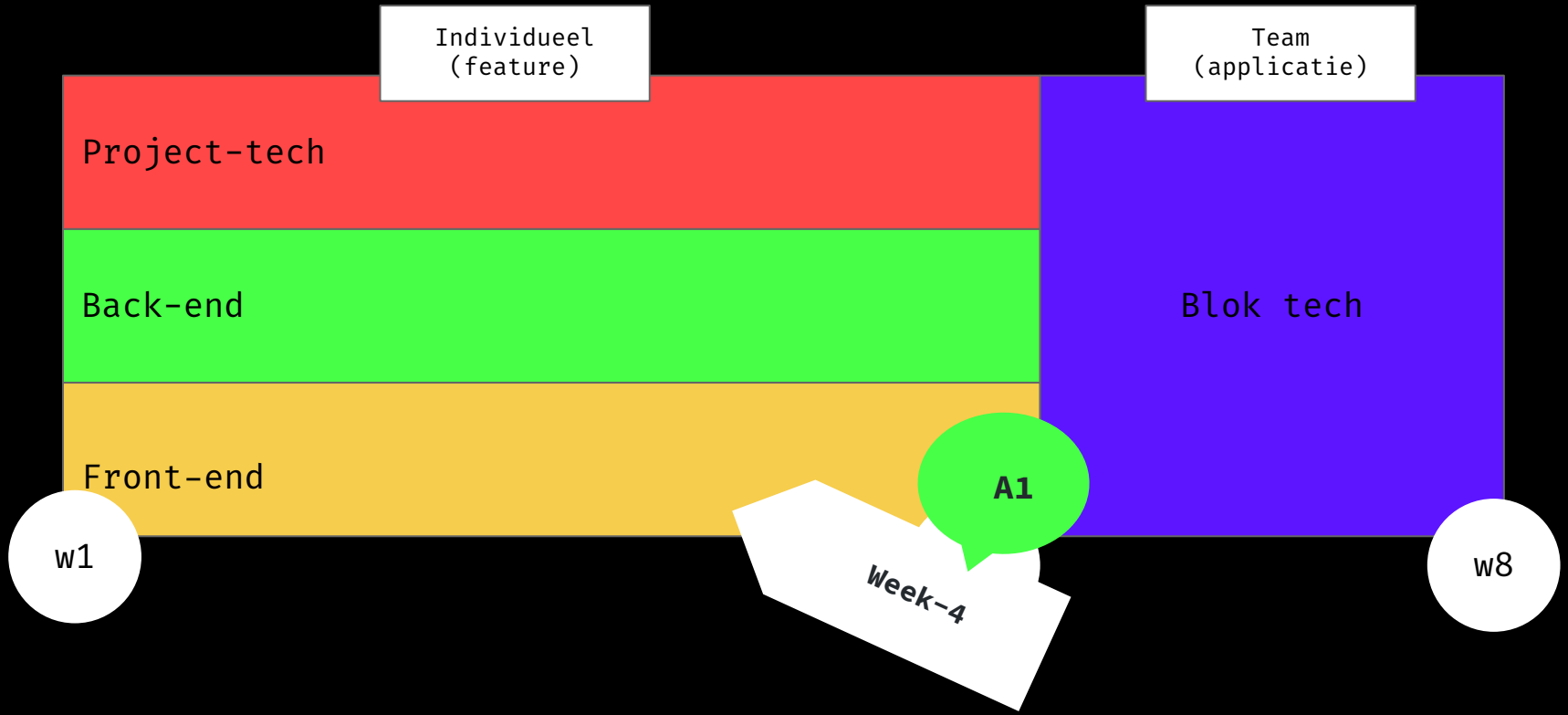
back-end

Database

lab 4/8

Stand-up!

Show what you did


Individueel
(feature)
Team
(applicatie)
Project-tech
Back-end
Front-end
Blok tech
A1
Week-4
w1
w8

A1

Praktische info

- Lees goed de info over ‘grading’ op github
  - Daar staat ook de rubric voor de beoordeling
  - En hoe je je voorbereid op het mondeling
- Lever uiterlijk <datum en tijd> een link naar je github repo + een .zip van je code in op DLO
- Schrijf je in op een tijdslot en zorg zelf dat er geen overlap is met het mondeling van project tech en frontend 2

