

# Veb programiranje

Marko Arsenovic  
Srdjan Sladojevic



# Uvod u MongoDB



# NoSQL

- Termin NoSQL – nastao 2009. godine
  - u opštem smislu, objedinjuje sve baze podataka i skladišta podataka koje ne slede primarne principe relacionih baza podataka



# NoSQL

- CAP teorema
  - sistem koji skladišti deljene podatke ne može obezbediti istovremeno zadovoljenje sledećih uslova
    - konzistentnost – eng. Consistency
    - raspoloživost – eng. Availability
    - tolerancija razdvojenosti – eng. Partition tolerance
- primenljiva na sisteme zasnovane na distribuiranoj arhitekturi



# NoSQL

- CAP teorema
  - konzistentnost
    - svako čitanje iz baze podataka kao rezultat ima najnoviju verziju podataka
  - raspoloživost
    - odziv sistema u garantovanim vremenskim okvirima
    - baza podataka će uvek biti dostupna
      - nezavisno kada je postavljen upit
  - postiže se
    - velikim brojem fizičkih servera
    - replikacijom podataka

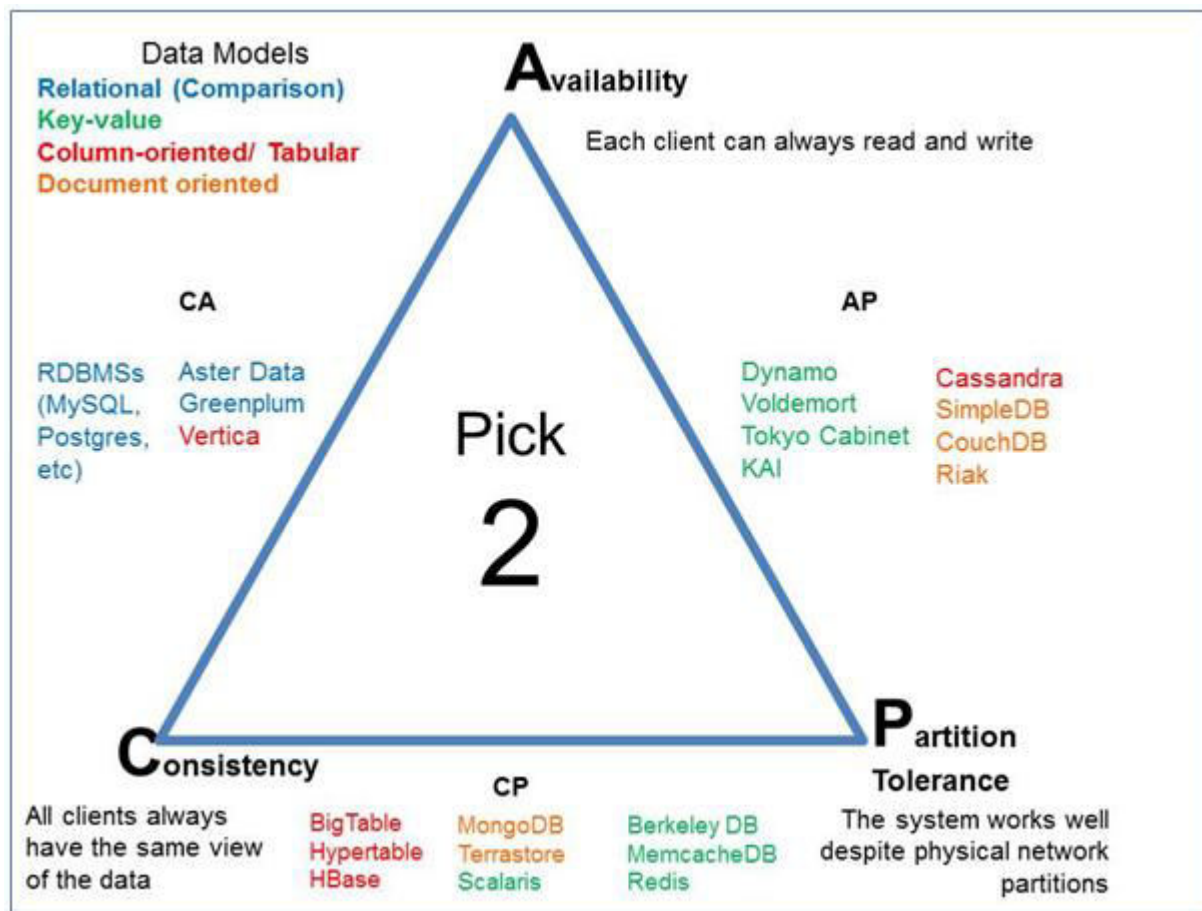


# NoSQL

- CAP teorema
- tolerancija razdvojenosti
  - nijedan skup otkaza, osim potpunog otkazivanja, ne sme da proizvede neispravan odziv sistema baze podataka
    - sistem mora da prihvata delimične otkaze i nastavlja ispravan rad
  - Razdvojenost
    - stanje komunikacione mreže kod koje su delovi sistema podeljeni u dve ili više particija
    - Između kojih ne postoji komunikacija
    - rešenje:
      - replikacija
      - odloženo (asinhrono) pisanje u trenutno nedostupne delove baze podataka



# NoSQL



# MongoDB

- NoSQL baza u C++
- cross-platform dokument orjentisana baza podataka
- Dokumenti su u BSON formatu
  - Binarna reprezentacija jednostavnih struktura podataka i asocijativnih listi
  - JSON-like (BSON – **B**inary **J**SON)
- Najpopularniji NoSQL DBMS, od [marta 2017.](#) peta po popularnosti DBMS (posle Oracle, MySQL, Microsoft SQL Server i PostgreSQL-a)
- Odlična podrška u Node.js aplikacijama (deo MEAN stack-a)

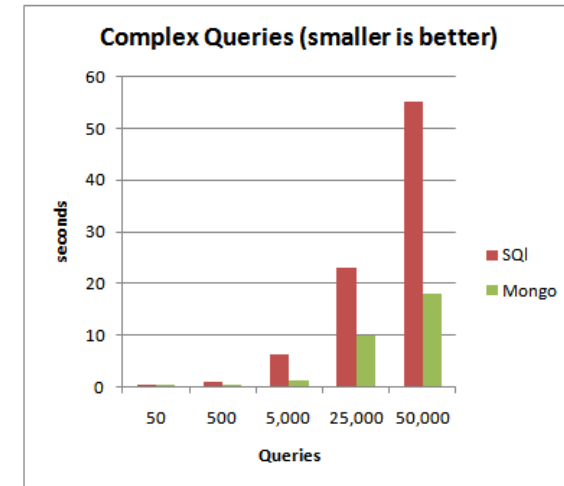
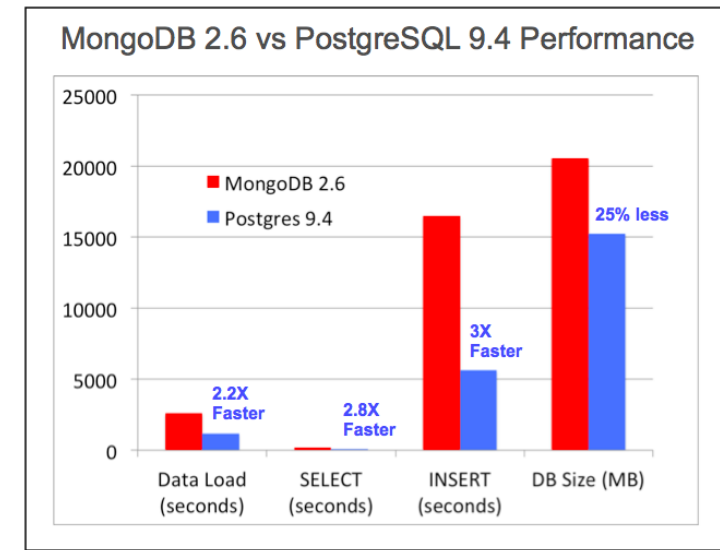




