OBLIKOVANJE BAZA PODATAKA

Vježbe 14

Document baze podataka (MongoDB)

- MongoDB je razvijen 2007. godine i tada je odlučeno da će biti otvorenog koda
- Primarno je osmišljen za spremanje podataka s web aplikacija
- MongoDB je *platform native* baza podataka - radi na različitim platformama (Linux, Mac OS, Windows, Solaris, …)
- Koristi horizontalno skaliranje - uvijek ima više poslužitelja, a ne samo jedan što je primarna ideja relacijskih baza podataka
- Dodavanjem većeg broja poslužitelja povećavamo učinkovitost baze podataka

Koncept *document* baze podataka (MongoDB)

- Kao što relacijske baze podataka imaju tablice, stupce i retke, tako i MongoDB ima neke koncepte pomoću kojih stvara bazu podataka
  - **Dokument** – ekvivalent retku u relacijskim bazama
  - **Kolekcija** – ekvivalent tablici u relacijskim bazama
  - **Baza podataka** – sadrži kolekcije dokumenata
  - **Ključ** – svaki dokument ima svoj jedinstveni ključ

Query API sintaksa

- Connection string za bazu podataka:

```
mongodb://[korisničko_ime:lozinka@]hostname[:port]
```

- Popis svih baza podataka:

```
show dbs
```

- Popis svih kolekcija:

```
show collections
```

- Korištenje baze podataka:

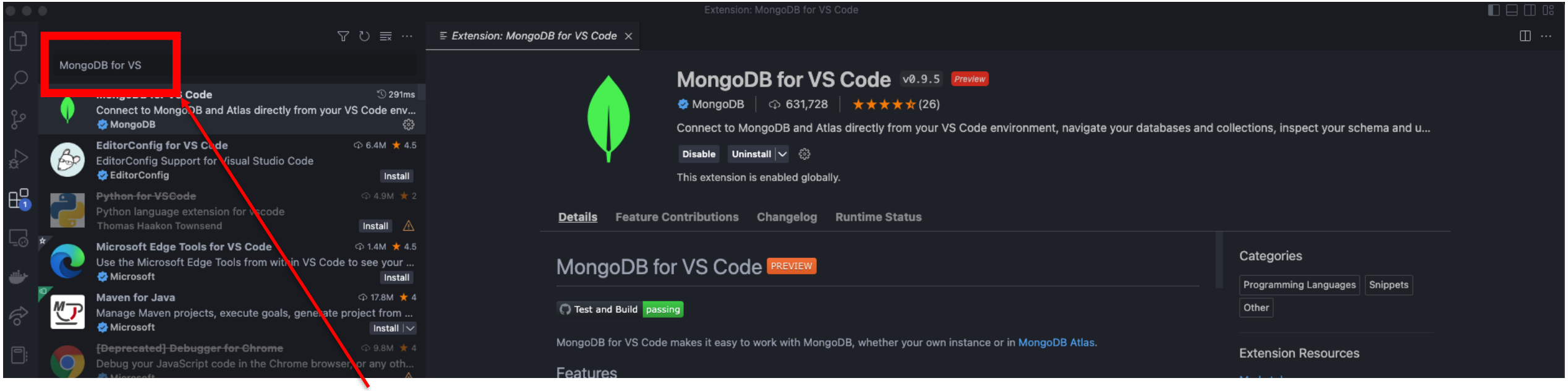
```
use [naziv_baze_podataka]
```

Zadaci: Osnove MongoDBa

1. Registrirajte se na sljedećem linku: https://www.mongodb.com/free-cloud-database.
2. Kreirajte server, korisnika i njegovu lozinku.
3. Povežite se u Mongo servis.
4. Ispišite popis postojećih baza podataka. Zapišite sve baze podataka koje ste pronašli na serveru.
5. Kreirajte bazu podataka **webApp** i kreirajte kolekciju: **users**

