OBLIKOVANJE BAZA PODATAKA

Predavanje 14

NoSQL - uvod

- **Not only SQL** (*NoSQL*) predstavlja *non-tabular* baze podataka koje pohranjuju podatke drugačije u odnosu na relacijske baze podataka

- NoSQL bazi nije toliko važan integritet podataka već je bitnija dostupnost podataka

- Za NoSQL baze koristi se često akronim BASE
  (*Basically Available, Soft-state, Eventual consistency*)

- NoSQL baze imaju odgovor za svaku akciju koja se događa
  - Odgovor može biti negativan ili pozitivan ovisno o tome jesu li podaci dohvaćeni ili ne (*Basically Available*)

NoSQL - uvod

- Baza neprekidno ažurira podatke kako bi vidjela postoji li možda promjena u podacima pa da ju odmah i promijeni (*Soft-state*)
  - U sustavima gdje je MongoDB u replikaciji. *Soft-state* je stanje u kojem podaci čvora nisu u potpunosti konzistentni s ostatkom klastera, ali čvor još uvijek može primati i odgovarati na operacije čitanja i pisanja.
  - Primjer 1: Čvor je tek dodan u klaster i još uvijek je u procesu sustizanja ostalih čvorova, može biti u mekom stanju dok u potpunosti ne sinkronizira svoje podatke.
  - Primjer 2: Čvor je doživio mrežni problem i ne može komunicirati s drugim čvorovima, može biti u mekom stanju dok se particija ne riješi.
  - S obzirom da NE postoje tradicionalne transakcije, postoje pravila koja se mogu definirati što će se desiti ako postoji dulja sesija koja upravlja dokumentima.
  - Dok je čvor u mekom stanju, može nastaviti prihvaćati operacije čitanja i pisanja, ali te se operacije možda neće odraziti na ostatak klastera. Nakon što čvor u potpunosti sinkronizira svoje podatke i može komunicirati s ostatkom skupa, prijeći će iz mekog stanja i postati potpuno konzistentan s ostatkom klastera.

- Podrazumijeva da će NoSQL baza na kraju postati dosljedna kada prestane primati naredbe (*Eventual consistency*), označava da sesija prema bazi traje samo za vrijeme trajanja naredbi

NoSQL - uvod

- Postoje četiri načina pohrane u NoSQL baze podataka:
  - **Document baze podataka (npr. *MongoDB*)**
    - Spremaju podatke u dokumente. Svaki dokument ima vrijednosti koje su obilježene tipovima podataka. Ovakav tip se najviše koristi zbog raznolikosti podataka
  - Key-Value stores (npr. *Redis*)
    - Svaki podatak ima svoj ključ i vrijednost. Podatak se dohvaća preko ključa. Ovakve baze se koriste kada se želi spremiti velika količina podataka, ali se ne koriste složeni upiti za dohvaćanje istih
  - Column-oriented baze podataka (npr. *Apache Cassandra*)
    - Podaci se spremaju u tablice, redove i stupce. Redovi mogu imati različite stupce. Koriste se kod pohrane velike količine podataka
  - Graph baze podataka (npr. *FlockDB*)
    - Spremaju podatke u čvorove i veze

Što i kada koristiti?

- **Relacijsku bazu podataka** bismo koristili kada imamo strukturirane podatke koje treba spremiti i pretraživati pomoću SQL-a (kada treba osigurati integritet podataka)

- **NoSQL bazu podataka** bismo koristili kada imamo veliku količinu nestrukturiranih ili polustrukturiranih podataka koje treba brzo spremati i kojima treba brzo i jednostavno pristupati

- Važno je napomenuti da ne postoji jedinstveno rješenje za sve potrebe i da će pravi izbor ovisiti o specifičnim zahtjevima aplikacije

- U nekim slučajevima možda će biti potrebno koristiti i relacijske i NoSQL baze podataka za spremanje različitih vrsta podataka

NoSQL baza podataka

- Može spremati velike količine podataka
- Često upiti nad NoSQL bazom podataka brže vrate rezultat nego što bi to napravila relacijska baza podataka
- Baza će primiti bilo kakve podatke i uvrstiti ih neovisno o tome kakvog su tipa
- NoSQL baze podataka često imaju vrlo veliku i brzo rastuću količinu podataka pa im je potrebno okruženje koje će to podržati. Iz tog razloga, za ovakve baze se koristi uvijek pohrana u oblaku

Volume, Velocity, Variety

- NoSQL baza podataka je opisana skraćenicom **3V**
  - ***VOLUME*** - volumen podataka, odnosno velike količine podataka
  - ***VELOCITY*** - brzina mijenjanja podataka, odnosno podaci nastaju velikom brzinom i brzo se obrađuju
  - ***VARIETY*** - različitost podataka, odnosno podaci nisu potpuni, mogu biti nestrukturirani ili polustrukturirani ovisno o izvoru podataka
    - Zbog raznolikosti podataka shema za ovakve baze nije fiksna

