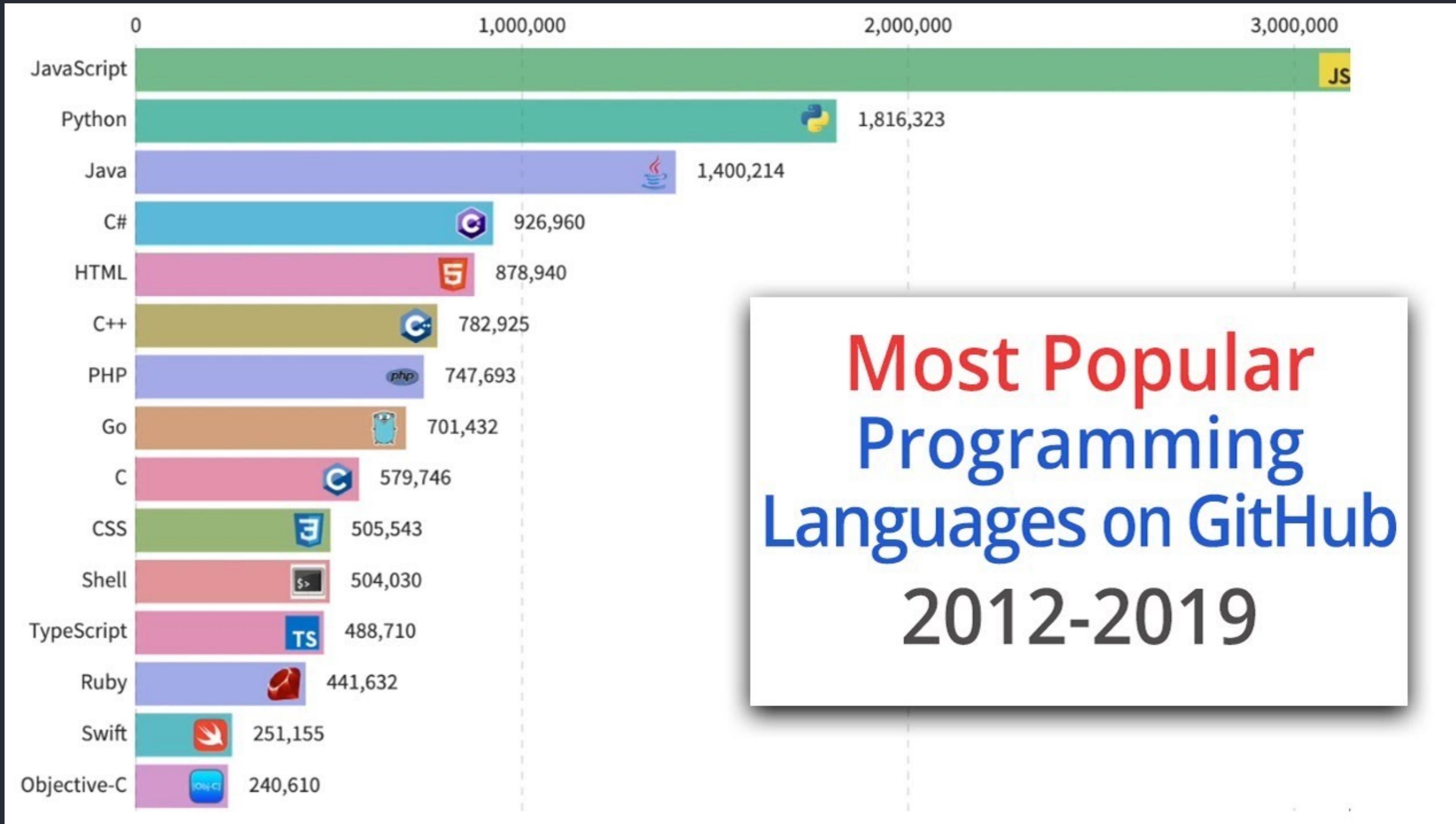
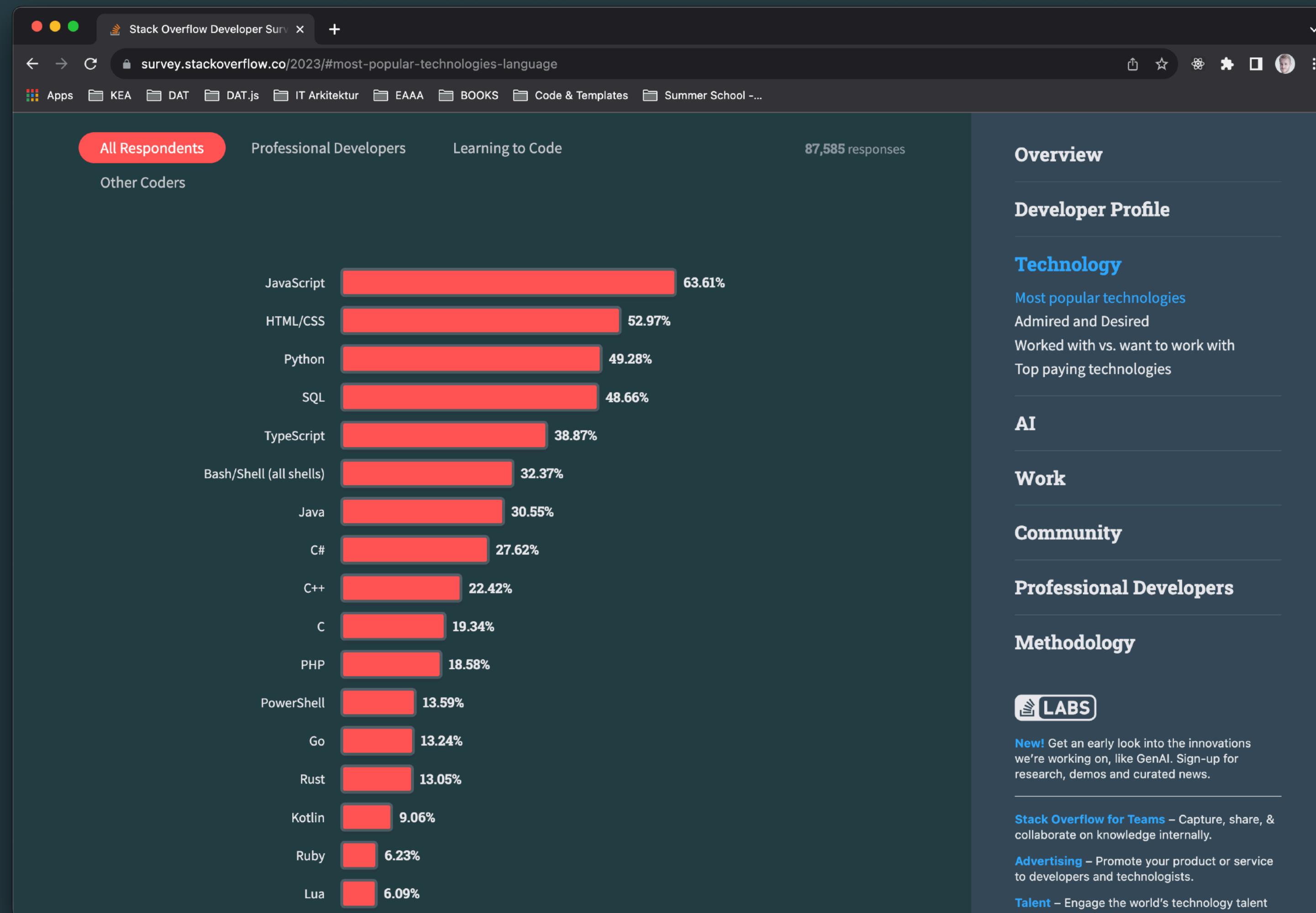


# Hvis I havde glemt det



# Most popular technologies



<https://survey.stackoverflow.co/2023/#most-popular-technologies-language>

# Integrated development environment



<https://survey.stackoverflow.co/2023/#section-worked-with-vs-want-to-work-with-integrated-development-environment>

# Most common web technologies



<https://survey.stackoverflow.co/2023/#most-popular-technologies-webframe>

“

The cool thing about JavaScript is  
that you can create stuff that will put  
a smile on your face, as a developer.

You can create stuff and it will  
visually look like something, and it'll  
do stuff, and it makes you feel good,  
it makes you fall in love with  
programming.