# MongoDB

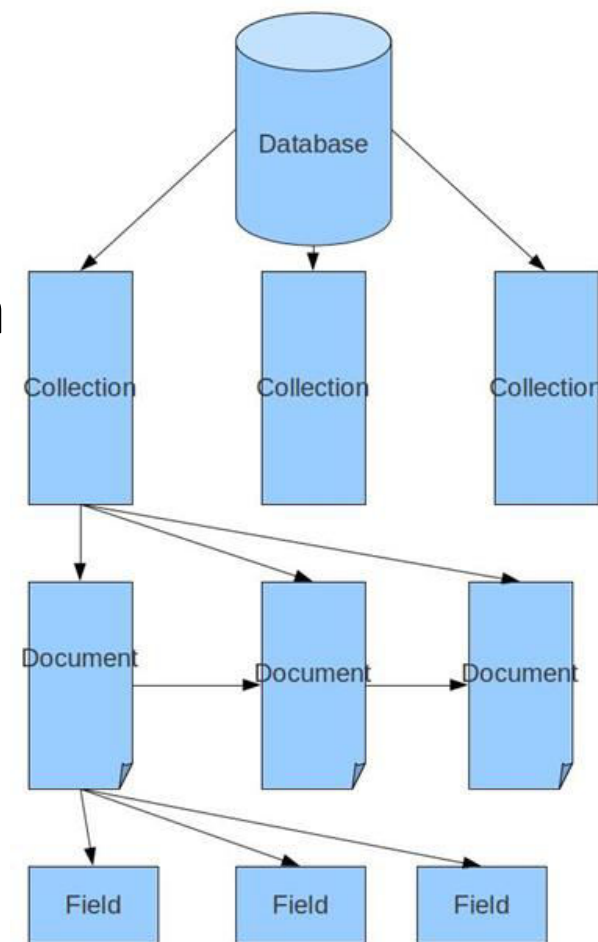
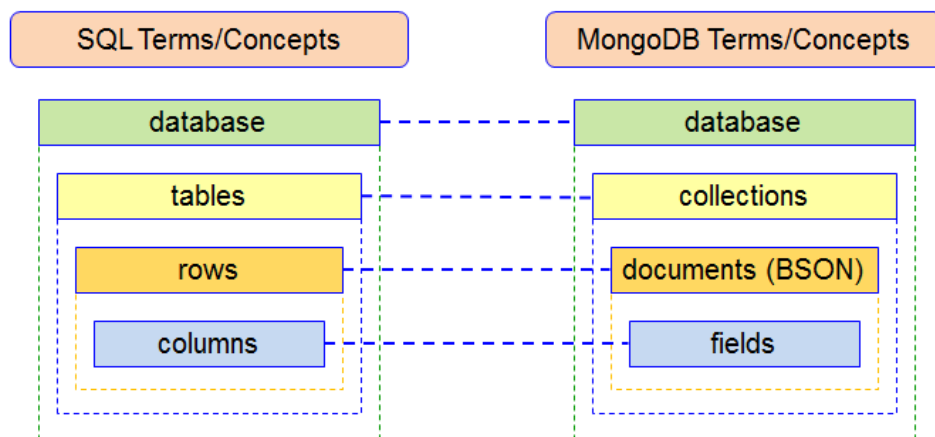
- Odlike:

- Ad hoc queries – po field-u, range queries, regex pretrage
- Indeksiranje – bilo kog field-a u dokumentu
- Replikacija – Master/Slave replikacija: Master Read/Write, Slave kopira podatke samo Read ili Backup - bez promene nad podacima
- Duplikacija podataka
- Load balancing – automatski konfigurisan
- Map reduce i aggregation alati
- JavaScript umesto procedura
- Schema-less db u C++
- Visoke performance
- Skladisti fajlove bilo koje velicine bez narušavanja steka
- Laka administracija



# MongoDB

- Svaka MongoDB instanca može da ima više baza podataka
- Svaka baza podataka može da ima više kolekcija
- Svaka kolekcija može da ima više dokumenata
- Svaki dokument može da ima više polja



# MongoDB

- Dokument: JSON dokument + par korisnih funkcionalnosti (na primer podrška za Date format)
- Kolekcija: kolekcija JSON dokumenata
  - Smeštanje dokumenata u istu kolekciju ne nameće shemu koju dokumenti moraju da zadovolje
  - Odsustvo sheme omogućuje da se jednostavno prave izmene u formatu dokumenata
    - Nedisciplinovanost u korišćenju ove osobine može da izazove velike probleme
- Baza podataka je skup kolekcija



# MongoDB

- Svaki dokument u kolekciji mora da ima vrednost za `_id` polje
- Mora da bude jedinstvena na nivou kolekcije
- Ukoliko se ne postavi vrednost za `_id`, MongoDB postavlja automatski generisani `ObjectID`
  - Heksadecimalni 24-cifreni broj
  - 539ed1d9f7da431c00026e18

0	1	2	3	4	5	6	7	8	9	10	11
time				machine			pid		inc		

- Zašto ne autoinkrement?
  - Teško je koristiti autoinkrement u distribuiranim sistemima (kako da znamo koja bila prošla vrednost ako je baza distribuirana?)



