back-end

Database

lab 4/8

Stand-up!

Review

In the lab y having a look at the assessment checklist and grading fellow students based on the rubric. A hat teachers ns and do some technical troubleshooting.

A1 peer rev
This is the peer r w document you will fill-in as duo. You'll perform the checks on the project of your fellow student. It's a good last check t see if everything is in order.

→ Peer Review

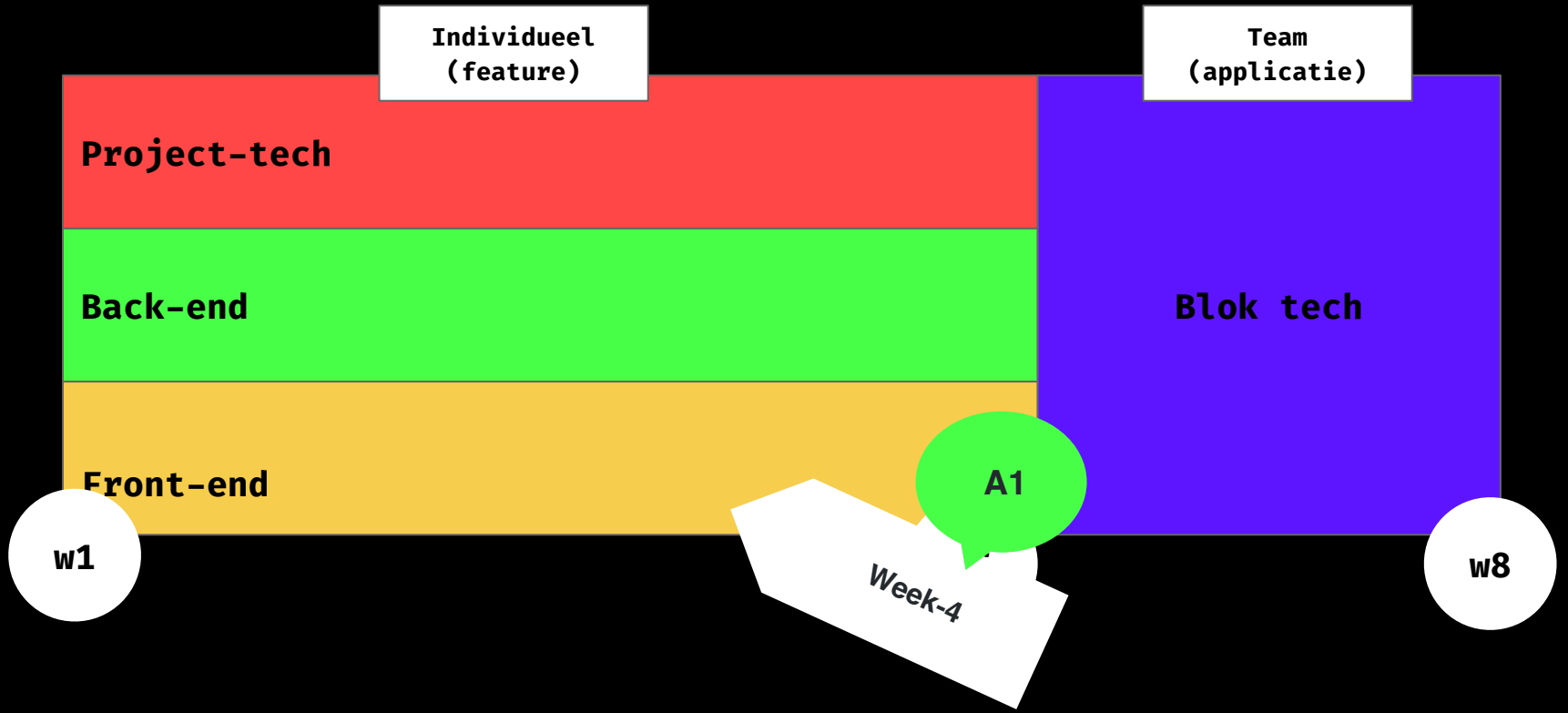
A1 rubric:
This is the rubric your teacher will grade you on during the assessment. Ask yourself upon completion if everything listed on the rubric is clear and that you understand each row and column, if not ask your teacher on MS Teams!