today

I. ~~Stand-up~~
II. ~~Info about A1~~
III. MongoDB
IV. Peer review
V. Crud

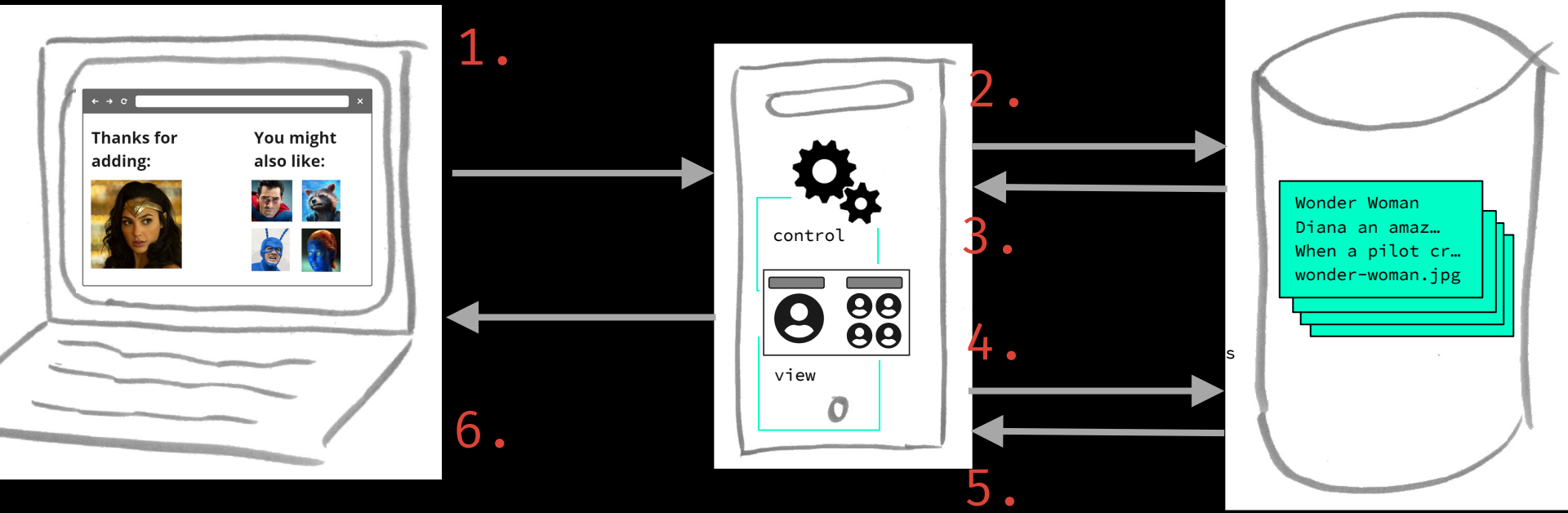
MongoDB

Storing data in db

Client
(browser)

Server
(Webserver met Node.js)

Database
(MongoDB)



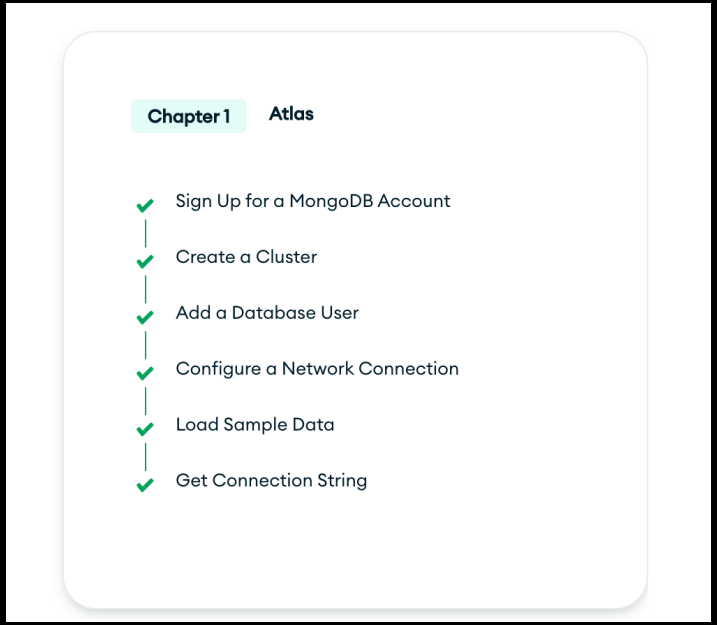
connect

mongodb

MongoDB (from hu**mong**ous) is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. MongoDB is developed by MongoDB Inc. […]

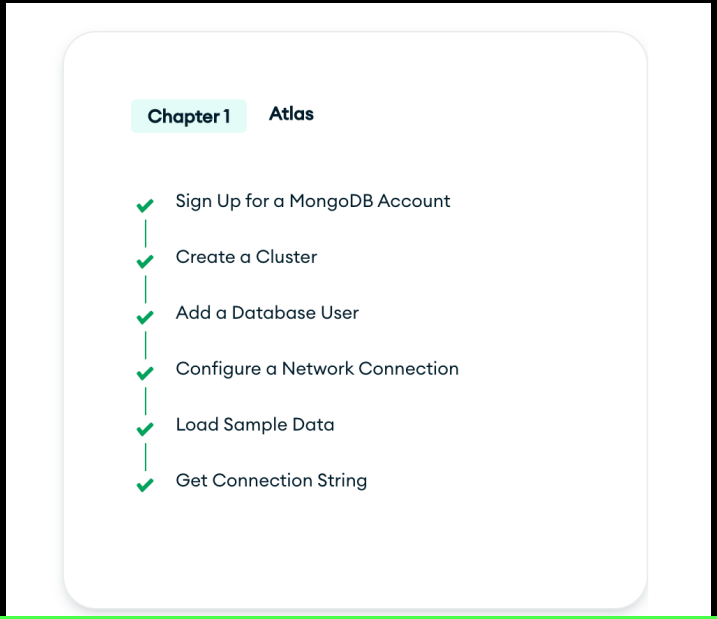
wikipedia.org

connect



Steps in the “MongoDB start with guide”

connect



Note: there are a lot of small steps involved. Read the Mongo guides very carefully. If you miss a step everything will be broken.

index.js

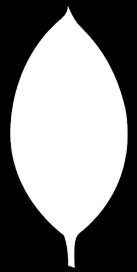
```
…

require('dotenv').config()

const { MongoClient, ServerApiVersion, ObjectId } = require('mongodb')
const uri = 'mongodb+srv://' + process.env.DB_USERNAME + ':' +
  process.env.DB_PASS + '@' +
  process.env.DB_HOST + '/' + process.env.DB_NAME +
  '?retryWrites=true&w=majority'

const client = new MongoClient(uri, { useNewUrlParser: true,
  useUnifiedTopology: true, serverApi: ServerApiVersion.v1 })

client.connect(err => {
  if (err) { throw err }
})

…
```

live demo .env en connection

bash

```
$ npm install mongodb d

+ mongodb@2.2.33
+ dotenv@4.0.0
added 11 packages in 4.022s

$
```

mongodb wraps **MongoDB** for **Node**



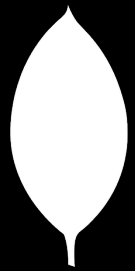
connect

mongodb

```
// Files
mongodb-server/
├─ node_modules/
├─ static/
│  ├─ index.css
│  ├─ index.js
│  └─ upload/
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  ├─ not-found.ejs
│  └─ tail.ejs
├─ .env
├─ index.js
└─ package.json
```

