

# NodeJS – MongoDB - Mongoose

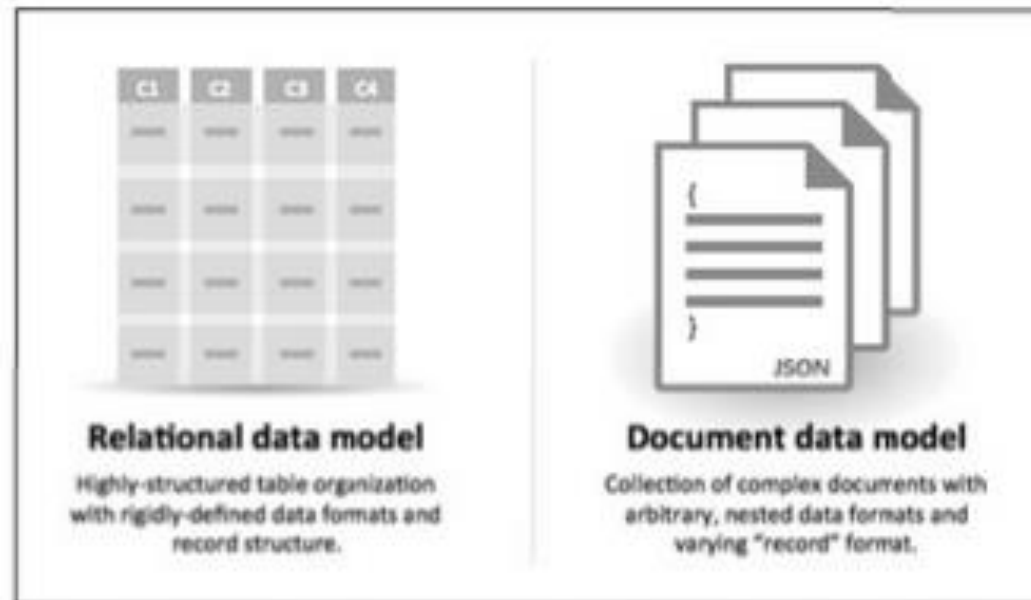


# Inleiding Databases en Node.Js

- Er bestaan verschillende drivers voor het samenwerken met verschillende databases
    - MySQL, SQL server, MongoDB, ...
  - Wij leggen de focus op MongoDB in combinatie met de npm-module mongoose
- 
- Download de voorbeelden uit Resources/CH12\_Voorbeelden

# Inleiding Databases en Node.js

- Document databases (MongoDB)
  - NoSQL database
  - Type database dat JSON-objecten rechtstreeks kan opslaan in de vorm van documenten



# Inleiding Databases en Node.js

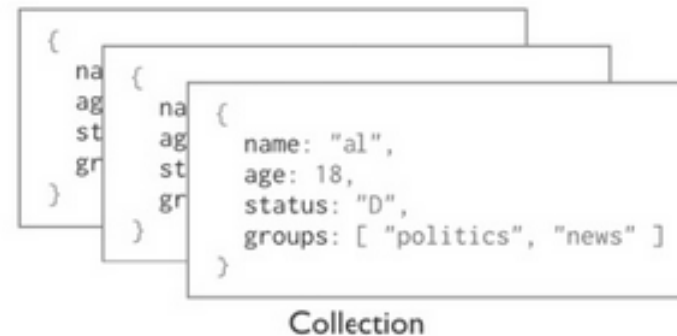
- Document databases (MongoDB)
  - Qua structuur zijn er geen voorgedefinieerde tabellen, records, joins en andere typische database-onderdelen, zoals in de RDBMS
  - De enige overeenkomsten tussen verschillende documenten in de database is dat ze een uniek (id)-veld hebben. In MongoDB heet dit letterlijk `_id` en is van het type `ObjectId()`

# Inleiding Databases en Node.js

- Document databases (MongoDB)
  - Het is een JSON-database die met JavaScript wordt bediend
  - Een document is een JSON-object
  - Een collection, een verzameling van documenten

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```

← field: value  
← field: value  
← field: value  
← field: value



# Inleiding Databases en Node.js

- Document databases (MongoDB)
  - Het veld `_id`
    - Wordt automatisch gemaakt bij het invoegen van nieuwe documenten
    - Het is een unieke sleutel met een random waarde
    - Vergelijkbaar met een primary key in RDBMS
    - Er is geen schema voorzien in MongoDB

```
{  
  "_id" : ObjectId("55ac8c69e16ba6e51644ce05"),  
  "titel" : "Web Development Library - AngularJS",  
  "auteur" : "Peter Kassenaar",  
  "ISBN" : "978059407879"  
}
```