Usporedba NoSQL i relacijskih baza podataka

- **Relacijske baze podataka**
  - Koriste model u kojem se podaci spremaju u tablice koje sadrže stupce i retke
  - Shema je jako stroga
  - Bitan je referencijalni integritet

- **Nerelacijske baze podataka**
  - Pružaju više različitih modela i kose se sa svime što relacijski modeli strogo definiraju (izrazito fleksibilan pristup)

Usporedba NoSQL i relacijskih baza podataka

- **Relacijske baze podataka**
  - ACID osobine
    - (**Atomicity**) Atomičnost: cijela transakcija završi uspješno ili neuspješno - bitan je referencijalni integritet
    - (**Consistency**) Konzistentnost: nakon što transakcija završi, podaci moraju biti u skladu sa shemom baze
    - (**Isolation**) Izoliranost: transakcije se izvršavaju neovisno
    - (**Durability**) Trajnost: mogućnost oporavka ukoliko dođe do pada sustava
- **Nerelacijske baze podataka**
  - Izgrađene na BASE osobinama
  - ACID osobine su manje bitne jer se upravlja s fleksibilnijim podacima

Usporedba NoSQL i relacijskih baza podataka

- **Relacijske baze podataka**
  - Učinkovitost relacijskih baza dijelom ovisi i o diskovnom sustavu
  - Kako bi bile učinkovitije, radi se optimizacija upita, indeksa te same strukture tablica unutar baze
  - Koristi SQL jezik
- **Nerelacijske baze podataka**
  - Učinkovitost nerelacijskih baza podataka temeljena je na funkciji hardverskog sklopa, mreži i aplikaciji koja ju koristi
  - Ne postoji standardni jezik (NoSQL ovisi o programskom jeziku kojeg koristimo)

Usporedba NoSQL i SQL baza podataka

	Relacijske baze podataka	Document baze podataka
Shema	Strogo definirana	Fleksibilna
Skaliranje	Vertikalno (moćniji poslužitelj)	Horizontalno (više poslužitelja)
ACID Transakcije	Podržava	Većina tehnologija ne podržava, MongoDB podržava
Joinovi	Potrebno	Podržava (ali postoje i druge metode)
Indexi	Potrebno	Podržava (koristi se za optimizaciju velikih upita)
Mapiranje podataka u objekt	Potreban ORM	Nije potreban ORM. MongoDB dokumenti mapiraju se izravno u strukture podataka

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

Document baze podataka (MongoDB)

- MongoDB je sustav za upravljanje bazama podataka koji koristi **model dokumenata** za upravljanje i dohvaćanje podataka

- **Shema baze podataka ne mora biti unaprijed definirana** već se podaci mogu dodati u bilo kojem obliku

- Uz svaki programski jezik dolazi biblioteka koja sadrži skup osnovnih funkcija za upravljanje podataka
- Podaci su pohranjeni u JSON, BSON ili XML formatu
- Kada se radi s većim količinama podataka i doda se novi poslužitelj, MongoDB će sam posložiti podatke

- **MongoDB je osjetljiv na velika i mala slova i na tipove podataka**

Document baze podataka (MongoDB)

- Razvojnim inženjerima olakšavaju se:
  - Pohrana i upiti prema bazi podataka jer se koristi isti format modela dokumenta koji koriste u vlastitom kodu aplikacije
  - Fleksibilan, polu-strukturiran i hijerarhijski dizajn podataka omogućuje agilniji razvoj
    - Nema potrebe za posebnim izmjenama dizajna atributa
    - Nema potrebe za migracijskim skriptama

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
  - **Baza podataka** – sadrži kolekcije dokumenata
  - **Kolekcija** – ekvivalent tablici u relacijskim bazama
  - **Dokument** – ekvivalent retku u relacijskim bazama
  - **Ključ** – svaki dokument ima svoj jedinstveni ključ

```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  }
  "hobbies": ["surfing", "coding"]
}
```

Dokument

- Dokument je uređeni skup ključeva s pridruženim vrijednostima
- Svaki programski jezik prikazuje dokumente na svoj način pomoću struktura podataka
  - MongoDB je predefiniran za JavaScript kojeg vežemo uz JSON (*JavaScript Object Notation*)
- Dokumenti su u formatu koji se naziva binarni JSON (BSON), što je zapravo binarni prikaz JSON dokumenta

Dokument

- Svaki dokument mora sadržavati polje (atribut) **_id** koje je rezervirano za primarni ključ
  - Ukoliko se prilikom unosa podataka ne definira polje _id, MongoDB će ga automatski napraviti
  - Jedini je zahtjev da vrijednost bude jedinstvena
  - Obično se u tu svrhu koristi *ObjectId*

- MongoDB ne sadrži relacije uobičajene u relacijskim bazama podataka, već odnose implementira na dva načina:
  - Pomoću ugniježđenih dokumenata
  - Pomoću referenci
  - VAŽNO: programer mora brinuti o jedinstvenim poljima (atributima) s kojima će povezivati podatke