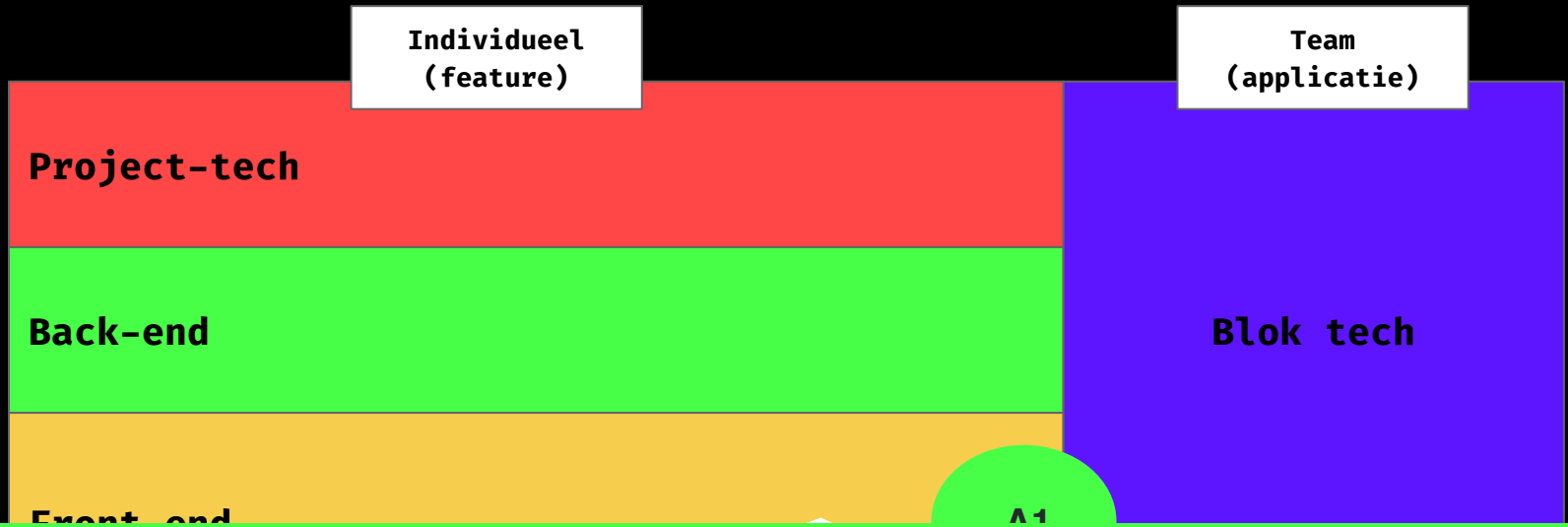
→ Rubric

Questions:

If you have any question you can reach the student-assistants and teachers on MS Teams. Make sure you are prepared:

peer review


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



Note: **last class before assessment.**

Next week is ‘hemelvaart’ and the week after is A1.

today

I. ~~Stand-up~~
II. A1 Grading
III. SQL vs NoSQL
IV. Connect
V. Crud

A1

course

goals

- You can build web apps with Node and use packages from NPM
- You can communicate over HTTP and understand client/server flow
- You can render data server-side with a templating engine
- You can store data in a database and update that data
- You can write documentation that other developers understand
- You can explain your code and the cohesion of your application

course

deliverables

- **Individual Prototype:** working **interactive feature** for serious relationships
- **Process book (wiki):** that provides insight into the weekly iterative process and your research

Assessment 1 - Individual

You've worked iteratively (*formative*) on your product and finish with an oral test (*summative*). You'll **show the feature** you've [illegible] for back-end based on your code in your repository and live version. *A teacher will try out your feature and look at the code.*

You will show you can create a quality project in which you apply the subject matter of this course and that you understand it.

You will answer questions in such a way as to demonstrate sufficient knowledge of our goals.

This is an individual assessment, so tests will be conducted between one teacher and one student.

> **This is an assessment, not another moment for feedback.** So you will be graded. There isn't much time for additional feedback or troubleshooting technical issues.

Preparation

Since we have limited time make sure you come to the assessment prepared:

- **Bring your computer** and make sure it's *charged* and connected to *Wifi*.
- Make sure your *webcam*, *microphone* and *screen sharing* **works in MS Teams**
- Have the latest version of your feature **ready in your browser**.
- Have the latest version of your code **ready on GitHub**.

