 Tema projektnog zadatka

Tema 1: Modelirati dokumentsku bazu podataka za provedbu ispita i izgraditi aplikaciju nad njom.

Model treba sadržavati podatke o studentima, nastavnicima, ispitima, sadržajima ispita i slične detalje – po želji ga obogatiti. Odrediti što će biti agregati, nemojte se uplesti u relacijsko modeliranje. Napuniti bazu s podacima o najmanje 100 ispita (generirati podatke). Koristiti mongoDB. Izgraditi aplikaciju koja će služiti za administraciju ispita, u programskom jeziku/okruženju po želji. Glavna funkcija aplikacije je rad s agregatnim modelom, ali razmisliti o i implementirati održavanje kvalitete pojedinih podataka. Napisati M/R upite za analizu uspješnosti na ispitima.

Iako je provođenje ispita zajedničko svojstvo svim fakultetima, ipak svaki fakultet (pa i odsjek unutar fakulteta) ima svoja, ponešto drugačija pravila prilagođena svome području djelovanja. Stoga smo odlučili izabrati jednog klijenta za kojeg kreiramo aplikaciju za administraciju ispita s pripadnom bazom. Mi smo odlučili kreirati bazu i aplikaciju za studente i nastavnike Matematičkog odsjeka Zagrebačkog PMF-a, jer taj način polaganja ispita najbolje poznajemo te tako najbolje znamo odrediti i razraditi funkcionalnosti aplikacije.

2 Korištena tehnologija



Slika 2-1. Shema postupka kreiranja aplikacije integrirane s bazom, s prikazom korištene tehnologije

Za realizaciju baze podataka korištena je dokumentaska NoSQL baza podataka MongoDB. Tijekom podjele zadataka na generiranje podataka i kreiranje aplikacije, korištena su dva pristupa MongoDB bazi. Za generiranje podataka korištena je MongoDB baza zapakirana u Docker kontejneru, tamo su generirane kolekcije i pripadni dokumenti te su zatim podaci izvezeni u CSV datoteke. Za implementaciju baze povezane s izrađenom aplikacijom korišten je MongoDB server instaliran direktno na naša računala, a za uvoz podataka u ovu bazu korištene su spomenute generirane CSV datoteke. Aplikacija je izrađena u programskom jeziku C#, u integriranom razvojnom okruženju Visual Studio (verzija 2019 Community). Logički dio kreiranja aplikacije razrađen je u programskom jeziku C#, a za vizualnu strukturu aplikacije korištene su Windows Forme, grafička klasna biblioteka ugrađena u Microsoft .NET Framework. Za spajanje MongoDB baze s programom Visual Studio korišten je MongoDB Driver za C#/.NET. Instaliran je unutar Visual Studio programa koristeći NuGet Package Manager koji se također nalazi unutar Visual Studio-a.

3 Modeliranje dokumentске baze podataka

Kreirana je dokumentska baza podataka *ispiti* koja sadrži tri kolekcije: *kolegiji*, *studenti* i *nastavnici*.

Svaki agregat kolekcije *kolegiji* opisuje jedan kolegij, a ukupno ih ima 37. U agregatu svakog kolegija navedena su polja: *isvu_sifra*, *naziv*, *semestar*, *ects_bodovi*, *izborna_grupa*, *nositelj*, *nositelj_oib*, *nastavni_sadrzaji* i *ispiti*. Jedan agregat kolekcije kolegij prikazan je na slici 3-1.

```
{
  "isvu_sifra": 92978,
  "naziv": "Grada računala",
  "semestar": "ljetni",
  "ects_bodovi": 5,
  "izborna_grupa": "ne",
  "nositelj": "Slobodan Ribarić",
  "nositelj_oib": 87920272420,
  "nastavni_sadrzaji": "Von Neumannov model računala. Značajke CISC i RISC arhitekture.",
  "ispiti": [
    {
      "ispit_id": 929781,
      "datum_odrzavanja": "2020-04-30",
      "vrijeme_odrzavanja": "9",
      "prijava_do": "2020-04-25",
      "odjava_do": "2020-04-28",
      "nacin_polaganja": "pismeno",
      "sadrzaj": "Provjera znanja zaključno s cjelinom virtualne memorije.",
      "trajanje": "120",
      "predavaonica": "A101"
    },
    {
      "ispit_id": 929782,
      "datum_odrzavanja": "2020-06-26",
      "vrijeme_odrzavanja": "9",
      "prijava_do": "2020-06-21",
      "odjava_do": "2020-06-24",
      "nacin_polaganja": "pismeno",
      "sadrzaj": "Provjera usvojenog znanja zaključno s RISC arhitekturom.",
      "trajanje": "120",
      "predavaonica": "A101"
    },
    {
      "ispit_id": 929783,
      "datum_odrzavanja": "2020-07-15",
      "vrijeme_odrzavanja": "9",
      "prijava_do": "2020-07-10",
      "odjava_do": "2020-07-13",
      "nacin_polaganja": "usmeno",
      "sadrzaj": "Usmena provjera cjelokupnog gradiva.",
      "trajanje": "20",
      "predavaonica": "A206"
    }
  ]
}
```

Slika 3-1. Agregat kolekcije kolegiji

Polje *nositelj* sadrži ime i prezime nastavnika koji je opisan jednim agregatom u kolekciji *nastavnici*. Lista *ispiti* sadrži podatke o svim ispitima za pojedini kolegij. Na matematičkom odsjeku su u pravilu tri ispita te se naša baza time i vodi. Svaki kolegij polaže se prolaskom tih triju ispita: prvi kolokvij, drugi kolokvij i završni ispit. Agregat ispita opisan je poljima: *ispit_id*, *datum_odrzavanja*, *vrijeme_odrzavanja*, *prijava_do*, *odjava_do*, *nacin_polaganja*, *sadrzaj*, *trajanje* i *predavaonica*. Polje *ispit_id* kreirano je tako da se na kraj podatka spremljenog u polje *isvu_sifra* doda znamenka 1, 2 ili 3, ovisno o kojem se ispitu radi. Budući da su polja *isvu_sifra* jedinstvena, ovime je osigurana i jedinstvenost polja *ispit_id*. U bazi je definirano 111 ispita.

