

SVEUČILIŠTE U SPLITU  
**PRIRODOSLOVNO MATEMATIČKI FAKULTET**

Dokumentacija projekta

**Baza podataka – Korisničke playliste putem  
MongoDB-a**

Josipa Mrša

Split, kolovoz, 2020.

## SADRŽAJ

1. Uvod.....	2
2. Značajke MongoDB baze podataka .....	3
3. Instalacija MongoDB okruženja .....	5
4. Baza podataka .....	6
5. Kreiranje i uništavanje baze podataka .....	7
6. Stvaranje i upravljanje kolekcijama u bazi podataka.....	8
7. Upravljanje podacima .....	9
7.1. Učitavanje podataka u bazu .....	10
7.2. Ažuriranje podataka i dokumenata .....	10
7.3. Brisanje podataka i dokumenata .....	12
8. Upiti na bazu .....	13
8.1. Metode pronalaska dokumenata .....	13
8.2. Ključne riječi .....	14
8.3. Indeksiranje .....	16
8.4. Agregacija.....	16
Popis slika .....	17

## 1. UVOD

Za izradu ovog projekta, uz *MongoDB* korišten je i programski jezik *C#*, a projekt je izveden u *Visual Studio 2019*. Tema projekta je izrada baze podataka korisničkih playlista.

*MongoDB* je tip *NoSQL* baze podataka koji se koristi za skladištenje velike količine podataka. Umjesto korištenja tradicionalnih relacijskih baza podataka, *MongoDB* se oslanja na kolekcije i datoteke kao način spremanja podataka, gdje se dokumenti sastoje od parova ključ-vrijednost te predstavljaju osnovnu jedinicu podataka u *MongoDB*-u. Kolekcije sadrže skupove dokumenata te funkcioniraju slično kao i relacijske tablice u bazi podataka.

## 2. ZNAČAJKE MONGODB BAZE PODATAKA

*MongoDB* baze podataka označava višeplatformnost, visoke performanse, dostupnost te laku skalabilnost. Mnoge kompanije oko svijeta koriste definirane clustere, od kojih neki broje i preko 100 čvorova i milijune dokumenata unutar baze podataka.

Svaki dokument može biti drukčiji i sa varijabilnim brojem polja, a veličina i sadržaj svakog dokumenta isto tako mogu biti različiti. Sama struktura dokumenta je više prilagođena programerima, a nalikuje na konstrukciju klasa i objekata. Polja se mogu stvarati bez prethodno zacrtane sheme, a model podataka unutar *MongoDB*-a omogućava prezentaciju hijerarhije relacija, kako bi se lakše mogli spremati nizovi i druge kompleksne kolekcije.

Ključne razlike među *MongoDB* i *RDBMS* su slijedeće:

RDBMS	MongoDB	Razlike
Tablice	Kolekcije	U RDBMS-u, tablica se sastoji od stupaca i redova u koje se spremaju podaci, dok se u MongoDB-u za to koriste kolekcije. One sadržavaju dokumente sa poljima, u obliku ključ-vrijednost parova.
Redovi	Dokumenti	U RDBMS-u red se koristi kao jedna stavka strukturiranih podataka u tablici, dok se u MongoDB-u podaci spremaju u dokumente.
Stupci	Polja	U RDBMS-u stupci su set podataka, a u MongoDB-u su to polja.
Spajanja	Embedded dokumenti	U RDBMS-u su podaci nekad raspršeni preko raznih tablica, pa kako bi se dao potpun pregled svih podataka, potrebna su spajanja (Join) tablica. U MongoDB-u se najčešće podaci spremaju u jednu kolekciju, ali su odvojeni korištenjem „embedded“ (ugrađenih) dokumenata, pa koncept spajanja ovdje ne postoji.

Isto tako, kod relacijskih baza podataka je potrebno primijeniti integritet podataka, što nije eksplicitno traženo kod *MongoDB*-a, te *RDBMS* zahtjeva da se podaci prvo normaliziraju da bi se spriječilo napuštene zapise i duplikate zapisa. Normalizacijom podataka se povećava potreba za više tablica, više spajanja, te na kraju više indeksa i ključeva, što može znatno utjecati na performancu same aplikacije. *MongoDB* s druge strane je fleksibilan i ne zahtjeva normalizaciju podataka. Stoga je odlična opcija za korištenje u situacijama gdje se barata s *Big Data* setovima, kod upravljanja sadržajem i isporukom, mobilnom i društvenom infrastrukturom, korisničkim podacima i kod *Data Hub*-ova.