”

**Lex Fridman**  
Computer Scientist

<https://www.youtube.com/watch?v=GLhyjVZp0cw&t=252s>

# Intro til full-stack programmering

2. semester

# Formål

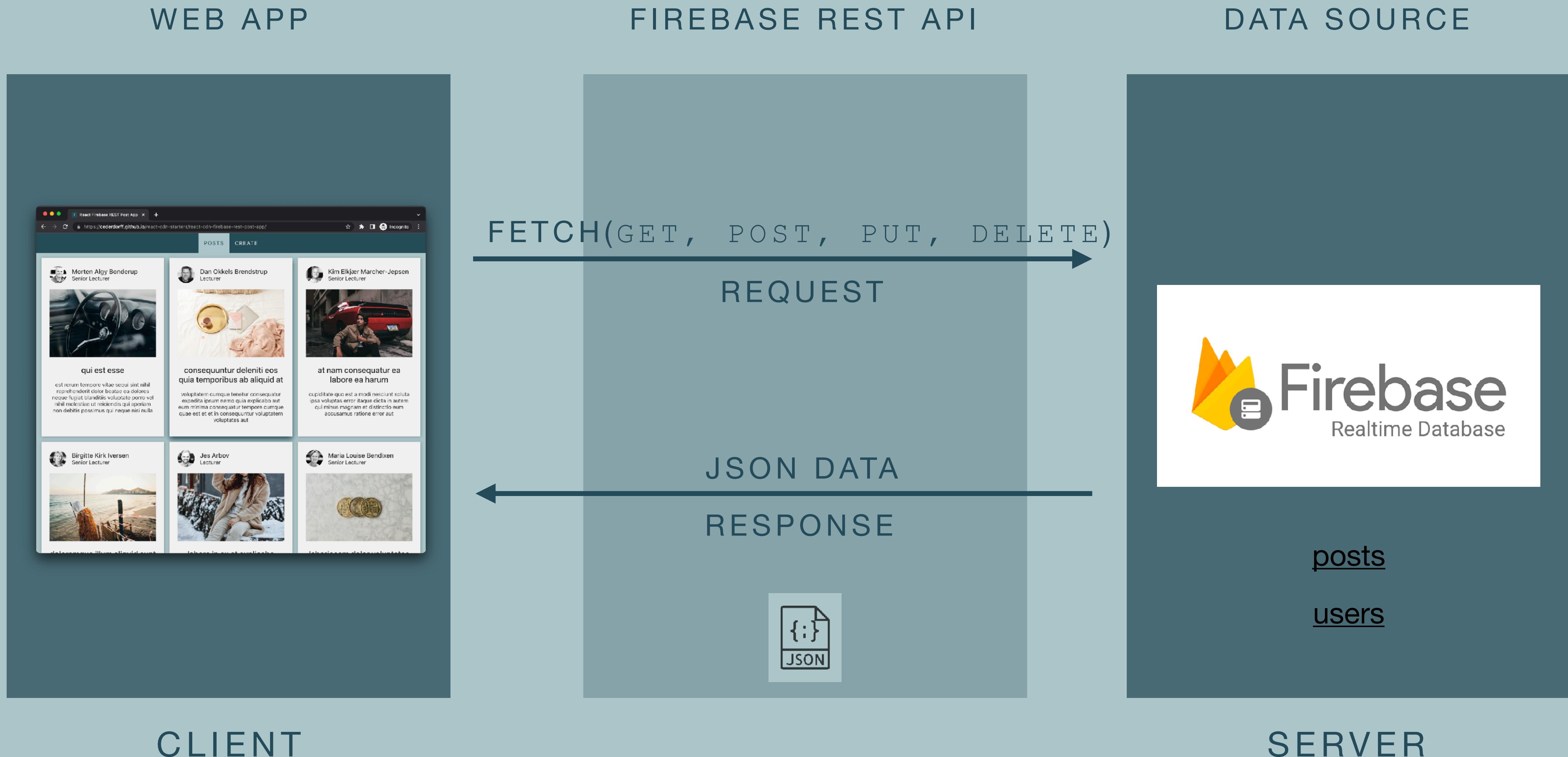
- Intro til backendprogrammering med Node.js og Express.js
- Udvikle et simpelt REST API
- Kobling mellem frontend (1. semester) og backend.
- Semesteroversigt og -indhold

1. Velkommen til 2. semester
2. Webudvikling og intro til dagens øvelse
3. Øvelse: Backendapp med Node.js og Express
4. Semesterstruktur og -oversigt



# Agenda

# Fetch, REST, HTTP Request & Response



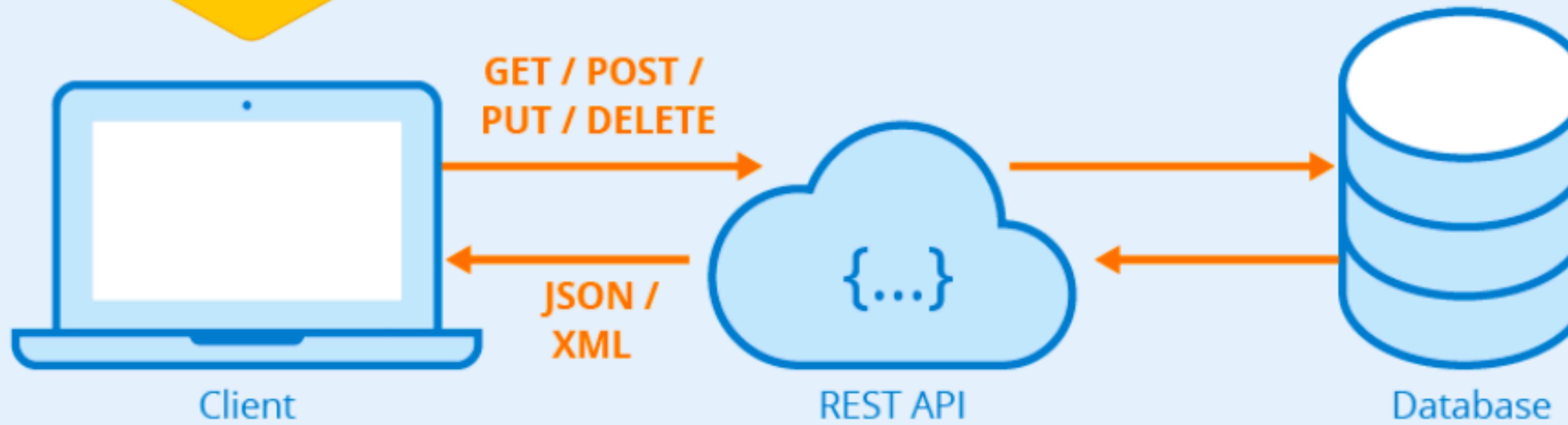
When someone asks you how to get data from a database in a js code