Svaki agregat kolekcije *studenti* opisuje jednog studenta, a ukupno ih ima 20. Agregat za opis studenta sadrži polja: *jmbag*, *ime*, *prezime*, *razina*, *smjer* i listu *kolegiji* koja sadrži podatke o kolegijima koje je student upisao. Primjer agregata kolekcije *studenti* nalazi se na slici 3-2. Podaci su generirani tako da je u prosjeku svaki student upisao 8 kolegija, odnosno trebao bi položiti u prosjeku 24 ispita u jednoj nastavnoj godini (zimskom i ljetnom semestru).

```

{
  "jmbag": 1191234996,
  "ime": "Ante",
  "prezime": "Antić",
  "razina": "diplomski",
  "smjer": "Računarstvo i matematika",
  "kolegiji": [
    {
      "isvu_sifra": 92979,
      "naziv": "Oblikovanje i analiza algoritama",
      "ects_bodovi": 5,
      "ispiti": [
        {
          "ispit_id": 929791,
          "status": "obavljen",
          "rezultat": "70%"
        },
        {
          "ispit_id": 929792,
          "status": "obavljen",
          "rezultat": "80%"
        },
        {
          "ispit_id": 929793,
          "status": "obavljen",
          "rezultat": "80%"
        }
      ],
      "ocjena": "4"
    },
    {
      "isvu_sifra": 92978,
      "naziv": "Građa računala",
      "ects_bodovi": 5,
      "ispiti": [
        {
          "ispit_id": 929781,
          "status": "obavljen",
          "rezultat": "80%"
        },
        {
          "ispit_id": 929782,
          "status": "prijavljen",
          "rezultat": null
        },
        {
          "ispit_id": 929783,
          "status": "neprijavljen",
          "rezultat": null
        }
      ],
      "ocjena": null
    }
  ]
},
]

```

Slika 3-2. Agregat kolekcije studenti
(radi čitljivosti, prikazani su ispiti samo dva upisana kolegija)

Polja navedena za identifikaciju kolegija su *isvu_sifra*, *naziv*, *ects_bodovi*, *ispiti* i *ocjena*. Polje *naziv* navedeno je da bi bilo dostupno kod ispisa odabira kolegija koje student želi pregledati. Polje *ects_bodovi* navedeno je u agregatu za svaki kolegij jer se M/R upitom računa prosjek studenta (i obični prosjek i težinski prosjek). Lista *ispiti* sadrži podatke o sva tri ispita kolegija, a svaki ispit je opisan poljima: *ispit_id*, *status* i *rezultat*, pri čemu *status* može biti *neprijavljen* (dodaje se svakom ispitu na početku te ostaje do trenutka kada student prijavi ispit), *prijavljen* (dodaje se ispitu nakon što se student prijavi na ispit) i *obavljen* (unos ga nastavnik u slučaju da je student izašao na ispit, bez obzira je li student položio ispit ili ne). *Rezultat* na ispitu ispisuje se u obliku postotka, a *ocjena* kao broj od 1 do 5.

Agregati kolekcije *nastavnici*, kojih ukupno ima 25, sadrže polja: *oib*, *ime*, *prezime* i *kolegiji*. Lista *kolegiji* je lista kolegija koje nastavnik predaje. Podaci su generirani tako da svaki nastavnik predaje od

1 do 3 kolegija. Primjer agregata kolekcije *nastavnici* nalazi se na slici 3-3. Za svaki kolegij koji predaje, nastavnik ima pravo promjene polja, brisanja ispita i dodavanja ispita.

```
{
  "oib": 26559220386,
  "ime": "Luka",
  "prezime": "Grubišić",
  "kolegiji": [
    {
      "isvu_sifra": 45688,
      "naziv": "Umjetna inteligencija"
    },
    {
      "isvu_sifra": 92949,
      "naziv": "Mreže računala"
    }
  ]
},
```

Slika 3-3. Agregat kolekcije *nastavnici*

Radi jednostavnosti, neki podaci iz baze (naziv, isvu šifra, ects bodovi, imena i prezimena nastavnika, nastavni sadržaji i ideje za sadržaje ispita, oznake predavaonica...) preuzeti su s [1].

3.1 Objašnjenje agregatnog modela

Kolekcije, odnosno agregati, su dizajnirani ovisno o podacima koji su predviđeni za prikazivanje na pojedinoj formi aplikacije. Iz tog razloga, za svakog nastavnika, odnosno studenta, postoji lista kolegija kojih je nositelj, odnosno lista kolegija koje je upisao, te se pri korištenju aplikacije ispisuju svi kolegiji vezani uz korisnika i omogućuje njihov odabir. Rezultati ispita svakog studenta i konačna ocjena kolegija pamte se u njegovom agregatu, u polju *ispiti* i polju *ocjena*, kako bi dohvat podataka za prikaz i računanje analize uspješnosti ispita bili jednostavniji.

Broj ispita za kolegij je tri, što je različito od standardnog poimanja ispita (dva zimski/ljetni roka i dva jesenska). Prilikom stvaranja baze i razmišljanjem o aplikaciji zamišljeno je da iste odgovaraju načinu polaganja kolegija na Prirodoslovno-matematičkom fakultetu u Zagrebu pa bi njegovi nastavnici i studenti bili idealni korisnici aplikacije.

3.2 Kreiranje podataka

Kolekcije su prvotno kreirane i napunjene korištenjem MongoDB upita *createCollection* i *insert*, a zatim su agregati izvezeni u CSV datoteke korištenjem naredbe *mongoexport*. To se pokazalo kao jednostavna razmjena podataka u projektnoj grupi. Import podataka u bazu povezanu s aplikacijom obavljao se naredbom *mongoimport*.

3.3 Modificiranje agregatnog modela

Prva važnija ideja za model bila je sljedeća: kolekcija *ispiti* koja bi sadržavala agregate ispita te kolekcija *korisnici* koja bi sadržavala agregate koji opisuju nastavnike i studente. No tada bi se podaci o kolegiju ponavljali u agregatima triju ispita. Također, u svakom agregatu ispita postojala bi (možda povećana) lista studenata koji su prijavili ispit. Dodatno, obje vrste korisnika aplikacije bile bi u istoj kolekciji, iako su pogledi, a onda i mogućnosti pri korištenju aplikacije, za nastavnika i studenta različiti. Iz navedenih razloga, kreiran je novi model, opisan u prethodnom dijelu poglavlja.