### 3. INSTALACIJA MONGODB OKRUŽENJA

Da bi se uopće moglo kreirati i raditi sa *MongoDB* bazama podataka, potrebno je najprije instalirati *MongoDB Community* okruženje, koje se besplatno može preuzeti sa službene web-stranice MongoDB. Nakon instalacije okruženja, potrebno je unutar *Command Prompta* definirati putanju na kojoj će *MongoDB* spremati sve kreirane baze i promjene – potrebno je na **C:** disku izraditi direktorij data, unutar kojeg treba postojati folder **db**. Cijela putanja može se napraviti na slijedeći način:

```
C:\>md data  
C:\>md data\db
```

Potom je potrebno povezati putanju gdje će *MongoDB* spremati podatke i baze. Naredba je slijedeća:

```
C:\Users\XYZ>d:cd C:\Program Files\MongoDB\Server\4.2\bin  
C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe --dbpath "C:\data"
```

Ako se unutar *Command Prompta* pokazuje „*Waiting for connections*“, postavljanje je uspješno, te je proces *MongoDB* uspješno pokrenut. Zadnja naredba koja nam treba je slijedeća:

```
C:\Program Files\MongoDB\Server\4.2\bin>mongo.exe
```

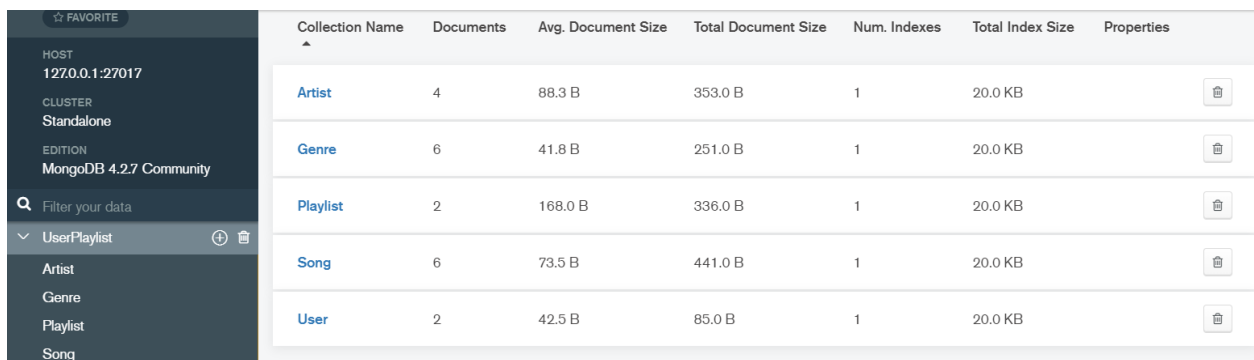
Tu se potom dobiva adresa na koju se može spojiti klijent, odnosno koja će se koristiti za implementaciju *C#* klijenta u ovom projektu. Kako bi se to moglo implementirati, *C#* jezik mora biti instaliran na računalu, te treba postojati adekvatno okruženje za razvijanje *C#* aplikacija. U ovom projektu korišten je *Visual Studio 2019*. Kako bi se moglo upravljati *MongoDB* bazom podataka u *C#* klijentu, potrebno je instalirati prikladne pakete koji omogućuju korištenje *MongoDB* baze podataka u *C#* aplikacijama: **MongoDB.Driver** i **MongoDB.Bson**. Ove pakete moguće je instalirati preko *NuGet Package Managera* unutar VS.

```
using MongoDB.Driver;  
using MongoDB.Bson;
```