# MongoDB installeren en in gebruik nemen

- MongoDB bestaat uit twee delen
  - Mongo Daemon: daemon die op de achtergrond draait, gegevens ophaalt en opslaat in databases. Deze moet als eerste gestart worden (**mongod.exe**)
  - Mongo shell: Dit is de CLI omgeving waarmee we opdrachten geven aan de daemon. Gebruik van databases, documenten toevoegen, documenten opvragen, ... (**mongo.exe**)

# MongoDB installeren en in gebruik nemen

- Installatiedetails voor de database zijn te vinden op [docs.mongodb.org](https://docs.mongodb.org) onder de rubriek “installation”



Afbeelding 6.5 De installatie van MongoDB wordt online uitgebreid toegelicht.



# MongoDB installeren en in gebruik nemen

- Ga naar <http://mongodb.org> en klik rechts boven op “get mongodb”
- Kies voor “server” en selecteer de community server en kies het juiste besturingssysteem
- Installeer de complete version (standaard in c:\Program files\mongodb)
- MongoDB Compass moet niet geïnstalleerd worden
- Maak een data directory. Hierin worden de databases opgeslagen
  - **De standaardlocatie is c:\data\db. Deze locatie nemen wij over. Maak zelf deze map in de verkenner.**

# MongoDB installeren en in gebruik nemen

---



## MongoDB starten als een service

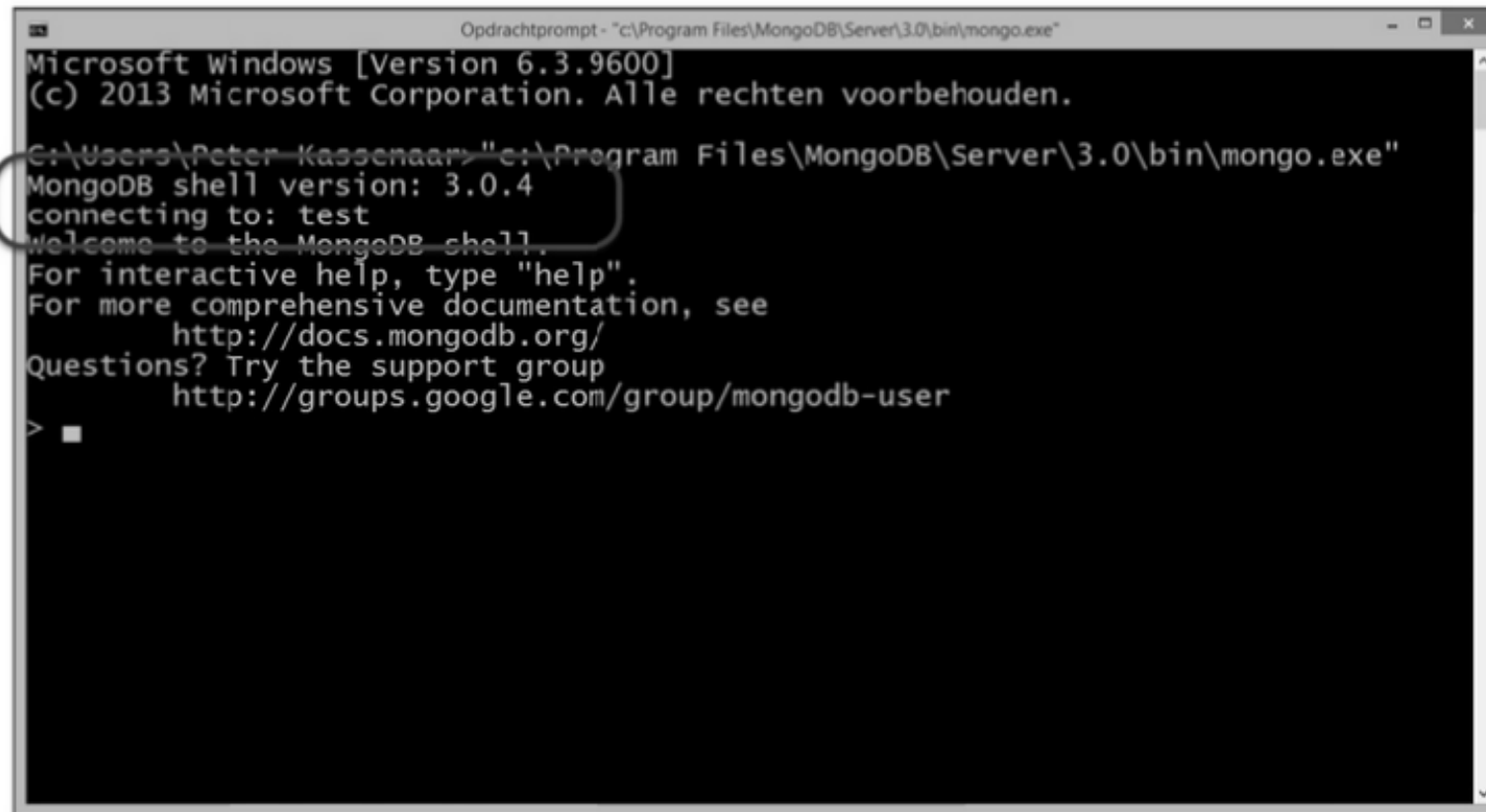
Let er op dat u telkens de daemon start als u aan het werk wilt met MongoDB. Het opdrachtvenster van de daemon moet geopend blijven (u mag het wel minimaliseren). Als u elke keer als de computer wordt gestart automatisch MongoDB wilt starten, kunt u Mongo installeren als een *service* in Windows. In de installatiedocumenten leest u hoe dat gaat.