Kolekcije

- Kolekcija je grupa dokumenata (ekvivalent tablici u relacijskim bazama podataka)

- Tablice u relacijskim bazama imaju definiranu shemu koja se mora poštivati, a kolekcije nemaju nikakvu shemu po kojoj bi se morali upisivati podaci

- Kod imenovanja kolekcija postoje ograničenja:
  1. Ne može biti prazan naziv
  2. Ne smije sadržavati NULL znak jer on označava kraj imena
  3. Ne smije početi riječju *system* jer se time označavaju sistemske kolekcije
  4. Ne smije sadržavati znak *$* u imenu

- Postoji mogućnost ugnježđivanja kolekcija, što olakšava i upravljanje dokumentima

Ograničene i sistemske kolekcije

- MongoDB ograničene kolekcije
  - Ograničena kolekcija je kolekcija fiksne veličine
  - Kad ograničena kolekcija dosegne svoju maksimalnu količinu dokumenata koji su spremljeni, oni će se obrisati i na njihovo mjesto će se spremiti novi dokumenti
  - U ograničenim kolekcijama se polje **_id** ne stvara automatski
  - Korisne ako se treba čuvati konstantna količina podataka, bez potrebe za čuvanjem ranijih promjena (npr. čuvaju se samo najsvježije promjene statusa na socijalnim mrežama)
  - Ne podržavaju ažuriranja kojima bi se veličina dokumenata povećala
  - Ne podržavaju brisanja, osim najstarijih dokumenata

- MongoDB sistemske kolekcije
  - *system.namespace* ispisuje sve imenske prostore kako bi se vidjele kolekcije i potkolekcije
  - *system.indexes* prikazuje sve indekse kreirane u bazi podataka

MongoDB baze podataka

- MongoDB spaja kolekcije u baze podataka

- Kod imenovanja baza podataka mora se paziti na određena pravila
  - Naziv ne smije biti prazan
  - Naziv ne smije sadržavati razmak, točku, znak *$*, kosu crtu, obrnutu kosu crtu ili NULL znak
  - Cijeli naziv mora biti napisan malim slovima

- MongoDB sadrži sistemske baze podataka: *admin*, *local* i *config*

Sistemske baze podataka

- ***Admin* baza**
  - Glavna baza podataka - ako se nekom korisniku dodijeli pravo nad ovom bazom on može mijenjati prava nad svim bazama
- ***Local* baza**
  - Baza koja se kopira prilikom kreiranja nove baze (kao sistemska baza *Model* u SQL Serveru)
- ***Config* baza**
  - Baza koja sprema podatke kada se MongoDB koristi u particijskom stanju
  - “Particijsko stanje” odnosi se na stanje kada jedan ili više čvorova ne mogu komunicirati s klasterom
  - U tom slučaju ne postoji glavni *Config* koji se sinkronizira između servera, nego se koristi posljednji *Config* koji je sinkroniziran

Query API sintaksa

- Connection string za bazu podataka:

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

- Popis svih baza podataka:

```
show dbs
```

- Korištenje baze podataka:

```
use [naziv_baze_podataka]
```

Query API sintaksa za kreiranje baza podataka

- Kreiranje baze podataka:

```
use [naziv_baze_podataka]
```

- Brisanje baze podataka:

```
use [naziv_baze_podataka]
db.dropDatabase
```

- Kreiranje kolekcije:

```
db.createCollection („[naziv_kolekcije]”)
```

- Brisanje kolekcije:

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

Kreiranje i brisanje baze podataka

Primjer 1: Bazu poslovnih partnera kreiramo upitom:

```
[naziv_baze] > use BazaPoslovnihPartnera
```

- Nakon USE naredbe iskoristit ćemo naredbu: **show dbs**
- Možemo primijetiti da baza podataka još ne postoji. Kako bi bazu podataka stvarno kreirali, potrebno je kreirati prvu kolekciju

Primjer 2: Kreiranje kolekcije:

```
[BazaPoslovnihPartnera] > db.createCollection („[naziv_kolekcije]”)
```

Primjer 3: Brisanje kolekcije:

```
[BazaPoslovnihPartnera] > db.[naziv_kolekcije].drop()
```

Primjer 4: Brisanje baze poslovnih partnera možemo izvršiti upitom:

```
[BazaPoslovnihPartnera] > db.dropDatabase
```