.env

```
DB_HOST=localhost
DB_PORT=27017
DB_NAME=mymoviewebsite
DB_USERNAME=dandevri
```



connect

mongodb

```
// Files
mongodb-server/
├─ node_modules/
├─ static/
│  ├─ index.css
│  ├─ index.js
│  └─ upload/
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  ├─ not-found.ejs
│  └─ tail.ejs
```

.env

```
DB_HOST=localhost
DB_PORT=27017
DB_NAME=mymoviewebsite
```

Note: Never ever put your **host and password in code or on GitHub**! People will be able to access your database!

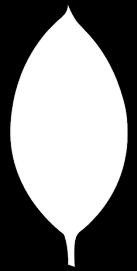
connect

mongodb

```
// Files
mongodb-server/
├─ node_modules/
├─ static/
│  ├─ index.css
│  ├─ index.js
│  └─ upload/
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  ├─ not-found.ejs
│  └─ tail.ejs
├─ .env
├─ .gitignore
├─ index.js
└─ package.json
```

.gitignore

```
node_modules/
.DS_Store
.env
```

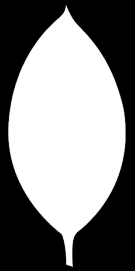


II MON

MONGODB

index.js

```
…

require('dotenv').config()

const { MongoClient, ServerApiVersion } = require('mongodb')
const uri = 'mongodb+srv://' + process.env.DB_USERNAME + ':' +
  process.env.DB_PASS + '@' +
  process.env.DB_HOST + '/' + process.env.DB_NAME +
  '?retryWrites=true&w=majority'

const client = new MongoClient(uri, { useNewUrlParser: true,
  useUnifiedTopology: true, serverApi: ServerApiVersion.v1 })

client.connect(err => {
  if (err) { throw err }
})

…
```



index.js

```
…

function add(req, res) {
  var id = slug(req.body.title).toLowerCase()

  data.push({
    id: id,
    title: req.body.title,
    plot: req.body.plot,
    description: req.body.des
  })

  res.redirect('/' + id)
}

…
```

title, **plot**, and **description** come from **name** attributes on form inputs

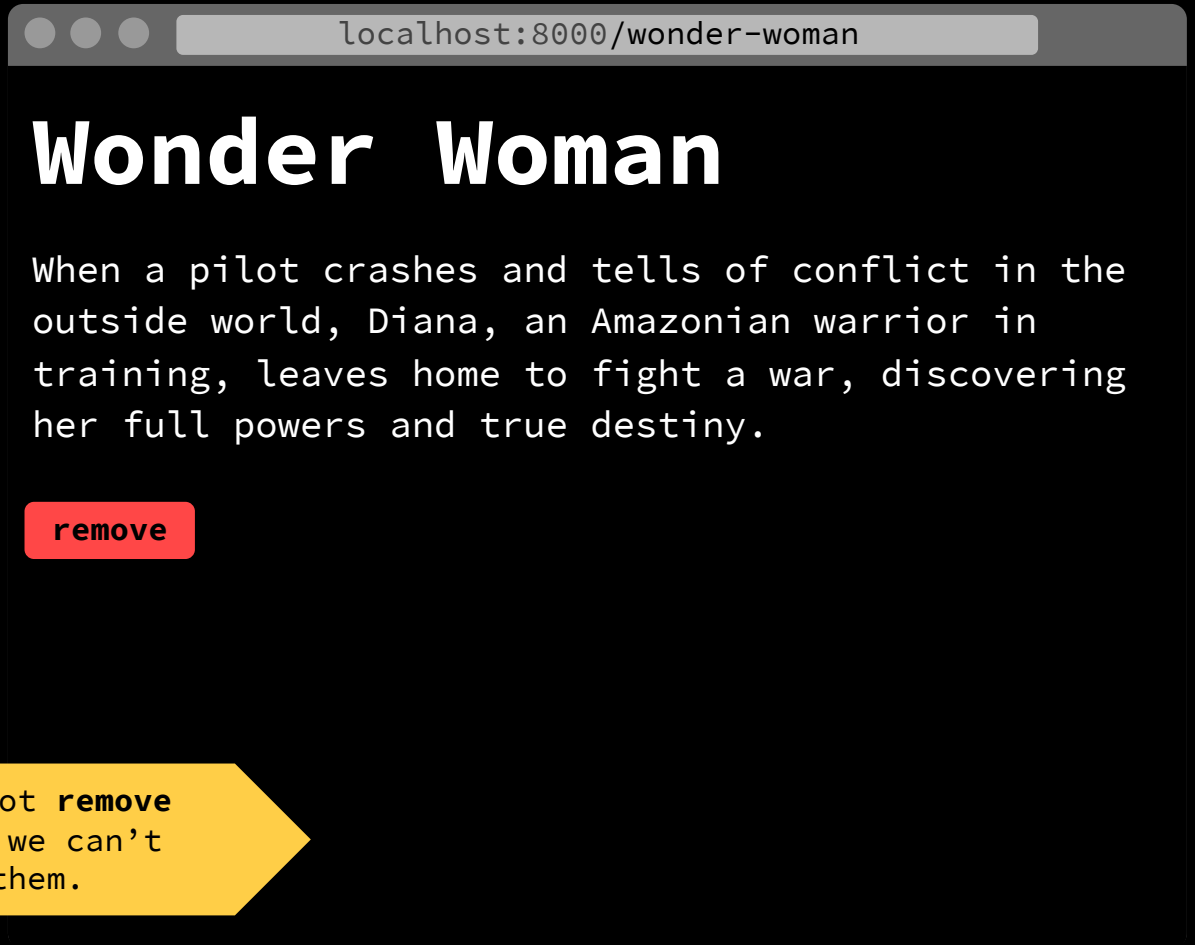
Express

Wonder Woman

When a pilot crashes and tells of conflict in the outside world, Diana, an Amazonian warrior in training, leaves home to fight a war, discovering her full powers and true destiny.