Uputa: Povezivanje na bazu podataka

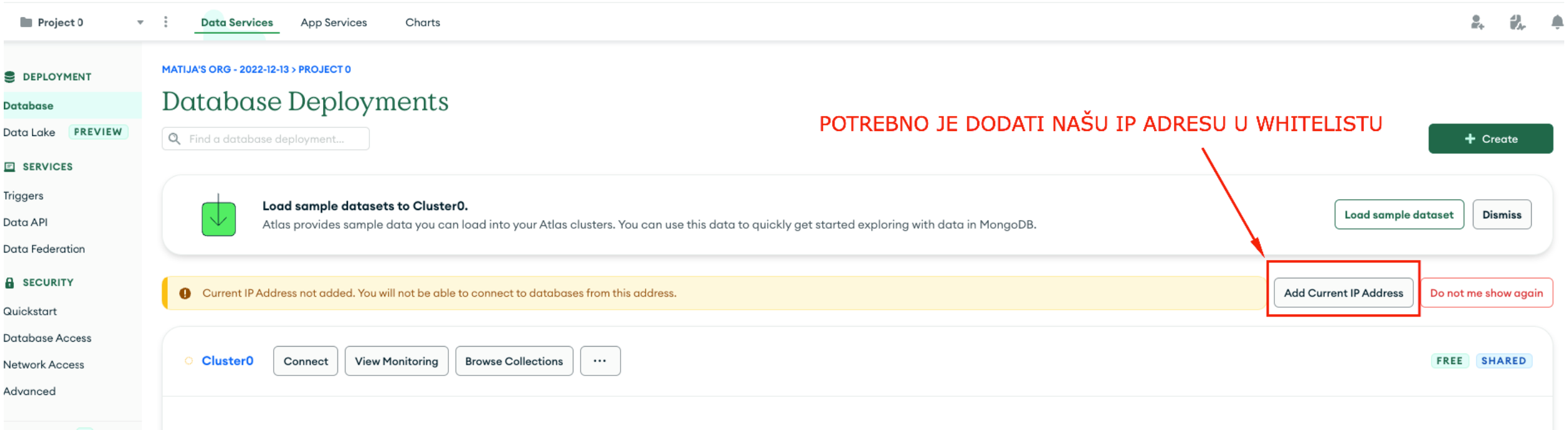
0. Potrebno je pokrenuti ***VS Code*** i instalirati ekstenziju:



UPISATI: MongoDB for VS

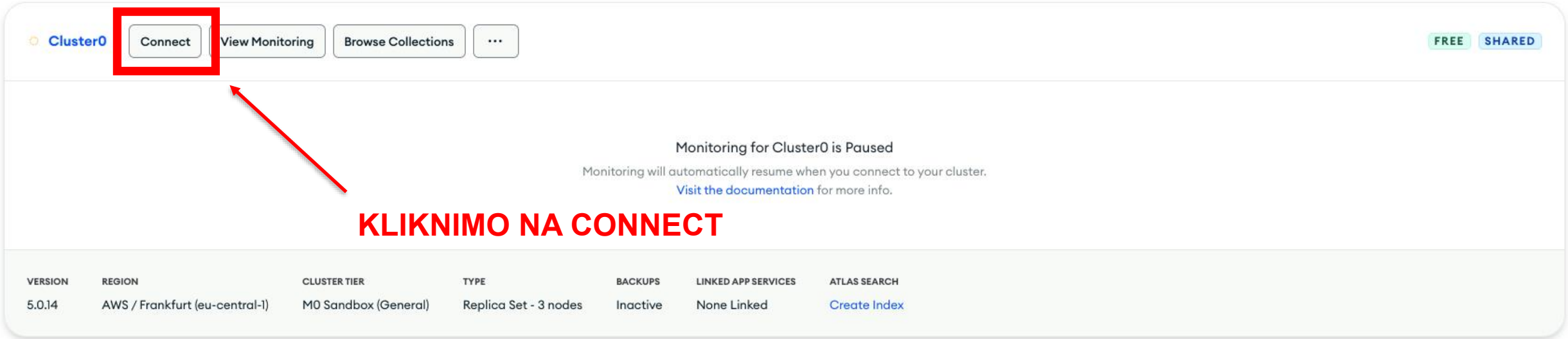
Uputa: Povezivanje na bazu podataka

1. Nakon registracije treba podesiti postavke koje će omogućiti povezivanje s bazom podataka



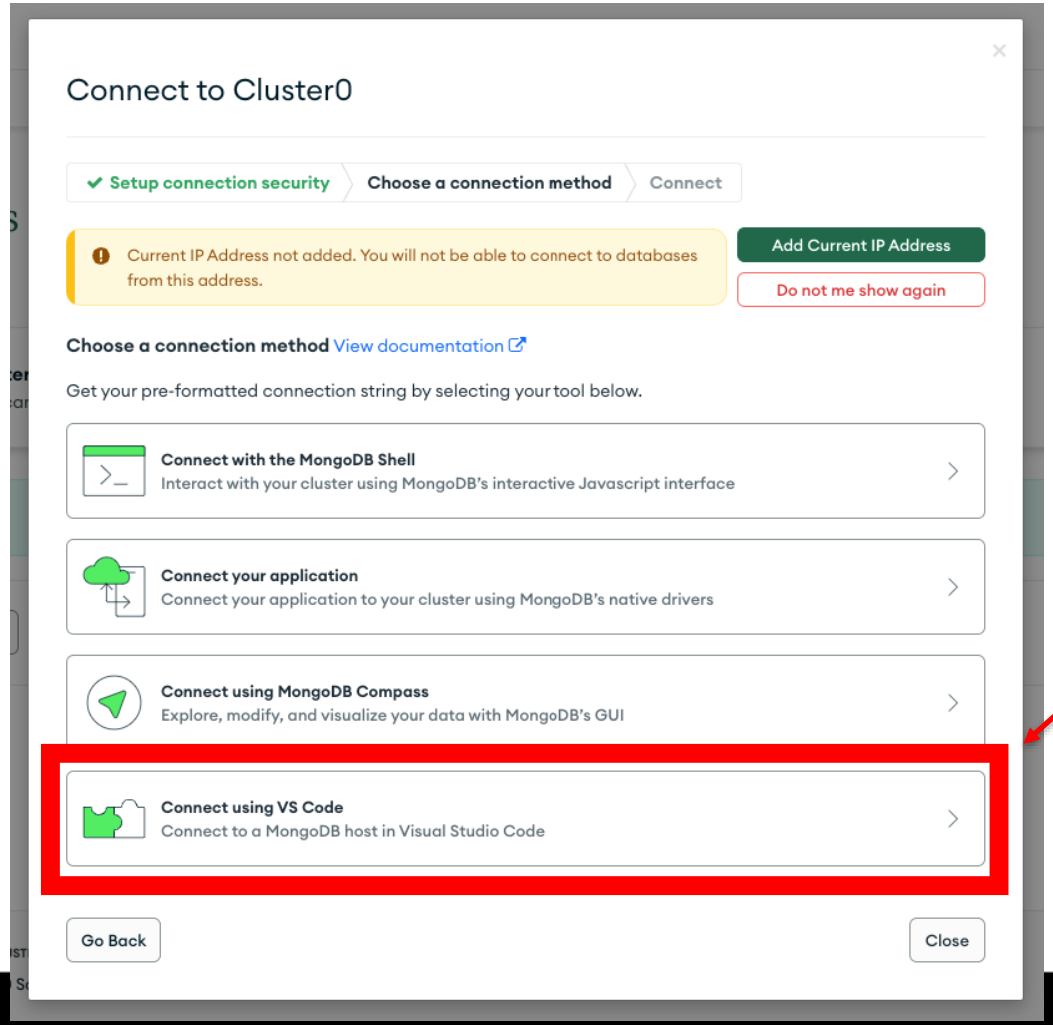
Uputa: Povezivanje na bazu podataka

2. Kliknuti na gumb: **Connect**



Uputa: Povezivanje na bazu podataka