---

# MongoDB installeren en in gebruik nemen

- MongoDB starten
  - Open een nieuw terminal venster. Start de MongoDB-deamon
    - Bij windows: `c:\program files\mongoDb\server\4.0\bin\mongod.exe`
    - Bij mac: `/Applications/mongodb/bin/mongod`
    - Gelukt indien de laatste regels “waiting for connnections” bevat.
  - Dit venster moet geopend blijven voor het runnen van de server!
  - Open een nieuw terminal venster en start de shell. Dit is mongo.exe in dezelfde folder

# MongoDB installeren en in gebruik nemen



```
Opdrachtprompt - "c:\Program Files\MongoDB\Server\3.0\bin\mongo.exe"
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\Peter Kassenaar>"c:\Program Files\MongoDB\Server\3.0\bin\mongo.exe"
MongoDB shell version: 3.0.4
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
    http://docs.mongodb.org/
Questions? Try the support group
    http://groups.google.com/group/mongodb-user
> ■
```

Afbeelding 6.6 De mongo-shell nadat hij voor de eerste keer gestart is.

# MongoDB installeren en in gebruik nemen

- U bent nu verbonden met de databaseserver en kunt databases en documenten gaan maken. De indeling is als volgt:
  - **Database** Er is meestal één database per app. Maar er kunnen meerdere databases bij een server worden opgeslagen.
  - **Collection** Een database bestaat uit verzamelingen (collections). In een collection worden de documenten opgeslagen. Je dat dit vergelijken met een tabel
  - **Document** Een document is één enkel item in de database. Het is best te vergelijken met een record uit een sql-database.

# Documenten opslaan/bewerken in de database

- Database maken en document toevoegen
  - Maak een nieuwe database

```
> use driesTest  
switched to db driesTest  
>
```

- Op de achtergrond heeft MongoDB in de data directory een aantal bestanden gemaakt.

# Documenten opslaan/bewerken in de database

- Database maken en documenten toevoegen
  - Volgende opdracht voegt een document in :

```
> db.users.insert({name: "Dries Swinnen"})
WriteResult({ "nInserted" : 1 })
>
```

- Een query uitvoeren: Opvragen van documenten

```
> db.users.find()
{ "_id" : ObjectId("5a5a6b67d7c6508a5f517806"), "name" : "Dries Swinnen" }
>
```

# Documenten opslaan/bewerken in de database

- Een query uitvoeren: andere zoekmethodes:

```
> db.users.find(ObjectId("5a5a6b67d7c6508a5f517806"))  
< { "_id" : ObjectId("5a5a6b67d7c6508a5f517806"), "name" : "Dries Swinnen" }  
>  
> db.users.find({name: 'Dries Swinnen'})  
< { "_id" : ObjectId("5a5a6b67d7c6508a5f517806"), "name" : "Dries Swinnen" }  
>
```