# MongoDB

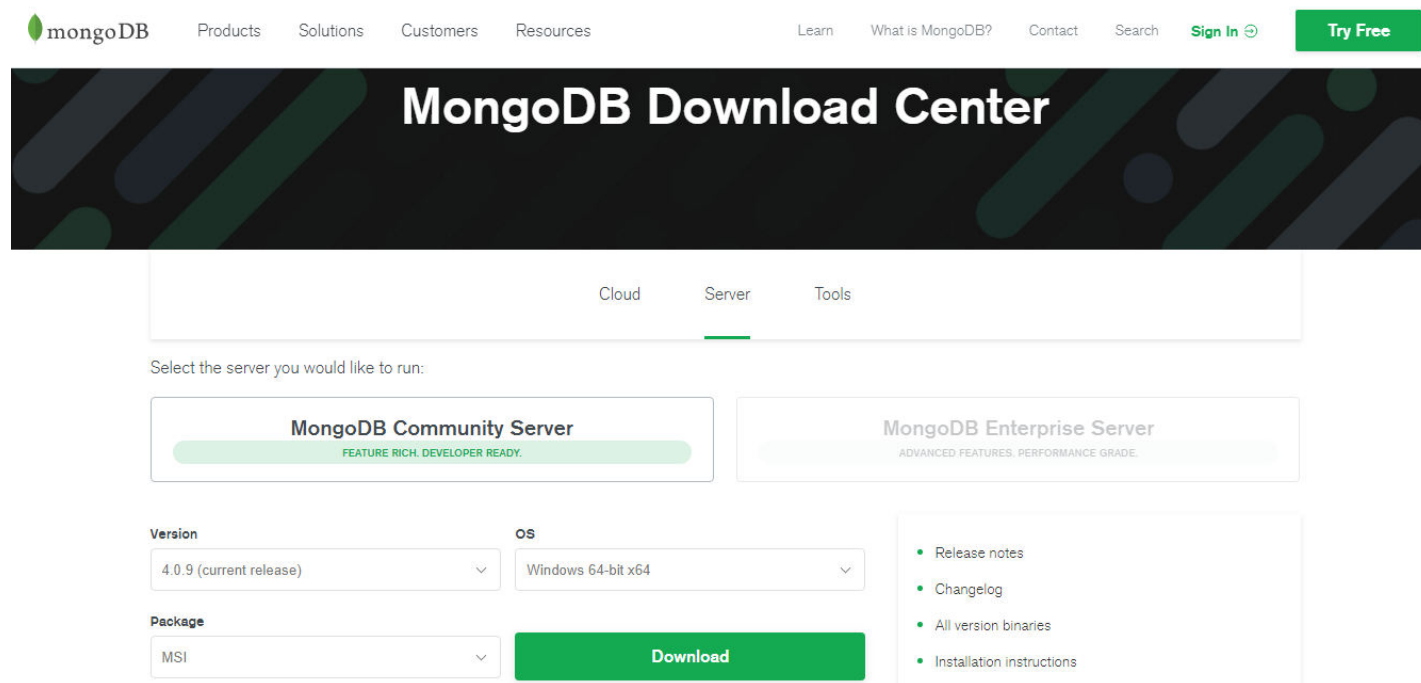
- Tipovi podataka

Data Types	Description
String	String is the most commonly used datatype. It is used to store data. A string must be UTF 8 valid in mongodb.
Integer	Integer is used to store the numeric value. It can be 32 bit or 64 bit depending on the server you are using.
Boolean	This datatype is used to store boolean values. It just shows YES/NO values.
Double	Double datatype stores floating point values.
Min/Max Keys	This datatype compare a value against the lowest and highest bson elements.
Arrays	This datatype is used to store a list or multiple values into a single key.
Object	Object datatype is used for embedded documents.
Null	It is used to store null values.
Symbol	It is generally used for languages that use a specific type.
Date	This datatype stores the current date or time in unix time format. It makes you possible to specify your own date time by creating object of date and pass the value of date, month, year into it.



# MongoDB

- Instalacija
- [mongodb.com](https://mongodb.com)
- Potreban je data/db direktorijum
- Default lokacija za MongoDB:
  - C:\Program Files\MongoDB
- Moze se promeniti u toku instalacije – Custom
- Default lokacija za data direktorijum:
  - C:\data\db
- Ako se koristi drugi direktorijum, potrebno je specificirati:
  - `mongod -- dbpath "{lokacija do direktorijuma}\data"`



# MongoDB

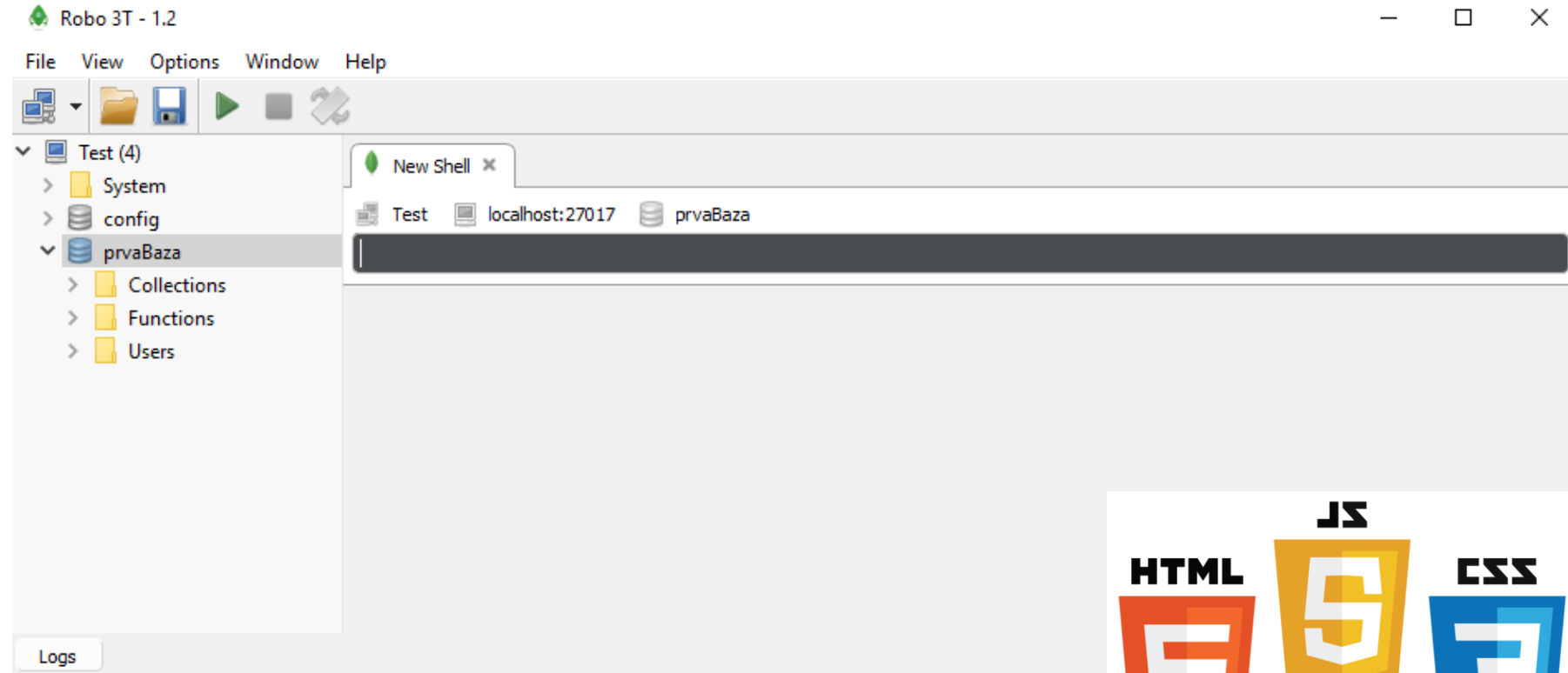
- Robo 3T – native, cross-platform MongoDB manager
- [robomongo.org](http://robomongo.org)

## Robo 3T

Robo 3T (formerly Robomongo) is the free lightweight GUI for MongoDB enthusiasts.