let's look at the **rubric for A1**

SQL & NoSQL

sql

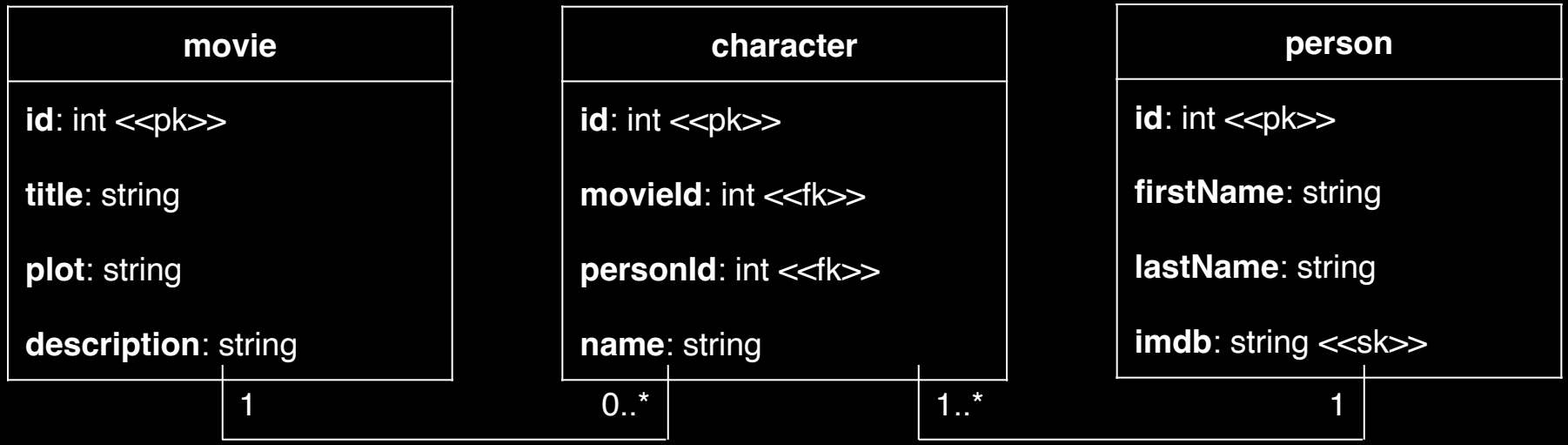
?

SQL (Structured Query Language) is a […] language used in programming and designed for managing data held in a relational database […].

wikipedia.org

sql

structure



nosql

?

A NoSQL (originally referring to “non SQL” or “non relational”) database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. […]

wikipedia.org

nosql

?

[…] Such databases have existed since the late 1960s, but did not obtain the “NoSQL” moniker until a surge of popularity in the early twenty-first century, triggered by the needs of Web 2.0. […]

