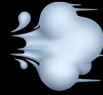


back-end

Teams, MVC and Sessions

lab 6/8



Adem in. Adem uit.

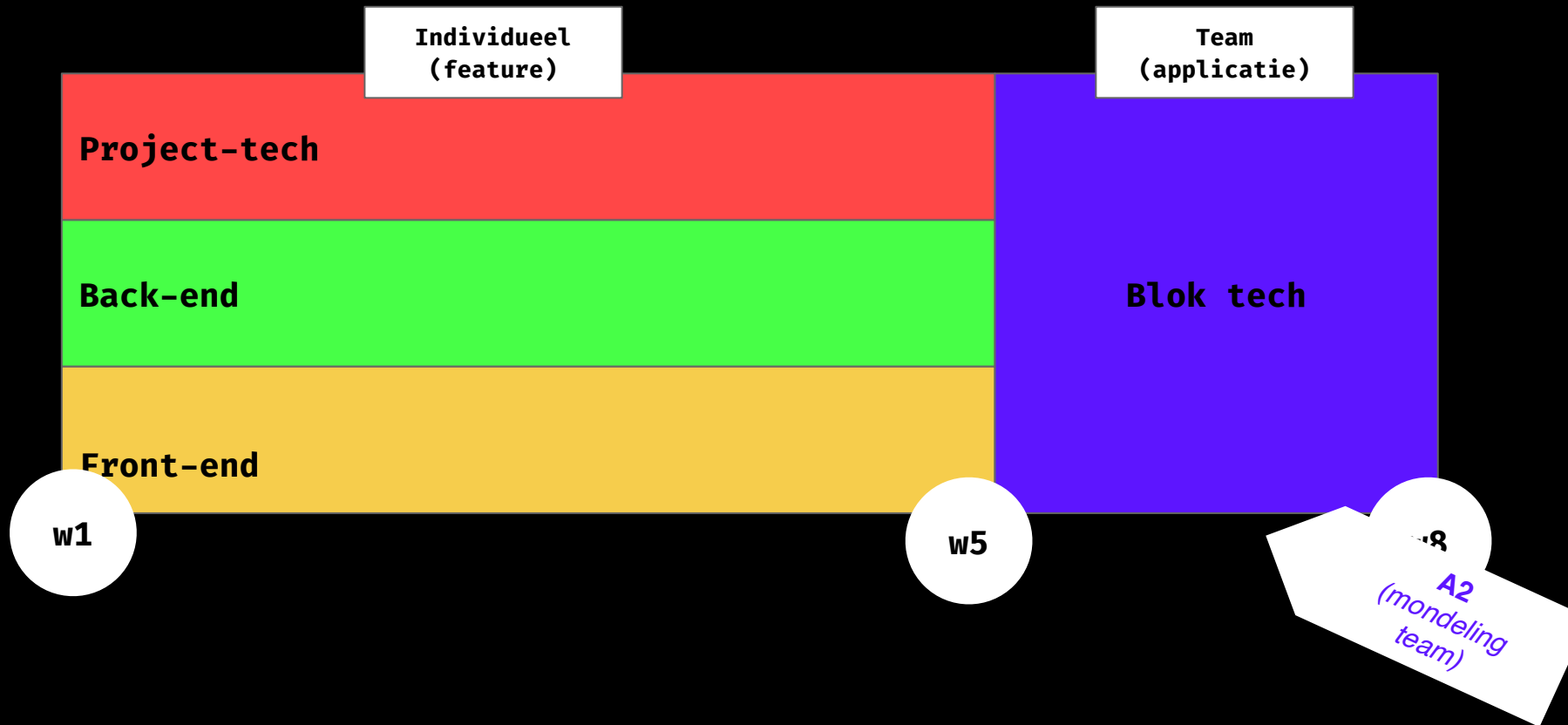
today

- I. Teams explanation
- II. Topics
- III. MVC Model
- IV. Sessions and Storage



Team Assignment

**tl;dr: in een sprint van ~2
weken als team de matching-
app verder uitwerken**



course

deliverables

- ❖ **Team prototype:** working **interactive matching application** with your team
- ❖ **Process book (wiki):** that provides insight into the **weekly iterative process** and the research

course

Grade

Team presentation at the end of the sprint to show the project. Then an assessment with each teacher from each course to show your contribution **to get an individual grade.**