3. Odabrati: **Connect using VS Code**



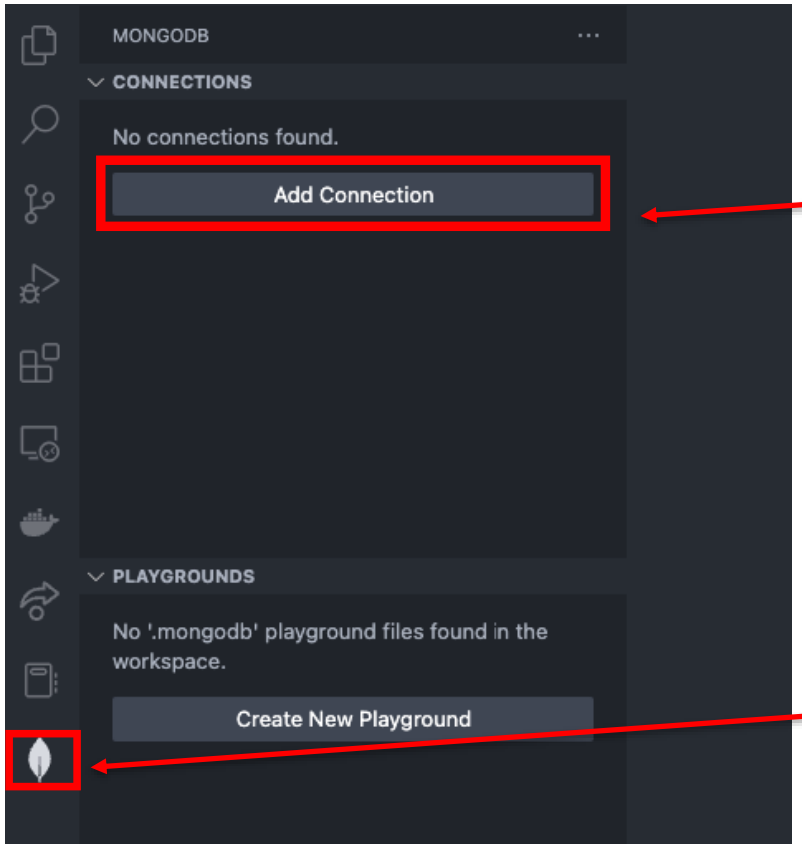
KLIKNIMO NA CONNECT USING VS CODE

Uputa: Povezivanje na bazu podataka

4. Kopirati *connection string*

Uputa: Povezivanje na bazu podataka

4. ***VS Code*:** iz lijevog izbornika odabrati **ikonicu Mongo** i nakon toga kliknuti na **Add Connection**