- GUI for MongoDB with embedded shell

Download Robo 3T



# MongoDB

- Primer MongoDB dizajna baze za veb-sajt klijenta:
- Zahtev:
  - Svaki post je jedinstven: jedinstven naslov, url i opis
  - Svaki post ima jedan ili više tagova
  - Svaki post ima ime publisher-a i ukupan broj lajkova
  - Svaki post ima 0 ili više komentara koji sadrže ime korisnika, poruku, vreme i datum i lajkove

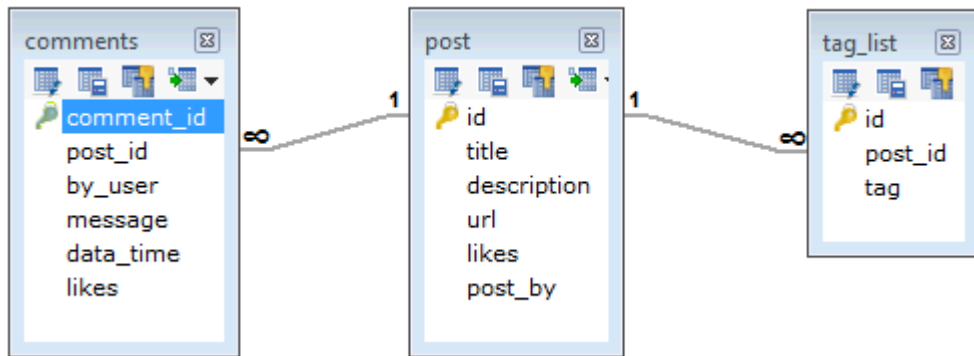




# MongoDB

- Primer MongoDB dizajna baze za vebsajt klijenta:

## 1. Primer tabela u relacionoj bazi podataka



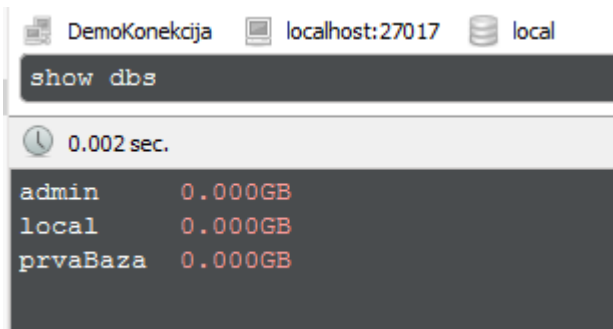
## 2. MongoDB dizajn:

```
{
  _id: POST_ID
  title: TITLE_OF_POST,
  description: POST_DESCRIPTION,
  by: POST_BY,
  url: URL_OF_POST,
  tags: [TAG1, TAG2, TAG3],
  likes: TOTAL_LIKES,
  comments: [
    {
      user: 'COMMENT_BY',
      message: TEXT,
      datecreated: DATE_TIME,
      like: LIKES
    },
    {
      user: 'COMMENT_BY',
      message: TEST,
      dateCreated: DATE_TIME,
      like: LIKES
    }
  ]
}
```

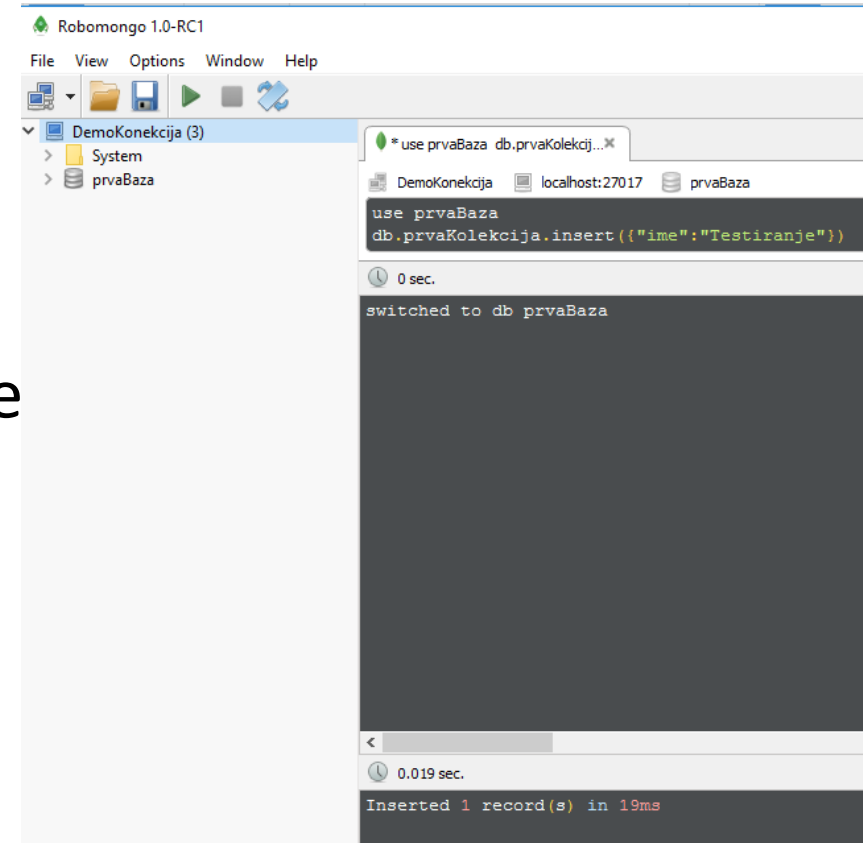


# MongoDB

- Kreiranje baze
- **use IME\_BAZE** ako postoji vratiće postojeću, ako ne kreiraće je
- Kreirati jedan document, inače neće baza biti u listi
- **db** proverava se trenutno čekirana baza
- **show dbs** proverava se lista baza podataka



```
DemoKonekcija localhost:27017 local
show dbs
0.002 sec.
admin    0.000GB
local    0.000GB
prvaBaza 0.000GB
```



```
Robomongo 1.0-RC1
File View Options Window Help
DemoKonekcija (3)
  System
  prvaBaza
*use prvaBaza db.prvaKolekcij...
DemoKonekcija localhost:27017 prvaBaza
use prvaBaza
db.prvaKolekcija.insert({"ime":"Testiranje"})
0 sec.
switched to db prvaBaza
0.019 sec.
Inserted 1 record(s) in 19ms
```



# MongoDB

## Brisanje baze i njenih fajlova - **dropDatabase()**

DemoKonekcija localhost:27017 prvaBaza

```
use prvaBaza  
db.dropDatabase()
```

0 sec.

switched to db prvaBaza

DemoKonekcija localhost:27017 local

```
show dbs
```

0.002 sec.

```
admin 0.000GB  
local 0.000GB
```

0.003 sec.

Key

▼ (1)

dropped

ok

Value

{ 2 fields }

prvaBaza

1.0

Type

Object

String

Double



# MongoDB

- Kreiranje kolekcije - **db.createCollection(name, options)**
- name – ime kolekcije, options su opcione: velicina, indeksiranje, tip dokumenta...

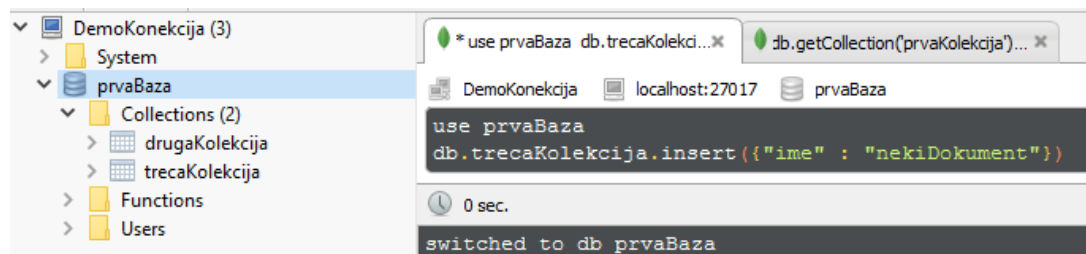
DemoKonekcija localhost:27017 prvaBaza

```
use prvaBaza
show collections
```

DemoKonekcija localhost:27017 prvaBaza

```
use prvaBaza
db.createCollection("drugaKolekcija")
```