*Slika 1 Uključeni paketi*

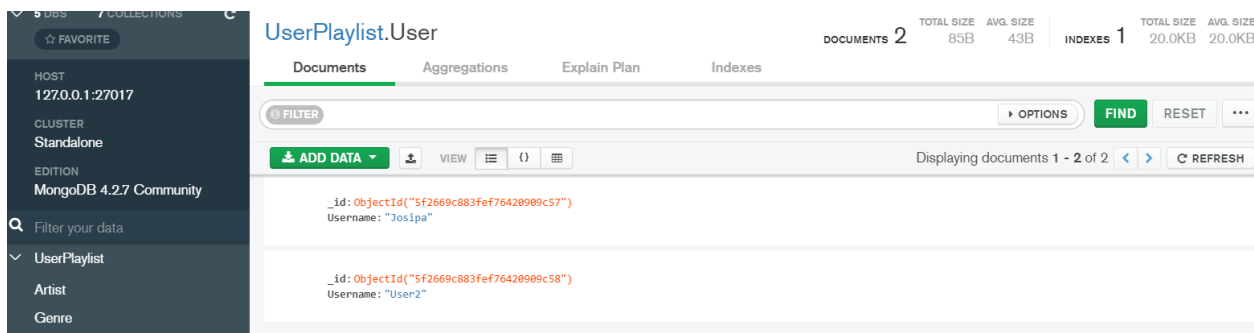
## 4. BAZA PODATAKA

Baza podataka koristi se kao fizički spremnik za zbirke. Svaka baza podataka dobiva vlastiti skup datoteka na datotečnom sustavu, a jedan *MongoDB* poslužitelj obično sadrži više baza podataka.



Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
Artist	4	88.3 B	353.0 B	1	20.0 KB	
Genre	6	41.8 B	251.0 B	1	20.0 KB	
Playlist	2	168.0 B	336.0 B	1	20.0 KB	
Song	6	73.5 B	441.0 B	1	20.0 KB	
User	2	42.5 B	85.0 B	1	20.0 KB	

Slika 2 Struktura baze podataka



UserPlaylist.User

DOCUMENTS 2 TOTAL SIZE 85B AVG. SIZE 43B INDEXES 1 TOTAL SIZE 20.0KB AVG. SIZE 20.0KB

Documents Aggregations Explain Plan Indexes

FILTER OPTIONS FIND RESET ...

ADD DATA VIEW

Displaying documents 1 - 2 of 2 REFRESH

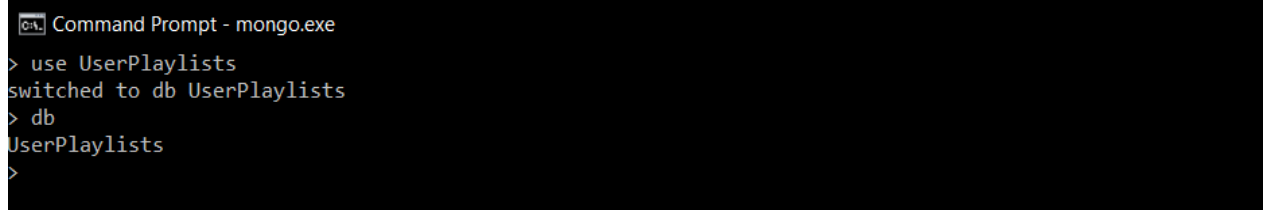
```

{ "_id": ObjectId("5f2669c883fef76428909c57"), "Username": "Josipa" }
{ "_id": ObjectId("5f2669c883fef76428909c58"), "Username": "User2" }
    
```

Slika 3 Sadržaj jedne od kolekcija u bazi podataka

## 5. KREIRANJE I UNIŠTAVANJE BAZE PODATAKA

Bazu podataka kreira se korištenjem naredbe `use`:

A screenshot of a Windows Command Prompt window titled "Command Prompt - mongo.exe". The text inside shows the following commands and their outputs: the prompt ">" followed by "use UserPlaylists", the output "switched to db UserPlaylists", the prompt ">" followed by "db", and the output "UserPlaylists". The prompt ">" is shown again at the end.

```
Command Prompt - mongo.exe
> use UserPlaylists
switched to db UserPlaylists
> db
UserPlaylists
>
```

*Slika 4 Stvaranje baze podataka*

Dakle, naredbom `use` stvara se baza `UserPlaylists`, a naredbom `db` se baza podataka aktivira. Ako ova baza podataka već postoji, vraća se postojeća baza podataka. Isto tako, naredbom `db` može se vidjeti koja je baza trenutno aktivna. Naredbom `show dbs` možemo vidjeti popis svih stvorenih baza podataka.