# Documenten opslaan/bewerken in de database

- Gegevens in documenten wijzigen
  - Hiervoor wordt de methode `.update()` gebruikt

```
> db.users.update({name: "Dries Swinnen"},{$set:{name: "John Doe"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.users.find()
{ "_id" : ObjectId("5a5a6b67d7c6508a5f517806"), "name" : "John Doe" }
```

- `$set` zorgt ervoor dat andere properties van dat document niet undefined worden

# Documenten opslaan/bewerken in de database

- Documenten verwijderen
  - Hiervoor wordt de methode `.remove()` gebruikt

```
> db.users.remove({})
WriteResult({ "nRemoved" : 1 })
>
```

# Documenten opslaan/bewerken in de database

- Meer CRUD-operaties
  - Meer info is terug te vinden op <https://docs.mongodb.com/manual/crud/>

```
db.<collectie>.find()  
db.<collectie>.insert()  
db.<collectie>.update()  
db.<collectie>.remove()
```

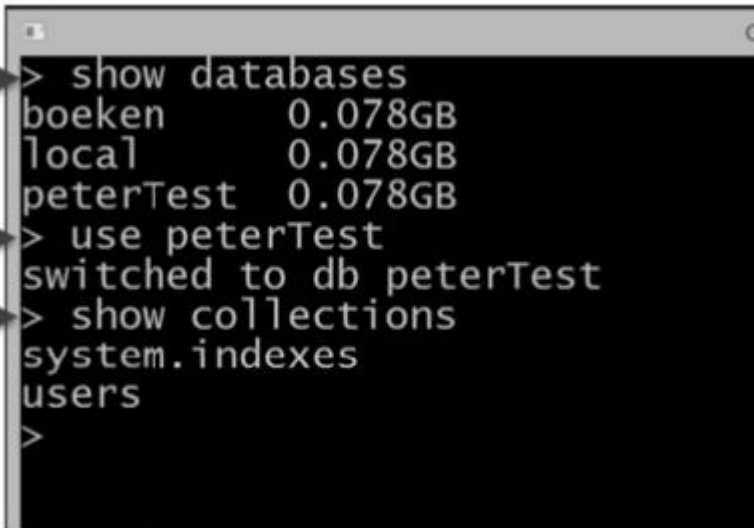
# Documenten opslaan/bewerken in dedatabase

- Databases opvragen

```
show databases  
show collections
```

- Drop databases

```
use <databasenaam>  
db.dropDatabase()
```



A screenshot of a MongoDB command prompt window. Three arrows point to the first three commands entered: 'show databases', 'use peterTest', and 'show collections'. The output shows three databases (boeken, local, peterTest) each 0.078GB, and then the collections (system.indexes, users) for the 'peterTest' database.

```
> show databases  
boeken      0.078GB  
local       0.078GB  
peterTest   0.078GB  
> use peterTest  
switched to db peterTest  
> show collections  
system.indexes  
users  
>
```

# Documenten opslaan/bewerken in de database

---



## JavaScript-bestand uitvoeren in MongoDB

In plaats van de opdrachten in de mongo-shell te typen, kunt u ze ook opslaan in een JavaScript-bestand en laden in de shell. De opdracht hiervoor is `load(mijnOpdrachten.js)`. In de paragraaf **Getting Started with the Mongo shell** in de online documentatie leest u meer van dit soort handige Mongo-tips.

---

# De rol van Mongoose

- Mongoose is een laag tussen Node.js en MongoDB en acteert als een Object Document Mapper of ODM
- Het vertaalt objecten (vanuit uw applicatie) naar documenten (voor gebruik in MongoDB) en omgekeerd
- Mongoose is een npm-module. Homepage en documentatie op [mongoosejs.com](https://mongoosejs.com)

# De rol van Mongoose

- Mongoose wordt aan de applicatie toegevoegd:

```
npm install mongoose --save
```

- Modellen en schema's: Basiscode voor een model 'user' ziet eruit als volgt:

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/driesTest', function(){
  console.log('MongoDb connected');
});

var User = mongoose.model('User', {name: String}); // model
var user = new User({name: 'Bob bobbie'}) // instance
user.save(function(err){
  console.log('user gemaakt');
});
```

# Een MongoDB CRUD app maken

- Voorbeeldcode: /1201



## CRUD?

CRUD is een bekende databaseterm en is een afkorting voor *Create*, *Retrieve*, *Update* en *Delete*, de meest voorkomende handelingen in databaseapplicaties.

---



# Een Mongoose CRUD app maken

- Centrale connectie met database
  - Maak een aparte file db.js voor de logica voor het verbinden met MongoDB
  - Zorg ervoor dat de mongoDB-deamon draait op de achtergrond

```
//db.js - logica voor verbinden MongoDB
var mongoose = require('mongoose');
var db = mongoose.connect('mongodb://localhost/boeken', function(){
  console.log('mongoose connected');
});
module.exports = db;
```

# Een Mongoose CRUD app maken

- Model voor boeken
  - Aparte file /models/boeken.js
  - Voorzien van een schema dat mapt naar de structuur van een collection
  - Koppeling aan een model
    - Uitbreiding van mongoose -- mongoose.model()

```
// boeken.js : Model voor boeken in MongoDB-database
var mongoose = require('mongoose');

const boekSchema = mongoose.Schema({
  titel : {type: String, required: true},
  auteur: {type: String, required: true},
  isbn  : {type: String, required: true},
  date  : {type: Date, required: true, default: Date.now}
});

var Boek = mongoose.model('Boek', boekSchema);
module.exports = Boek;
```

# Een Mongoose CRUD app maken

Verder ziet u dat het type voor een boek tamelijk strikt is vastgelegd. Dit doen we door velden op te geven en het type van het veld in het Mongoose-model te definiëren. Uiteraard gebeurt dit met een JavaScript-objectnotatie.

---



## Hoofdletter

Omdat een variabele `Boek` hier als het ware als een klasse fungeert, schrijven we deze met een hoofdletter (deze techniek wordt ook gebruikt in programmeertalen als Java en C#). Instanties van die klasse (de eigenlijke boek-objecten) worden later met een kleine letter gemaakt.

---

Omdat de variabele `Boek` is afgeleid van een Mongoose-object, heeft het ook methods als `.find()` en `.save()`. Dit gaat zeker nog van pas komen, bijvoorbeeld om nieuwe boeken in de database op te slaan.

# Een Mongoose CRUD app maken

- API-endpoints maken in de server
  - Gebruik maken van de Express module uit het vorige hoofdstuk.

```
// server.js - applicatie voor het ophalen en
// opslaan van boeken in MongoDB
var express = require('express');
var bodyParser = require('body-parser');
var db = require('./db');
var Boek = require('./models/boeken');

var app = express();
app.use(bodyParser.json());

// 1. Eenvoudige instructie
app.get('/api', function (req, res) {
  res.json({'Gebruik': 'voer een GET of POST-call uit naar /boeken'});
});

// 2. POST-endpoint: nieuw boek in de database plaatsen.
app.post('/api/boeken', function (req, res, next) {
  // TODO
});

app.listen(3000, function () {
  console.log('server gestart op poort 3000');
});
```

- Best opsplitsen a.d.h.v. Express router!

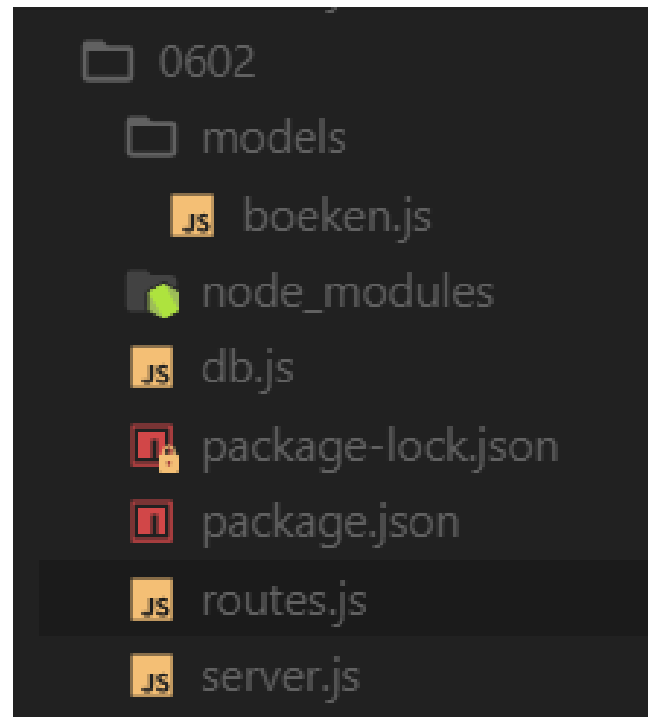
# Een Mongoose CRUD app maken

- `bodyParser.json()`
  - In het vorige hoofdstuk maakte we gebruik van `bodyParser.urlencoded(...)`
  - Binnen Angular >2 gebruikt de `HttpClientModule` standaard JSON om een request te versturen
  - In onze backend geven we dus aan dat de body van onze binnenkomende request van het type JSON is via de middleware functie.