wikipedia.org

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

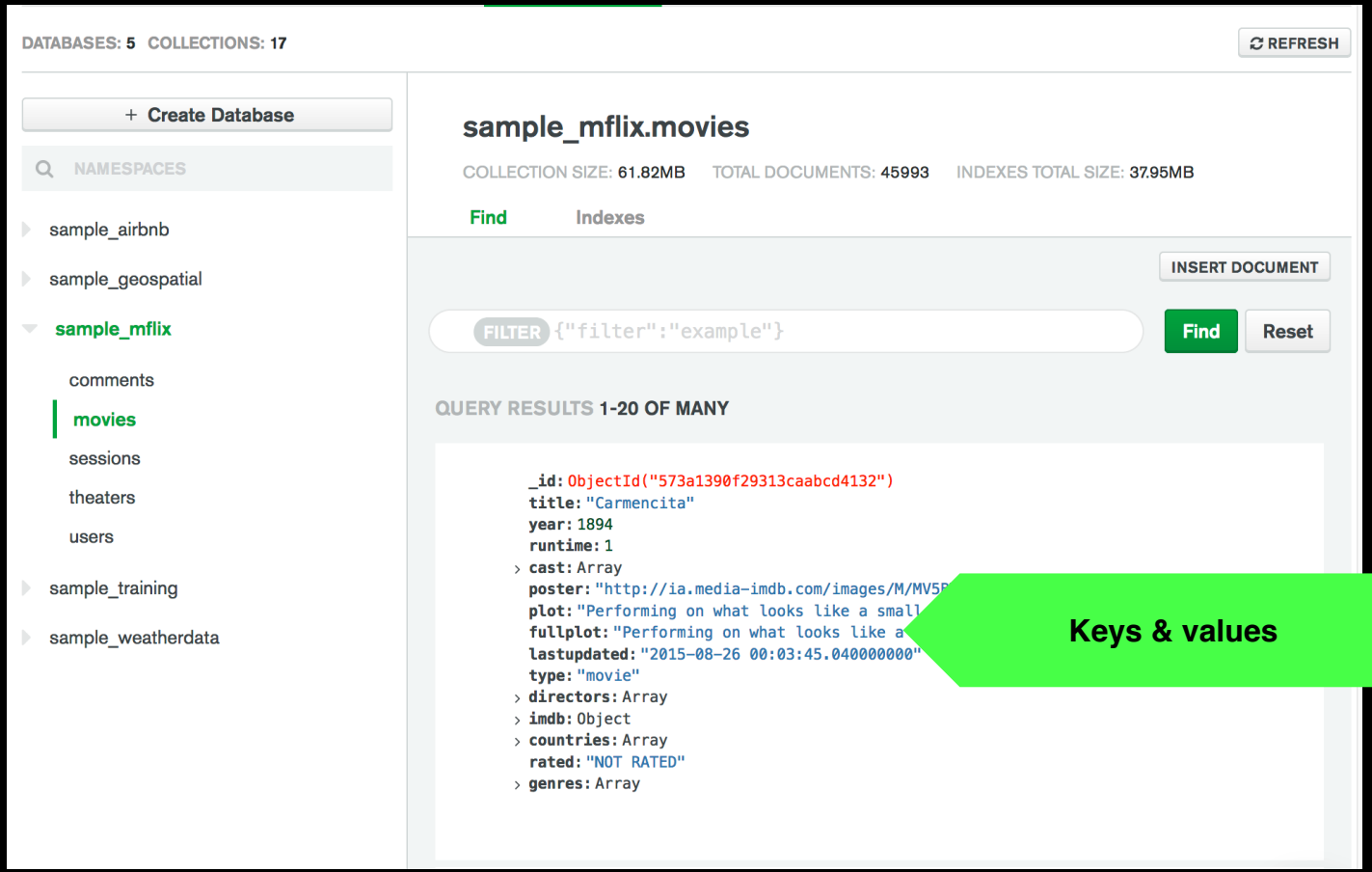
mongodb

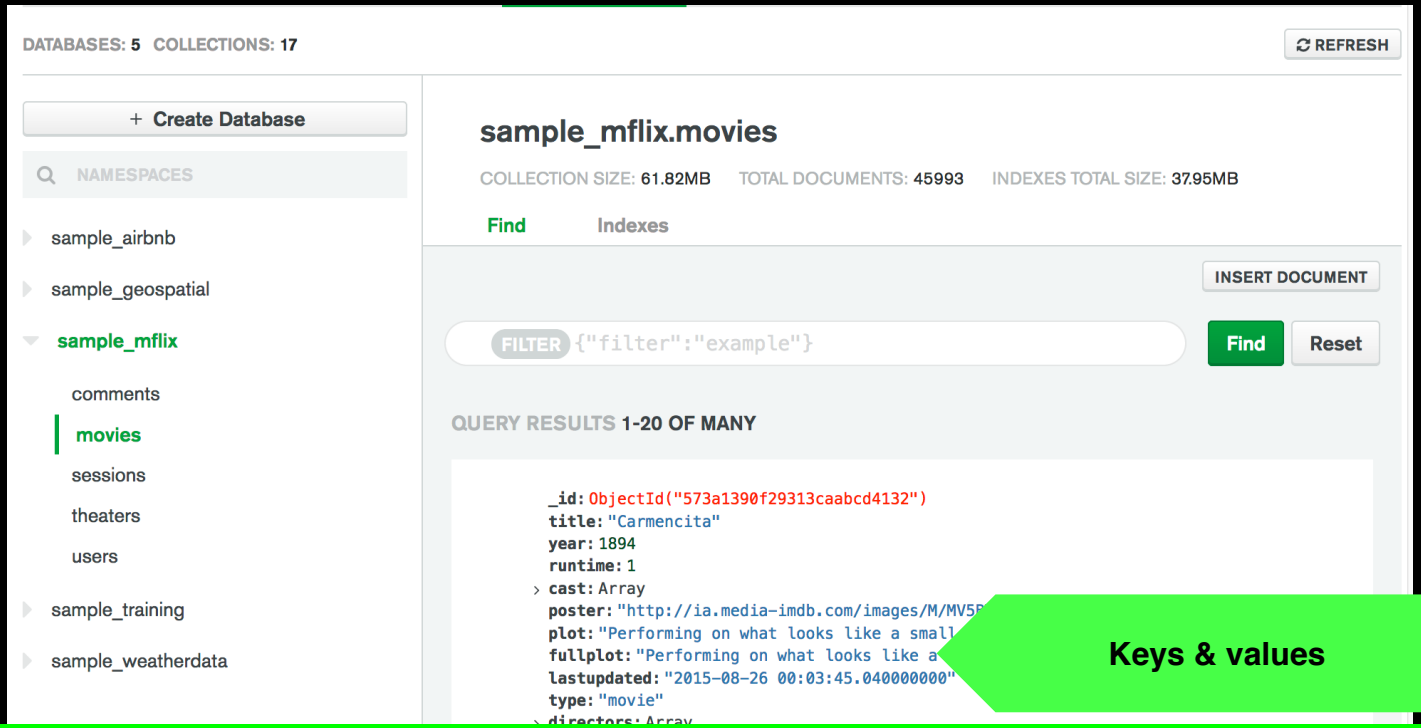
?