Spajanje na stvorenu bazu podataka u *C#* klijentu može se odraditi na slijedeći način:

A screenshot of a code editor showing a C# method named DatabaseConnect. The code includes a comment in green, and two lines of code that create a MongoClient and get a database instance. The code is as follows:

```
public static void DatabaseConnect()
{
    // Povezivanje na bazu podataka

    MongoClient mgClient = new MongoClient("mongodb://localhost:27017");
    userPlaylistDatabase = mgClient.GetDatabase("UserPlaylist");
}
```

*Slika 5 Spajanje na bazu podataka iz klijenta*

Instancira se novi `MongoClient` preko adrese dobivene u samoj *MongoDB* aplikaciji. Potom se poziva prethodno stvorena baza podataka.

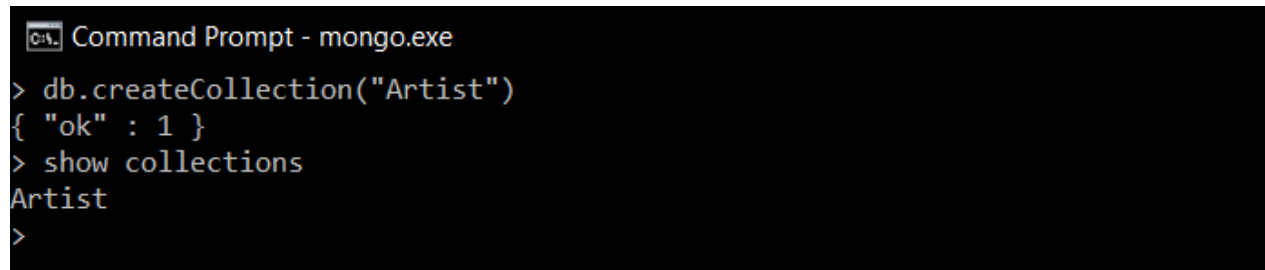
Za izbrisati cijelu bazu podataka, može se koristiti naredba `db.dropDatabase()`. Prije korištenja naredbe treba se uvjeriti da je upravo trenutna aktivna baza ona koja se želi obrisati.



## 6. STVARANJE I UPRAVLJANJE KOLEKCIJAMA U BAZI PODATAKA

Zbirkom se smatra skup *MongoDB* dokumenata, te predstavlja ono što bi klasična tablica predstavljala u *RDBMS*-u. Shema agnostik su, odnosno nemaju strogu shemu, te dokumenti, koji se smatraju ekvivalentom redu u *RDBMS*-u, mogu imati različita polja. Svi dokumenti u zbirci imaju sličnu ili povezanu svrhu.

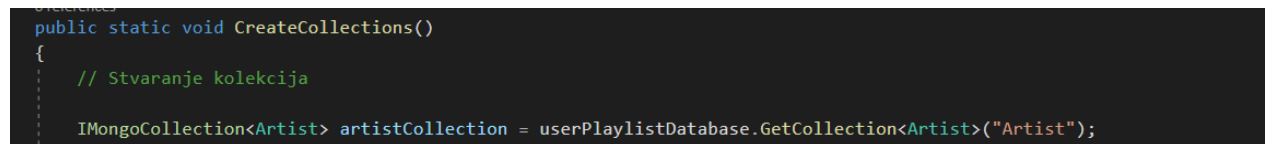
Naredba za kreiranje kolekcije unutar neke baze podataka u *MongoDB*-u je `db.createCollection()`, kojom se stvara zbirka u aktivnoj bazi podataka:



```
Command Prompt - mongo.exe
> db.createCollection("Artist")
{ "ok" : 1 }
> show collections
Artist
>
```

Slika 6 Stvaranje kolekcije

Unutar *C#* klijenta također je moguće stvarati kolekcije. To se može ostvariti na sljedeći način:



```
public static void CreateCollections()
{
    // Stvaranje kolekcija
    IMongoCollection<Artist> artistCollection = userPlaylistDatabase.GetCollection<Artist>("Artist");
}
```

Slika 7 Stvaranje kolekcije u *C#* klijentu

Putem klase `MongoCollection` stvara se objekt koji sprema objekte tipa `Artist`, i preko poveznice na bazu podataka s korisničkim playlistama dohvati se kolekcija `Artist`. Ako ona ne postoji, stvorit će se ta kolekcija unutar baze podataka.

Ako se neku kolekciju želi obrisati, koristi se naredba `db.ImeKolekcije.drop()`.

