

Mapping og Databaseklient

med



Sequelize

A screenshot of the PhpStorm IDE interface showing a dark-themed code editor. The title bar says "Arnold Franciscus > main.js > main.js". The left sidebar shows files like "index.php" and "main.js". The main editor area displays the following code:

```
</script>
</head>
<body>
<main>
  <section class="main--section">
    <nav>
      <ul>
        <li id="left--item">AF</li>
        <li class="right--items"><a href="#section--content" class="pink--button scroll" onclick="scrollToSection('content')>Hey, ik ben Arnold!
</a></li>
        <li class="right--items"><a href="#work--section" class="pink--button scroll" onclick="scrollToSection('work')>Ik ben een front-end developer en student applicatieontwikkeling
</a></li>
        <li class="right--items"><a href="#" class="pink--button scroll" onclick="scrollToSection('about')>Over mij
</a></li>
      </ul>
    </nav>
    
    <h1 class="section--header">
      Hey, ik ben Arnold!
    </h1>
    <p class="section--text">
      Ik ben een front-end developer en student applicatieontwikkeling
    </p>
    <a href="#work--section" class="pink--button scroll" onclick="scrollToSection('work')>Work
</a>
    
  </section>
  <section class="work--section" data-aos="fade-up" data-aos-duration="2s">
    <h2>Work</h2>
    <ul>
      <li>Front-end developer</li>
      <li>Student applicatieontwikkeling</li>
    </ul>
  </section>
  <section class="about--section" data-aos="fade-up" data-aos-duration="2s">
    <h2>About me</h2>
    <ul>
      <li>Name: Arnold Franciscus</li>
      <li>Age: 21</li>
      <li>Location: Den Haag, Netherlands</li>
    </ul>
  </section>
</main>
</body>
```

The bottom status bar shows "1: TDD" and "Terminal". The bottom right corner of the screen shows "MacBook Pro".

Formål

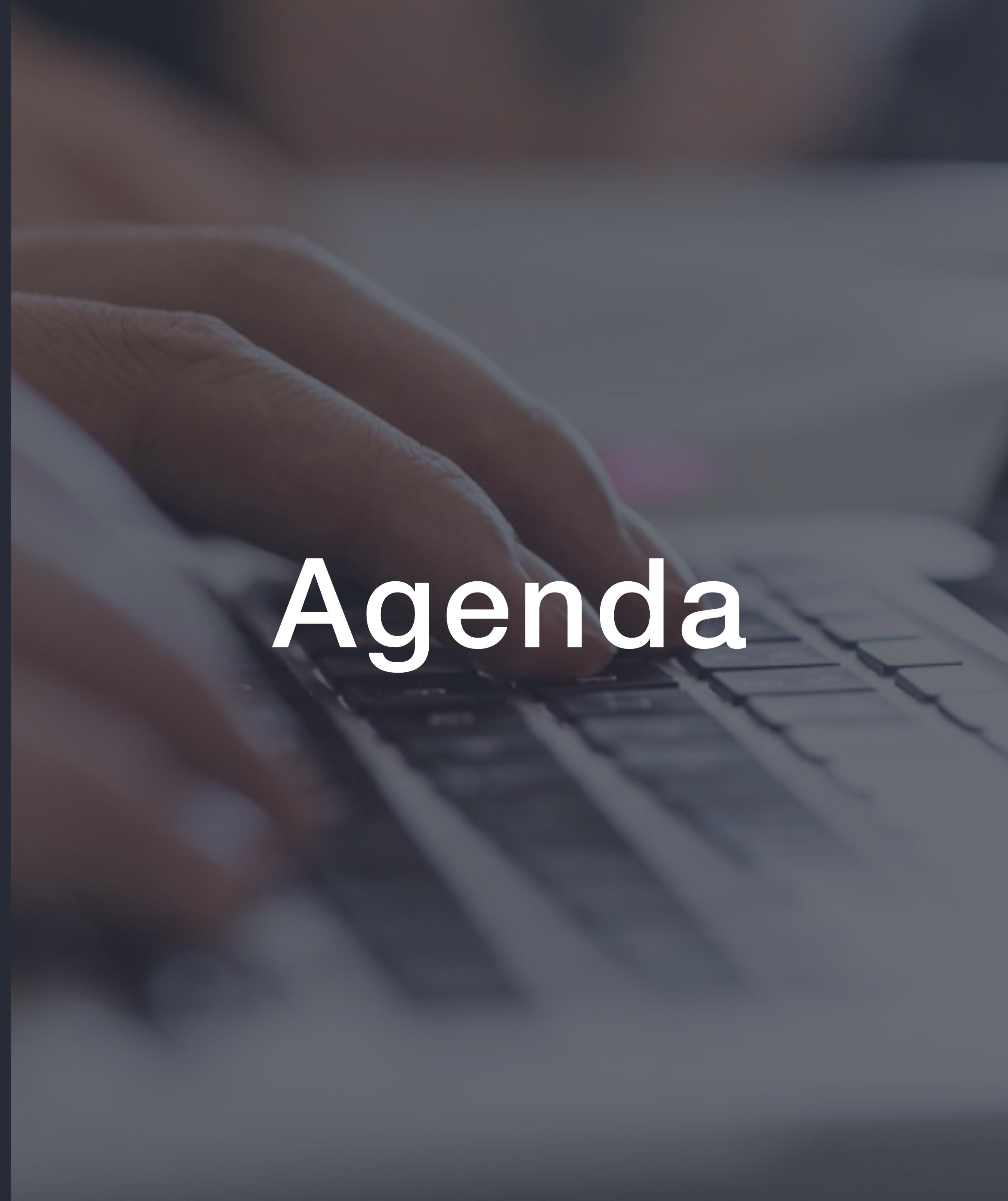
- Forståelse for hvordan man kan implementere en databaseklient med Node.js, Express.js og MySQL2
- Introduktion til Object Relational Mapping (ORM) - hvordan vi kan mappe objekter til database og omvendt
- At kunne anvende Sequelize som ORM for MySQL