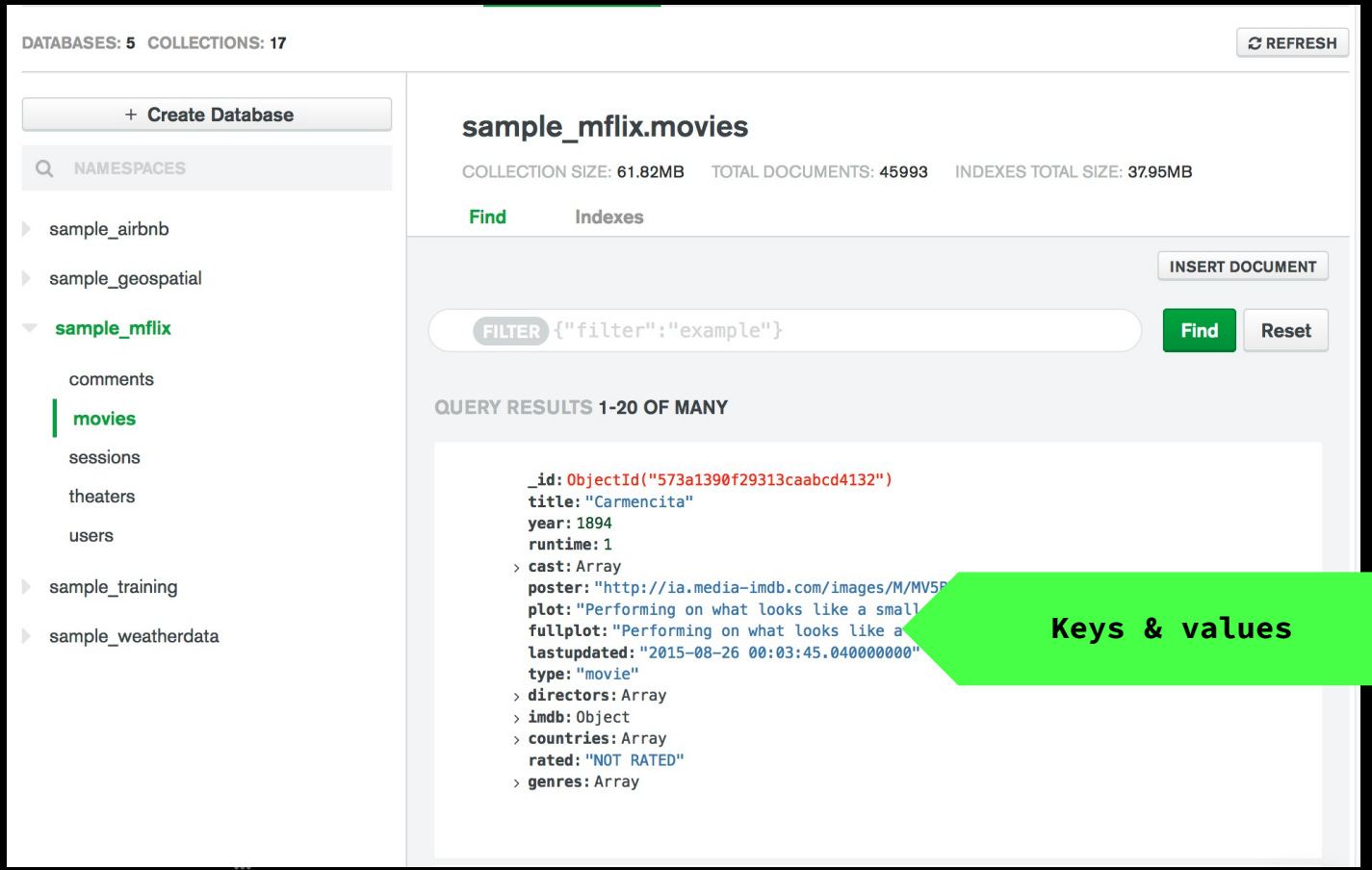
…and we can submit the form!