## 7. UPRAVLJANJE PODACIMA

U ovom projektu podaci su dohvaćeni iz prikladnih *JSON* datoteka, te pretvoreni u objekte prema zadanoj shemi. Postoji pet tipova objekata:

- **User** – podaci o korisnicima (korisničko ime)
- **Genre** – podaci o žanrovima (ime žanra)
- **Artist** – podaci o izvođačima (ime izvođača i lista žanrova njegovih skladbi)
- **Song** – podaci o pjesmama (ime pjesme i trajanje pjesme)
- **Playlist** – podaci o korisničkim listama (ime, ukupno trajanje playliste, broj skladbi, broj dijeljenja od drugih korisnika, lista žanrova)

Uz nabrojena polja za svaki objekt, svaki od njih ima i polje `_id`, koje će *MongoDB* kasnije koristiti za dodjeljivanje jedinstvenog ID-a, odnosno reference. Podaci su dohvaćeni na slijedeći način:

```
public static void GenerateData()
{
    // Generiranje podataka iz JSON-a u objekt

    var master = File.ReadAllText("./JSON/Master.json");
    UserPlaylistData = JsonConvert.DeserializeObject<UserPlaylistData>(master);

    Artists = UserPlaylistData.Artists;
    Genres = UserPlaylistData.Genres;
    Playlists = UserPlaylistData.Playlists;
    Songs = UserPlaylistData.Songs;
    Users = UserPlaylistData.Users;
}
```

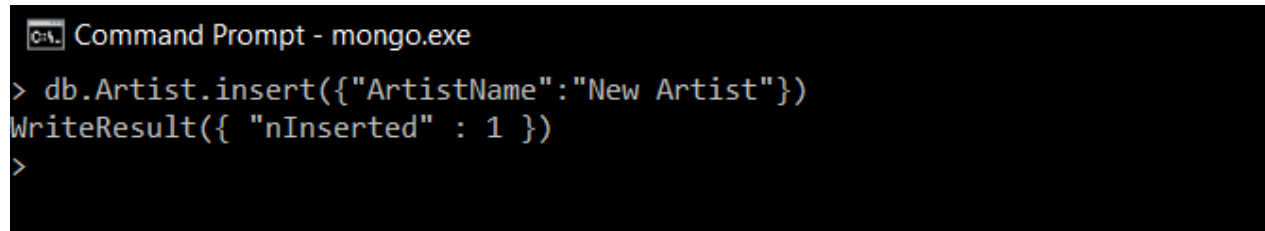
Slika 8 Generiranje podataka

*JSON* datoteke nalaze se u `bin/Debug` folderu ovog projekta. Za pretvorbu *JSON*-a u *C#* objekt korištena je biblioteka `Newtonsoft.Json`, koju se također može instalirati preko *NuGet Package Managera*. Dohvati se sadržaj direktorija u kojem se nalaze *JSON* datoteke, te se prikladna datoteka pretvori u prikladan objekt.

## 7.1. Učitavanje podataka u bazu

Podatke je moguće učitati u *MongoDB* aplikaciji putem naredbe `db.ImeKolekcije.insert()`.

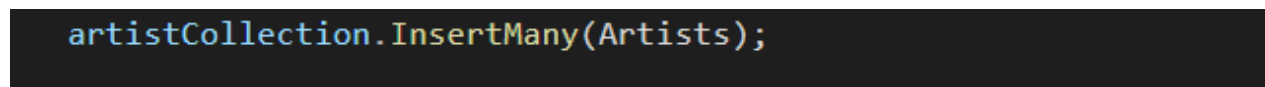
Naredba prima parametar objekta u *JSON* obliku, pa bi umetanje zapisa u bazu izgledalo ovako:



```
C:\> Command Prompt - mongo.exe
> db.Artist.insert({"ArtistName":"New Artist"})
WriteResult({ "nInserted" : 1 })
>
```

Slika 9 Umetanje podataka

Umetanje preko C# klijenta izgleda ovako:



```
artistCollection.InsertMany(Artists);
```

Slika 10 Umetanje podataka u C# klijentu

S naredbom `InsertMany()` unose se svi zapisi iz neke kolekcije. Isto tako postoji naredba `InsertOne()` gdje se u kolekciju unosi samo jedan zapis.