Improve	Standard	Advanced	
	Application		
	The topic the student picked is implemented in the matching app and the code is of decent		
	Research		
	The student documented the research they did for the topic in their wiki, shows relevant decisions and reflects on the quality of the feature contributed		
	Understanding		
	The student can explain every part of their code, how everything works together and can offer alternatives		

Implementation topic and app

Onderzoek naar topic in wiki

Topics

- [difference between development and production \(article\)](#)

Assignments

Topic

topic

Pick one topic from the list below and work on that topic for your team assessment.

Synopsis

- Homework
- Time: 10:00h
- Goals: subgoal 10,
- Due: before week 7

Description

This and next week you'll pick 1 topic (2 in total) to work on for the team assessment. This can be things like security enhancements, application structure or performance optimizations.

→ [List of topics you can pick from](#)

Something not on the list? Pitch a topic to your teacher!

Hand-in

1. Push your changes:

Hand in your progress in your repository on GitHub under your username.

2. Create an issue:

Mark this assignment as complete by opening an issue on our [GitHub issue tracker](#). Fill in the issue template with the correct information. Include what you did in the description of the issue.

Pick (minimum) one
topic from the list

Add your own!

- [difference between development and production \(article\)](#)

Assignments

Topic

topic

Pick one topic from the list below and work on that topic for your team assessment.

Synopsis

- Homework
- Time: 10:00h
- Goals: subgoal 10,
- Due: before week 7

Description

Pick (minimum) one
topic from the list

Make sure each member of
your team has a **different topic!**
And you **can't pick a topic you already did.**
Divide-and-conquer

Assignment ($\pm 30m$)



Have a look at the topics and discuss with your team members which one you'll want to work on.

- *Does the topic match with the feature you've created?*
- *Is the topic something you want to learn?*
- *Discuss with your team which topic you'll work on.*

MVC Model

MVC Model

?

Model–view–controller (MVC) is a software design pattern commonly used for developing user interfaces that divide the **related program logic into three interconnected elements**. *This is done to separate internal representations of information from the ways information is presented to and accepted from the user.*

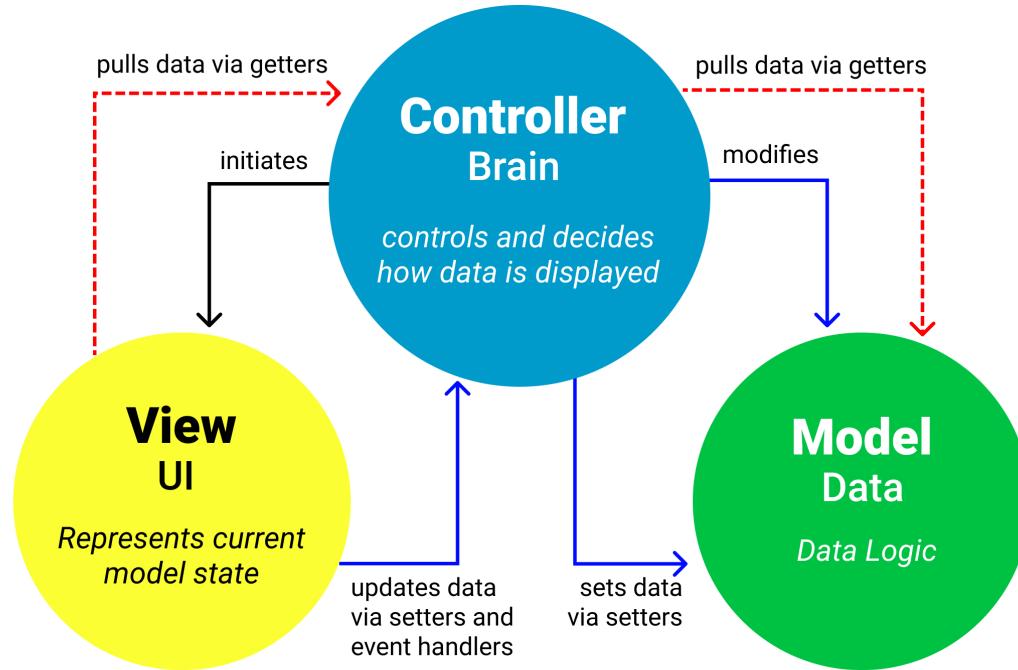
[wikipedia.org](https://en.wikipedia.org/wiki/Model-view-controller)

MVC Model

?