```
app.use(bodyParser.json());
```

# Een Mongoose CRUD app maken

- Structuur backend applicatie



# Een Mongoose CRUD app maken

- POST-endpoint aanvullen met onze code:
  - In aparte file routes.ts

```
router.post('/boeken', (req, res) => {  
  let boek = new Boek({  
    titel: req.body.titel,  
    auteur: req.body.auteur,  
    isbn: req.body.isbn  
  });  
  
  boek.save((err, boek) => {  
    if(err)  
      res.send(err);  
  
    res.json(boek);  
  });  
});
```

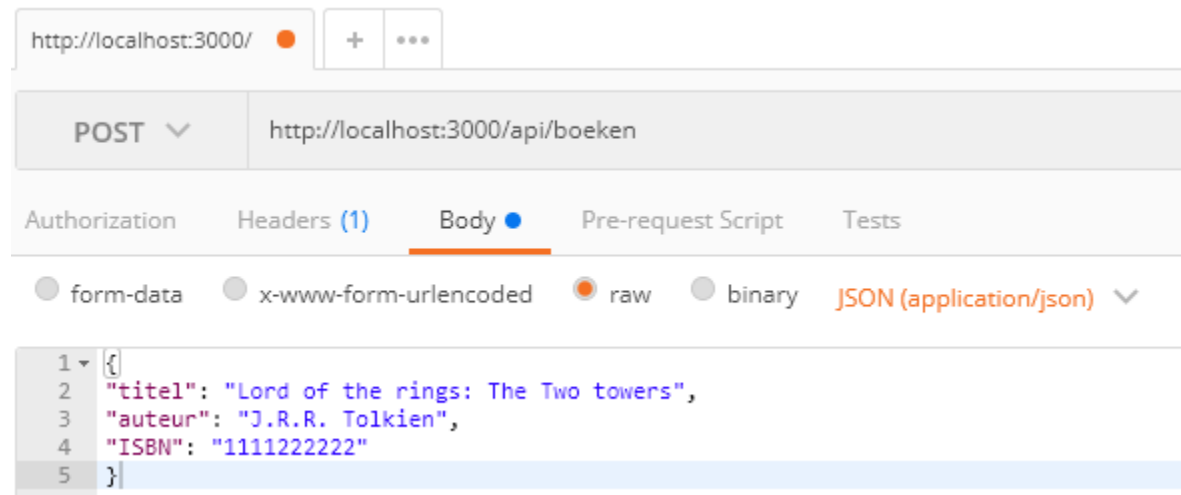
# Een Mongoose CRUD app maken

- In deze code gebeurt het volgende:
  - Eerst wordt een nieuw boekobject gemaakt op basis van het model. De velden worden uit req.body gehaald
  - Omdat het boekobject is afgeleid van het Mongoose-object, heeft het een methode `.save()`. Hiermee wordt het boek opgeslagen in de database.
    - De `_id` is automatisch gegenereerd door MongoDB
  - Het resultaat wordt in Node afgehandeld door een callbackfunctie.



# Een Mongoose CRUD app maken

- Code testen met Postman
  - Zorg ervoor dat de MongoDB-deamon is gestart
  - Start de node.js-server (node server.js)



# Een Mongoose CRUD app maken

- In de mongo shell kunnen we vervolgens checken of het boekobject inderdaad in de database is opgeslagen:

```
> use boeken  
switched to db boeken
```

```
> show collections  
boeks  
system.indexes  
> db.boeks.find()  
{ "_id" : ObjectId("55adecb8d1b4de802d73a097"), "titel" : "Handboek Word 2013",  
  "auteur" : "Peter Kassenaar", "ISBN" : "877542342", "date" : ISODate("2015-07-21  
T06:54:48.997Z"), "__v" : 0 }  
>
```

- Mongoose heeft zelf de collectie “boeks” gemaakt (afgeleide van het Engels)

# Een Mongoose CRUD app maken

- Voorbeeldcode: /1202
- GET-requests verwerken
  - Dezelfde endpoint, ander soort request
  - Voorbeeldcode: /1202

```
router.get('/boeken', (req, res) => {  
  Boek.find((err, boeken) => {  
    if(err)  
      res.send(err);  
    res.json(boeken);  
  });  
});
```

# Een MongoDB CRUD app maken

In deze code wordt rechtstreeks de method `.find()` op de klasse Boek aangeroepen. Omdat geen parameter is meegegeven (alleen een callbackfunctie), worden alle boeken geretourneerd.



**Afbeelding 6.14** Er zijn inmiddels enkele boeken in de database geplaatst. Met een GET-request worden ze opgehaald.

# Een Mongoose CRUD app maken

---



## Specifiek boek ophalen

Wilt u een specifiek boek ophalen (met een aangegeven id, of van een bepaalde auteur)? Zorg dan voor een routeparameter en geef deze als object mee aan de callbackfunctie. De notatie wordt dan iets als `Boek.find({ id: req.params.id }, function()...)`. Ook in de volgende paragraaf wordt een routeparameter gebruikt. Hij wordt ingezet om een bepaald document uit de database te kunnen verwijderen.

---

# Een Mongoose CRUD app maken

- Delete-requests verwerken
  - app.delete()
  - Gebruik van parameters (zie hoofdstuk 12)
    - Req.params.<paramnaam>

```
app.delete('/api/boeken/:id', function(req, res, next){
  Boek.remove({_id: req.params.id}, function(err, removed){
    if(err) {
      return next(err);
    }
    console.log(req.params.id);
    res.json(removed);
  });
});
```

# Een Mongoose CRUD app maken

The screenshot shows a REST client interface with the following details:

- URL Bar:** `http://localhost:3000/` with a red status indicator, a plus sign, and a menu icon.
- Environment:** `No Environment`
- Method:** `DELETE` (with a dropdown arrow)
- URL:** `http://localhost:3000/api/boeken/5a5a7a4611812ca704316aa9`
- Params:** A button labeled `Params`
- Send:** A blue button labeled `Send` with a dropdown arrow
- Tabs:** `Authorization` (selected), `Headers (1)`, `Body` (with a blue dot), `Pre-request Script`, and `Tests`
- Type:** A dropdown menu showing `No Auth`
- Body Tab:** `Body` (selected), `Cookies`, `Headers (6)`, and `Test Results`
- Status:** `Status: 200 OK`
- Format:** Buttons for `Pretty`, `Raw`, and `Preview`; a dropdown for `JSON`; and a refresh icon.
- Response Body:**

```
1 {  
2   "n": 1,  
3   "ok": 1  
4 }
```

# Een Mongoose CRUD app maken

- Put-request verwerken
  - app.put()
  - Gebruik `_id` om boek op te halen & te updaten

```
app.put('/boeken', (req, res) => {
  Boek.findById(req.body._id, (err, boek) => {
    boek.titel = req.body.titel;
    boek.auteur = req.body.auteur;
    boek.isbn = req.body.isbn;
    boek.date = req.body.date;

    boek.save((saveErr, saveBoek) => {
      if(saveErr)
        res.send(saveErr);