## 7.2. Ažuriranje podataka i dokumenata

Za ažuriranje podataka i dokumenata u zbirci, u *MongoDB*-u koriste se metode `update()` i `save()`. Metoda `update()` ažurira vrijednosti u postojećim dokumentima, a `save()` zamjenjuje postojeći dokument sa dokumentom navedenim kao parametar metode. Kod metode `update()` postoji parametar `multi`, koji označava da se više redova treba ažurirati sa navedenom vrijednošću. Na kraj reda potrebno je dodati `{ „multi“ : „true“ }` kako bi se više redova ažuriralo.

```

> db.Artist.find()
{ "_id" : ObjectId("5f2baff683fef74100c3bad4"), "ArtistName" : "Mark Markee", "Genres" : [ "Rock", "Pop" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad5"), "ArtistName" : "Peter Parker", "Genres" : [ "Rock", "Folk" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad6"), "ArtistName" : "Andrea Andrei", "Genres" : [ "Folk", "Blues" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad7"), "ArtistName" : "George Georg", "Genres" : [ "Jazz", "Blues" ] }
> db.Artist.update({"ArtistName" : "Mark Markee"}, {$set: {"ArtistName": "Derek Derrick"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Artist.find()
{ "_id" : ObjectId("5f2baff683fef74100c3bad4"), "ArtistName" : "Derek Derrick", "Genres" : [ "Rock", "Pop" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad5"), "ArtistName" : "Peter Parker", "Genres" : [ "Rock", "Folk" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad6"), "ArtistName" : "Andrea Andrei", "Genres" : [ "Folk", "Blues" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad7"), "ArtistName" : "George Georg", "Genres" : [ "Jazz", "Blues" ] }
>

```

Slika 11 Ažuriranje vrijednosti u postojećim dokumentima

```

> db.Artist.find()
{ "_id" : ObjectId("5f2baff683fef74100c3bad4"), "ArtistName" : "Derek Derrick", "Genres" : [ "Rock", "Pop" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad5"), "ArtistName" : "Peter Parker", "Genres" : [ "Rock", "Folk" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad6"), "ArtistName" : "Andrea Andrei", "Genres" : [ "Folk", "Blues" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad7"), "ArtistName" : "George Georg", "Genres" : [ "Jazz", "Blues" ] }
{ "_id" : ObjectId("5f2bb2b171879715a9de4d44"), "ArtistName" : "New Artist", "Genres" : [ "Rock", "Folk", "Blues" ] }
> db.Artist.save({"_id":ObjectId("5f2bb2b171879715a9de4d44"), "ArtistName":"Richard Richardson", "Genres":["Rock", "Folk", "Blues"]})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Artist.find()
{ "_id" : ObjectId("5f2baff683fef74100c3bad4"), "ArtistName" : "Derek Derrick", "Genres" : [ "Rock", "Pop" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad5"), "ArtistName" : "Peter Parker", "Genres" : [ "Rock", "Folk" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad6"), "ArtistName" : "Andrea Andrei", "Genres" : [ "Folk", "Blues" ] }
{ "_id" : ObjectId("5f2baff683fef74100c3bad7"), "ArtistName" : "George Georg", "Genres" : [ "Jazz", "Blues" ] }
{ "_id" : ObjectId("5f2bb2b171879715a9de4d44"), "ArtistName" : "Richard Richardson", "Genres" : [ "Rock", "Folk", "Blues" ] }
>

```

Slika 12 Zamjena dokumenata

Kod C# klijentske aplikacije koristi se **Builders<T>** klasa sa raznim pomoćnim metodama za rad s modelima, te svojstvo **Update** u kombinaciji sa naredbom **Set()** preko kojih se definira nova vrijednost za atribut iz postojećeg dokumenta. Potom preko naredbe **FindOneAndUpdate<T>()** ažuriramo jedan dokument sa novom vrijednošću.

```

// Ažuriranje podataka

IMongoCollection<User> userCollection = userPlaylistDatabase.GetCollection<User>("User");

var update = Builders<User>.Update.Set(x => x.Username, "Josipa");
var rezultat = userCollection.FindOneAndUpdate<User>(x => x.Username == "User1", update);

```

Slika 13 Ažuriranje podataka u C# klijentu

### 7.3. Brisanje podataka i dokumenata

Za brisanje dokumenata u *MongoDB* koristi se metoda `remove()`, koja koristi dva parametra. Prvi označava kriterij brisanja, a drugi je zastavica (`justOne`) koji označava koliko dokumenata će se izbrisati. To znači da ako zastavica nije postavljena na `true`, brišu se svi dokumenti koji odgovaraju tom kriteriju; u protivnom se briše samo jedan.

