MongoDB

Created	@February 12, 2023 12:45 PM
Type	Backend
Materials	https://www.youtube.com/playlist?list=PL4cUxeGkcC9h77dJ-QJlwGlZlTd4ecZOA
Reviewed	☐

▼ What is it?

- Is a database to store different data for a website
- Is a NoSQL database, and it uses
  - Collections - users, blogs...
  - Document - each entry of a collection

    It gives an _id object to each document
- It uses a JSON-like structure to store data called BSON
- Can be used
  - Locally - installed in your own server
  - Cloud-based - MongoDB atlas
- usually, run in port 27017

▼ Getting Started

To use it locally

- Download and install the MongoDB community edition to your device from mogodb.com
- Install MongoDB compass to have the “GUI”
- If you want to work on a raw MongoDB you can download “Mongosh” shell to interact, but we don’t usually use that we make that from our application

- connect to the MongoDB with the connection key
- then we can create, delete and use databases

▼ Interacting using MongoSH

First, install Mongosh `npm i -g mongosh`

▼ Basic commands

- `mongosh` switch to MongoSH from any terminal
- `show dbs` to see all the databases
- `db` to see the current database
- `use <database-name>` to switch to another database
- `show collections` to see all the collections of the current database
- `help` to see all the commands
- `exit` to exit Mongosh

▼ Manipulating Database

- **Insert a single doc** `db.<c-name>.insertOne(<an object>)`

  If there is no collection by that name it will automatically create it.
- **Insert multiple docs** `db.<c-name>.insertMany(<an array of objects>)`
- **Get all docs** `db.<c-name>.find()`
  - It will print only the first 20 docs
  - `it` to print the next 20 docs
- **Get docs by filtering** `db.<c-name>.find(<filters as an object>)` to filter the what we want.

  Eg - `db.users.find({age: 21, gender: "M"})`
  - to specify which properties to be displayed `db.<c-name>.find(<filters as an object>, {property1: 1, property2: 1})`
- **Get a single doc** `db.<c-name>.findOne(<filters as an object>)`

- **Delete a single doc** `db.<c-name>.deleteOne({filter})` we usually use _id
- **Delete multiple doc** `db.<c-name>.deleteMany({filter})`
- **Update a single doc** `db.<c-name>.updateOne({filter}, {$set: {values to be updated}})` to increment use `$inc` , to remove a property `$pull` , to add a property `$push` instead of `$set`

▼ Functions of find()

we can combine multiple functions to find desired result

- `count()` to count the docs returned by find

  Eg `db.blogs.find().count()`
- `limit(<n>)` to limit the result to the n numbers
- `sort({property1: 1})` to sort based on that property
  - 1 - ascending
  - -1 - descending

▼ Operators

Always starts with $

- `$gt` greater then
- `$lt`
- `$gte`
- `$lte`

  Eg `db.books.find({rating: {$lte: 7}})`
- `$or` one of the filters

  Eg `db.books.find({$or: [{rating: 8}, {name: 'Abenezer"})`
- `$in` when docs have a property with a value listed in the array

  Eg `db.books.find({author: {$in: ["Kebede", "girma", "Chala"]}})`
- `$nin` the opposite of $in

▼ Interacting from node.js

▼ Install mongo db

```
npm install mongodb --save
```

▼ Connecting with mongo db

`create two functions (usually in a separate file and import it from the main api/ server) to connect and get database

In db.js

```
const { MongoClient } = require('mongodb')
let dbConnection
connectToDb: (cd) => { //cb is a call back to be executed after the connection
    MongoClient.connect('<connection key>')
        .then(client => {
            dbConnection = client.db()
            return cb()
        })
        .catch(err => <something>)
    }


getDb: () => dbConnection
```

In app.js

```
const { connectToDb, getDb } = require('./db')
let db
connectToDb() {
    if (!err) {
        app.listen(3000)
        }
    db = getDb()
    }
```

▼ Get the docs

```
let array = []
```

`db.collectioin('<c-name>')`

`.find()`

`.forEach(doc = array.push(doc))`

`.then(<do something after all of it is done>))`

▼ Pagination

to fetch docs in a bunch if they are many

▼ Mongoose