- Kreiranje kolekcije automatski prilikom kreiranja dokumenta

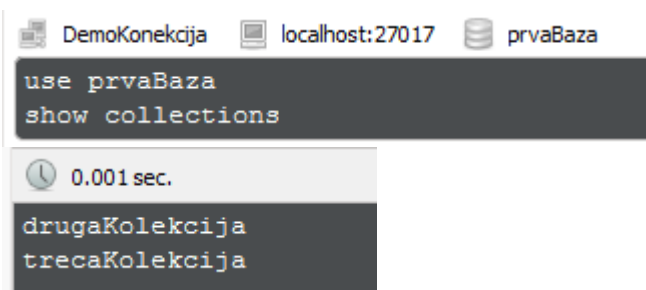


Field	Type	Description
Capped	Boolean	(Optional) If it is set to true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. If you specify true, you need to specify size parameter also.
AutoIndexID	Boolean	(Optional) If it is set to true, automatically create index on ID field. Its default value is false.
Size	Number	(Optional) It specifies a maximum size in bytes for a capped collection. If capped is true, then you need to specify this field also.
Max	Number	(Optional) It specifies the maximum number of documents allowed in the capped collection.



# MongoDB

- Brisanje kolekcije i njenih indeksa - **db.ime\_kolekcije.drop()**



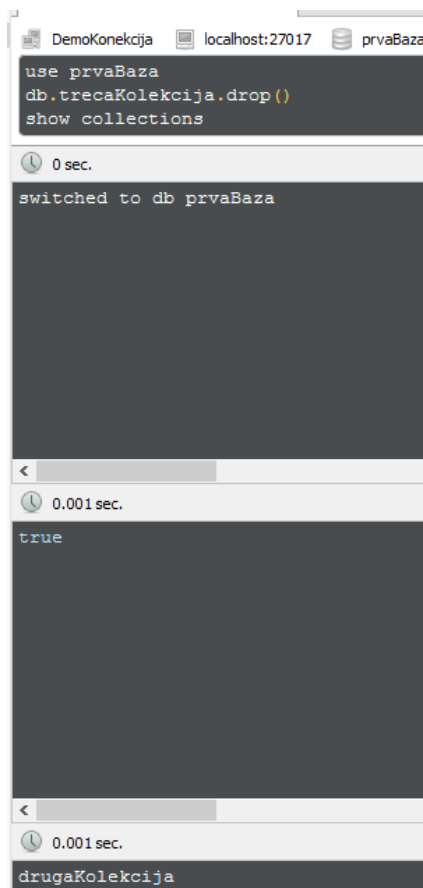
DemoKonekcija localhost:27017 prvaBaza

```
use prvaBaza
show collections
```

0.001 sec.

```
drugaKolekcija
trecaKolekcija
```

This screenshot shows the MongoDB Shell interface. The top bar indicates the connection to 'prvaBaza' on 'localhost:27017'. The command prompt shows 'use prvaBaza' and 'show collections'. The execution time is 0.001 seconds. The output lists two collections: 'drugaKolekcija' and 'trecaKolekcija'.



DemoKonekcija localhost:27017 prvaBaza

```
use prvaBaza
db.trecaKolekcija.drop()
show collections
```

0 sec.

switched to db prvaBaza

0.001 sec.

```
true
```

0.001 sec.

```
drugaKolekcija
```

This screenshot shows the MongoDB Shell interface after dropping the 'trecaKolekcija' collection. The top bar indicates the connection to 'prvaBaza' on 'localhost:27017'. The command prompt shows 'use prvaBaza', 'db.trecaKolekcija.drop()', and 'show collections'. The execution time for the drop command is 0 seconds. The output shows 'switched to db prvaBaza'. The execution time for the 'show collections' command is 0.001 seconds. The output is 'true', indicating the collection was successfully dropped. The execution time for the 'show collections' command is 0.001 seconds. The output lists only one collection: 'drugaKolekcija'.



# MongoDB

- CRUD
- Dodavanje novog dokumenta u kolekciju-  
**db.ime\_kolekcije.insert(document)**

```
use prvaBaza
db.kursevi.insert(
  {
    course: "veb programiranje",
    details: {
      duration: "6 meseci",
      Trainer: "M.A"
    },
    Students: [ { name: "Perica", id: 1 }, { name: "Jovica", id: 2 } ],
    category: "MongoDB"
  }
)
```

```
use prvaBaza
db.kursevi.find()

0 sec.
switched to db prvaBaza

kursevi 0.001 sec.

/* 1 */
{
  "_id" : ObjectId("58c6ce92fbecaafd28610185"),
  "course" : "veb programiranje",
  "details" : {
    "duration" : "6 meseci",
    "Trainer" : "M.A"
  },
  "Students" : [
    {
      "name" : "Perica",
      "id" : 1.0
    },
    {
      "name" : "Jovica",
      "id" : 2.0
    }
  ],
  "category" : "MongoDB"
}
```



# MongoDB

- CRUD
- Dodavanje vise dokumenata, prosledjuje se niz dokumenata **insert()** metodi

```
use prvaBaza
var sviKursevi =
[
  {
    course: "MongoDB",
    details: {
      duration: "6 meseci",
      Trainer: "M.A"
    },
    Students: [ { name: "Perica", id: 1 }, { name: "Jovica", id: 2 } ],
    category: "veb programiranje"
  },
  {
    course: "Node.js",
    details: {
      duration: "6 meseci",
      Trainer: "M.A"
    },
    Students: [ { name: "Perica", id: 1 }, { name: "Jovica", id: 2 } ],
    category: "veb programiranje"
  },
  {
    course: "Full Stack JS",
    details: {
      duration: "6 meseci",
      Trainer: "M.A"
    },
    Students: [ { name: "Perica", id: 1 }, { name: "Jovica", id: 2 } ],
    category: "veb programiranje"
  }
];
db.kursevi.insert(sviKursevi)
```



# MongoDB

- CRUD
- Dodavanje vise dokumenata – **bulk insert**
- Inicijalizuje se prvo bulk operation builder za “kursevi” kolekciju – odrzava listu operacija koje trebaju da se izvrse
- Definisu se insert operacije
- Sa **execute()** se izvrsavaju operacije iz liste

```
DemoKonekcija localhost:27017 prvaBaza
use prvaBaza
var bulk = db.kursevi.initializeUnorderedBulkOp();
bulk.insert(
  {
    course: "Osnove JavaScript-a",
    details: {
      duration: "6 meseci",
      Trainer: "M.A"
    },
    Students: [ { name: "Perica", id: 1 }, { name: "Jovica", id: 2 } ],
    category: "veb programiranje"
  }
);
bulk.execute();
```





# MongoDB

- CRUD
- Azuriranje dokumenata -  
**`db.ime_kolekcije.update(kriterijum_odabira, azurirani_podaci)`**

```
DemoKonekcija localhost:27017 prvaBaza
use prvaBaza
db.kursevi.update({'course':'Java'},{$set: {'course':'android'}});
db.kursevi.find()
```

0 sec.

switched to db prvaBaza

0.003 sec.

Updated 1 existing record(s) in 3ms

0.001 sec.