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. […]

JSON like **documents** (no relations)

wikipedia.org

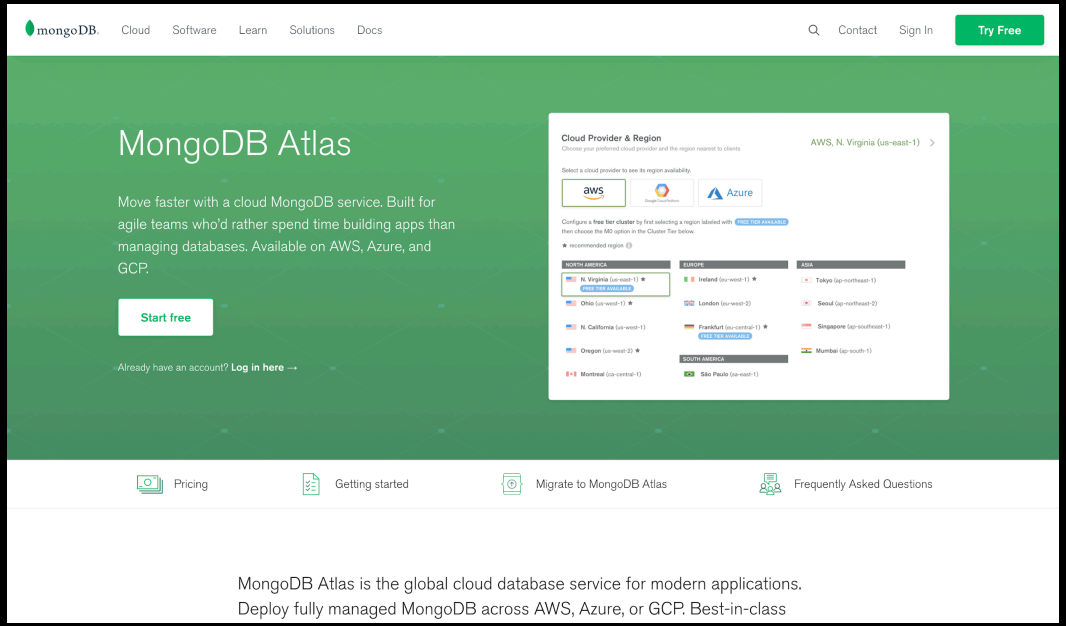

DATABASES: 5 COLLECTIONS: 17
REFRESH
+ Create Database
NAMESPACES
sample_airbnb
sample_geospatial
sample_mflix
comments
movies
sessions
theaters
users
sample_training
sample_weatherdata
sample_mflix.movies
COLLECTION SIZE: 61.82MB
TOTAL DOCUMENTS: 45993
INDEXES TOTAL SIZE: 37.95MB
Find
Indexes
INSERT 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/MV5B
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