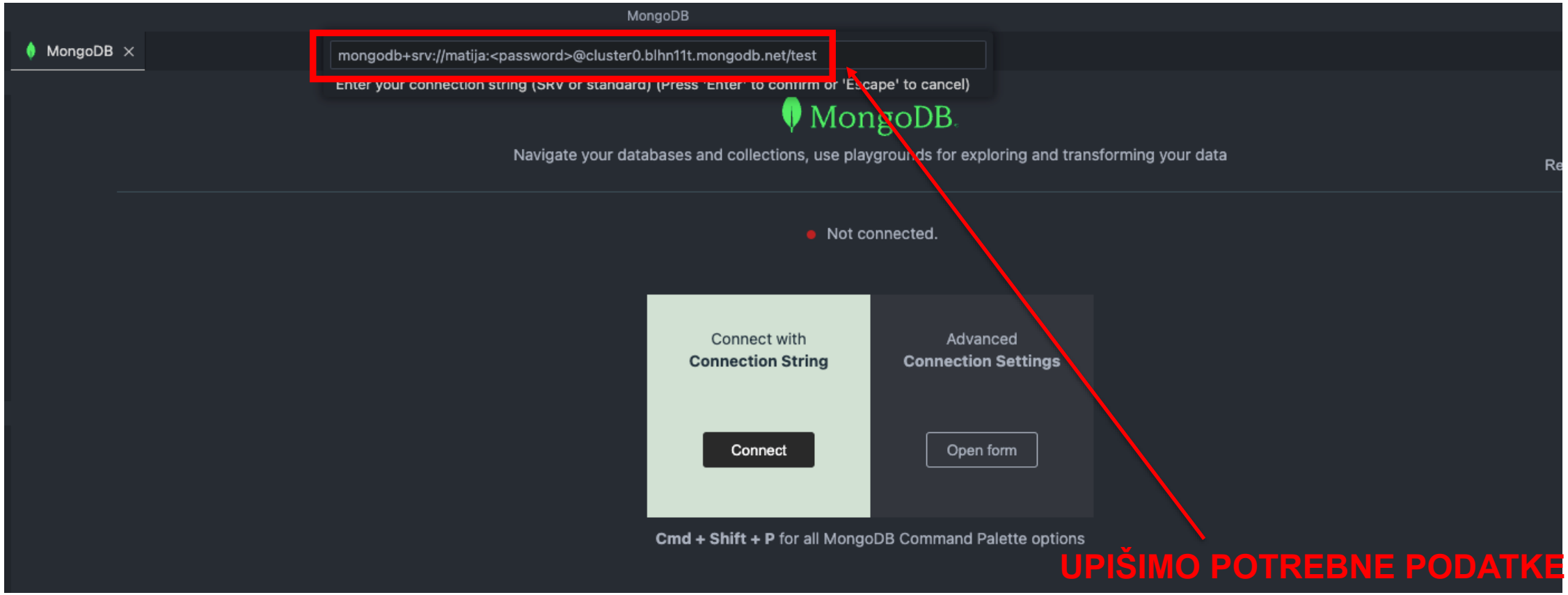


2. KLIKNIMO NA ADD CONNECTION

1. KLIKNIMO NA MONGO IKONICU

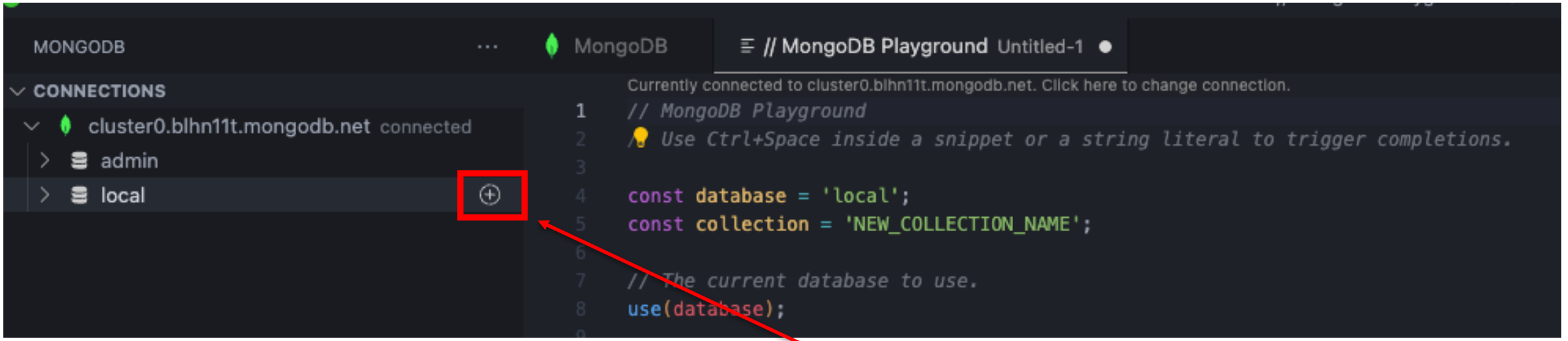
Uputa: Povezivanje na bazu podataka

5. Zalijepiti *connection string,* promijeniti lozinku i kliknuti ***Connect***



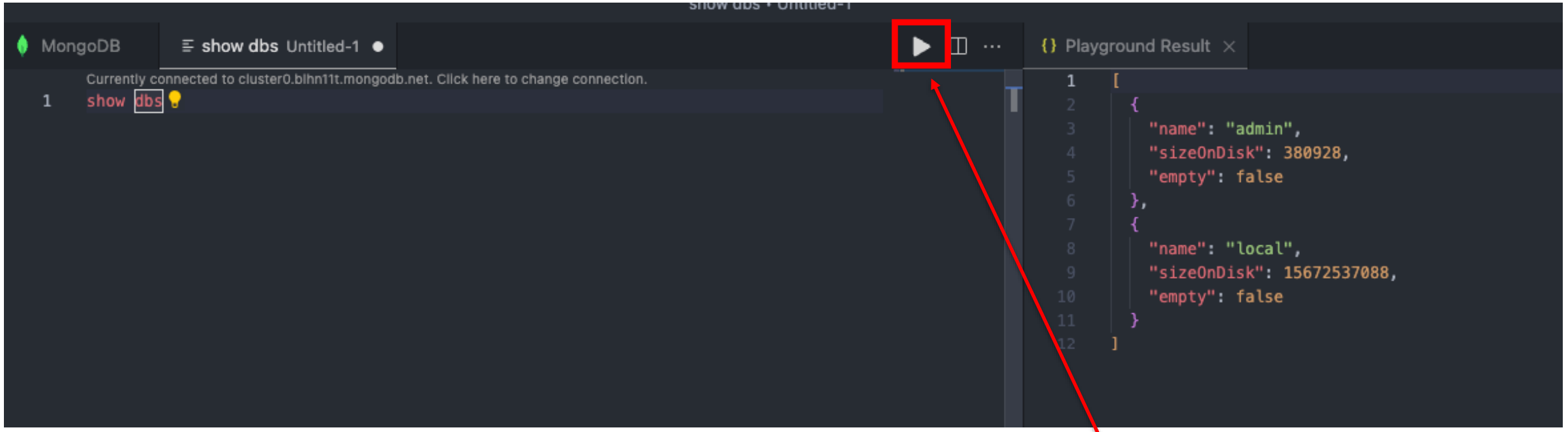
Uputa: Povezivanje na bazu podataka