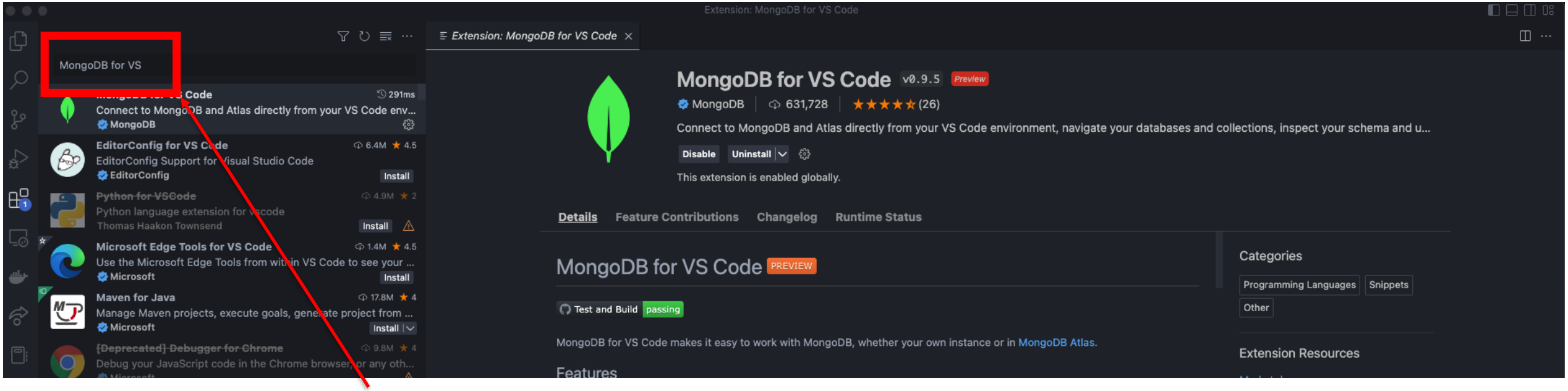
Napomena: moramo se nalaziti u bazi podataka koju brišemo

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 **Algebra** i kreirajte kolekciju: **Student**

Uputa: Povezivanje na bazu podataka

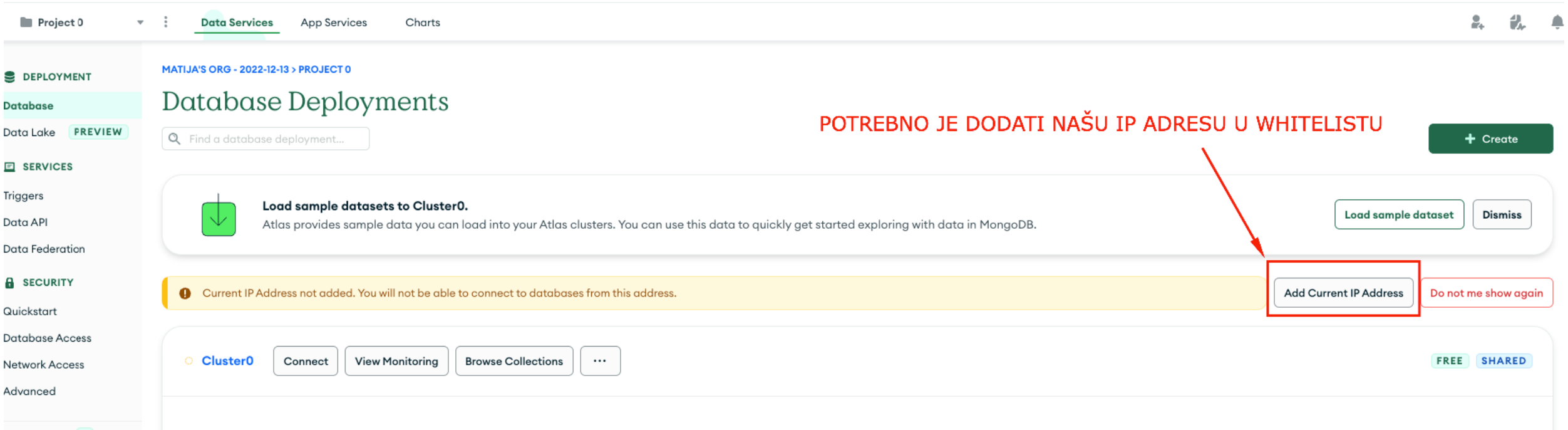
0. Potrebno je pokrenuti ***VS Code*** i instalirati ekstenziju:



UPISATI: MongoDB for VS

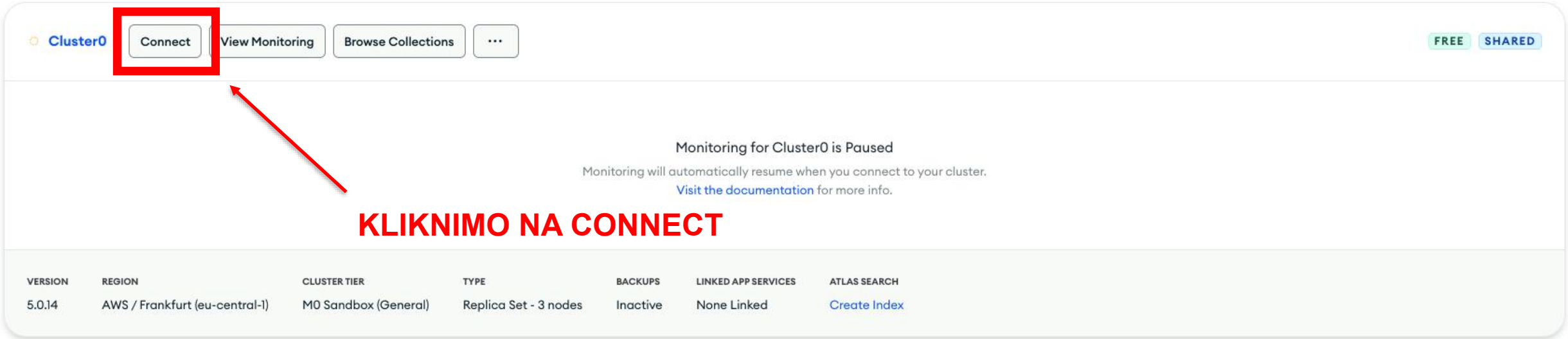
Uputa: Povezivanje na bazu podataka

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



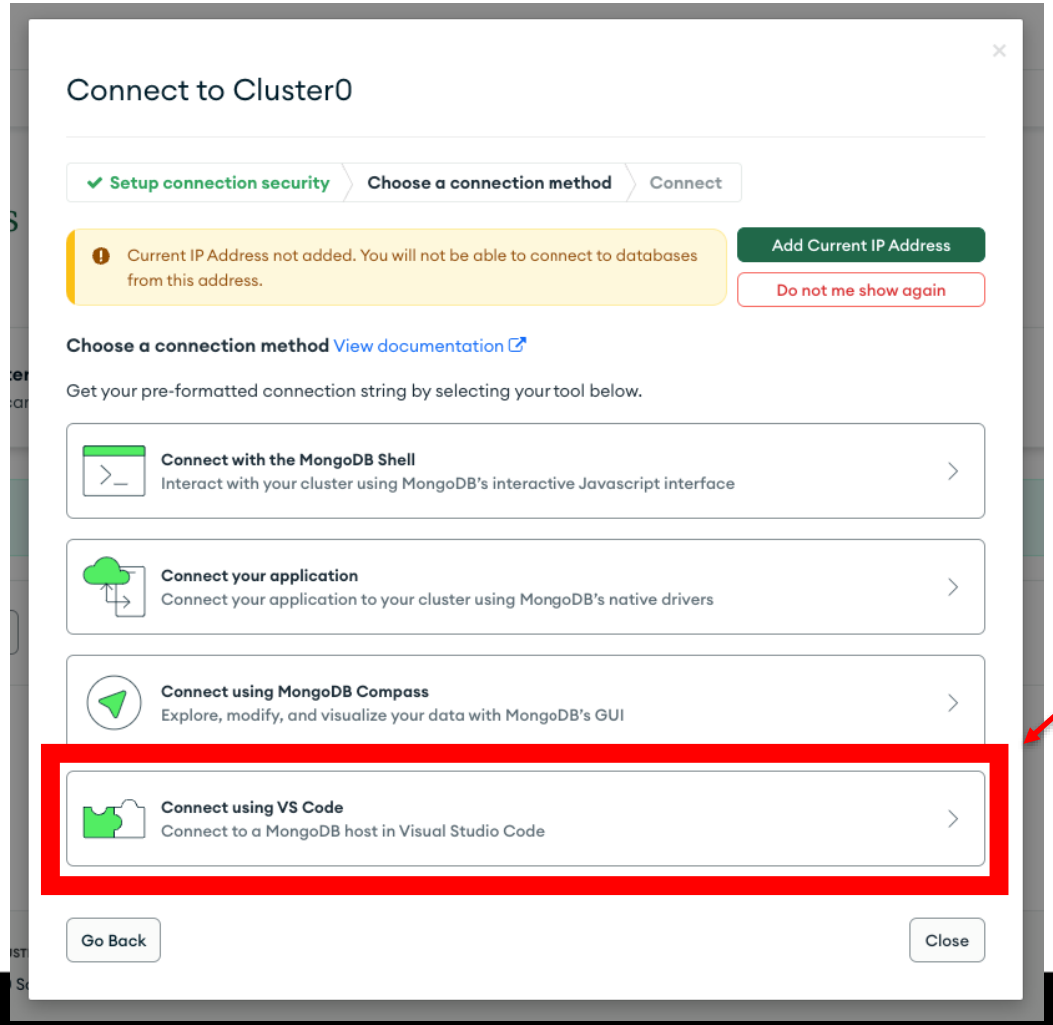
Uputa: Povezivanje na bazu podataka

2. Kliknuti na gumb: **Connect**



Uputa: Povezivanje na bazu podataka

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



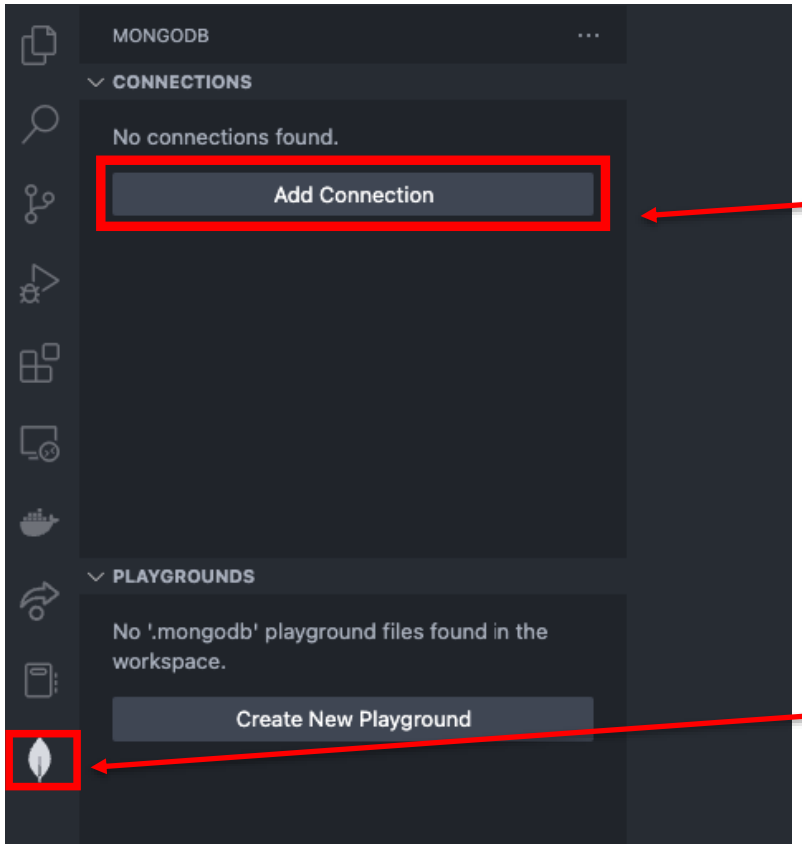
KLIKNIMO NA CONNECT USING VS CODE

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**

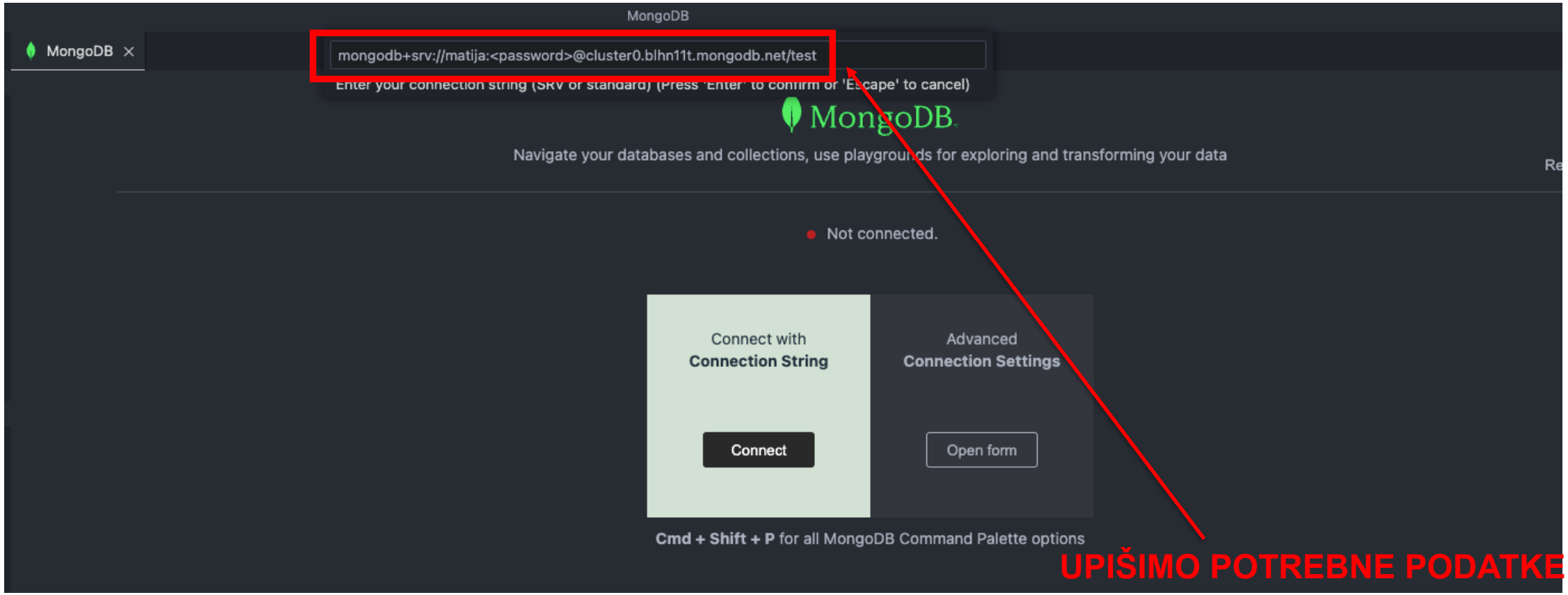


2. KLIKNIMO NA ADD CONNECTION

1. KLIKNIMO NA MONGO IKONICU

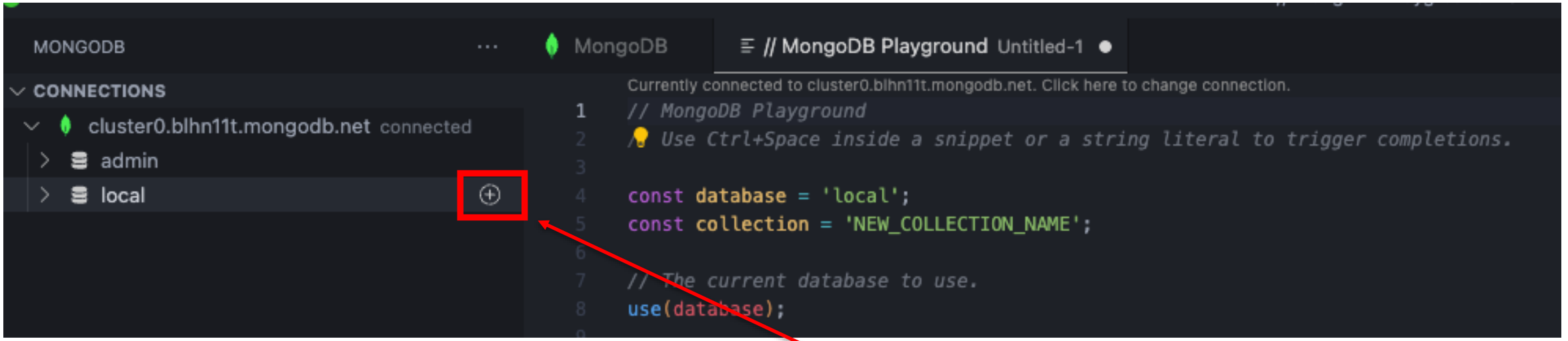
Uputa: Povezivanje na bazu podataka

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



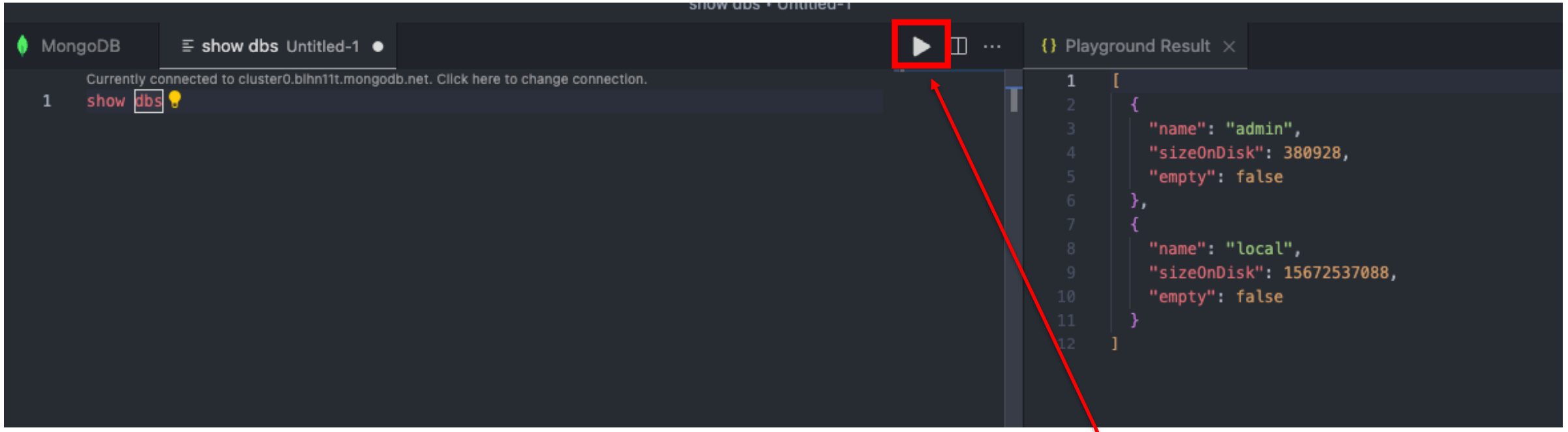
Uputa: Povezivanje na bazu podataka

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



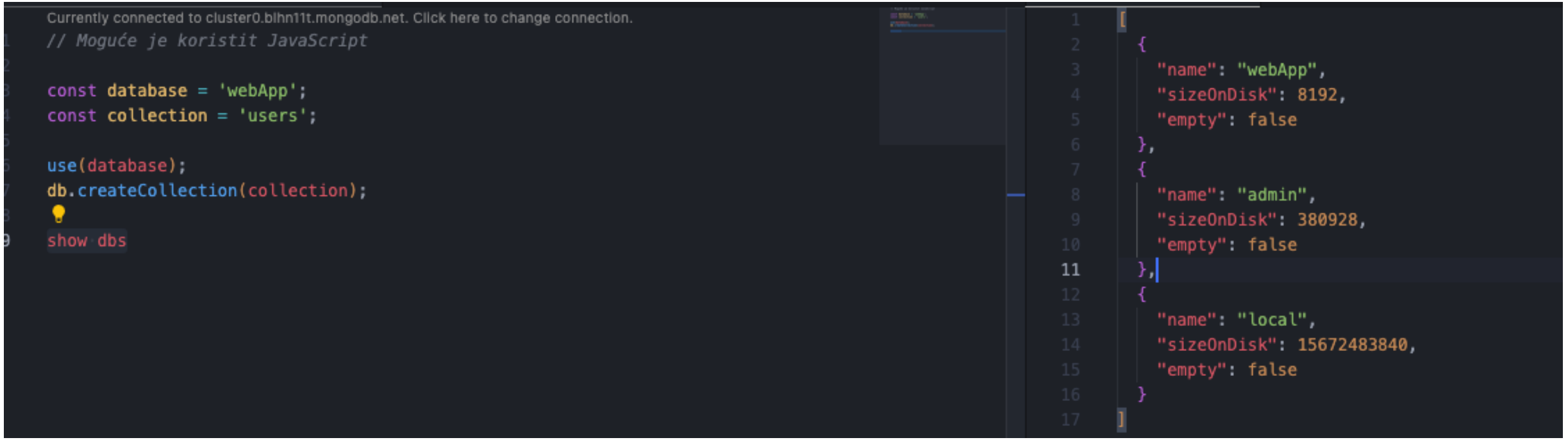
KLIKNIMO NA PLUS

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>
)
```