Key	Value
> (1) ObjectId("58c6ce92fbecaafd28610185")	{ 5 fields }
> (2) ObjectId("58c6cfd4fbecaafd28610186")	{ 5 fields }
> (3) ObjectId("58c6cfd4fbecaafd28610187")	{ 5 fields }
> (4) ObjectId("58c6cfd4fbecaafd28610188")	{ 5 fields }
> (5) ObjectId("58c6d31bfecaafd28610189")	{ 5 fields }
▼ (6) ObjectId("58c6d3aefbcaafd2861018a")	{ 5 fields }
_id	ObjectId("58c6d3aefbcaafd2861018a")
course	android
> details	{ 2 fields }
> Students	[ 2 elements ]
category	programski jezici



Brisanje samo prvog koji zadovolji kriterijum

# MongoDB

- CRUD
- Brisanje podataka
- **db.ime\_kolekcije.remove (kriterijum\_brisanja)**

DemoKonekcija localhost:27017 prvaBaza

```
use prvaBaza
db.kursevi.remove( { category : "programski jezici" } )
```

0 sec.

switched to db prvaBaza

0.002 sec.

Removed 1 record(s) in 2ms

- Brisanje svih

DemoKonekcija localhost:27017 prvaBaza

```
use prvaBaza
db.kursevi.remove({})
```

DemoKonekcija localhost:27017 prvaBaza

```
use prvaBaza
db.kursevi.remove( { category : "veb programiranje" }, 1 )
```

0 sec.

switched to db prvaBaza

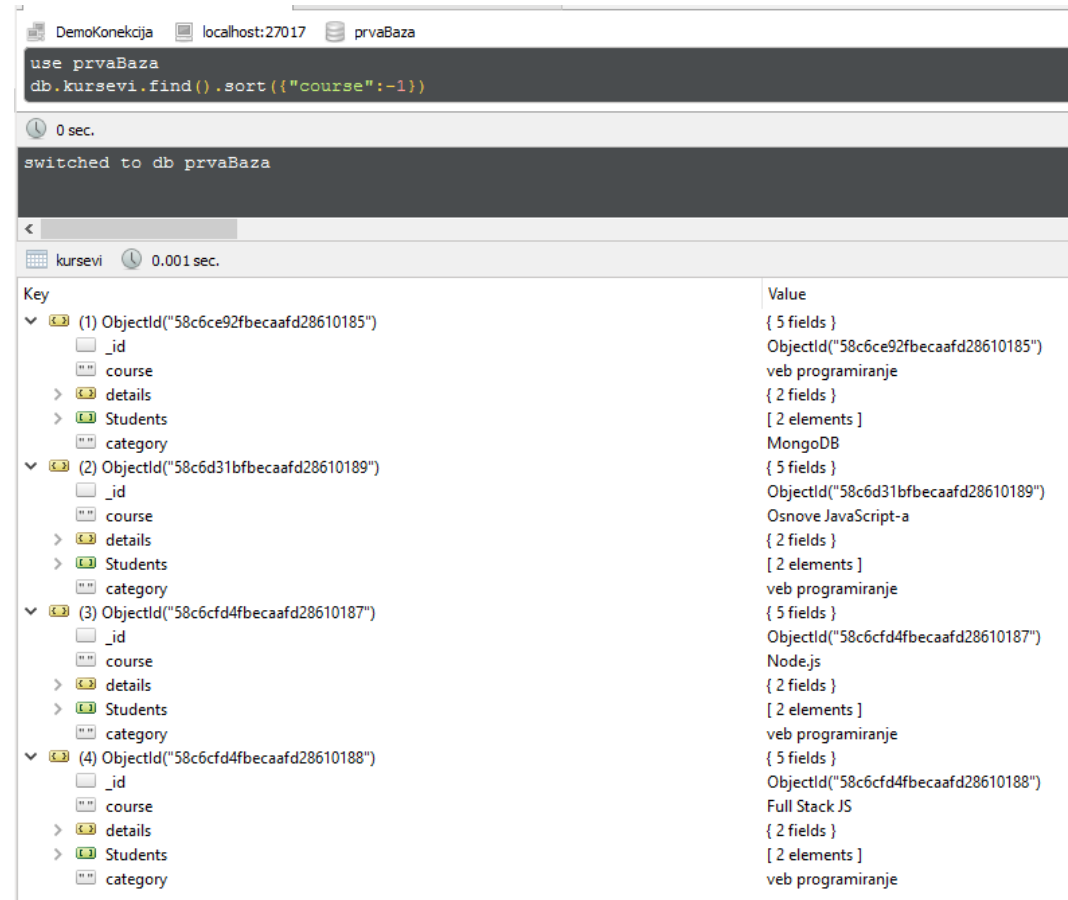
0.001 sec.

Removed 1 record(s) in 1ms



# MongoDB

- Upiti
- **`db.ime_kolekcije.find({})`** - izvlaci sve dokumente iz kolekcije
- **`db.ime_kolekcije.find().limit(broj)`** - izvlaci ogranicen broj iz kolekcije
- **`db.ime_kolekcije.find().limit(broj).skip(broj)`** preskace document
- **`db.ime_kolekcije.find().sort({kljuc:1})`** - sortira 1 rastuce, -1



# MongoDB i Node.js

```
server.js
1 //import mongodb native driver-a
2 var mongodb = require('mongodb');
3
4 //MongoClient - interfejs za konekciju na mongodb server
5 var MongoClient = mongodb.MongoClient;
6
7 //URL konekcije
8 var url = 'mongodb://localhost:27017/prvaBaza';
9
10 //connect metoda za konekciju na server
11 MongoClient.connect(url, { useNewUrlParser: true }, function(err, client) {
12   if (err) {
13     console.log('Greska: ' + err);
14   } else {
15     console.log('Konekcija uspostavljena ' + url);
16
17     var db = client.db('prvaBaza');
18
19     //Kreiranje kursa
20     var course = {
21       course: "MongoDB + Node.js",
22       details: {
23         duration: "6 meseci",
24         trainer: "M.A"
25       },
26       students: [{name: "Perica", id:1}, {name: "Jovica", id:2}],
27       category: "veb programiranje"
28     };
29
30     //Insert
31     db.collection('kursevi').insertOne(course, function(err, result){
32       if(err){
33         console.log(err);
34       }else{
35         console.log('Kreiran dokument');
36       }
37       //Zatvaranje konekcije
38       client.close();
39     });
40   }
41 });
```

Primer: **INSERT**

- mongodb paket – drajver za MongoDB iz Node.JS-a

npm install mongodb

Key	Value
(1) ObjectId("58c6da308d2f3f2f10a3d3ec")	{ 5 fields }
_id	ObjectId("58c6da308d2f3f2f10a3d3ec")
course	MongoDB + NodeJS
details	{ 2 fields }
Students	[ 2 elements ]
category	veb programiranje

```
C:\WINDOWS\system32\cmd.exe
E:\MongoDB_Primeri>node server.js
Konekcija uspostavljena mongodb://localhost:27017/prvaBaza
Kreiran dokument
```



# MongoDB i Node.js

```
server.js
1 //import mongodb native driver-a
2 var mongodb = require('mongodb');
3
4 //MongoClient - interfejs za konekciju na mongodb server
5 var MongoClient = mongodb.MongoClient;
6
7 //URL konekcije
8 var url = 'mongodb://localhost:27017/prvaBaza';
9
10 //connect metoda za konekciju na server
11 MongoClient.connect(url, { useNewUrlParser: true }, function(err, client) {
12   if (err) {
13     console.log('Greska: ' + err);
14   } else {
15     console.log('Konekcija uspostavljena ' + url);
16
17     var db = client.db('prvaBaza');
18
19     //Update
20     db.collection('kursevi').updateOne({course: 'MongoDB + Node.js'},{$set: {course:'Node.js'}}, function(err, numUpdated){
21       if(err){
22         console.log(err);
23       } else if (numUpdated){
24         console.log('Update uspesan');
25       } else {
26         console.log('Kreiran dokument');
27       }
28       //Zatvaranje konekcije
29       client.close();
30     });
31   }
32 });
```

## Primer: UPDATE

DemoKonekcija localhost:27017 prvaBaza

```
db.getCollection('kursevi').find({})
```

kursevi 0.001 sec.