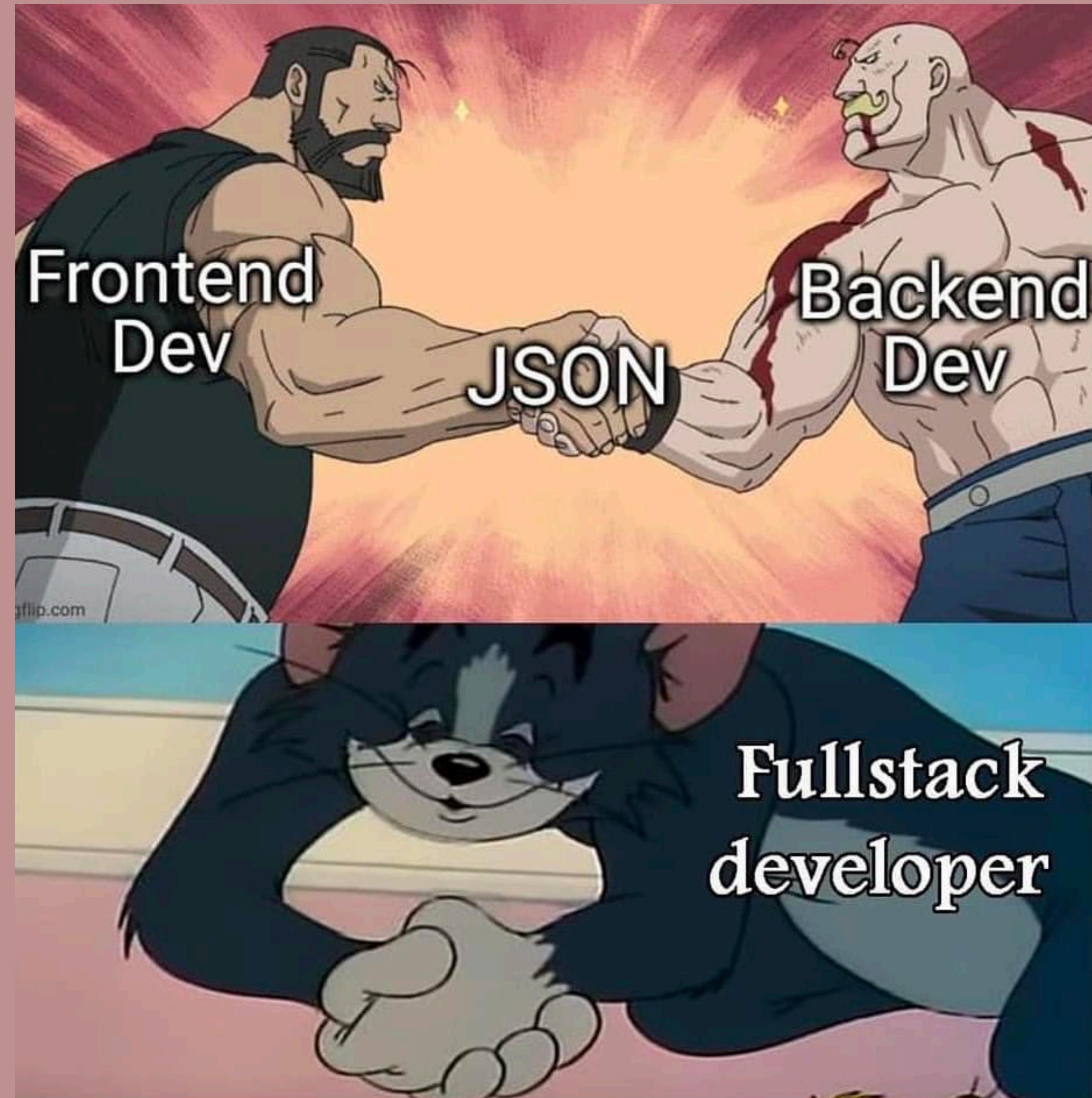
made with mematic



# Firebase

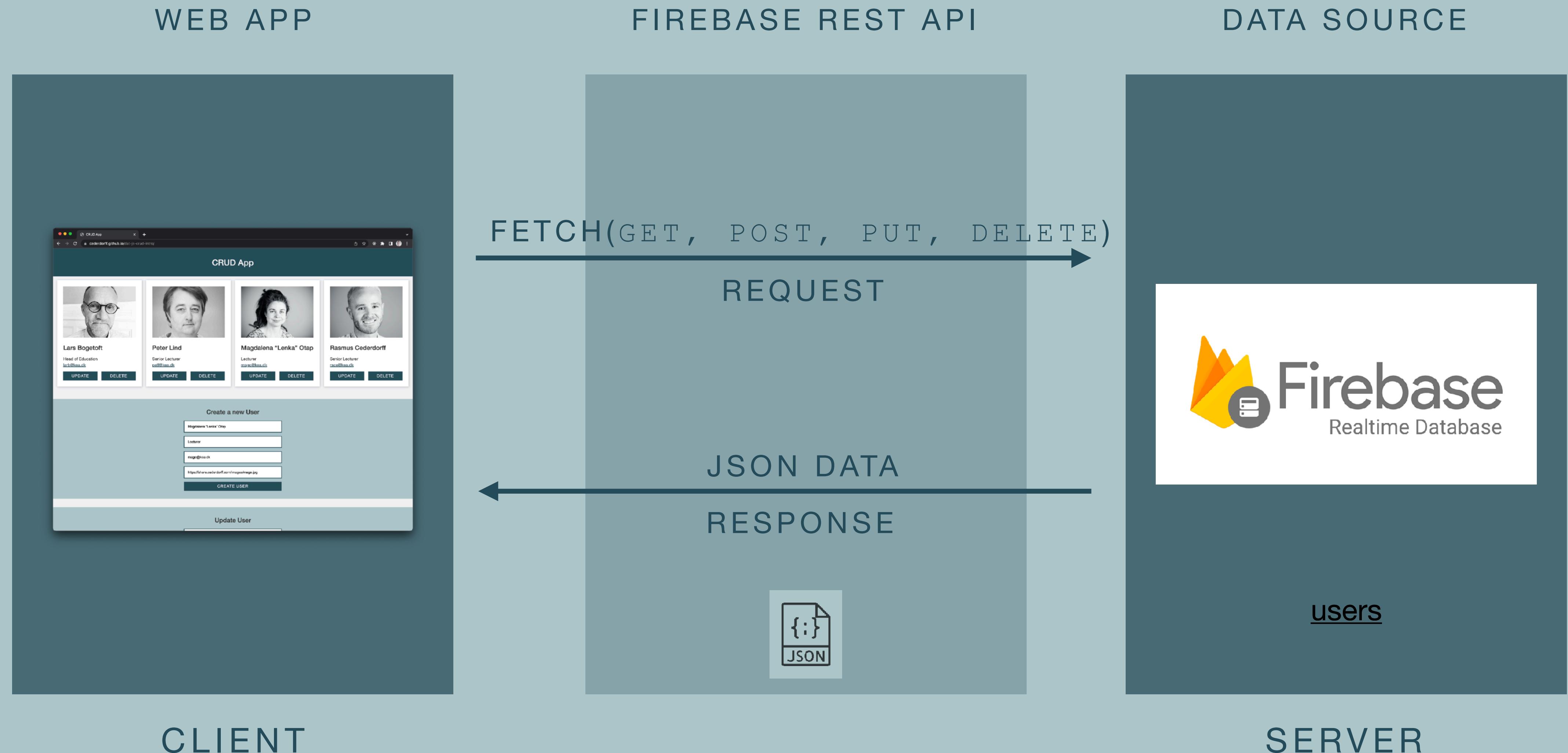




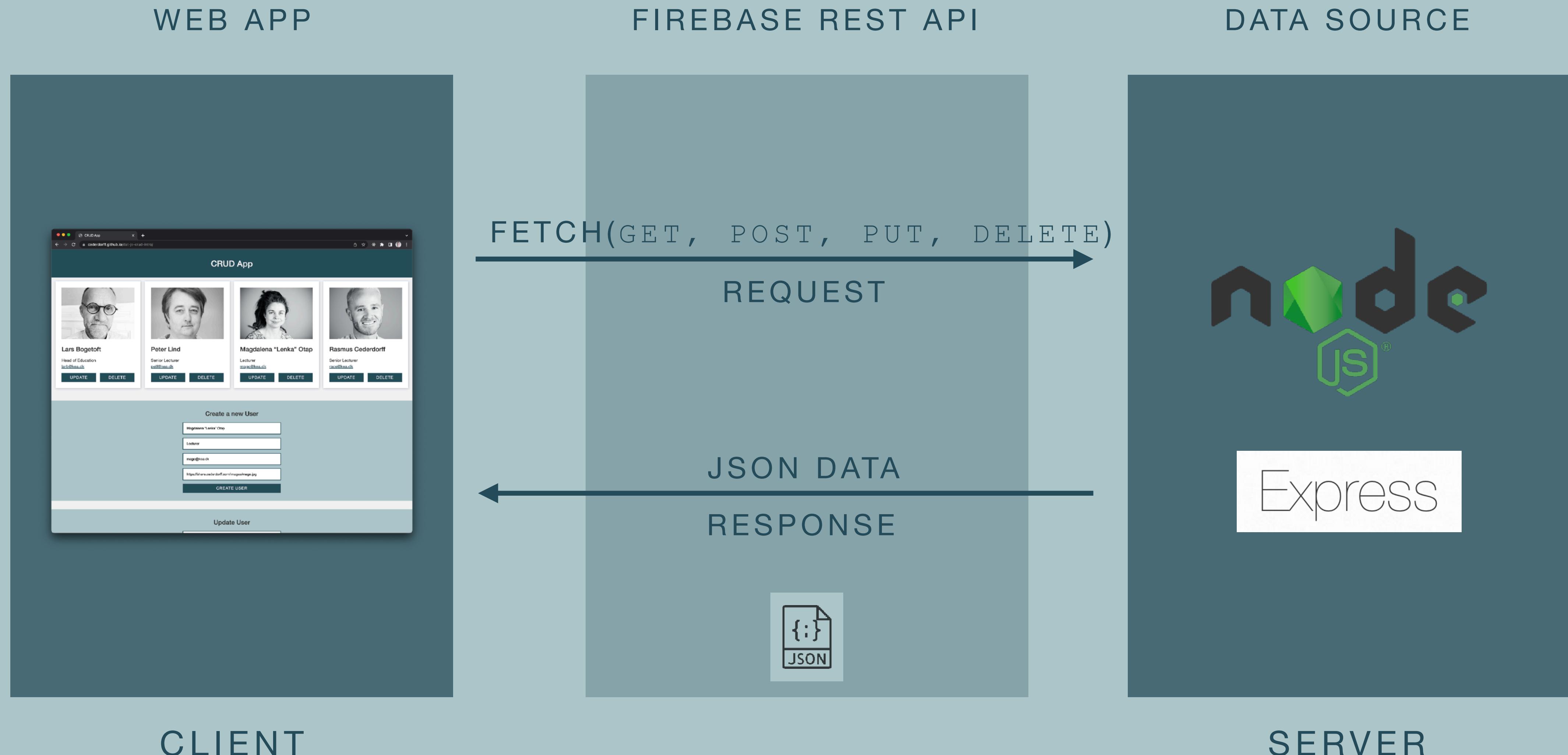


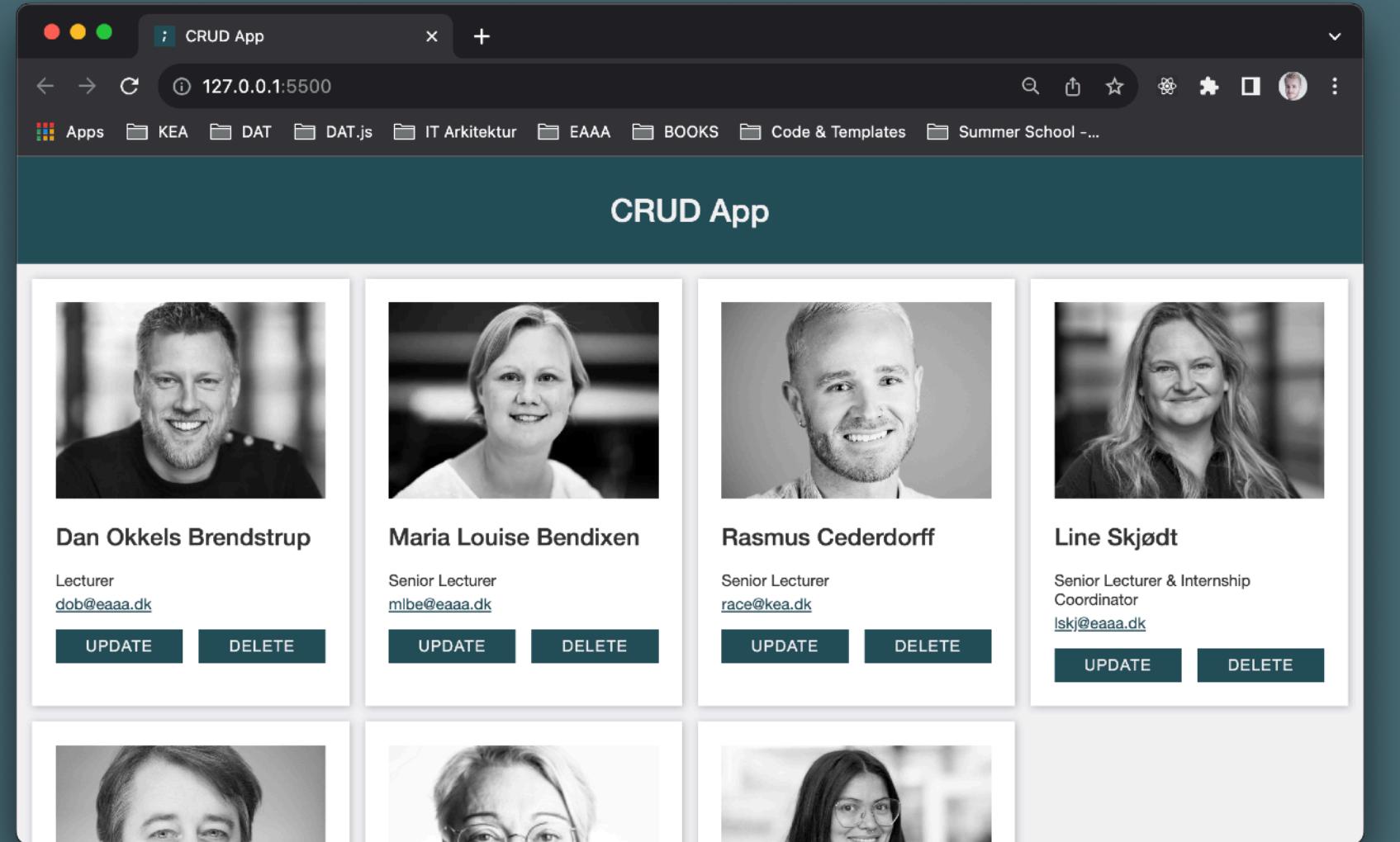
<https://www.instagram.com/p/CVqbCzgsZUF/>

# Web Development



# Web Development

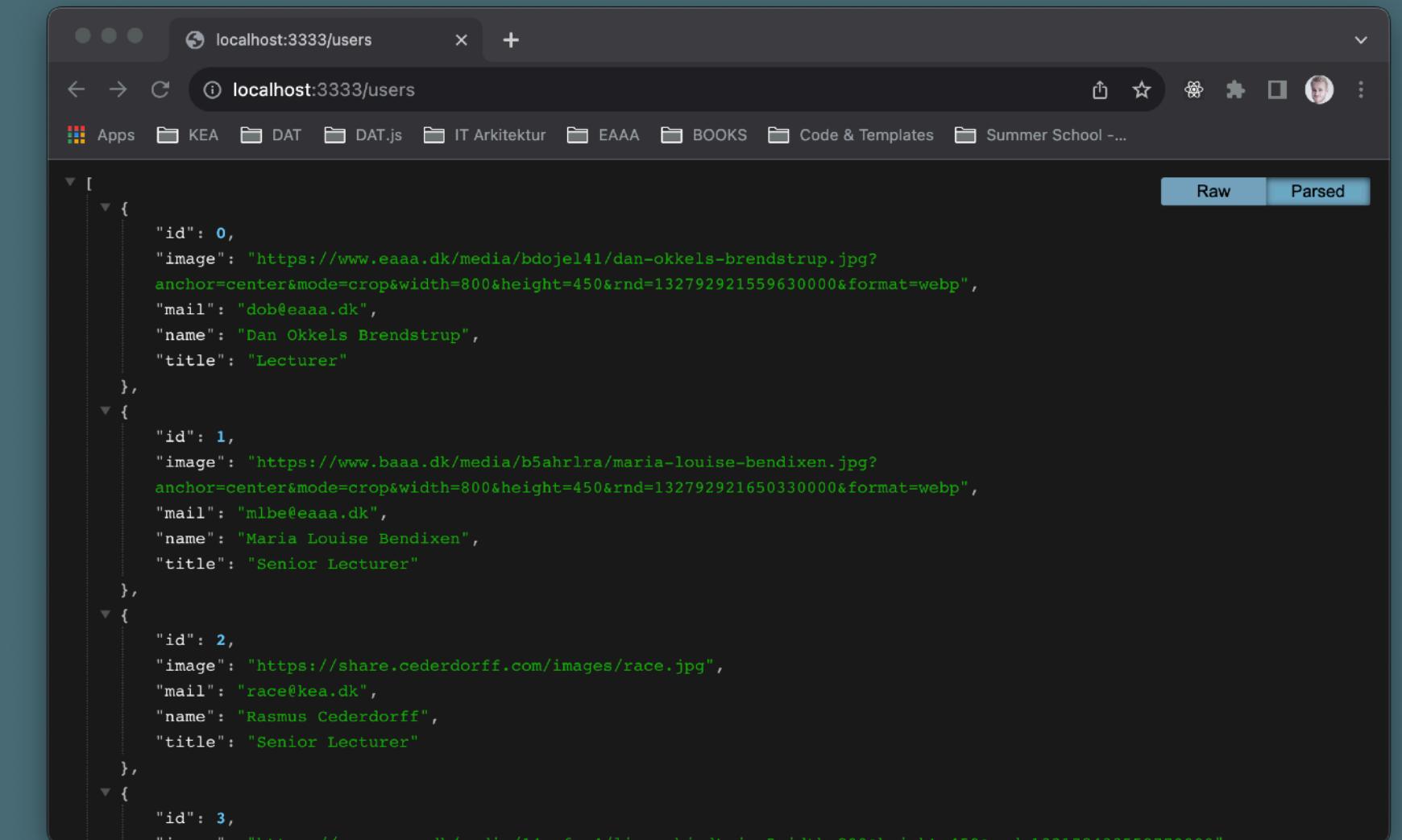




```
app.js — dat-js-crud-intro
JS app.js > ...
25
26 // Read (GET) all users from backend using REST API
27 async function readUsers() {
28   const response = await fetch(`${endpoint}/users`);
29   const data = await response.json();
30   return data;
31 }
32
33 // Create HTML and display all users from given list
34 function displayUsers(list) {
35   // reset <section id="users-grid" class="grid-container">...</section>
36   document.querySelector("#users-grid").innerHTML = "";
37   //loop through all users and create an article with content for each
38   for (const user of list) {
39     document.querySelector("#users-grid").insertAdjacentHTML(
40       "beforeend",
41       /*html*/
42       <article>
```