6. Nakon uspješnog povezivanja kliknuti na gumb **+** za pisanje upita



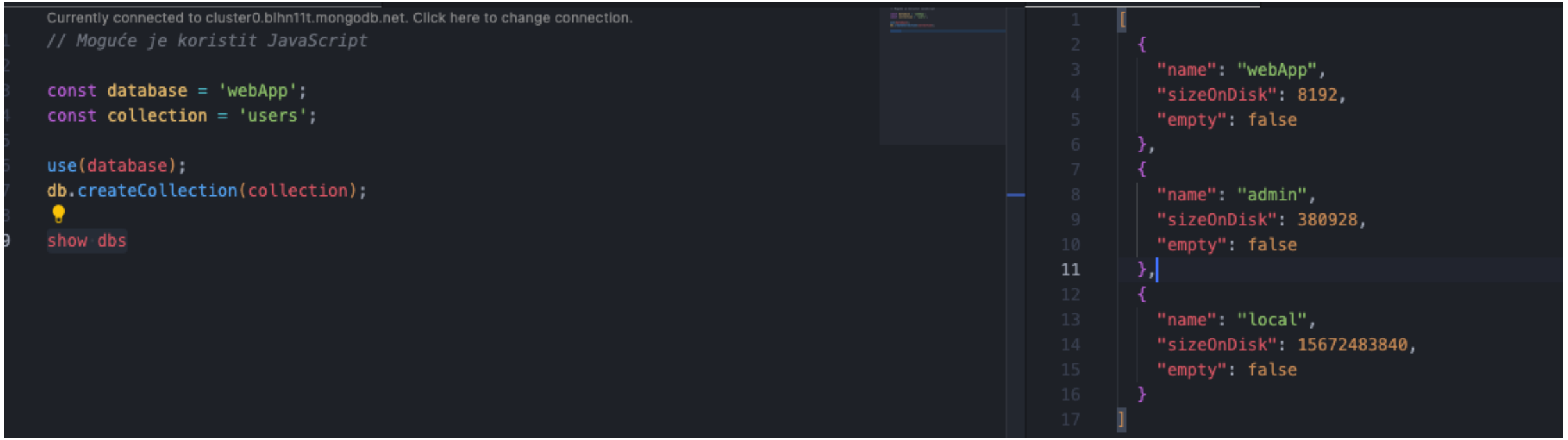
KLIKNIMO NA PLUS

Uputa: Ispišite popis baza podataka



KLIKNIMO RUN ZA POKRETANJE UPITA

Uputa: Kreirajte bazu podataka “*webApp*” i kreirajte kolekciju: users



INSERT QUERY API sintaksa

- **Primjer 1:** Upisivanje jednog podatka

```
db.[naziv_collectiona].insertOne(
 <tijelo_JSONa>
)
```