U C# klijentu brisanje dokumenata se odrađuje na slijedeći način:

```
// Brisanje zapisa  
  
IMongoCollection<Genre> genreCollection = userPlaylistDatabase.GetCollection<Genre>("Genre");  
genreCollection.DeleteOne<Genre>(x => x.GenreName == "Folk");
```

*Slika 14 Brisanje dokumenata preko C# klijenta*

Naredbom `DeleteOne()` briše se jedan zapis koji odgovara navedenom kriteriju.

## 8. UPITI NA BAZU

Postoji nekoliko načina izrade upita za dokumente u *MongoDB* zbirci.

### 8.1. Metode pronalaska dokumenata

Za pronalazak zapisa koriste se metode `find()` i `findOne()`. Metoda `find()` izlistava sve dokumente iz *MongoDB* zbirke na nestrukturiran način, dok metoda `findOne()` vraća samo jedan dokument kao rezultat. Da bi rezultati metode `find()` bili pregledniji, može se koristiti metoda `pretty()`, koja formatira dokument na ekranu na čitljiviji način.

```
> db.Song.find().pretty()
{
  "_id" : ObjectId("5f2baff883fef74100c3bae0"),
  "SongName" : "Happy Song",
  "SongDuration" : "00:05:10"
}
{
  "_id" : ObjectId("5f2baff883fef74100c3bae1"),
  "SongName" : "Bad Song",
  "SongDuration" : "00:05:10"
}
{
  "_id" : ObjectId("5f2baff883fef74100c3bae2"),
  "SongName" : "Lazy Song",
  "SongDuration" : "00:05:10"
}
```

Slika 15 Ispis svih zapisa u kolekciji

```
> db.Song.findOne()
{
  "_id" : ObjectId("5f2baff883fef74100c3bae0"),
  "SongName" : "Happy Song",
  "SongDuration" : "00:05:10"
}
>
```

Slika 16 Ispis jednog zapisa iz kolekcije

U obje metode može se proslijediti kriterij pretrage, ili ostaviti prazno da se dobiju bilo koji zapisi.

```
> db.Song.find({"SongDuration":"00:05:10"}).pretty()
{
  "_id" : ObjectId("5f2baff883fef74100c3bae0"),
  "SongName" : "Happy Song",
  "SongDuration" : "00:05:10"
}
{
  "_id" : ObjectId("5f2baff883fef74100c3bae1"),
  "SongName" : "Bad Song",
  "SongDuration" : "00:05:10"
}
{
  "_id" : ObjectId("5f2baff883fef74100c3bae2"),
  "SongName" : "Lazy Song",
  "SongDuration" : "00:05:10"
}
>
```

Slika 17 Prosljeđivanje kriterija

U C# klijentu zapis može se pronaći na sljedeći način:

```
// Nadi izvođača po imenu
Console.WriteLine("Svi izvođači imena \"Mark Markee\": ");

artistCollection.Find(x => x.ArtistName == "Mark Markee")
    .ForEachAsync<Artist>(x => Console.WriteLine("{0} -> {1}", x.ArtistName, string.Join(", ", x.Genres)));
Thread.Sleep(1000);

Console.WriteLine();
```

Slika 18 Pronalazak zapisa

Metodom `Find()` pronalaze se zapisi koji odgovaraju navedenom kriteriju, a onda za svaki pronađeni zapis ispisivaju se željene vrijednosti.

## 8.2. Ključne riječi

Ako se u metodi `find()` proslijedi više ključeva odvojenih sa zarezom, *MongoDB* to tretira kao operaciju **AND**. Da bi se ispisao željeni dokument, oba ili svi navedeni kriteriji (ključevi) moraju

biti zadovoljeni (odnosno, postojati). Ovo se može postići i korištenjem ključne riječi `$and`. Ako se treba ispisati dokument koji ispunjava bar jedan uvjet, koristi se ključna riječ `$or`.

```
> db.Playlist.find({$and:[{"PerformanceCount":4},{"ShareCount":12}]}).pretty()
{
  "_id" : ObjectId("5f2baff783fef74100c3bade"),
  "PlaylistName" : "Playlist1",
  "PlaylistDuration" : "00:15:30",
  "PerformanceCount" : 4,
  "ShareCount" : 12,
  "Genres" : [
    "Rock",
    "Pop",
    "Folk"
  ]
}
```