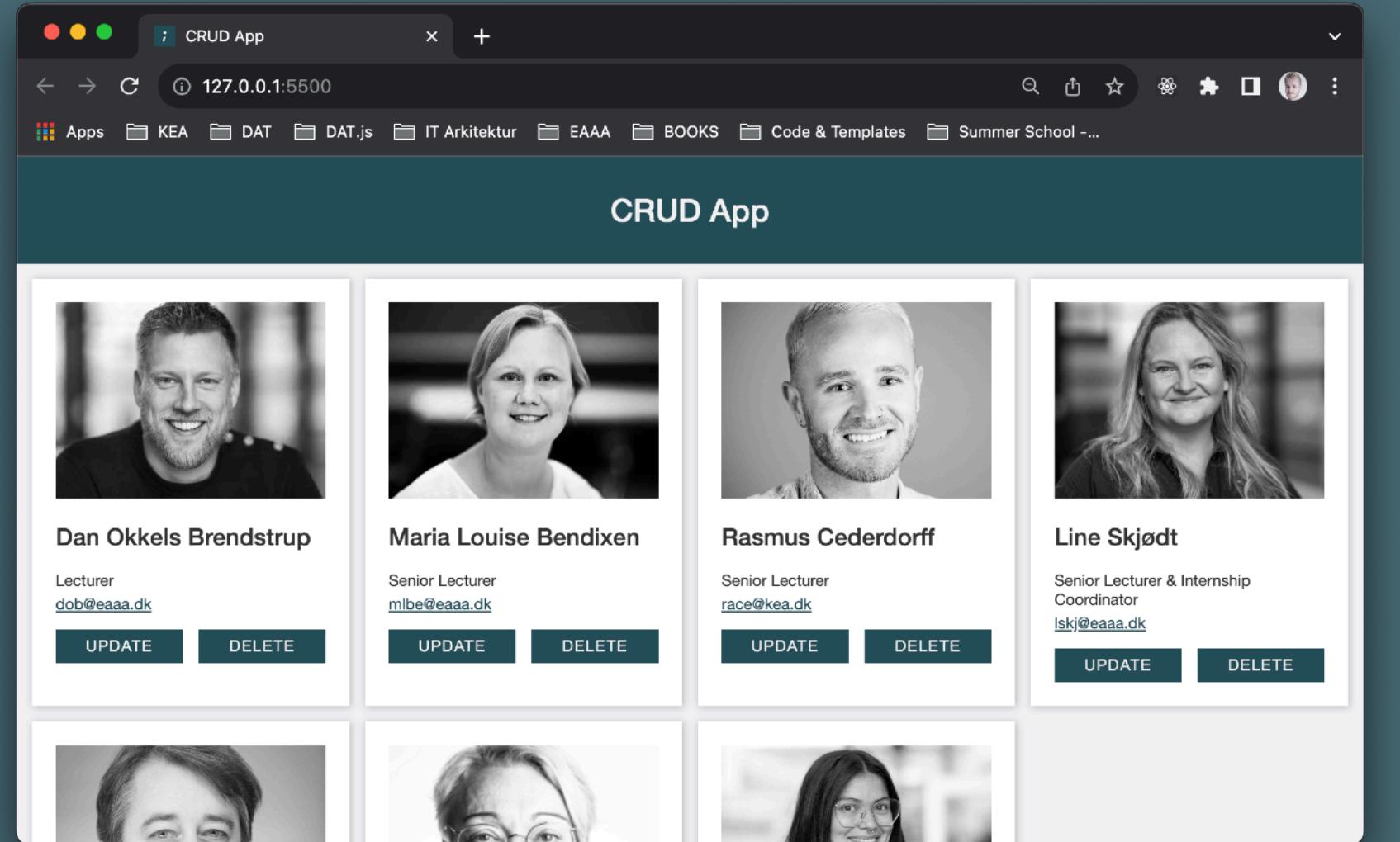
Frontend

REQUEST



```
app.js — node-users-rest-api
JS app.js > ⚡ app.put("/users/:id") callback
19 // READ all users - "/users" : GET
20 app.get("/users", async (request, response) => {
21   const data = await fs.readFile("data.json");
22   const users = JSON.parse(data);
23   response.json(users);
24 });
25
26 // CREATE user - "/users" : POST
27 app.post("/users", async (request, response) => {
28   const newUser = request.body;
29   newUser.id = new Date().getTime();
30   console.log(newUser);
31
32   const data = await fs.readFile("data.json");
33   const users = JSON.parse(data);
34
35   users.push(newUser);
36   console.log(newUser);
```

Backend



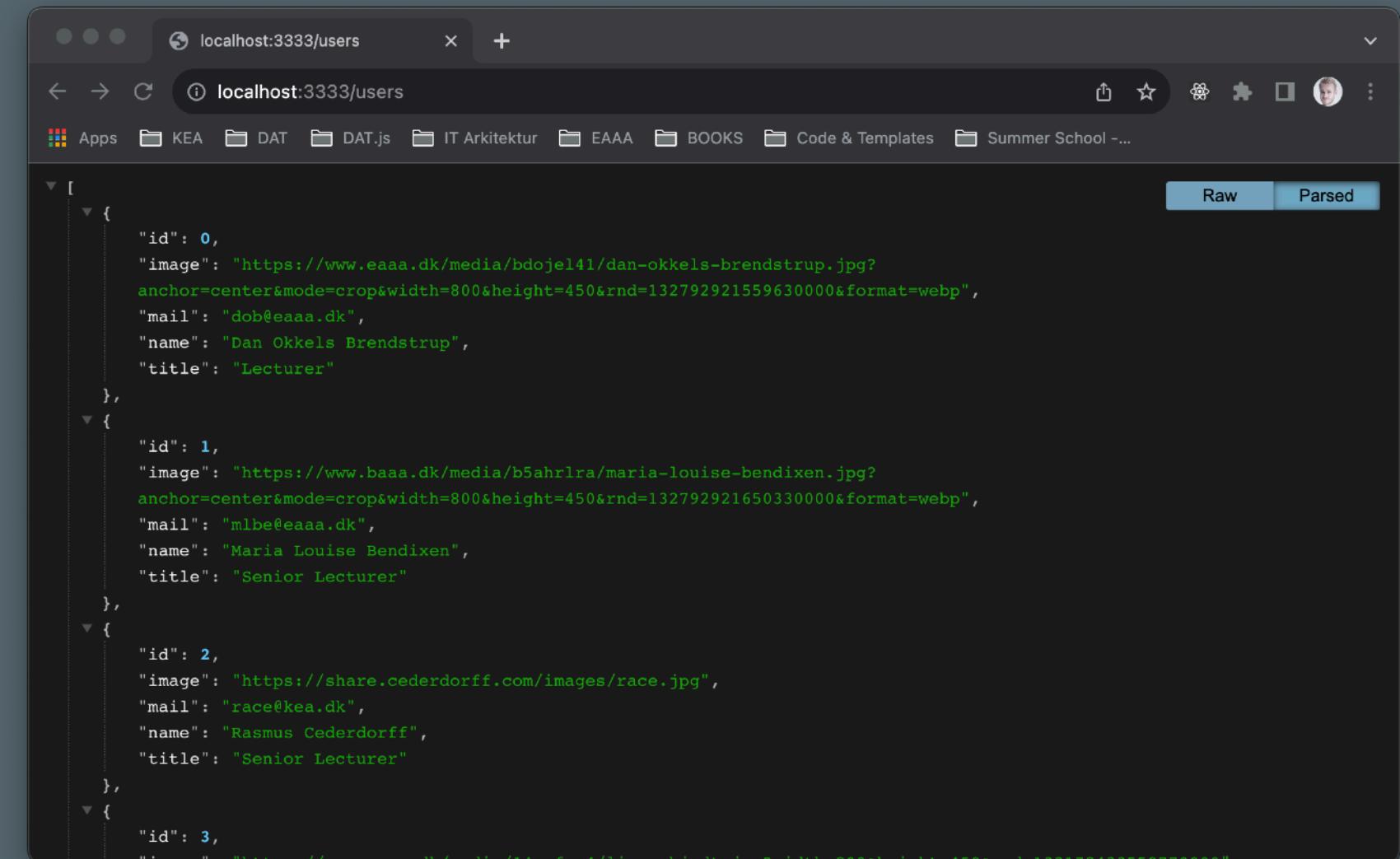
```
JS app.js
JS app.js > ...
25
26 // Read (GET) all users from backend using REST API
27 async function readUsers() {
28   const response = await fetch(`${endpoint}/users`);
29   const data = await response.json();
30   return data;
31 }
32
33 // Create HTML and display all users from given list
34 function displayUsers(list) {
35   // reset <section id="users-grid" class="grid-container">...</section>
36   document.querySelector("#users-grid").innerHTML = "";
37   //loop through all users and create an article with content for each
38   for (const user of list) {
39     document.querySelector("#users-grid").insertAdjacentHTML(
40       "beforeend",
41       /*html*/
42       <article>
```