- ❖ **Model**; data structures and relations
- ❖ **View**; user interface (front-end)
- ❖ **Controller**; reacts to events

MVC Architecture Pattern



master 20 branches 0 tags

Go to file

Add file

Code



HappyPantss Merge pull request #54 from StanBankras/staging

fbac353 on Apr 14, 2020

211 commits

helpers	Profile picture uploading works	2 years ago
middleware	Added authentication to routes so users must be logged in to go the...	2 years ago
public	profile pictures	2 years ago
routes	Updated gitignore	2 years ago
services	Now not matching same genders	2 years ago
views	Added required to all inputs	2 years ago
.editorconfig	Editorconfig added & linting errors fixed	2 years ago
.eslintrc	Added ESLint setup	2 years ago
.gitignore	Basic express setup	2 years ago
LICENSE	Initial commit	2 years ago
README.md	Update README.md	2 years ago
package-lock.json	api request now able to be sent and that data is being rendered on t...	2 years ago
package.json	api request now able to be sent and that data is being rendered on t...	2 years ago
server.js	Added styling to form button and fixed a chat issue	2 years ago
websockets.js	Merged	2 years ago



README.md

About

Amatch - Dating app project by Sergio Eijben, Merlijn Bergevoet & Stan Bankras

Readme

MIT License

0 stars

1 watching

1 fork

Releases

No releases published

Packages

No packages published

Contributors 3



standrd Stan



mbergevoet



HappyPantss Sergio Eijben

mongoose

elegant **mongodb** object modeling for **node.js**

[read the docs](#)[discover plugins](#)

Version 6.2.7



Let's face it, **writing MongoDB validation, casting and business logic boilerplate is a drag**. That's why we wrote Mongoose.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/test');

const Cat = mongoose.model('Cat', { name: String });

const kitty = new Cat({ name: 'Zildjian' });
kitty.save().then(() => console.log('meow'));
```

Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

Assignment ($\pm 30m$)



Research more about the MVC Model, write it down and try to explain it in your own terms.

- *Can you find real-life examples?*
- *Are there similar 'architecture' models based on mvc?*
- *How does MVC work in a node.js / mongodb project?*

Sessions

Sessions

?

A session is a **temporary and interactive information** interchange between two or more communicating devices, or between a computer and user (see login session).

A session is established at a certain point in time, and then 'torn down' - brought to an end - at some later point.

Sessions

?

Client-side sessions use cookies and cryptographic techniques to maintain state **without storing as much data on the server.**

When presenting a dynamic web page, the server sends the current state data to the client (web browser) in the form of a cookie. The client saves the cookie in memory or on disk.

Sessions

?

HTTP is stateless connection protocol, that is, the server cannot differentiate between different connections of different users. [..]

Once a client connects first time to a server, the server generates a new session ID, which later will be sent to the client as cookie value. And from now on, **this session id will identify that client connection.**

Gratis verzending vanaf 20,-

Bezorging dezelfde dag, 's avonds of in het weekend*

Gratis retourneren

Select Ontdek nu de 4 voordelen

bol.com

Waar ben je naar op zoek?



Welkom
Danny



Categorieën ▾

Cadeaus & Inspiratie ▾

Aanbiedingen ▾

Zakelijk

Cadeaukaart

Bestelstatus

Klantenservice

NL ▾

Select-Deals

Huis- & tuininspiratie

Elektronica Acties

Genieten in de tuin

Nieuwe collectie mode >

Mijn winkelwagentje

Bestel nu en je bestelling wordt gratis verzonden!



Clean Code

Robert Martin

Engelstalig - Paperback

★★★★★ (23)

Op voorraad. Voor 23:59 uur besteld,
dinsdag in huis

+ **Select** bezorgopties

Aantal

1 ▾

€ 46,99

Verplaats naar verlanglijstje

Verwijder

Select

Neem **Select** erbij voor €9,99 p.j.
en krijg meer gemak en voordeel

> Wat is Select?



> Cadeaukaartcode invoeren

Totaal artikelen (1)

€ 46,99

Verzendkosten ⓘ

Geen

Totaal

€ 46,99

> Verder naar bestellen

Sessions

Examples

- ❖ Are used to check if **somebody has logged in**
- ❖ **Show recently visited** product on webshop
- ❖ **Multi-page forms** to avoid database calls

Sessions

Examples

Local Storage	Session Storage	Cookies
The storage capacity of local storage is 5MB/10MB	The storage capacity of session storage is 5MB	The storage capacity of Cookies is 4KB
As it is not session-based, it must be deleted via javascript or manually	It's session-based and works per window or tab. This means that data is stored only for the duration of a session, i.e., until the browser (or tab) is closed	Cookies expire based on the setting and working per tab and window
The client can only read local storage	The client can only read local storage	Both clients and servers can read and write the cookies
There is no transfer of data to the server	There is no transfer of data to the server	Data transfer to the server is exist
There are fewer old browsers that support it	There are fewer old browsers that support it	It is supported by all the browser including older browser

sessions

set-up

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