3.4 Testiranje agregatnog modela

U početku je kreirana „testna“ baza podataka, koja se također sastojala od navedenih kolekcija, ali koja je sadržavala po samo nekoliko agregata u svakoj kolekciji. Time se omogućilo da se model lako provjeri, te po potrebi promijeni i nadopuni. Tijekom kreiranja i testiranja aplikacije naišli smo na određene poteškoće te smo početnu bazu morali mijenjati – uvedena su nova polja u pojedine kolekcije (zbog lakšeg pristupa podacima ili zbog obogaćivanja baze) te su promijenjeni tipovi pojedinih podataka (na primjer umjesto *INT* u *string*, zbog lakšeg baratanja podacima u C#-u). Neka od polja koja su naknadno dodana su *nastavni_sadrzaji*, *sadrzaji*, *predavaonica* i *oib* nastavnika.

4 Aplikacija DeltaExam

Ime aplikacije je inspirirano imenom našeg projektnog tima – delta, te okosnicom naše aplikacije – ispitima (eng. *exam*).

4.1 Konekcija MongoDB-a i Visual Studio-a

S instaliranim MongoDB Driver-om, Visual Studio se lako poveže s lokalnom MongoDB bazom na sljedeći način:

1. Kreiranje Mongo klijenta za instanciranje veze s bazom:
`static MongoClient client = new MongoClient();`
2. Dohvaćanje baze *ispiti*
`static IMongoDatabase db = client.GetDatabase("ispiti");`
3. Dohvaćanje kolekcije *nastavnici* iz baze *ispiti*
`static IMongoCollection<Nastavnik> collection_kolegiji = db.GetCollection<Kolegij>("nastavnici");`

Za dohvaćanje jednog dokumenta u kolekciji *nastavnici*, potrebno je kreirati klasu u kojoj će svaki element iz dokumenta odgovarati objektu u C#-u. To ne mora nužno biti objekt kao instanca klase, već to može biti *int*, *string*, *double*, ..., a može (tj. mora) biti klasa ako imamo ugniježdene dokumente s više polja, te svaki taj ugniježđeni dokument mora biti predstavljen novom, odgovarajućom klasom s pripadnim elementima. Tako će se kolekcija *nastavnici*, u C# realizirati kao lista objekata *Nastavnik*. MongoDB Driver funkcionira na bazi BSON dokumenata. Na primjer, dokument kolekcije *nastavnici* (primjer agregata prikazan na slici 3-3), je u C#-u implementiran ovako:

```
class Nastavnik
{
    [BsonId]
    public ObjectId Id { get; set; }

    [BsonElement("oib")]
    public Int64 Oib { get; set; }

    [BsonElement("ime")]
    public String Ime { get; set; }

    [BsonElement("prezime")]
    public String Prezime { get; set; }

    [BsonElement("kolegiji")]
    public List<KolegijNastavnika> KolegijiNastavnika { get; set; }
}
```

```
class KolegijNastavnika
{
    [BsonElement("isvu_sifra")]
    public int Isvu_sifra { get; set; }

    [BsonElement("naziv")]
    public String Naziv { get; set; }
}
```

Za CRUD operacije postoje implementirane funkcije omogućene MongoDB Driver-om.

Read operacija se može izvesti na nekoliko načina, neki koje smo koristili su:

```
List<Nastavnik> list_nastavnici = collection_nastavnici.AsQueryable().ToList<Nastavnik>();
foreach (Nastavnik n in list_nastavnici)
    Console.WriteLine( n.Ime );
```

```
var nastavnici = collection_nastavnici.Find(n => n.Oib == OibNastavnika).ToList();
foreach (var n in nastavnici)
    Console.WriteLine( n.Ime );
```

Pokazat ćemo *update* operaciju na primjeru promjene polja *status* u kolekciji *studenti*. To polje se nalazi u ugniježđenim listama – u kolekciji *studenti* nalazi se polje *kolegiji* koje je lista dokumenata, i svaki taj dokument sadrži polje *ispiti* koje je također lista drugih dokumenata. U toj zadnjoj listi *ispiti* nalazi se polje *status* koje želimo promijeniti:

```
var updateDef = Builders<Student>.Update.Set("kolegiji.$[i].ispiti.$[j].status", "prijavljen");

var arrayFilters =
    new List<ArrayFilterDefinition> {
        new JsonArrayFilterDefinition<Student>(@'{i.isvu_sifra':" + kolegij_isvu + "'}"),
        new JsonArrayFilterDefinition<Student>(@'{j.ispit_id':" + ispit_id + "'}"),
    };

var updateOptions = new UpdateOptions { ArrayFilters = arrayFilters };

collection_studenti.UpdateOne(
    s => s.Jmbag == JmbagStudenta &&
    s.KolegijiStudenta.Any(k => k.Isvu_sifra == kolegij_isvu &&
        k.IspitiStudenta.Any(i => i.Ispit_id == ispit_id)),
    updateDef,
    updateOptions
);
```

Operacije *create* i *delete* su realizirane na sličan način pa ih nećemo navesti, za *create* se koristi funkcija `Builders<BsonDocument>.Update.AddToSet`, a za *delete* funkcija `Builders< BsonDocument >.Update.PullFilter`.

4.2 Zahtjevi i funkcionalnosti aplikacije

Glavni zahtjev je administracija ispita. Pod administraciju ispita spada sljedeće: kreiranje ispita, mijenjanje ispita, brisanje ispita, te prijava i odjava ispita. Posljednje dvije opcije omogućene su jednoj vrsti korisnika aplikacije - studentima - a ostale tri vrsti korisnika aplikacije - profesorima.

Profesorima su omogućene još neke funkcionalnosti: unos rezultata ispita za pojedinog studenta, unos rezultata ispita, unos ocjene kolegija, pregled kolegija koje drži, pregled ispita na svojim kolegijima.