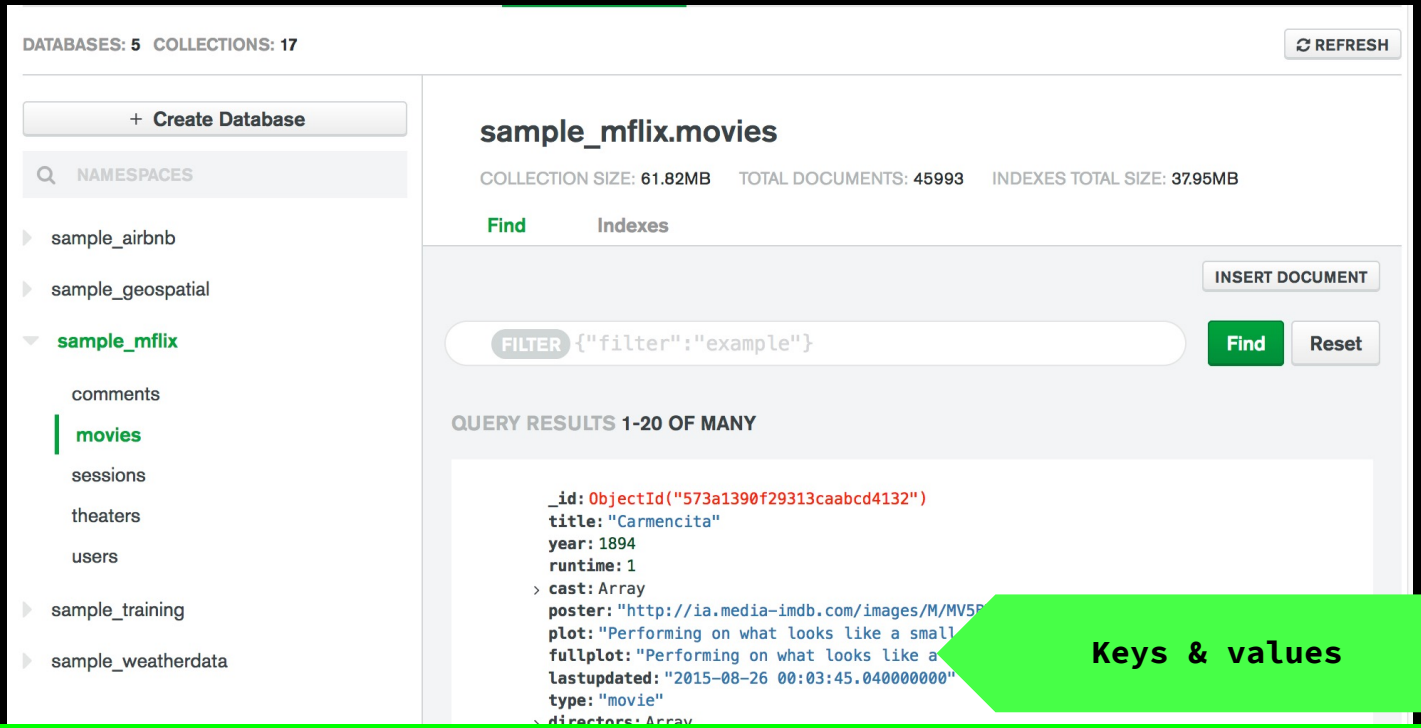
Express



…but we cannot **remove** movies? And we can't **update** them.

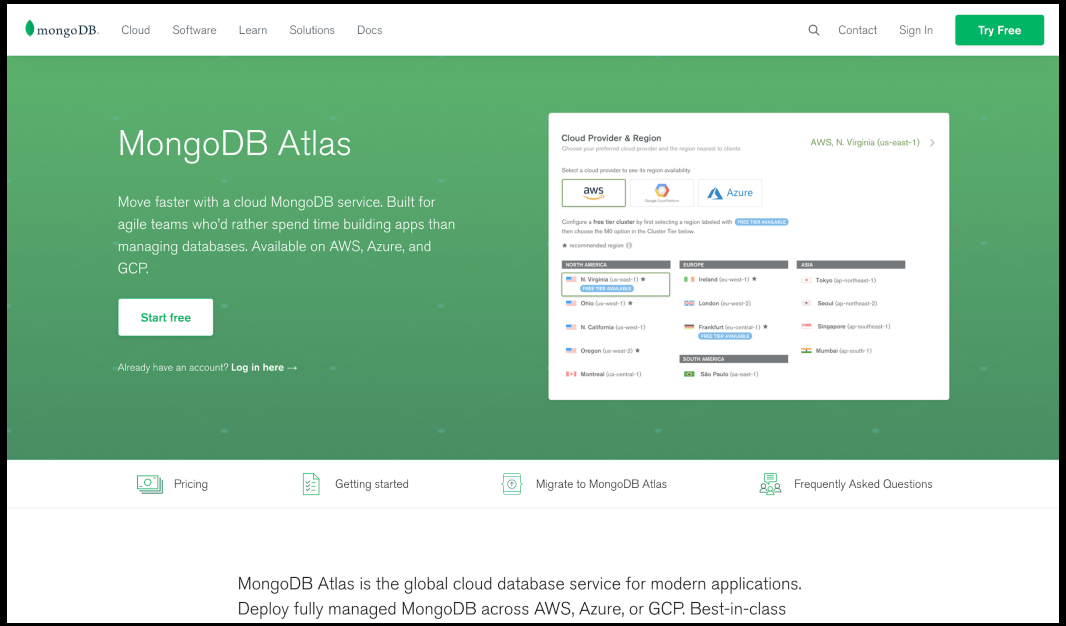
Express


DATABASES: 5 COLLECTIONS: 17
REFRESH
+ Create Database
NAMESPACES
sample_airbnb
sample_geospatial
sample_mflix
comments
movies
sessions
theaters
users
sample_training
sample_weatherdata
sample_mflix.movies
COLLECTION SIZE: 61.82MB
TOTAL DOCUMENTS: 45993
INDEXES TOTAL SIZE: 37.95MB
Find
Indexes
INSERT DOCUMENT
FILTER {"filter":"example"}
Find
Reset
QUERY RESULTS 1-20 OF MANY
_id: ObjectId("573a1390f29313caabcd4132")
title: "Carmencita"
year: 1894
runtime: 1
cast: Array
poster: "http://ia.media-imdb.com/images/M/MV5
plot: "Performing on what looks like a small
fullplot: "Performing on what looks like a
lastupdated: "2015-08-26 00:03:45.040000000"
type: "movie"
directors: Array
imdb: Object
countries: Array
rated: "NOT RATED"
genres: Array
Keys & values



Think about how you want to structure you data.
This is called data modelling.

mongodb atlas



https://www.mongodb.com/cloud/atlas

Break!