Slika 19 Ispis dokumenata koji zadovoljavaju sve kriterije

```
> db.Playlist.find({$or:[{"PerformanceCount":4},{"ShareCount":15}]}).pretty()
{
  "_id" : ObjectId("5f2baff783fef74100c3bade"),
  "PlaylistName" : "Playlist1",
  "PlaylistDuration" : "00:15:30",
  "PerformanceCount" : 4,
  "ShareCount" : 12,
  "Genres" : [
    "Rock",
    "Pop",
    "Folk"
  ]
}
{
  "_id" : ObjectId("5f2baff783fef74100c3badf"),
  "PlaylistName" : "Playlist2",
  "PlaylistDuration" : "00:12:40",
  "PerformanceCount" : 3,
  "ShareCount" : 15,
  "Genres" : [
    "Folk",
    "Jazz",
    "Blues"
  ]
}
```

Slika 20 Ispis dokumenata koji zadovoljavaju bar jedan od navedenih kriterija



U C# ovakav kriterij se može definirati i na slijedeći način:

```
Console.WriteLine("Sve playliste sa brojem skladbi većim od 3 i brojem dijeljenja manjim od 15: ");

playlistCollection.Find(x => x.PerformanceCount > 3 && x.ShareCount < 15)
    .ForEachAsync<Playlist>(x => Console.WriteLine("{0} -> {1}", x.PlaylistName, x.PlaylistDuration));
Thread.Sleep(1000);

Console.WriteLine();
```

Slika 21 Ispis pomoću kriterija u C# klijentu

### 8.3. Indeksiranje

Indeksi služe za učinkovito izvršavanje upita. Da nema indeksa, *MongoDB* bi trebao skenirati svaki dokument zbirke za pronalazak dokumenata koji odgovaraju izrazu upita, što je vrlo neučinkovito i zahtjevno, pošto se može raditi o velikom volumenu podataka. Indeksi su posebne strukture podataka koje pohranjuju mali dio skupa podataka u obliku koji je jednostavan za prijelaz. Pohranjuje vrijednost određenog polja ili skupa polja, koji su poredani prema vrijednosti polja navedenih u indeksu. Za izradu indeksa koristi se metoda `ensureIndex()`.

```
> db.Artist.ensureIndex({"ArtistName":1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
```

Slika 22 Indeksiranje

### 8.4. Agregacija

Ova operacija obrađuje zapise podataka te vraća izračunate rezultate. Skupne operacije grupiraju vrijednosti iz više dokumenata skupa te mogu izvesti različite operacije na grupiranim podacima da bi se vratio jedan rezultat.

```
> db.Song.aggregate([{$group:{_id:"SongDuration",num:{$sum:1}}}]
{ "_id" : "SongDuration", "num" : 6 }
>
```

Slika 23 Agregacija

## POPIS SLIKA

Slika 1 Uključeni paketi .....	5
Slika 2 Struktura baze podataka .....	6
Slika 3 Sadržaj jedne od kolekcija u bazi podataka .....	6
Slika 4 Stvaranje baze podataka.....	7
Slika 5 Spajanje na bazu podataka iz klijenta .....	7
Slika 6 Stvaranje kolekcije .....	8
Slika 7 Stvaranje kolekcije u C# klijentu .....	8
Slika 8 Generiranje podataka .....	9
Slika 9 Umetanje podataka.....	10
Slika 10 Umetanje podataka u C# klijentu .....	10
Slika 11 Ažuriranje vrijednosti u postojećim dokumentima .....	11
Slika 12 Zamjena dokumenata .....	11
Slika 13 Ažuriranje podataka u C# klijentu .....	11
Slika 14 Brisanje dokumenata preko C# klijenta .....	12
Slika 15 Ispis svih zapisa u kolekciji .....	13
Slika 16 Ispis jednog zapisa iz kolekcije.....	13
Slika 17 Prosljeđivanje kriterija .....	14
Slika 18 Pronalazak zapisa .....	14
Slika 19 Ispis dokumenata koji zadovoljavaju sve kriterije .....	15
Slika 20 Ispis dokumenata koji zadovoljavaju bar jedan od navedenih kriterija .....	15
Slika 21 Ispis pomoću kriterija u C# klijentu.....	16
Slika 22 Indeksiranje .....	16
Slika 23 Agregacija .....	16