Studentima je omogućen pregled kolegija te pripadnih ispita (uz spomenutu prijavu i odjavu ispita), te pregled težinskog i običnog prosjeka.

Prva zamisao aplikacije je bila kao na sljedećoj slici.

1. forma/prozor - Odabir studenta/nastavnika
- koristimo kolekciju Studenti/Nastavnici za provjeru

Ime

Prezime

STUDENT

NASTAVNIK

2. forma/prozor - Ispis naziva kolegija koje polaže/predaje
- koristimo kolekciju Studenti/Nastavnici za pronalazak kolegija (ovisno o odabranom na prethodnoj formi)

KOLEGIJ 1

KOLEGIJ 2

KOLEGIJ 3

3. forma/prozor - Za odabrani kolegij ispisujemo sve dostupne ispite
ZA STUDENTA - agregat za studenta i kolekcija Kolegiji

	Datum održavanja	Status	Rok za prijavu/odjavu	Rezultat
Ispit 1	15.06.2020.	prijavljen	12.06.2020.	<input type="button" value="ODJAVI"/>
Ispit 2	18.06.2020.	neprijavljen	15.06.2020.	<input type="button" value="PRIJAVI"/>
Ispit 3	10.05.2020.	položen		70%
KONAČNA OCJENA				<input type="text"/>

4. forma/prozor - ZA NASTAVNIKA

ISPIS SVIH ISPITA NA KOLEGIJU

ISPIS SVIH STUDENATA NA KOLEGIJU

5. forma/prozor - ZA NASTAVNIKA - Za odabrani kolegij ispisujemo sve dostupne ispite te omogućujemo dodavanje novog ispita - kolekcija Kolegiji

	Datum održavanja	
Ispit 1	17.06.2020.	<input type="button" value="PREGLED PO STUDENTIMA"/>
Ispit 2	20.06.2020.	<input type="button" value="PREGLED PO STUDENTIMA"/>
<input type="button" value="DODAJ NOVI ISPIT"/>		

6. forma/prozor - ZA NASTAVNIKA - Za odabrani kolegij ispisujemo sve studente koji su prijavili ispit (omogućiti pohranu nakon datuma održavanja) - ažuriramo rezultate u agregatu za studenta

JMBAG studenta	Ime i prezime studenta	Rezultat	Ažuriraj rezultat	
1191235678	Student 1	75%	<input type="text"/>	<input type="button" value="POHRANI"/>
1191236678	Student 2		<input type="text" value="80%"/>	<input type="button" value="POHRANI"/>

7. forma/prozor - ZA NASTAVNIKA - Za odabrani kolegij dodaj novi ispit - ažuriraj kolekcije Kolegiji i Studenti

UNESI POTREBNE PODATKE

Datum održavanja

Prijava ispita od

Prijava ispita do

Način polaganja

Trajanje

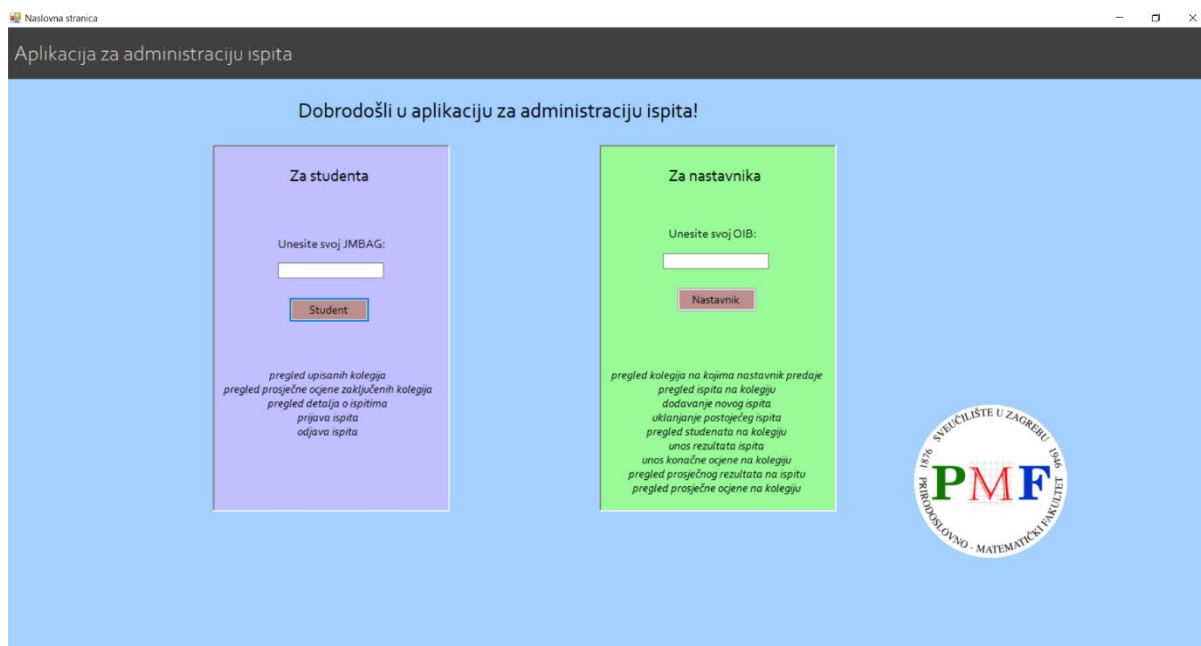
8. forma/prozor - ZA NASTAVNIKA - Za odabrani kolegij ispiši sve studente i njihove rezultate na svim ispitima te omogući upis ocjene za kolegij - agregati za studente

JMBAG studenta	Ime i prezime studenta	Rezultat na ispitu 1	Rezultat na ispitu 2	Unesi ocjenu	
1191235678	Student 1	75%	65%	<input type="text"/>	<input type="button" value="POHRANI"/>
1191236678	Student 2	80%	70%	<input type="text"/>	<input type="button" value="POHRANI"/>