Frontend

REQUEST



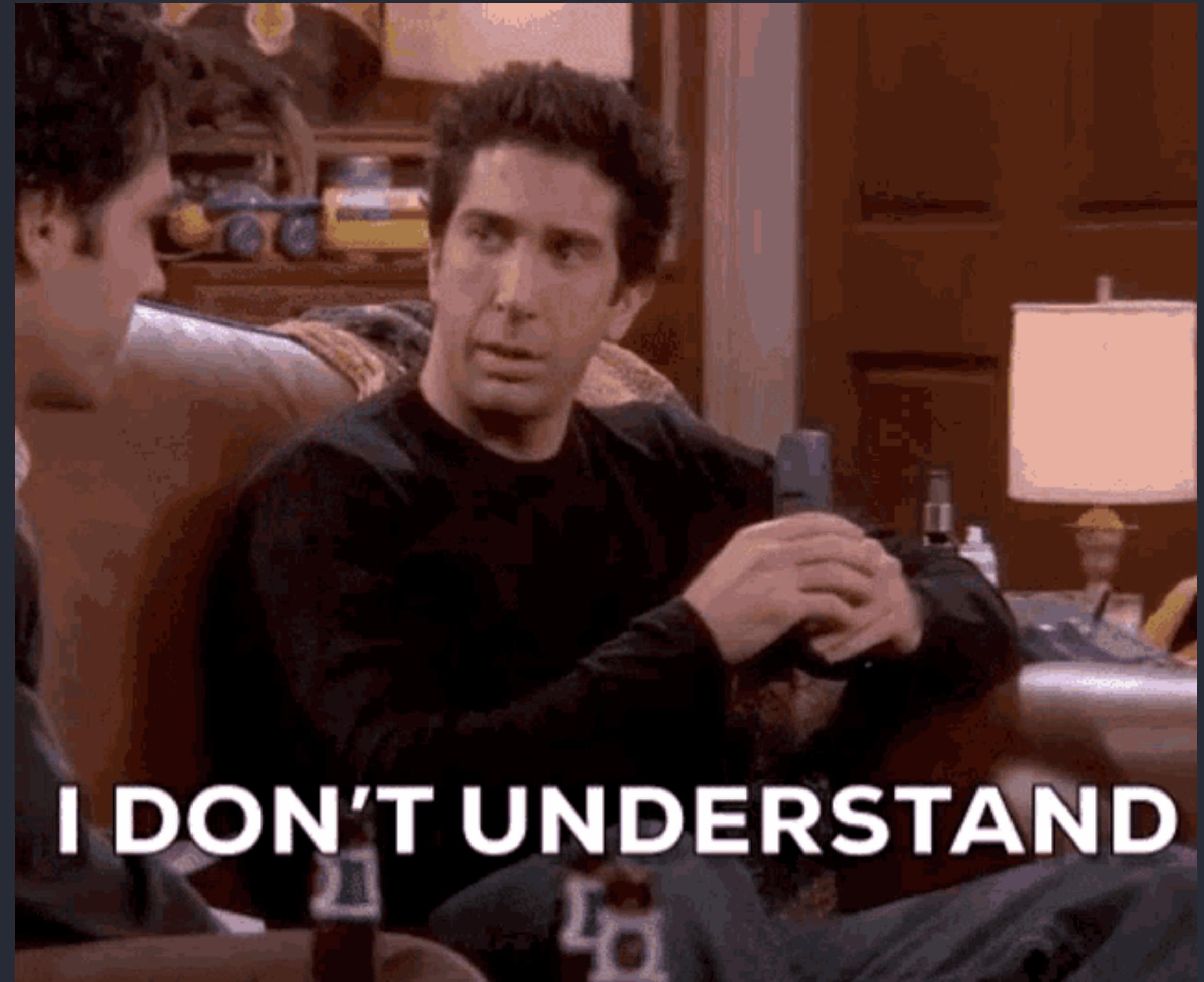
RESPONSE



```
JS app.js
JS app.js > ⚡ app.put("/users/:id") callback
19 // READ all users - "/users" : GET
20 app.get("/users", async (request, response) => {
21   const data = await fs.readFile("data.json");
22   const users = JSON.parse(data);
23   response.json(users);
24 });
25
26 // CREATE user - "/users" : POST
27 app.post("/users", async (request, response) => {
28   const newUser = request.body;
29   newUser.id = new Date().getTime();
30   console.log(newUser);
31
32   const data = await fs.readFile("data.json");
33   const users = JSON.parse(data);
34
35   users.push(newUser);
36   console.log(newUser);
```

Backend

Please,  
Do not try to  
understand  
everything!



I DON'T UNDERSTAND

# Øvelse

## Introduction to Backend

### Development with

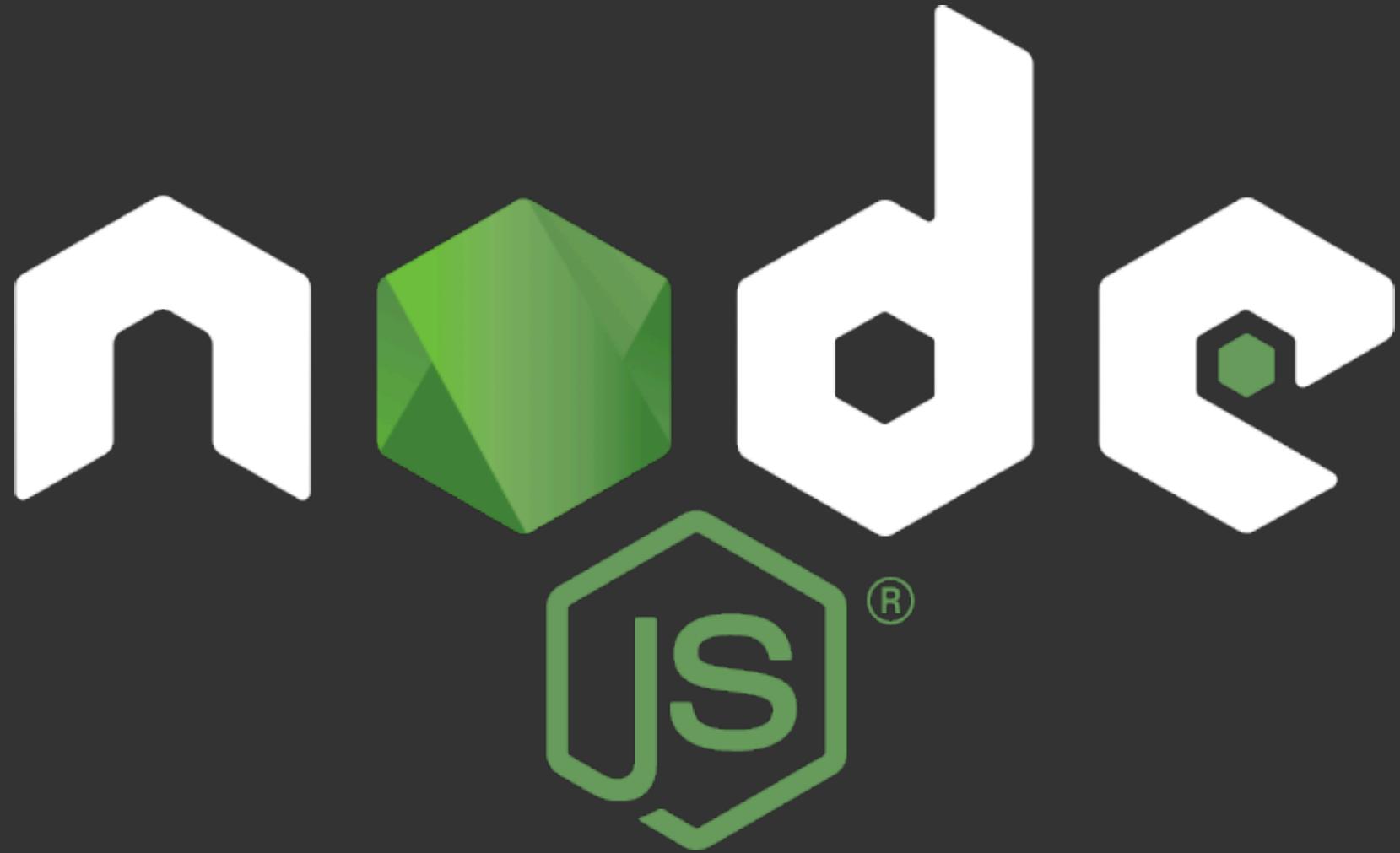
### Node.js & Express.js

The screenshot shows a web application titled "CRUD App" running in a browser window. The main content area displays four user profiles in a grid:

- Lars Bogetoft**  
Head of Education  
[larb@kea.dk](mailto:larb@kea.dk)  
[UPDATE](#) [DELETE](#)
- Peter Lind**  
Senior Lecturer  
[pell@kea.dk](mailto:pell@kea.dk)  
[UPDATE](#) [DELETE](#)
- Magdalena "Lenka" Otap**  
Lecturer  
[mago@kea.dk](mailto:mago@kea.dk)  
[UPDATE](#) [DELETE](#)
- Rasmus Cederdorff**  
Senior Lecturer  
[race@kea.dk](mailto:race@kea.dk)  
[UPDATE](#) [DELETE](#)

Below this, there are two sections:

- Create a new User**  
Form fields:
  - 
  - 
  - 
  -[CREATE USER](#)
- Update User**  
Form fields:
  - 
  - 
  - 
  -



- Server-side JavaScript runtime environment.
- Executes JavaScript outside of the browser.

# Run JavaScript, not just in browsers



A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows a project folder named "NODE-JS-FUN" containing an "app.js" file. The main editor area displays the following code:

```
const yourFavoriteLecturer = {  
  name: "Peter Lind",  
  mail: "petl@kea.dk"  
};  
console.log(yourFavoriteLecturer);
```

The terminal at the bottom shows the output of running the script:

```
race@RACEs-MacBook-Pro node-js-fun % node app.js  
{ name: 'Peter Lind', mail: 'petl@kea.dk' }  
race@RACEs-MacBook-Pro node-js-fun %
```

The status bar at the bottom indicates the code is in "JavaScript" mode, has 0 errors and 0 warnings, and shows the current position is "Ln 7, Col 1".

# Express

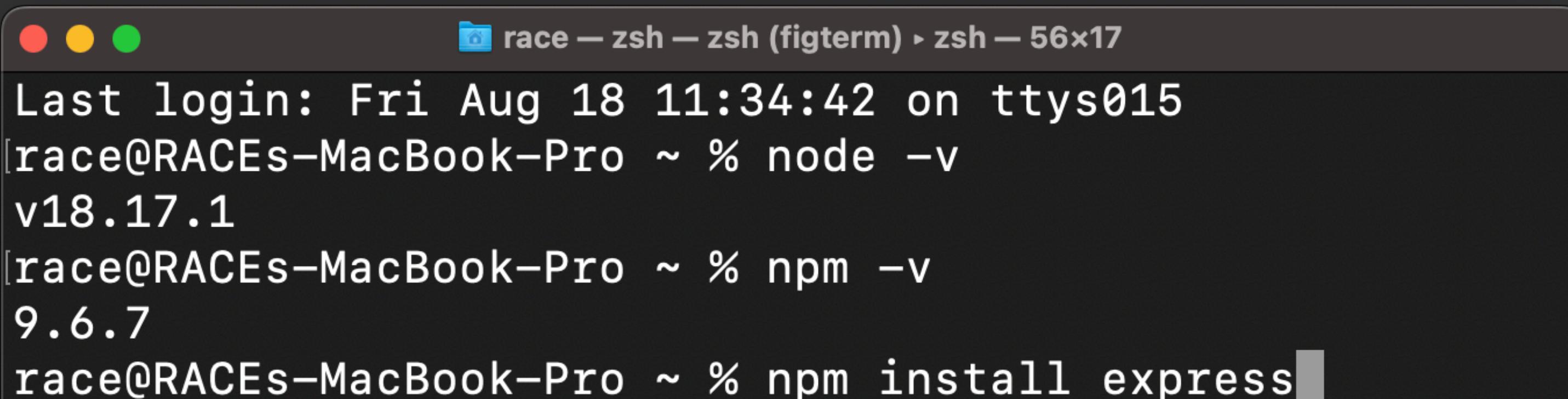


- A popular, minimalist web application framework for building APIs and web applications using Node.js.
- It simplifies the process of building web applications and APIs by providing a set of tools and utilities.

# Command Line Interface

... a program that allows users to type text commands to execute functions.

npm uses CLI to install software & packages.



A screenshot of a macOS terminal window titled "race — zsh — zsh (figterm) ▸ zsh — 56x17". The window shows the following command history:

```
Last login: Fri Aug 18 11:34:42 on ttys015
[race@RACEs-MacBook-Pro ~ % node -v
v18.17.1
[race@RACEs-MacBook-Pro ~ % npm -v
9.6.7
race@RACEs-MacBook-Pro ~ % npm install express█
```

EXPLORER

node-express-rest-users

New Terminal ⌘T

Split Terminal ⌘⌥⌘7

Run Task...

Run Build Task... ⌘⌥B

Run Active File

Run Selected Text

Show Running Tasks...

Restart Running Task...

Terminate Task...

Configure Tasks...

Configure Default Build Task...

node\_modules

.env

.gitattributes

.gitignore

app.js

data.json

package-lock.json

package.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE COMMENTS

race@RACEs-MacBook-Pro node-express-rest-users % node -v  
v18.17.1  
race@RACEs-MacBook-Pro node-express-rest-users % npm -v  
9.6.7  
race@RACEs-MacBook-Pro node-express-rest-users % npm install express