Back-end Peer Review

peer review for assessment 1

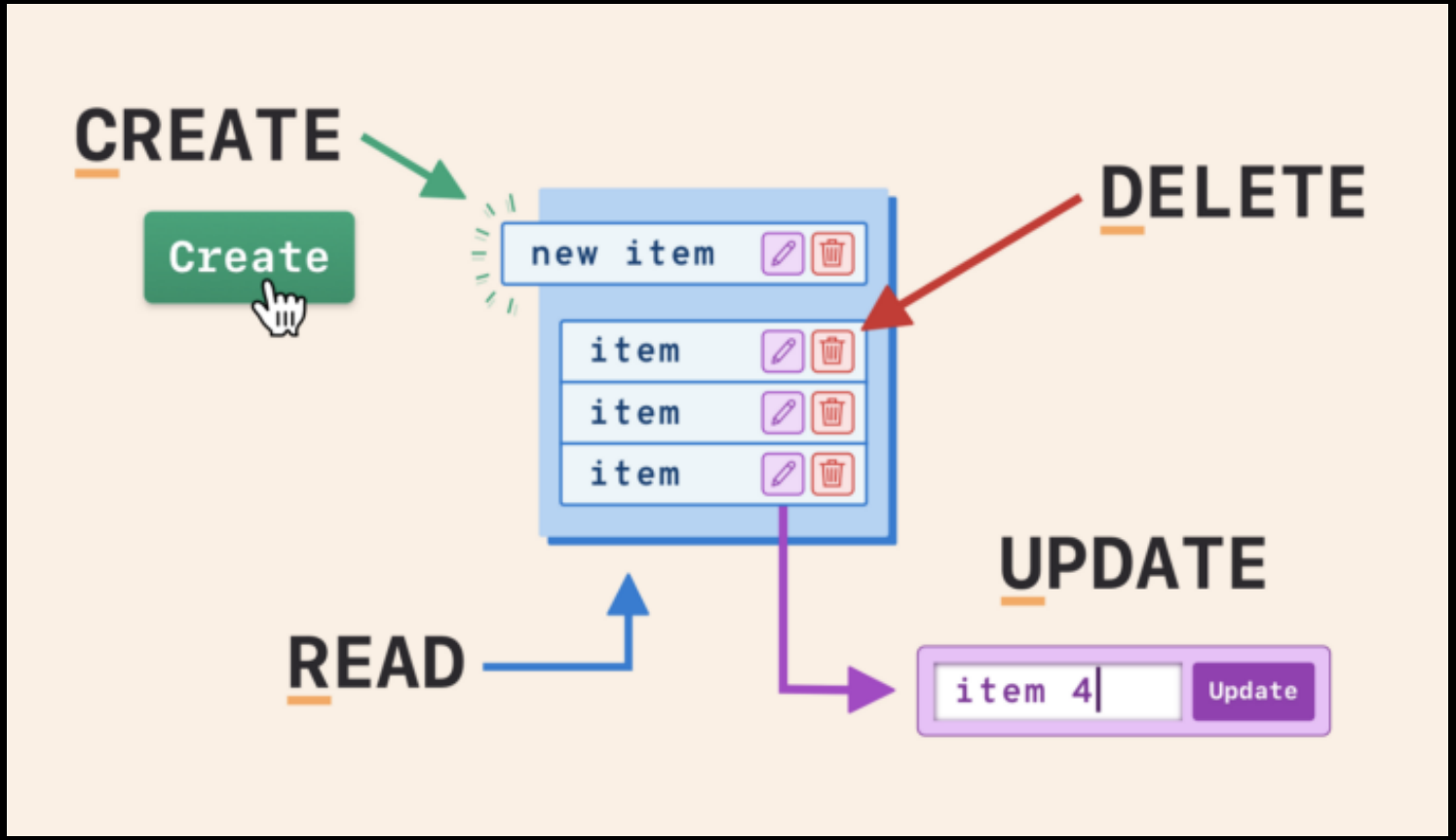
Reviewer	
Reviewee	
Class	
Teacher	

Checklist

Project is documented and has a readme.md	

a1-peer-review_student.docx

Crud

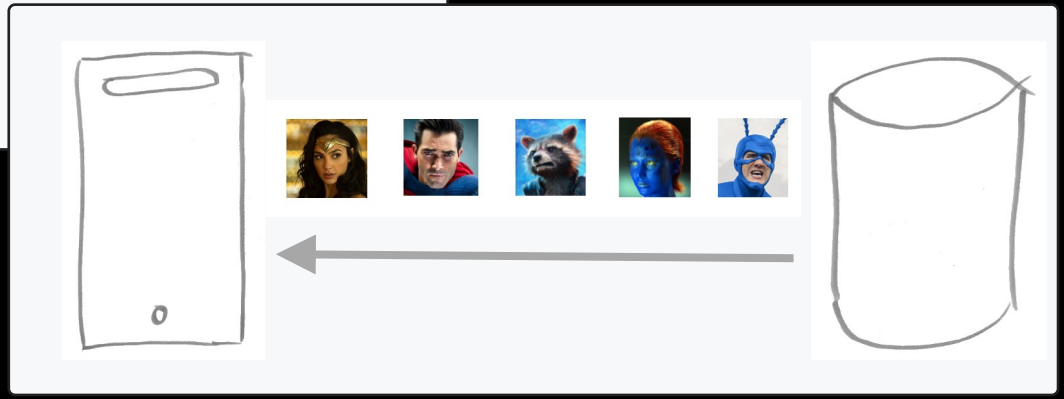


Source: CRUD Operations Explained by Avelon Pang, June 14, 2021

find

index.js

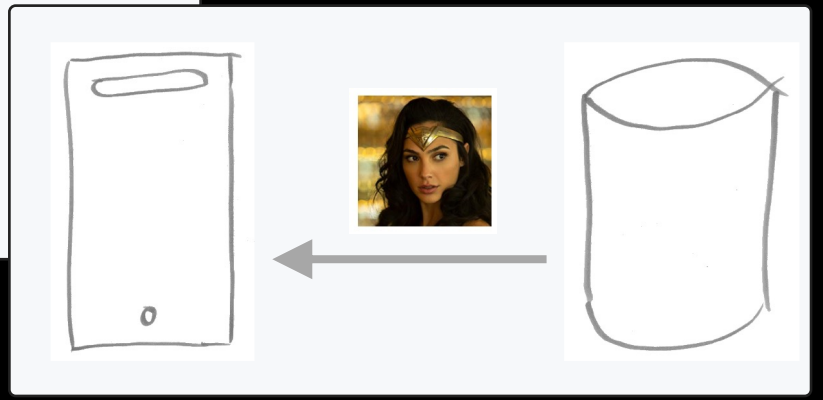
```
…
const db = client.db(process.env.DB_NAME)

function movies(req, res, next) {
  db.collection('movie').find().toArray(done)

  function done(err, data) {
    if (err) {
      next(err)
    } else {
      res.render('list.ejs', {data: data})
    }
  }
}

…
```



findOne

index.js

```
…

const db = client.db(process.env.DB_NAME)

function movie(req, res, next) {
  db.collection('movie').findOne({
    title: req.params.title
  }, done)

  function done(err, data) {
    if (err) {
      next(err)
    } else {