Slika 4-1. Početna ideja formi aplikacije

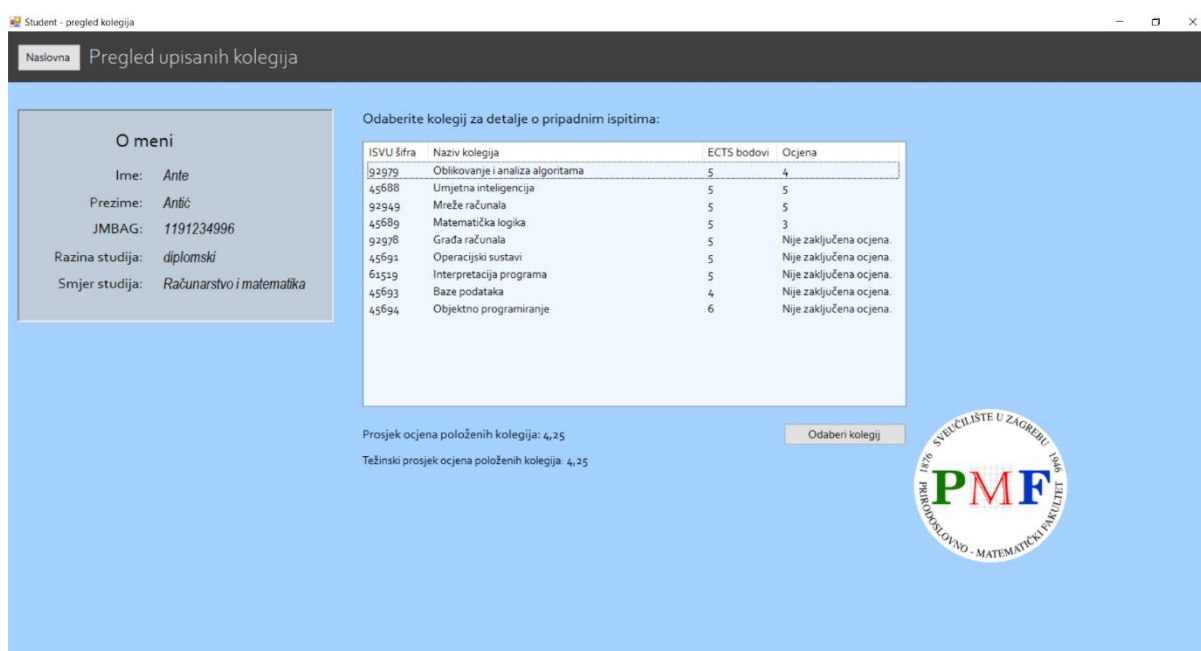
Slijedi objašnjenje formi uz naglašavanje značajnijih modifikacija koje su se dogodile nakon skice.

Prva forma predstavlja svojevrsnu *login* stranicu. Na prvoj formi umjesto imena i prezimena kao identifikacijske značajke ipak koristimo JMBAG za studenta i OIB za nastavnika, pošto je jedinstvenost podatka u bazi najvažnija karakteristika svake baze podataka. Provjerava se pravilan unos identifikacije (je li unesen broj) te se javlja upozorenje o nepravilnom unosu. Zatim se provjerava postoji li navedeni student/nastavnik u bazi, te se prelazi na formu 2 ako postoji, odnosno javlja se upozoravajuća poruka ako ne postoji te se korisnika moli da provjeri uneseni JMBAG/OIB. Sljedeći korak u razradi aplikacije bi bilo uvođenje lozinke, koje smo za ovaj projekt ostavili sa strane zbog zahtjevnosti generiranja hashiranih lozinke.



Slika 4-2. Forma 1

Druga forma služi kao svojevrsan profil ulogiranog korisnika, s lijeve strane se ispišu podaci o studentu/nastavniku te se desno ispišu kolegiji koje student pohađa, odnosno nastavnik predaje. Student može vidjeti svoj prosjek položenih kolegija, obični i težinski prosjek. Prosjeci su dobiveni M/R-om, postupak i faze su opisane u poglavlju 5.



Slika 4-3. Forma 2 – za studenta


Studentovim odabirom kolegija na drugoj formi, prelazi se na **treću formu** na kojoj se nalazi popis pripadnih ispita. Kraj svakog ispita postoji gumb za prijavu ili odjavu ispita, u slučaju da je zadovoljen kriterij statusa (ispit mora biti prijavljen da bi se mogao odjaviti, odnosno neprijavljen da bi se mogao prijaviti) i kriterij poštivanja roka za prijavu/odjavu. Ako nisu zadovoljeni kriteriji, ne postoji gumb. Student vidi i svoju konačnu ocjenu na kolegiju, ako je zaključena.

Student - pregled ispita na kolegiju

Naslovna Natrag Kolegij: *Oblikovanje i analiza algoritama*

ID ispita	Ispit	Datum održavanja	Status	Rok za prijavu	Rok za odjavu	Rezultat	Ako nije prošao rok, prikazat će se gumb za prijavu/odjavu
929791	Provjera usvojenog znanja prvog dijela gradiva.	2019-11-29	obavljen	2019-11-25	2019-11-27	70%	Ispit je već obavljen.
929792	Provjera usvojenog znanja drugog dijela gradiva.	2020-02-07	obavljen	2020-02-02	2020-02-04	80%	Ispit je već obavljen.
929793	Predaja i obrana domaće zadaće.	2020-02-14	obavljen	2020-02-09	2020-02-12	80%	Ispit je već obavljen.

Konačna ocjena: 4



Slika 4-4. Forma 3

Nastavnikovim odabirom kolegija na drugoj formi, prelazi se na **četvrtu formu** gdje su detalji o kolegiju te se nalaze dvije glavne opcije – pregled i administracija ispita te pregled i administracija rezultata studenata na ispitima na kolegiju.

Nastavnik - kolegij

Naslovna Natrag Kolegij: *Operacijski sustavi*

O kolegiju

Naziv: *Operacijski sustavi*

ISVU šifra: *45691*

ECTS bodovi: *5*

Semestar: *ljetni*

Izborna grupa: *ne*

Nositelj: *Leonardo Jelenković*


Za popis svih ispita i detalja o njima, za dodavanje novih ispita, uklanjanje postojećih, za pregled i unos rezultata ispita po studentima, za pregled prosječnog rezultata po ispitu, izaberite.