Key	Value
(1) ObjectId("58c6da308d2f3f2f10a3d3ec")	{ 5 fields }
_id	ObjectId("58c6da308d2f3f2f10a3d3ec")
course	NodeJS
details	{ 2 fields }
Students	[ 2 elements ]
category	veb programiranje

C:\WINDOWS\system32\cmd.exe

```
E:\MongoDB_Primeri>node server.js
Konekcija uspostavljena mongodb://localhost:27017/prvaBaza
Update uspesan
```



# MongoDB i Node.js

## Primer: Pretraga

```
server.js
1 //import mongodb native driver-a
2 var mongodb = require('mongodb');
3
4 //MongoClient - interfejs za konekciju na mongodb server
5 var MongoClient = mongodb.MongoClient;
6
7 //URL konekcije
8 var url = 'mongodb://localhost:27017/prvaBaza';
9
10 //connect metoda za konekciju na server
11 MongoClient.connect(url, { useNewUrlParser: true }, function(err, client) {
12   if (err) {
13     console.log('Greska: ' + err);
14   } else {
15     console.log('Konekcija uspostavljena ' + url);
16
17     var db = client.db('prvaBaza');
18
19     //Pretraga
20     var criteria = {
21       'details.duration' : '6 meseci'
22     };
23
24     db.collection('kursevi').find(criteria).toArray(function(err, result){
25       if(err){
26         console.log(err);
27       } else if (result.length){
28         console.log('Pronadjen: ', result);
29       } else {
30         console.log('Nema dokumenta!');
31       }
32
33       //Zatvaranje konekcije
34       client.close();
35     });
36   }
37 });
```

C:\WINDOWS\system32\cmd.exe

```
E:\MongoDB_Primeri>node server.js
Konekcija uspostavljena mongodb://localhost:27017/prvaBaza
Pronadjen: [ { _id: 58c6da308d2f3f2f10a3d3ec,
  course: 'Node.JS',
  details: { duration: '6 meseci', Trainer: 'M.A' },
  Students: [ [Object], [Object] ],
  category: 'veb programiranje' } ]
```



# MongoDB i Node.js

- [Mongoose](#)
- MongoDB rukuje jednostavnim JSON dokumentima
- Poslovna logika je smeštena u aplikativni sloj
- Mongoose ODM nam omogućuje jednostavnu konverziju između dokumenata i JavaScript objekata
  - Podaci
  - Metoda za validaciju
  - Poslovna logika



# MongoDB i Node.js

- Mongoose
- Express REST Primer

URL	HTTP Verb	POST Body	Result
/api/Kursevi	GET	Prazan	Vraća sve kurseve
/api/Kursevi	POST	JSON String	Novi kurs dodat
/api/Kursevi/:id	GET	Prazan	Vraća jedinstven kurs
/api/Kursevi/:id	PUT	JSON String	Ažurira postojeći kurs
/api/Kursevi/:id	DELETE	Prazan	Briše postojeći kurs





# MongoDB i Node.js

- Mongoose
- Struktuiranje podataka
- Ne može se skladištiti ništa što ne odgovara šemi
- Šema služi kao deklaracija kako treba da izgledaju podaci

```
1  var mongoose=require('mongoose');
2  var Schema=mongoose.Schema;
3
4  var kursSchema=new Schema({
5      naslov:String,
6      semestar:'String',
7      izvodjac:'String',
8      kategorija:'String'
9  });
10
11 module.exports=mongoose.model('Kurs',kursSchema);
12
```



# MongoDB i Node.js

- Mongoose
- Modeli su prave klase kreirane u JavaScript-i

```
1  var mongoose=require('mongoose');
2  var Schema=mongoose.Schema;
3
4  var kursSchema=new Schema({
5      naslov:String,
6      semestar:'String',
7      izvodjac:'String',
8      kategorija:'String'
9  });
10
11  module.exports=mongoose.model('Kurs',kursSchema);
12
```



# MongoDB i Node.js

- Mongoose
- Upiti
- Upiti ka bazi su zahtevni prema resursima
- Zato se koriste kroz **Callback**-ove
- Šalje se zahtev, obrađuju se drugi poslovi, kada se dobije povratna vrednost upita, izvršava se callback funkcija
- Upiti uvek u formatu:  
**function(error, result) {...}**

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
var Kurs=require('../models/kursevi');
var express=require('express');

//konfigurisanje ruta

var router=express.Router();

router.route('/Kursevi')
  .get(function(req,res){
    Kurs.find(function(err,kurs){
      if(err)
        res.send(err);
      res.json(kurs);
    });
  })
  .post(function(req,res){
    var kurs= new Kurs(req.body);
    kurs.save(function(err){
      if(err)
        res.send(err);
      res.send({message:'Kurs je dodat'});
    });
  });
```

Iščitavanje

Dodavanje



# MongoDB i Node.js

- Mongoose
- Primer

```
27 router.route('/:id')
28   .put(function(req,res) {
29     Kurs.findOne({_id:req.params.id},function(err,kurs) {
30
31       if(err)
32         res.send(err);
33
34       for(prop in req.body) {
35         kurs[prop]=req.body[prop];
36       }
37
38       //snimanje kursa
39       kurs.save(function(err) {
40         if (err)
41           res.send(err);
42
43         res.json({ message: 'Kurs je azuriran!' });
44       });
45     });
46   });
47
48   .get(function(req,res) {
49     Kurs.findOne({_id:req.params.id},function(err, kurs) {
50       if(err)
51         res.send(err);
52
53       res.json(kurs);
54     });
55   });
56
57   .delete(function(req,res) {
58     Kurs.remove({
59       _id: req.params.id
60     }, function(err, Kurs) {
61       if (err)
62         res.send(err);
63
64       res.json({ message: 'Kur je uspesno obrisan' });
65     });
66   });
67
68
69 module.exports=router;
```

Ažuriranje

Pretraga po kriterijumu

Brisanje



# MongoDB i Node.js

- Mongoose
- Testiranje primera

## Baza

The screenshot shows the MongoDB Compass interface. On the left, a sidebar displays the database structure: 'DemoKonekcija (4)' containing 'System', 'kurseviDB', 'Collections (1)', 'kurs', 'Functions', and 'Users'. The 'kurs' collection is selected. The main panel shows a query: `db.getCollection('kurs').find({})` with a result set containing one document. The document has the following fields: `_id` (ObjectId), `naslov` (Veb programiranje), `semestar` (I), `izvodjac` (M.A), `kategorija` (Uvod u MongoDB), and `_v` (0).

Key	Value
(1) ObjectId("58c7273bc8a9df9c2cbdf8c8")	{ 6 fields }
<code>_id</code>	ObjectId("58c7273bc8a9df9c2cbdf8c8")
<code>naslov</code>	Veb programiranje
<code>semestar</code>	I
<code>izvodjac</code>	M.A
<code>kategorija</code>	Uvod u MongoDB
<code>_v</code>	0

## Dodavanje

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:8000/api/Kursevi`. The 'Body' tab is selected, showing a JSON payload: `{ "naslov": "Veb programiranje", "semestar": "I", "izvodjac": "M.A", "kategorija": "Uvod u MongoDB" }`. The bottom bar shows the 'Body' tab with a 'Pretty' button and a 'JSON' dropdown. The response is displayed in the 'Body' tab, showing a JSON object: `{ "message": "Kurs je dodat" }`.



# MongoDB i Node.js

- Mongoose
- Testiranje primera

## Baza

DemoKonekcija localhost:27017 kurseviDB