```
bash
$ npm install express-session
+ express-session@1.15.6
added 5 packages in 1.749s
$
```

sessions

set-up

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


express-session handles sessions
through cookies

```
bash
$ npm install express-session
+ express-session@1.15.6
added 5 packages in 1.749s
$
```

sessions

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

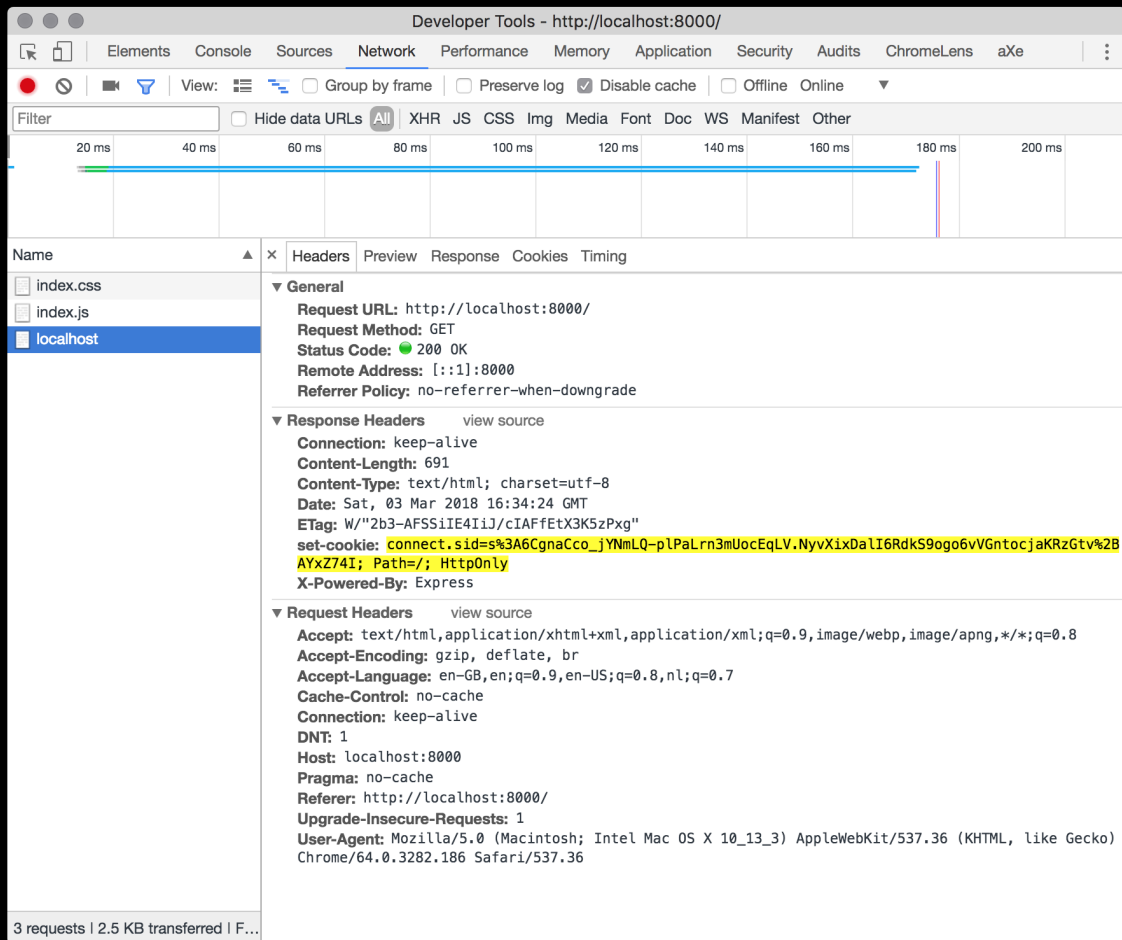
set-up

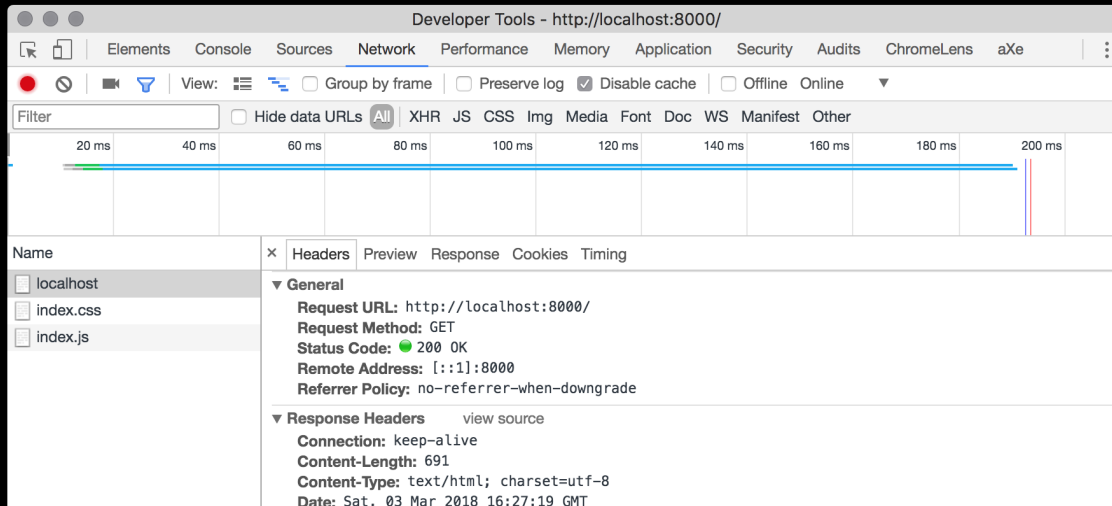


```
.env
DB_HOST=localhost
DB_USER=root
DB_NAME=mymoviewebsite
DB_PASSWORD=mypassword
SESSION_SECRET=ilikecats
```

index.js

```
...  
var session = require('express-session')  
  
...  
  
express()  
...  
  
  .use(session({  
    resave: false,  
    saveUninitialized: true,  
    secret: process.env.SESSION_SECRET  
  }))  
...  
  .listen(8000)
```





Localhost uses cookies

sluiten

view/list.ejs

```
<% include head.ejs %>
<title>Movies - My movie website</title>
<h1>Movies</h1>
<p>
  <% if (user) { %>
    Hello <%= user.username %>!
  <% } else { %>
    <a href=/log-in>Log in</a>
    or
    <a href=/sign-up>Sign up</a>
  <% } %>
</p>

...

<% if (user) { %>
  <a href=/add>Add a movie</a>
<% } %>
<% include tail.ejs %>
```


Assignment ($\pm 30m$)



Think of use cases in your team application where you could use sessions.

- *Where do you need temporary storage?*
- *Login / Register flow?*
- *Multi-page forms?*

Questions?



Feedback?



exit;

see you in **lab-7!**