ISPITI NA KOLEGIJU

Za popis svih studenata na kolegiju te pregled njihovih rezultata, za unos konačne ocjene studenta na kolegiju te za pregled prosječne ocjene kolegija, izaberite.

STUDENTI NA KOLEGIJU

Nastavni sadržaji:
Slojevita hijerarhijska struktura operacijskih sustava. Prekidni način rada.



Slika 4-5. Forma 4

Odabirom prvog gumba na četvrtoj formi, otvara se **peta forma** s pregledom svih ispita i njihovih detalja na odabranom kolegiju. Na formi se nalaze mogućnosti za unos novog ispita u bazu, izmjenu postojećeg, brisanje postojećeg te pregled ispita po studentima gdje će se moći administrirati i rezultati ispita. Odabirom uklanjanja ispita, ispit se briše u bazi.

Nastavnik - pregled ispita na kolegiju


Naslovna Natrag Kolegij: *Operacijski sustavi*

Popis ispita:

ID ispita	Sadržaj ispita	Datum održavanja	Vrijeme održavanja	Trajanje	Prijava do	Odjava do	Način polaganja	Predavaonica
456911	Pismena provjera gradiva, sinkronizacija dretvi uključena.	2020-04-30	9:00	90 minuta	2020-04-25	2020-04-27	pismeno	201
456912	Pismena provjera cjelokupnog gradiva.	2020-06-26	9:00	90 minuta	2020-06-21	2020-06-24	pismeno	202
456913	Usmena provjera znanja. Obrana domaćih zadaća po potrebi.	2020-07-10	9:00	25 minuta	2020-07-01	2020-07-05	usmeno	A102

Za detalje i unos rezultata studenata, odaberite ispit i kliknite: PREGLED PO STUDENTIMA

DODAJ NOVI ISPIT
IZMIJENI ISPIT
UKLONI ISPIT



Slika 4-6. Forma 5

Odabirom pregleda ispita po studentima, prikazuje se **šesta forma** na kojoj nastavnik može unijeti novi rezultat ispita za studenta, odnosno promijeniti postojeći u slučaju pogrešnog unosa. Za to mora biti zadovoljen logički kriterij da se rezultat unosi nakon datuma provedbe ispita. Prikazan je i prosječni rezultat ispita za unesene rezultate, dobiven M/R-om (postupak i faze su opisane u poglavlju 5).

Nastavnik - pregled ispita po studentima


Naslovna Natrag Kolegij: *Operacijski sustavi*

Popis studenata prijavljenih za ispit: *Pismena provjera gradiva, sinkronizacija dretvi uključena. (456911)*

JMBAG	Ime i prezime	Rezultat
1191235679	Erna Emić	80%
1191235114	Iva Ivić	90%
1191238000	Mirko Minić	60%
1191234996	Ante Antić	80%
1191239996	Klara Klarić	60%

Za unos ili izmjenu rezultata studenta, odaberite studenta s popisa, unesite rezultat u donju kućicu te ga pohranite: POHRANI

Prosječni rezultat ispita za unesene rezultate: 74%



Slika 4-7. Forma 6

Odabirom unosa ili promjene ispita na petoj formi, otvara se **sedma forma** na kojoj nastavnik unosi nove vrijednosti te se time baza ažurira.

Nastavnik - dodaj novi ispit

Naslovna Natrag Kolegij: *label7*

Unesite potrebne podatke:

Datum održavanja: *datum oblika "YYYY-MM-DD"

Vrijeme održavanja: *vrijeme oblika "HH:MM"

Prijava ispita do: *datum oblika "YYYY-MM-DD"

Odjava do: *datum oblika "YYYY-MM-DD"


Način polaganja:

Sadržaj:

Trajanje (u minutama):

Predavaonica:

SPREMI



Slika 4-8. Forma 7

Odabirom drugog gumba na četvrtoj formi, otvara se **osma forma** s konačnim pregledom svih ispita po studentu, mogućnošću unosa/promjene konačne ocjene kolegija te pregledom prosječne ocjene kolegija dobivenom M/R-om (postupak i faze su opisane u poglavlju 5).

Nastavnik - pregled studenata na kolegiju

Naslovna Natrag Kolegij: *Oblikovanje i analiza algoritama*

Popis studenata na kolegiju:

JMBAG	Ime	prezime	Ispit1	Ispit2	Ispit3	Ocjena
1191235114	Iva	Ivić	50%	70%	70%	3
1191234996	Ante	Antić	70%	80%	80%	4
1191235996	Klara	Klarić	50%	60%	50%	3
1191235988	Josip	Josić	45%	55%	55%	2
1191235998	Ivana	Ivanić	50%	45%	60%	2

Prosječna ocjena kolegija: 2,8


Za unos ili izmjenu ocjene studenta, označite studenta na popisu te unesite ocjenu u donju kućicu:

Ime:

JMBAG:

Ocjena:

Unesi



Slika 4-9. Forma 8

Svim formama su dodani gumbi za „natrag“ i odlazak na „početnu stranicu“, za što vjerodostojniju realizaciju aplikacije.

5 Map/Reduce upiti

Pisanje Map/Reduce upita za analizu uspješnost na ispitima dio je zadatka. Nakon testiranja baze i izrade aplikacije napisani su M/R upiti za četiri vrste prosjeka: težinski prosjek studenta obzirom na sve položene kolegije, obični prosjek studenta obzirom na sve položene kolegije, prosjek svakog od ispita za kolegij te cjelokupni prosjek kolegija.

Sve M/R upite izvodimo nad kolekcijom *studenti* jer kolekcije *nastavnici* i *kolegiji* više sadrže općenite podatke, dok su u kolekciji *studenti* svi rezultati ispita i ocjene kolegija. Polja u agregatima kolekcije *studenti* čiji je jednostavan dohvat bio bitan za upite su *ocjena*, *rezultat* i *ects_bodovi* za samo računanje, te naravno polja za identifikaciju studenta, kolegija i ispita: *oib*, *isvu_sifra*, *ispit_id*.