- **Primjer 3:** Upisivanje podatka (metoda koja objedinjuje prve dvije)

```
db.[naziv_collectiona].insert(
  <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_collectiona].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, …)
```

UPDATE QUERY API sintaksa

- **Primjer 1:** Ažuriranje podataka

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

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

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

DELETE QUERY API sintaksa

- **Primjer 1:** Brisanje podataka

```
db.[naziv_kolekcije].remove(
   {naziv_atributa: “vrijednost_atributa”},
   (true/false)
)
```

- posljednji *boolean* označava treba li atribut odgovarati vrijednosti, potrebno je paziti zato što je zadan vrijednost ***false***

```
db.[naziv_kolekcije].deleteOne({naziv_atributa: “vrijednost_atributa”});
```

```
db.[naziv_kolekcije].deleteMany({naziv_atributa: “vrijednost_atributa”});
```

Zadaci: CRUD u MongoDB

1. U kolekciju **Student** upišite nekoliko dokumenata s atributima:
   id, *Firstname*, *Lastname*, *Administrator, CityID*
2. U kolekciju **City** upišite nekoliko dokumenata s atributima:
   IDCity, CityName
3. Ispišite sve studente
4. Ispišite prvog studenta administratora
5. Ispišite sve studente administratore prikazujući samo njihovo ime
6. Ispišite sve studente administratore prikazujući sve podatke osim njihovog imena
7. Promijenite nekom studentu ime I označite ga kao administratora
8. Svim studentima ukinite oznaku administratora
9. Obrišite prvog studenta prema imenu
10. Obrišite sve studente koji nisu administratori
11. Obrišite 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

- **Primjer: osnovna lookup metoda**

```
db.Student.aggregate([
    {
      $lookup: {
        from: 'City',                    // Target collection
        localField: 'CityID’,    // Field from the input documents
        foreignField: 'IDCity',          // Field from the target documents
        as: 'StudentsFromCities'    // Alias for the joined documents
      }
    }
]);
```

LOOKUP QUERY API sintaksa

- **Primjer: lookup metoda s unwind**

```
db.Student.aggregate([
    {      $lookup: {
                    from: 'City',
                    localField: 'CityID',
                    foreignField: 'IDCity’,
                    as: 'StudentsFromCities'      }    },
    {      $unwind: '$StudentsFromCities'  // Unwind the array created by $lookup    },
    {       $project: {
          _id: 0,
          Firstname: 1,
          Lastname: 1,
          'StudentsFromCities.CityName': 1
        }    }  ]);
```

LOOKUP QUERY API sintaksa

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

```
db.Student.aggregate([
  {      $lookup: {
            from: 'City',
            localField: 'CityID',
            foreignField: 'IDCity',
            as: 'StudentsFromCities'      }  },
  { $unwind: '$StudentsFromCities' },
  {      $merge: {into: 'StudentsFromCitiesCombined', whenMatched: 'replace’,
               whenNotMatched: 'insert'}}   ]);
db.StudentsFromCitiesCombined.find({});
```