```
db.getCollection('kurs').find({})
```

kurs 0.001 sec.

Key	Value
(1) ObjectId("58c7273bc8a9df9c2cbdf8c8")	{ 6 fields }
_id	ObjectId("58c7273bc8a9df9c2cbdf8c8")
naslov	Veb programiranje
semestar	I
izvodjac	M.A
kategorija	Uvod u MongoDB i Node.js
_v	0

## Iscitavanje

localhost:8000/api/Kursevi

```
[{"_id":"58c7273bc8a9df9c2cbdf8c8","naslov":"Veb programiranje","semestar":"I","izvodjac":"M.A","kategorija":"Uvod u MongoDB","__v":0}]
```

## Azuriranje

http://localhost:8000/ +

PUT http://localhost:8000/api/Kursevi/58c7273bc8a9df9c2cbdf8c8

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "naslov": "Veb programiranje",
3   "semestar": "I",
4   "izvodjac": "M.A",
5   "kategorija": "Uvod u MongoDB i Node.js"
6 }
7
8
9
```

Body Cookies Headers (5) Tests

Pretty Raw Preview JSON

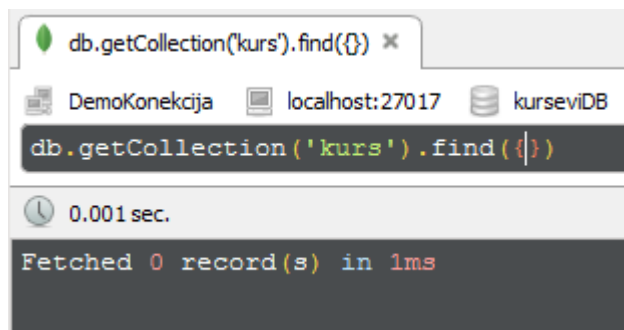
```
1 {
2   "message": "Kurs je azuriran!"
3 }
```



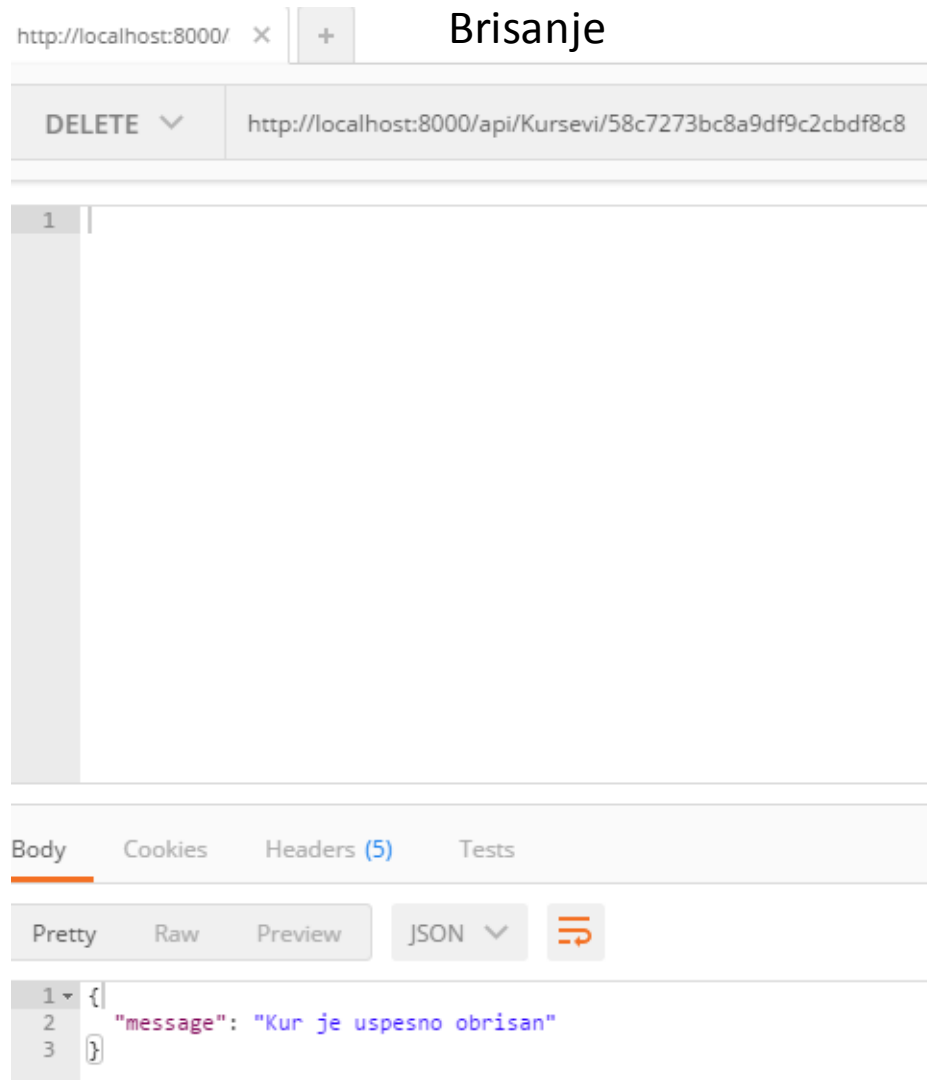
# MongoDB i Node.js

- Mongoose
- Testiranje primera

## Baza



A screenshot of the MongoDB Shell interface. The top bar shows the command `db.getCollection('kurs').find({})`. Below it, the connection details are `DemoKonekcija`, `localhost:27017`, and `kurseviDB`. The command `db.getCollection('kurs').find({})` is entered in the input field. The execution time is `0.001 sec.`. The result is `Fetches 0 record(s) in 1ms`.



A screenshot of the Postman application showing a DELETE request. The URL is `http://localhost:8000/api/Kursevi/58c7273bc8a9df9c2cbdf8c8`. The response body is `{ "message": "Kur je uspesno obrisan" }`.



# MongoDB i Node.js

- ZADATAK
- Napraviti REST API za rukovođenje knjigama unutar biblioteke
- Svaka knjiga ima naslov, id autora, da li je iznajmljena, kojoj kategoriji pripada, broj strana
- Svaki autor ima ime, godinu rođenja, nacionalnost, žanr
- Svaka kategorija ima ime, naziv sektora gde se nalazi ta kategorija u biblioteci
- Omogućiti:
  - Dodavanje, brisanje, ažuriranje i pregled: autora, knjiga i kategorija unutar biblioteke
  - Pretragu knjiga po autoru ili kategoriji
  - Pretragu autora po kategoriji