5.1 Obični i težinski prosjek studenta

Na fakultetima je uobičajeno uz obični prosjek imati i težinski prosjek ocjena u ovisnosti o ECTS bodovima kolegija. Filtrirani dokument za ulaz u map fazu je zapravo jedan dokument – onaj za pripadnog studenta. Njegove ocjene u listi kolegija tretiramo kao „zasebne“ objekte koje treba mapirati i reducirati. U prosjeke ulaze samo zaključene ocjene pa pazimo da preskačemo *null* vrijednosti na početku mapiranja. Ukratko, u reduce i finalize fazama zbrajamo ocjene i dijelimo ih brojem ocjena za obični prosjek, odnosno zbrajamo umnoške ocjena s ECTS bodovima i dijelimo zbrojem ECTS bodova za težinski prosjek. Prikazujemo map-reduce proces za težinski prosjek studenta:

```
var map = function(){
  for (var idx = 0; idx < this.kolegiji.length; idx++){
    if( !this.kolegiji[idx].ocjena ) continue;

    var key = this.jmbag;

    var ocjena_int = parseInt(this.kolegiji[idx].ocjena);
    var value = { ocjena: ocjena_int, ects: this.kolegiji[idx].ects_bodovi };

    emit(key, value);
  }
};

var reduce = function(keyProdName, values){
  reducedVal = { rezultat: 0, ects: 0 };

  for(var i = 0; i < values.length; i++){
    reducedVal.ects += values[i].ects;
    reducedVal.rezultat += (values[i].ocjena * values[i].ects);
  }

  return reducedVal;
};

var finalize = function(key, reducedVal){
  reducedVal.avg = reducedVal.rezultat/reducedVal.ects;
  return reducedVal;
};

db.studenti.mapReduce(
  map,
  reduce,
  {
    out: { merge: "tezinskiProsjekStudenta" },
    finalize: finalize,
    query: { jmbag: 1191265765 }
  }
);
```

Slika 5-1. M/R upit za računanje težinskog prosjeka ocjena studenta s navedenim JMBAG-om

Umjesto direktnog unosa M/R upita u bazu, integrirali smo te upite unutar aplikacije te ih dinamički izvodimo u C#-u kada se određena forma s prosjekom treba prikazati. Rezultat M/R-a smo definirali da želimo primiti *inline*, a ne unijeti u bazu, pošto se dinamičkim unosima i promjenama baze u aplikaciji mijenjaju i prosječne vrijednosti te tako ne bismo imali ažurirane prosječne vrijednosti u svakom trenutku. Sve M/R upite na projektu smo prvo istestirali direktno u bazi, te ih onda integrirali u C#, tako da se svi M/R upiti zapravo izvršavaju na ovaj način. Pokazujemo kako izgleda dobivanje običnog prosjeka M/R-om u C#-u:

```
double prosjecnaOcjena = 0;

string mapf = @"
function(){
    for (var idx = 0; idx < this.kolegiji.length; idx++){
        if( !this.kolegiji[idx].ocjena ) continue;

        var key = this.jmbag;

        var ocjena_int = parseInt(this.kolegiji[idx].ocjena);
        var value = {count: 1, ocjena: ocjena_int};

        emit(key, value);
    }
}";

string reducef = @"
function(keyProdName, values){
    reducedVal = { count: 0, rezultat: 0 };

    for(var i = 0; i < values.length; i++){
        reducedVal.count += values[i].count;
        reducedVal.rezultat += values[i].ocjena;
    }

    return reducedVal;
}";

string finalizef = @"
function(key, reducedVal){
    reducedVal.avg = reducedVal.rezultat/reducedVal.count;
    return reducedVal;
}";

IMongoCollection<Student> collection = db.GetCollection<Student>("studenti");
BsonJavaScript map = new BsonJavaScript(mapf);
BsonJavaScript reduce = new BsonJavaScript(reducef);
FilterDefinitionBuilder<Student> filterBuilder = new FilterDefinitionBuilder<Student>();
FilterDefinition<Student> filter = filterBuilder.Where(s => s.Jmbag == jmbagStudenta);
MapReduceOptions<Student, BsonDocument> options = new MapReduceOptions<Student, BsonDocument>
{
    Filter = filter,
    MaxTime = TimeSpan.FromMinutes(1),
    Finalize = new BsonJavaScript(finalizef),
    OutputOptions = MapReduceOutputOptions.Inline,
    Verbose = true
};
try
{
    var results = collection.MapReduce(map, reduce, options).ToList();

    // Ako postoji avg ocjena (ako je zaključena za bar jedan kolegij, onda ce postojati avg)
    if (results.Count > 0)
    {
        foreach (BsonValue elem in results)
        {
            prosjecnaOcjena = double.Parse( elem["value"]["avg"].ToString() );
        }
    }
}
catch (Exception ex)
{
    Console.WriteLine($"Exception occurred {ex.Message}");
}
```

Slika 5-2. M/R upit za računanje običnog prosjeka ocjena studenta s navedenim JMBAG-om, u C#-u

Dobivene prosjeke ispisujemo na formi 2 (slika 4-3) jer tu student ima pregled svih upisanih kolegija.

5.2 Prosjek ispita

U prosjek ispita ubrajamo rezultate koji su uneseni, pazimo da preskačemo *null* vrijednosti. U fazi mapiranja pripremamo i šaljemo rezultat ispita i količinu 1, s ID-jem ispita kao ključem. Pošto je rezultat ispita u našoj bazi spremljen kao *string* u obliku „80%“, trebamo mu otkloniti zadnji znak „%“ (funkcija *slice*), te parsirati *string* „80“ u *INT* 80. U reduce i finalize fazama se već standardno zbrajaju rezultati i dijele s brojem rezultata.

```

var map = function(){
  for (var idx = 0; idx < this.kolegiji.length; idx++){
    for(var j = 0; j < this.kolegiji[idx].ispiti.length; j++){
      if( !this.kolegiji[idx].ispiti[j].rezultat ) continue;

      var key = this.kolegiji[idx].ispiti[j].ispit_id;

      var rezultat = this.kolegiji[idx].ispiti[j].rezultat;
      var rezultat_bez_postotka = rezultat.slice(0, -1);
      var rezultat_broj = parseInt(rezultat_bez_postotka);
      var value = { count: 1, rezultat: rezultat_broj };

      emit(key, value);
    }
  }
};

var reduce = function(keyProdName, values){
  reducedVal = { count: 0, rezultat: 0 };

  for(var i = 0; i < values.length; i++){
    reducedVal.count += values[i].count;
    reducedVal.rezultat += values[i].rezultat;
  }

  return reducedVal;
};

var finalize = function(key, reducedVal){
  reducedVal.avg = reducedVal.rezultat/reducedVal.count;
  return reducedVal;
};

db.studenti.mapReduce(
  map,
  reduce,
  {
    out: { merge: "prosjeciIspita" },
    finalize: finalize
  }
);

```

Slika 5-3. M/R upit za računanje prosječnog rezultata svakog ispita

Ovaj rezultat prikazujemo nastavniku u formi 6 (slika 4-7), pošto su tamo vidljivi rezultati svih studenata koji su pristupili ispitu.