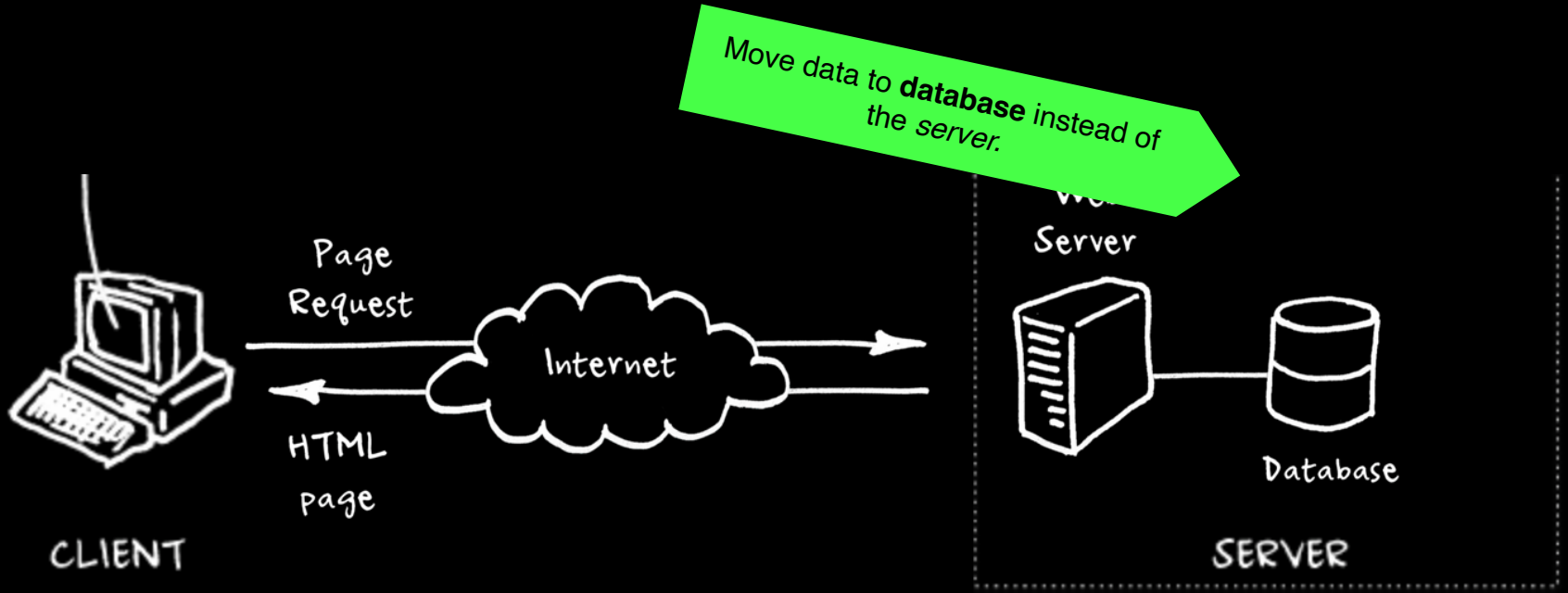
https://www.mongodb.com/cloud/atlas

Assignment (±20m)

Data model your database and draw / write how you are going to structure your database.

- Look at your interface, what inputs are you using?
- What data types do you need?
- What relations between tabels do you need?

Connect


Move data to **database** instead of the *server.*
Page Request
HTML page
Internet
Server
Database
CLIENT
SERVER

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. […]

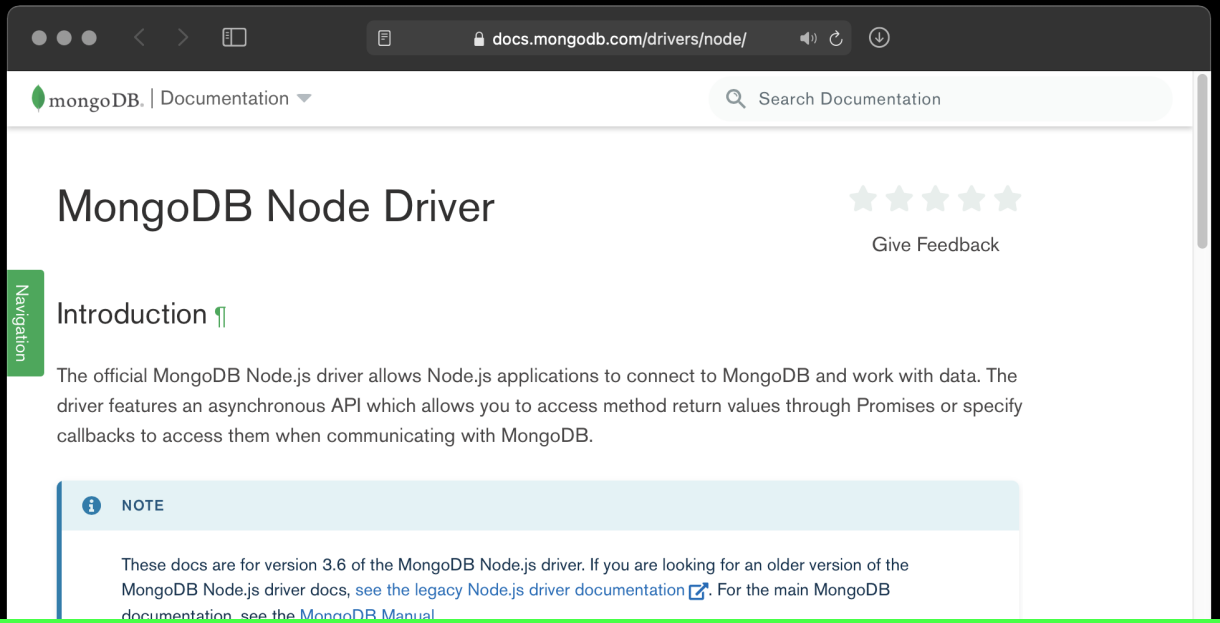
wikipedia.org

connect

mongodb

- JavaScript can be used in queries, aggregation function
- Map-reduce can be used for batch processing of data and aggregation operations
- Manage massive increases in new, rapidly changing data types

https://www.mongodb.com/compare/mongodb-mysql

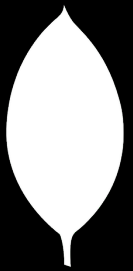


Note: there are a lot of small steps involved. Read the Mongo guides very carefully. If you miss a step everything will be broken.