- **Primjer 2:** Upisivanje više podatka

```
db.[naziv_collectiona].insertMany(
 <tijelo_JSONa>, <tijelo_JSONa>, …
)
```

UPITI

- Više načina izvođenja
  - *Ad hoc* upiti nad dokumentima pomoću metoda **find** ili **findOne**
  - Moguće korištenje uobičajenih operatora (rasponi, nejednakosti, ograničenja…)
  - Filtriranje s **$where**
  - Upiti vraćaju pokazivač na sve potrebne dokumente
  - Postoje operacije nad pokazivačem poput preskakanja određenog broja rezultata, ograničavanje broja podataka koji se vraća, sortiranje i sl.

SQL	MongoDB
SELECT * FROM filmovi WHERE kategorija = „akcija“.	db.filmovi.find({kategorija:„akcija“})
SELECT * FROM filmovi WHERE kategorija = „akcija“ AND godina <= 2000	db.filmovi.find({kategorija: „akcija“, godina: {$lte: 2000}})

UPITI

- **Find** metoda je ekvivalentna SELECT naredbi u relacijskim bazama podataka

- Primjer mogućih operatora
  - **$eq** (*equal*) - usporedba zadane vrijednosti i vrijednosti iz dokumenta
  - **$ne** (*not equal*) - kada određena vrijednost nije jednaka onoj u dokumentu
  - **$gt** (*greater than*) i **$lt** (*less than*) - uspoređuju je li zadana vrijednost veća ili manja od vrijednosti u dokumentu
  - **$gte** (*greater than or equal*) i **$lte** (*less than or equal*) - provjerava je li vrijednost veća / manja ili jednaka zadanoj
  - **$in** (*included*) - kada se želi pronaći dokument sa zadanom vrijednosti
  - **$ni** (*not included*) - kada se želi pronaći dokument bez zadane vrijednosti
  - **$and**, **$not**, **$nor** i **$or** - logički operatori
  - **$exists** - vraća sve dokumente koji imaju zadano polje iz upita
  - **$type** - vraća dokumente koji imaju polja istog tipa zadanog u upitu

SELECT QUERY API sintaksa

- **Primjer 1:** Dohvaćanje podataka

```
db.[naziv_kolekcije].find(upit, projekcija, …)
```

`upit`

– **Neobavezno**. Određuje filtar. Ako treba vratiti sve dokumente iz kolekcije, ovaj parametar izostaviti ili proslijediti prazan dokument ({}).

`projekcija`

– **Neobavezno.** Određuje atribute koji se vraćaju. Za vraćanje svih atributa izostaviti ovaj parametar.

- **Primjer 2:** Dohvaćanje jednog podatka

```
db.[naziv_kolekcije].findOne(upit, projekcija, …)
```

- **Primjer 3:** Dohvaćanje s lijepim ispisom (ispis kroz više linija umjesto u jednoj liniji)

```
db.[naziv_kolekcije].find(upit, projekcija, …).pretty()
```

UPDATE QUERY API sintaksa

- **Primjer 1:** Ažuriranje podataka

```
db.[naziv_kolekcije].updateOne(
   {naziv_atributa: „trenutna_vrijednost_atributa”},
   {$set: {naziv_atributa: „nova_vrijednost_atributa”}}
)
```

- **Primjer 2:** Ažuriranje samo jednog podataka

```
db.[naziv_kolekcije].updateMany(
   {naziv_atributa: „trenutna_vrijednost_atributa”},
   {$set: naziv_atributa: „nova_vrijednost_atributa”}
)
```

DELETE QUERY API sintaksa

- **Primjer 1:** Brisanje podataka

```
db.[naziv_kolekcije].deleteOne({ naziv_atributa: 'vrijednost' });
```

```
db.[naziv_kolekcije].deleteMany({ naziv_atributa: 'vrijednost' });
```

- **Primjer 2:** Brisanje kolekcije

```
db.[naziv_kolekcije].drop();
```

Zadaci: CRUD u MongoDB

1. Kreirajte baze podataka **Algebra** s kolekcijom **Student**

2. U kolekciju **Student** upišite **studenta** s atributima I vrijednostima:
   FirstName: Maja
   LastName: Majić
   Administrator: False

2. U kolekciju **Student** upišite dodatna tri studenta, od kojih barem jedan mora imati vrijednost True u atributu Administrator.