5.3 Prosjek kolegija

U prosjek kolegija ubrajamo ocjene koje su zaključene pa pazimo da preskačemo *null* vrijednosti. U fazi mapiranja pripremamo i šaljemo ocjenu kolegija i količinu 1, s ISVU šifrom kao ključem. Budući da je ocjena kolegija u našoj bazi spremljena kao *string*, trebamo ju parsirati u *INT*. U reduce i finalize fazama se već standardno zbrajaju rezultati i dijele s brojem rezultata.


```

var map = function(){
  for (var idx = 0; idx < this.kolegiji.length; idx++){
    if( !this.kolegiji[idx].ocjena ) continue;

    var key = this.kolegiji[idx].isvu_sifra;

    var ocjena_int = parseInt(this.kolegiji[idx].ocjena);
    var value = {count: 1, ocjena: ocjena_int};

    emit(key, value);
  }
};

var reduce = function(keyProdName, values){
  reducedVal = { count: 0, rezultat: 0 };

  for(var i = 0; i < values.length; i++){
    reducedVal.count += values[i].count;
    reducedVal.rezultat += values[i].ocjena;
  }

  return reducedVal;
};

var finalize = function(key, reducedVal){
  reducedVal.avg = reducedVal.rezultat/reducedVal.count;
  return reducedVal;
};

db.studenti.mapReduce(
  map,
  reduce,
  {
    out: { merge: "prosjeciKolegija" },
    finalize: finalize
  }
);

```

Slika 5-4. M/R upit za računanje prosječne zaključne ocjene svih kolegija

Prosječnu ocjenu kolegija prikazujemo nastavniku u formi 8 (slika 4-9), jer su tamo vidljive sve ocjene studenata na kolegiju.

6 Poteškoće i rješavanje

U početku rada smo imali poteškoće kod modeliranja baze, odnosno određivanja agregata, kako je objašnjeno u poglavlju 3.3. Pazeći da ne zalutamo u relacijsko modeliranje, pokušavali smo modelirati što manje kolekcija. Naposljetku smo odabrali model za koji smatramo da je najpogodniji za aplikaciju, razlikuje se od relacijskog modela te u svakoj kolekciji imamo sve ono što nam je bitno za dohvat u radu aplikacije.

Kod povezivanja MongoDB i C# javio se problem prepoznavanja *null* vrijednosti kod polja koja su inače imala *integer* ili *double* vrijednosti, no primijetili smo da se taj problem ne događa ako polje ima *string* kao tip, stoga su ocjene i rezultati u kolekciji *studenti* pohranjeni kao *stringovi*, zato što su to jedina polja koja mogu poprimati vrijednost *null*.

Veći problem je bio shvatiti kako dohvaćati i baratati podacima nakon uspješne konekcije te kako provoditi CRUD operacije. Najveći problem je bio shvatiti kako dohvaćati elemente iz ugniježđenih

listi dokumenata. Trebalo je shvatiti kako funkcioniraju Builder objekti kojima se služi Mongo.Client. Jedna od prepreka jest to što je na Internetu malo rješenja za noviju verziju MongoDB Drivera koju smo koristili. Naime, rješenja se većinom odnose na stariju verziju te ona više ne funkcioniraju.

Jedan od problema je bio i shvatiti kako M/R upiti funkcioniraju. Dolazilo je do nepotrebnih grešaka s rijetkim odgovorima na Internetu, kao na primjer da će se M/R izvrstjeti, ali vratiti prazan rezultat.

7 Ideje za poboljšanja

1. Dodati autentifikaciju studenta, odnosno nastavnika.
2. Dodati formu za studenta da vidi koji je način polaganja kolegija (koliki postotak konačne ocjene je ostvariv na kojem ispitu), te trajanje i predavaonicu u kojoj se piše neki ispit (podaci su generirani i već dostupni u kolekciji *kolegiji* za svaki ispit, no nismo imali vremena za implementaciju njihova prikaza u aplikaciji).
3. Razlikovati ispis kolegija po semestrima i godinama.

8 Zaključak

Napravili smo dokumentsku bazu podataka koristeći MongoDB za administraciju ispita pri čemu smo se „istrenirali“ u modeliranju NoSQL baze tako da možemo pohraniti veći broj informacija. U isto vrijeme cilj nam je bio da nam te informacije budu istodobno jednostavno i brzo dohvatljive unutar formi aplikacije, ali da pazimo da ne zalutamo u relacijsko modeliranje. Kako smo odlučili koristiti C# za izradu aplikacije, bilo je važno naučiti kako povezati MongoDB i C#, te također obratiti pažnju na tip podataka koja imaju polja u bazi, i bitno primijetiti kako se MongoDB ponaša s *null* vrijednostima. Naučili smo da je s dokumentskom NoSQL bazom lako baratati u objektno-orijentiranom jeziku, zato što svaki dokument može biti realiziran klasom, odnosno objektom. Također smo se uvjerali koliko je bitno testirati aplikaciju na malom broju podataka zbog čestih izmjena nailaskom na različite prepreke, s obzirom na to da smo koristili nove alate za koje ne znamo unaprijed kako se ponašaju zajedno.

Reference

- [1] <https://www.math.pmf.unizg.hr/>
- [2] <https://stackoverflow.com/>
- [3] <https://docs.mongodb.com/>