      res.render('detail.ejs', {data: data})
    }
  }
}

…
```



update

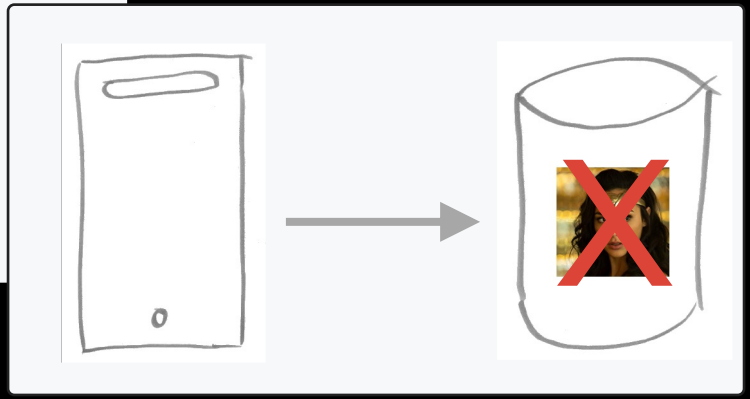
index.js

```
const { MongoClient, ObjectId } = require('mongodb')
const db = client.db(process.env.DB_NAME)

function update(req, res, next) {
  db.collection('movie').updateOne({
          _id: ObjectId(req.body.id),
          {$set: {plot: req.body.plot}}
  }, done)

  function done(err, data) {
    if (err) {
      next(err)
    } else {
      res.render('movie_updated.ejs', {title: req.body.title})
    }
  }
}
```

delete

index.js

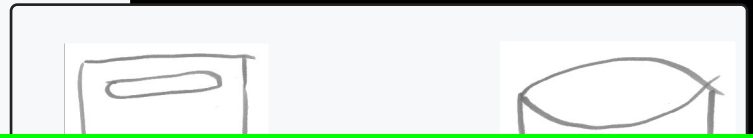
```
const { MongoClient, ObjectId } = require('mongodb')
const db = client.db(process.env.DB_NAME)

…

function remove(req, res, next) {
  db.collection('movie').deleteOne({
    _id: ObjectId(req.body.id)
  }, done)

  function done(err) {
    if (err) {
      next(err)
    } else {
      res.json({status: 'ok'})
    }
  }
}

…
```