1. Download & install Node.js & npm: [nodejs.org/en/](https://nodejs.org/en/)
2. Check your version of npm & Node.js

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

AZURE

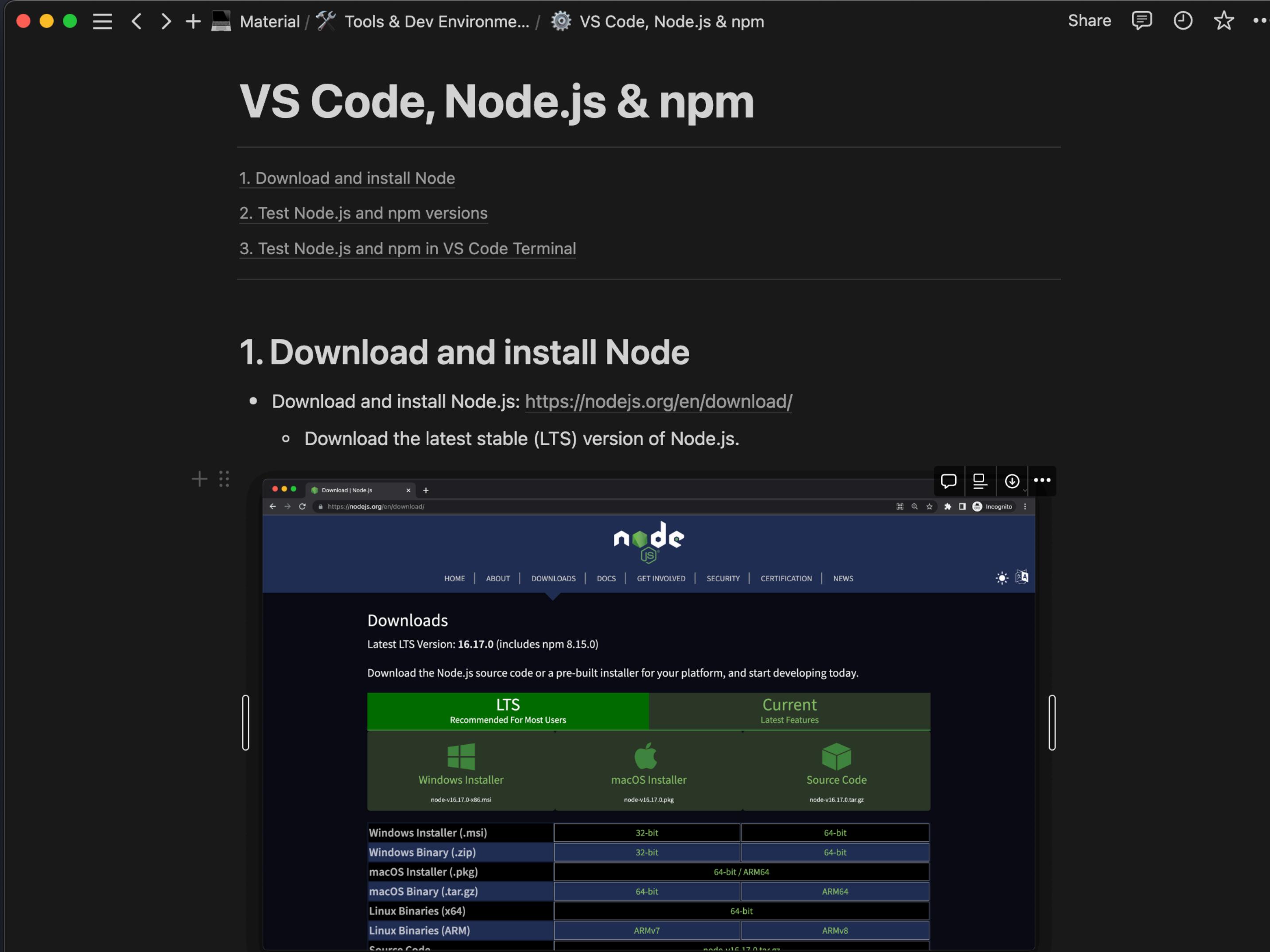
COMMENTS

---

```
race@RACEs-MacBook-Pro node-express-rest-users % node -v  
v18.17.1
```

```
race@RACEs-MacBook-Pro node-express-rest-users % npm -v  
9.6.7
```

# VS Code, Node.js & npm



The screenshot shows a browser window with a dark theme. The address bar indicates the page is about "VS Code, Node.js & npm". The main content area has a large heading "VS Code, Node.js & npm". Below it is a numbered list:

1. Download and install Node
2. Test Node.js and npm versions
3. Test Node.js and npm in VS Code Terminal

Under the heading "1. Download and install Node", there is a bulleted list:

- Download and install Node.js: <https://nodejs.org/en/download/>
  - Download the latest stable (LTS) version of Node.js.

The screenshot also includes a screenshot of the Node.js download page itself, showing the LTS and Current sections with various installer and source code options for Windows, macOS, and Linux.

LTS	Current	
Recommended For Most Users	Latest Features	
Windows Installer node-v16.17.0-x64.msi	macOS Installer node-v16.17.0.pkg	
Source Code node-v16.17.0.tar.gz		
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v16.17.0.tar.gz	

A close-up photograph of a person's hands interacting with a black Wi-Fi router. One hand is holding a blue Ethernet cable and is in the process of plugging it into one of the four gold-colored ports on the front panel of the router. The router has three external black antennas. The background is blurred.

# What's a Router?

# What's a Router?

*“It is the piece of software in charge to organize the states of the application, switching between different views.”*

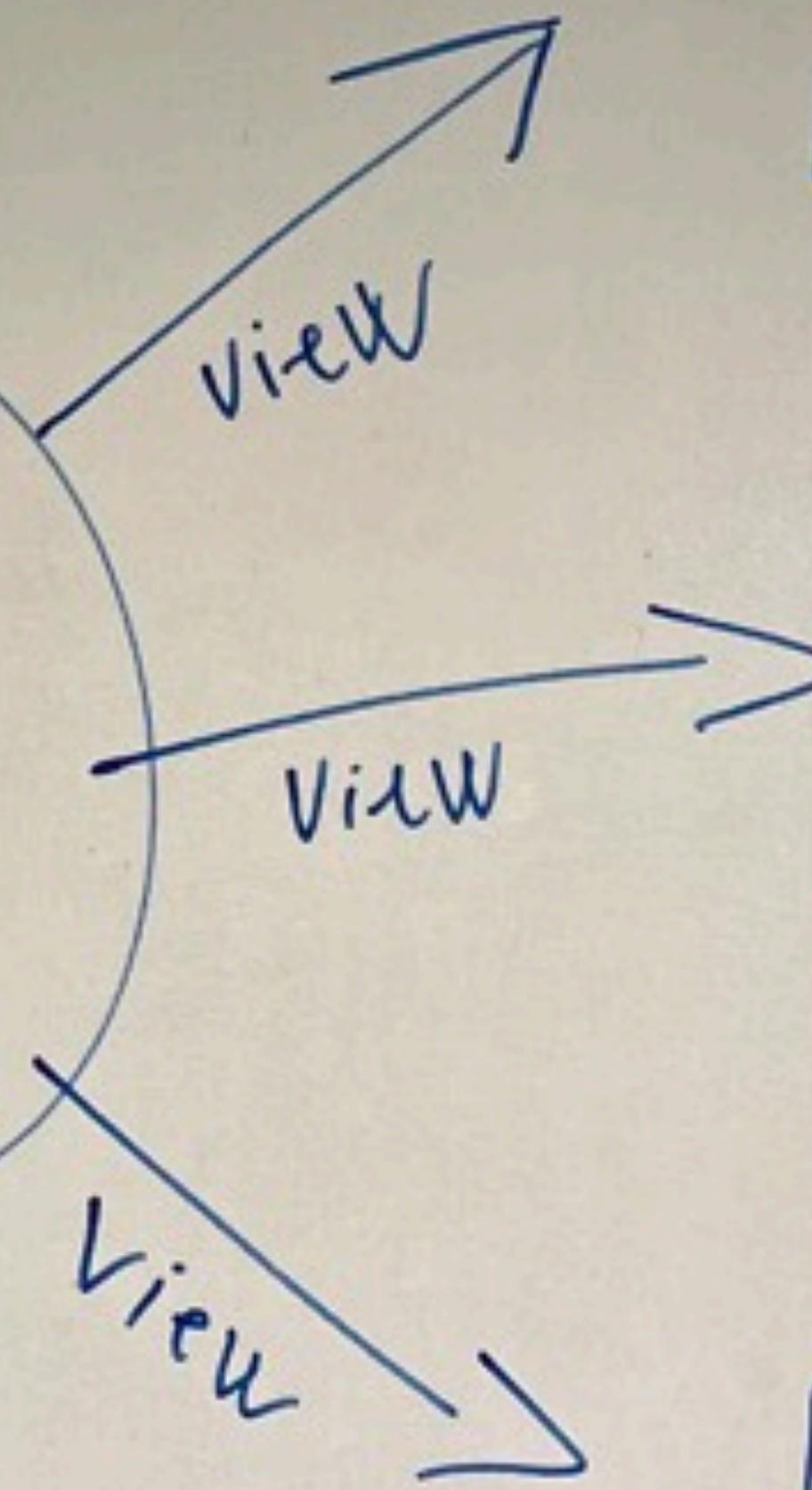
Routes organize how your application handles different requests, making your code structured and maintainable.

[https://medium.com/@fro\\_g/routing-in-javascript-d552ff4d2921](https://medium.com/@fro_g/routing-in-javascript-d552ff4d2921)



/ create  
route

Router  
(router.js)



users page

create page

update page

# Routes in Express.js

In Node.js with Express.js, routes are a fundamental concept that helps you define **how your web application responds to different types of HTTP requests** at different URLs. Think of routes as the paths your application can take based on the URLs that users request.

# Routes in Express.js

- Routes determine how your application responds to specific URLs and HTTP methods.
- Each route combines an HTTP method (like GET, POST) with a URL path.
- When a client request matches a route, a corresponding route handler function is executed to process the request and send a response.

```
import express from "express";
const app = express();

// Define routes using app.METHOD(PATH, HANDLER)
app.get("/posts", (req, res) => {
    // return all posts
});

app.post("/posts", (req, res) => {
    const post = req.body;
    // Create a new post based on the data in req.body
});

app.put("/posts/:id", (req, res) => {
    const postId = req.params.id;
    const post = req.body;
    // Find and update the post with postId and data from req.body
});

app.delete("/posts/:id", (req, res) => {
    const postId = req.params.id;
    // Find and delete the post with postId
});

// ...and so on

app.listen(3000, () => {
    console.log("Server is running on port 3000");
});
```

# Code Every Day