bash

```
$ npm install mongodb dotenv

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

$
```

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



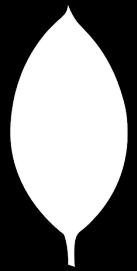
connect

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
```



connect

mongodb

```
// Files
mongodb-server/
 ├── node_modules/
 ├── static/
 |    ├── index.css
 |    ├── index.js
 |    └── upload/
 ├── view/
 |    ├── add.ejs
 |    ├── detail.ejs
 |    ├── head.ejs
 |    ├── list.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!

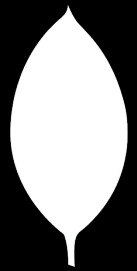
connect

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
```

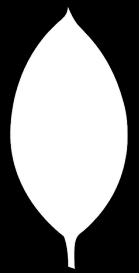
.gitignore

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



index.js

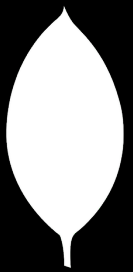
```
...
var multer = require('multer')
var mongo = require('mongodb')

require('dotenv').config()

var db = null
var url = 'mongodb://' + process.env.DB_HOST + ':' + process.env.DB_PORT

mongo.MongoClient.connect(url, function (err, client) {
  if (err) throw err
  db = client.db(process.env.DB_NAME)
})

...
```



Crud

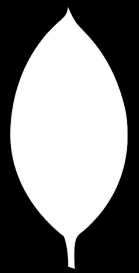
index.js

```
...

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})
    }
  }
}

...
```



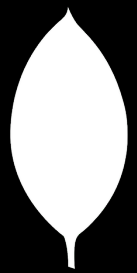
index.js

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

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

...
```



index.js

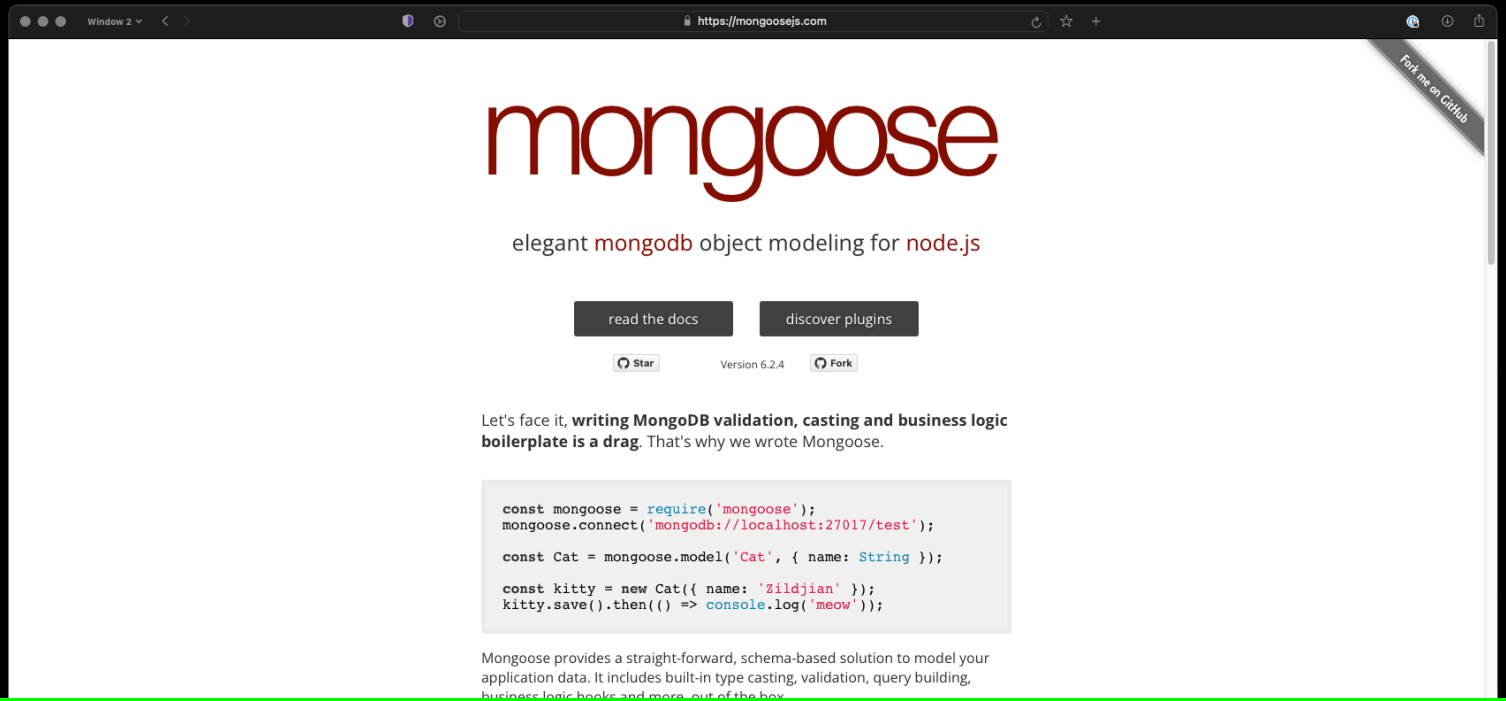
```
...

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)
    }
  }
}
```