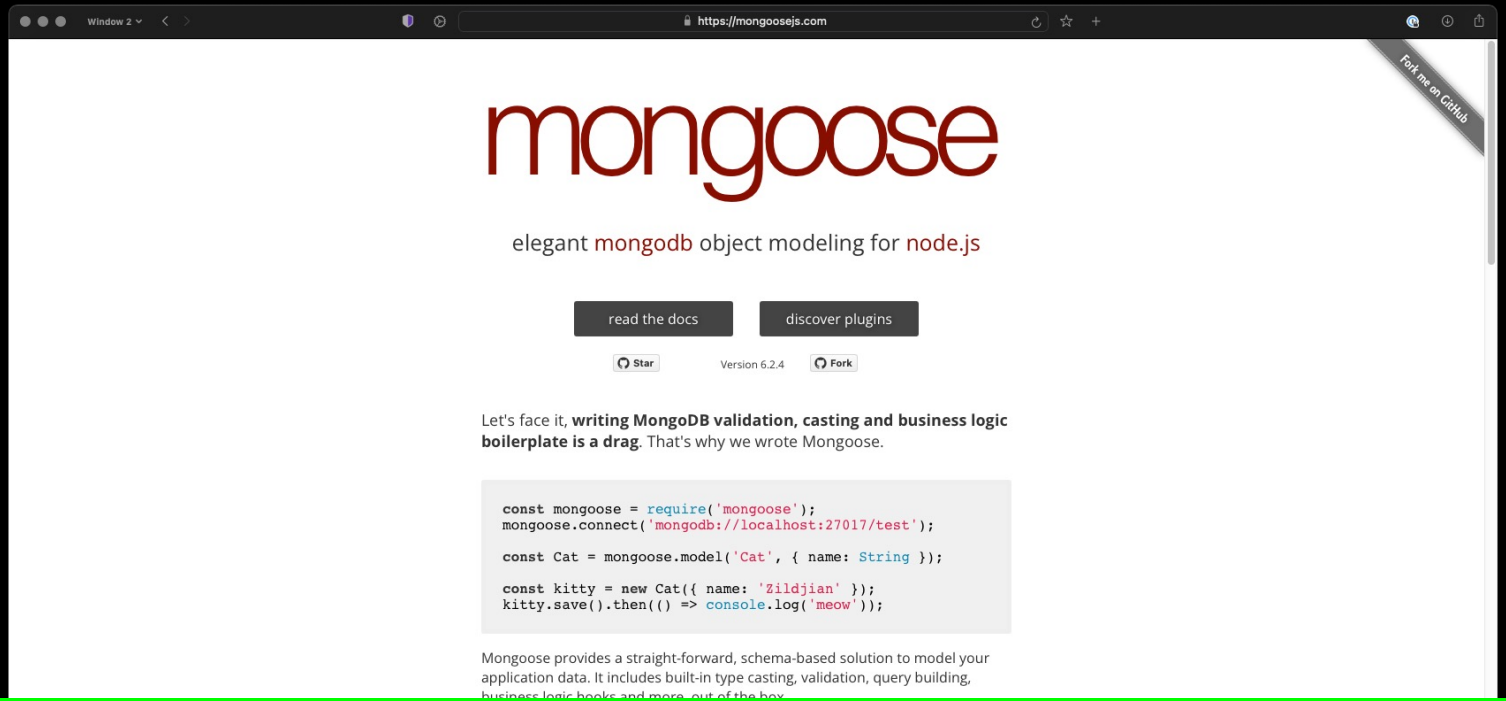
delete

index.js

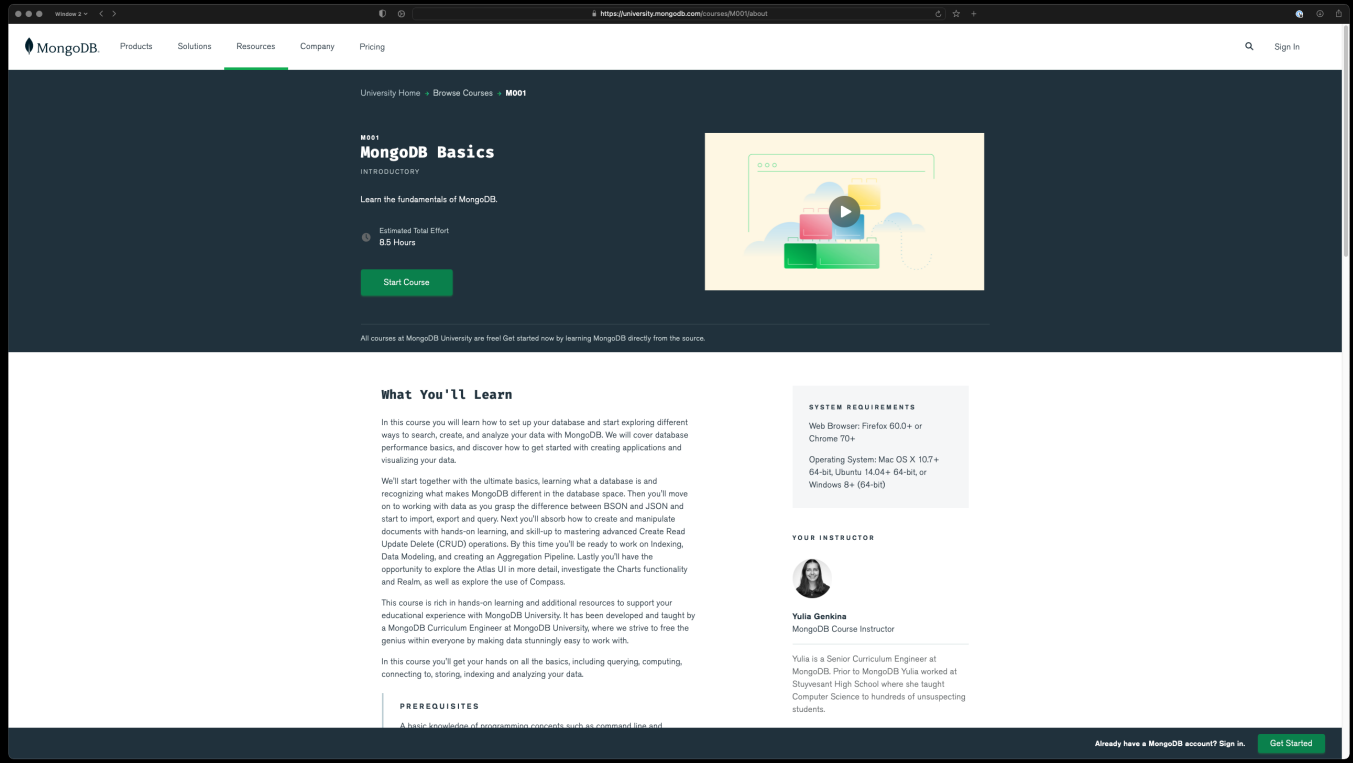
```
…
function remove(req, res, next) {
  var id = req.params.id
  db.collection('movie').deleteOne({
    _id: mongo.ObjectID(id)
  }, done)

  function done(err) {
    if (err) {
      next(err)
    } else {
      res.json({status: 'ok'})
    }
  }
}

…
```



Note: don’t just use find - to pass you’ll also need
Update or insert or delete



Pick mongoDB (default driver) over Mongoose.



MongoDB University

```
function add(req, res, next) {
  db.collection('movie').updateOne({
      _id: ObjectID(req.body._id),
      {$set: {textProfile: req.body.description}}
  }, done)

  function done(err, data) {
    if (err) {
      next(err)
    } else {
      res.redirect('/' + data.insertedId)
    }
  }
```

live demo crud operations

exit;

see you in lab-5!