      res.send(saveBoek);
    });
  });
});
```



# Een Mongoose CRUD app maken

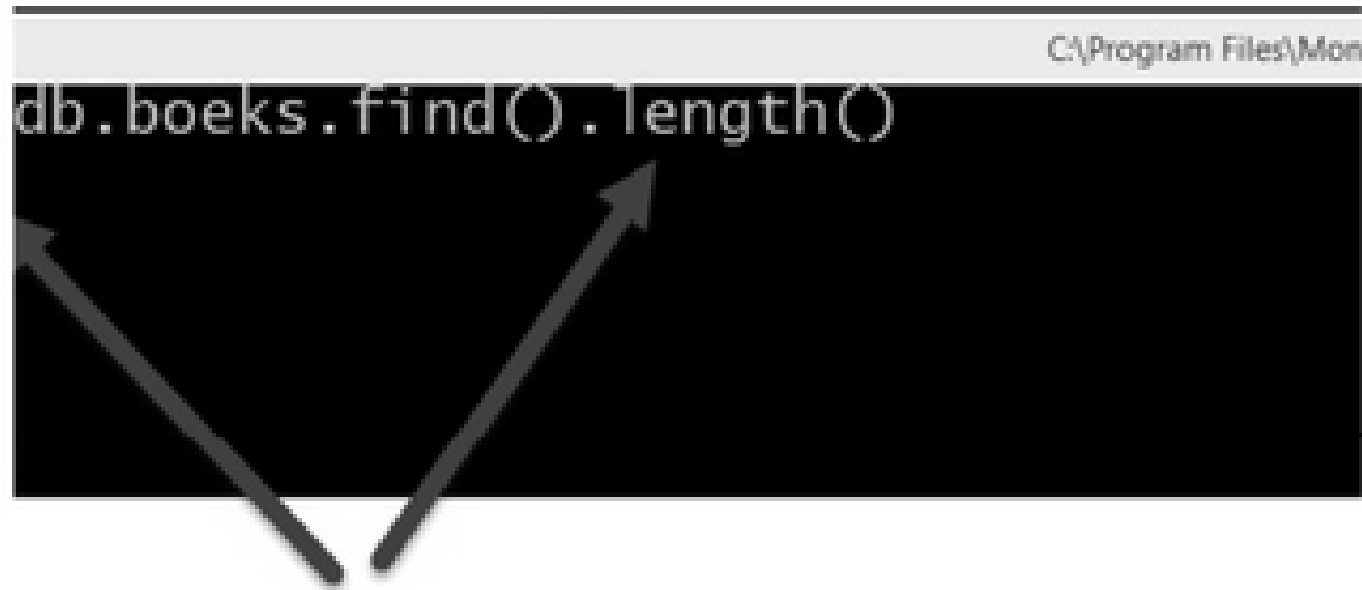


## Aantal items opvragen

Met de MongoDB-functie `.length()` is op te vragen hoeveel documenten in een bepaalde collectie aanwezig zijn. Dit is vergelijkbaar met de eigenschap `.length` voor JavaScript-arrays. Alleen is het in MongoDB een functie die wordt uitgevoerd op het resultaat van `.find()`, dus moet u een hakenpaar gebruiken. De complete notatie is bijvoorbeeld `database.boeks.find().length()`.

---

# Een Mongoose CRUD app maken



```
C:\Program Files\Mon
db.boeks.find().length()
```

Afbeelding 6.16 *Na het verwijderen zijn nog drie boeken over in de database.*

# Een Mongoose CRUD app maken

- Onze API beschikbaar maken voor een Angular Front-end
  - Cross-Origin Resource Sharing (CORS)
  - Toevoegen van localhost:4200 voor alle soorten requests via middleware functie in server.js

```
app.use(function(req, res, next){  
  res.setHeader('Access-Control-Allow-Origin', 'http://localhost:4200');  
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');  
  res.setHeader("Access-Control-Allow-Headers", "Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");  
  next();  
});
```

# Een Angular frontend applicatie voorzien

- Een Angular frontend applicatie bouwen  
ng new mongooseFrontend
- Voorbeeldcode: 1202\_frontend
- Bevat volgende elementen:
  - BoekService: aanspreken van de API
  - Boek model: model voor het mappen van boeken
  - AppComponent: Weergeven van onderstaande components
  - BoekListComponent: Weergeven van boeken in een tabel
  - AddBoekComponent: Formulier voor het toevoegen van een boek
  - EditBoekComponent: Formulier voor het bewerken van een boek

# Een Angular frontend applicatie voorzien

Titel:

Auteur:

ISBN:

Datum uitgave:

Titel	Auteur	ISBN nummer	Datum uitgave	Acties
The hobbit	J.R.R. Tolkien	123-45678-90	2019-01-18T00:00:00.000Z	<input type="button" value="edit"/> <input type="button" value="delete"/>
Divergent	Veronica Roth	123-45678-90	2019-01-17T13:30:31.298Z	<input type="button" value="edit"/> <input type="button" value="delete"/>
Testboek123	Testauteur	111-1111111-11	2019-01-17T13:31:04.436Z	<input type="button" value="edit"/> <input type="button" value="delete"/>