3. Ispišite sve studente.
4. Ispišite samo prvog administratora.
5. Ispišite samo imena svih studenata koji nisu administrator, a potom I sve osim imena onih koji nisu administratori.

Zadaci: CRUD u MongoDB

6. Ažurirajte podatke tako da Pero postane administrator I promijenite mu ime u Peroslav.
7. Ažurirajte podatke tako da svim administratorima stavite Admnistrator: False
8. Izbrišite prvu Maju.
9. Izbrišite sve student koji nisu administrator.
10. Izbrišite cijelu kolekciju Student.

Spajanja kolekcija

- U MongoDBu postoje opcije za rad s podacima koja se spajaju iz više kolekcija, slično kao što se to radi u SQL bazama podataka s JOIN naredbama

  1. **$lookup** - omogućuje spajanje podataka iz dvije kolekcije u jednom pogledu. Koristi se u slučajevima kada se podaci iz dvije kolekcije trebaju spojiti na temelju nekog zajedničkog polja.
  2. **$graphLookup** - omogućuje spajanje podataka u obliku grafa. Koristi se za pretraživanje podataka u više kolekcija i rekurzivno spajanje podataka u obliku stabla ili grafa.
  3. **$merge** - omogućuje spajanje podataka iz dvije ili više kolekcija u jednu kolekciju. Koristi se za spajanje podataka iz više izvora u jedinstvenu kolekciju za lakše upravljanje i pretraživanje.
  4. **$facet** - omogućuje izvođenje više agregacijskih radnji u jednom pogledu. Koristi se za spajanje podataka iz više kolekcija i izvođenje više agregacijskih radnji nad tim podacima u jednom pogledu.
  5. **Kombiniranje naredbi** - moguće je kombinirati različite naredbe za spajanje podataka.

- Treba imati na umu da ovakvi pristupi spajanju podataka nisu isti kao u SQL bazama podataka i da mogu zahtijevati više koraka i kompliciranije upite
- Treba pažljivo razmotriti koja je opcija najprikladnija pri radu s podacima u MongoDB

LOOKUP QUERY API sintaksa

- Naredba **$lookup** dodaje niz podataka iz pridruženog dokumenta
- Usporedivo s JOIN iz SQL-a

- **Primjer 1:** Spajanja

```
db.[naziv_kolekcije].aggregate ([{
   $lookup:
     {
       localField: „<naziv_kolekcije-atribut>”,
    from: „<naziv_kolekcije_za_spajanje>”,
       foreignField: „<naziv_kolekcije_za_spajanje-atribut>”,
       as: „<naziv_novog_polja_koje_će_se_pojaviti_u_ispisu>”
     }
}])
```

LOOKUP QUERY API sintaksa

1. Kreirati kolekciju **roles** i upisati novi podatak s atributima Naziv i UserEmail.

- **Primjer 1:** Spajanja

```
db.users.aggregate ([{
  $lookup:
   {
    localField: „Email”,
    from: „roles”,
    foreignField: „UserEmail”,
    as: „Role”
   }
}])
```

LOOKUP QUERY API sintaksa

- Naredba **$merge** spaja podatke u novu kolekciju, vrlo često se koristi kada želimo ispraviti podatke

- **Primjer 1:** Spajanja

```
db.[naziv_kolekcije].aggregate ([{
   $merge:
     {
       into: „<naziv__nove_kolekcije>”,
        whenMatched: „<koja_akcija_se_treba_dogoditi>”,
        whenNotMatched: „<koja_akcija_se_treba_dogoditi>”
     }
}])
```

LOOKUP QUERY API sintaksa

1. Kreirajmo novu kolekciju iz: **users** i **roles**.

- **Primjer 1:** Spajanja

```
db.users.aggregate([
{
    $lookup: {
         from: "roles",
         localField: "Email",
         foreignField: "UserEmail",
         as: "roles"
    }},
{ $unwind: "$roles” },
{
   $merge: {
    into: "combined",
    whenMatched: "replace",
    whenNotMatched: "insert"}} ])
```