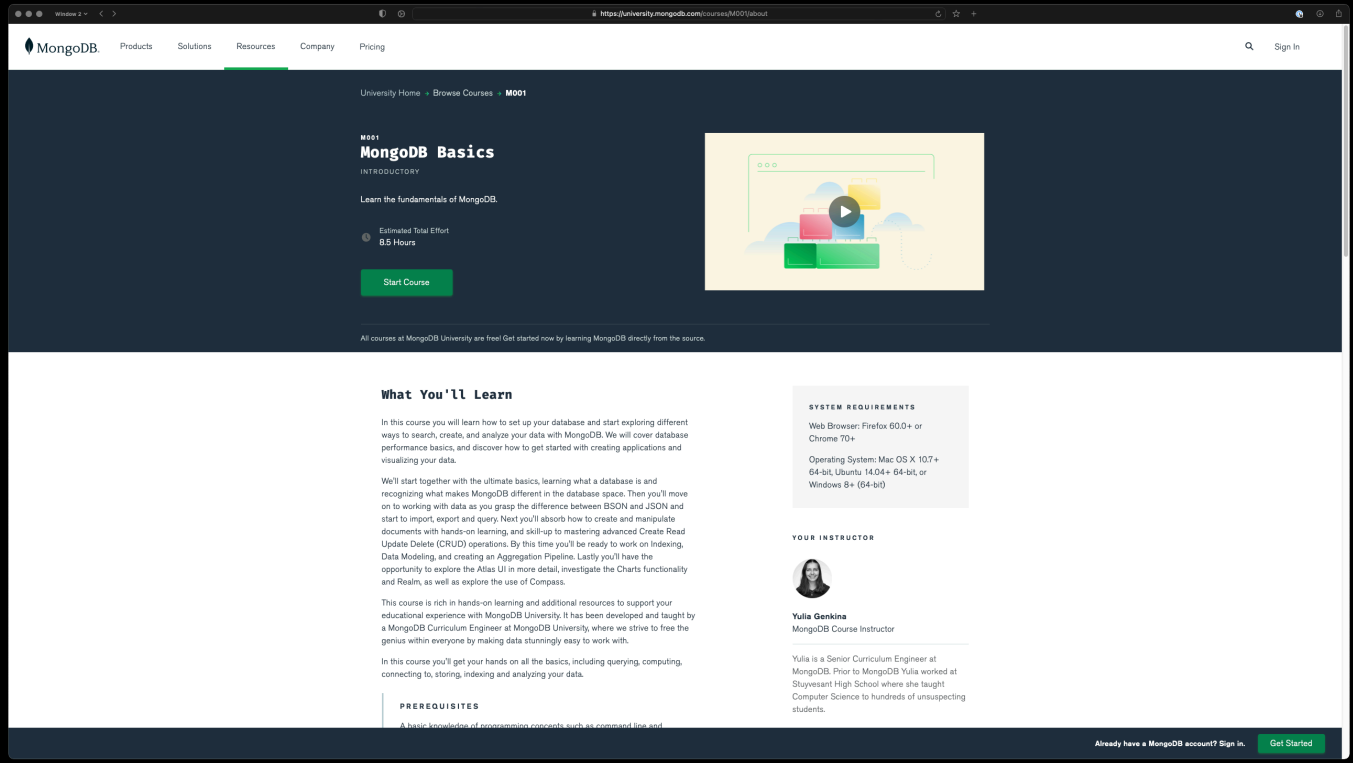
index.js

```
...
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: for A1 to pass you'll need to **find and update**



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**!