1. Intro til dagens undervisning

- Client-server
- Objekter og arrays
- CRUD og REST API

2. Node.js og MySQL

3. Object Relational Mapping (ORM) med Sequelize

A blurred background image showing a close-up of a person's hands typing on a dark-colored computer keyboard. The hands are positioned as if they are writing code.

Agenda

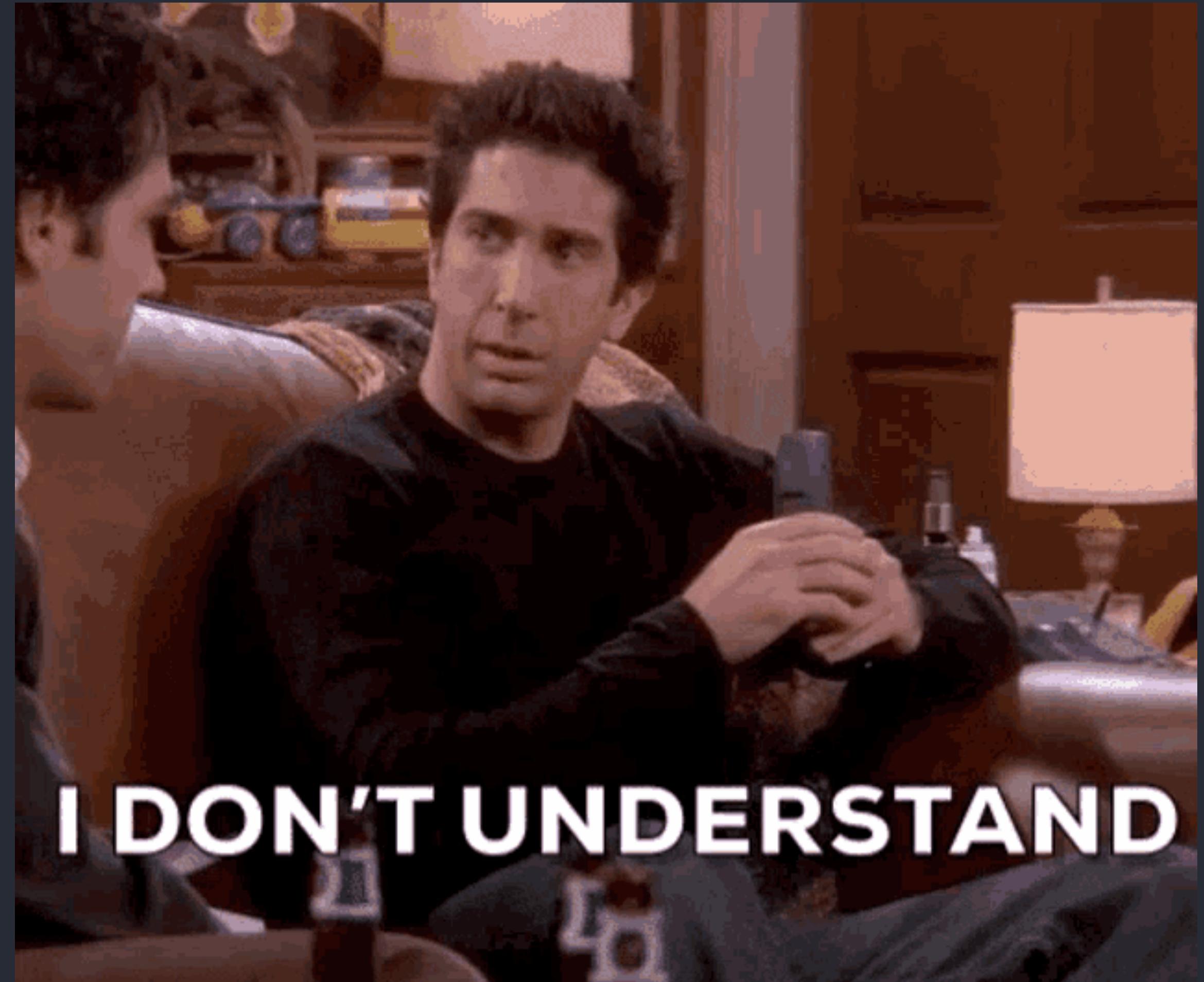


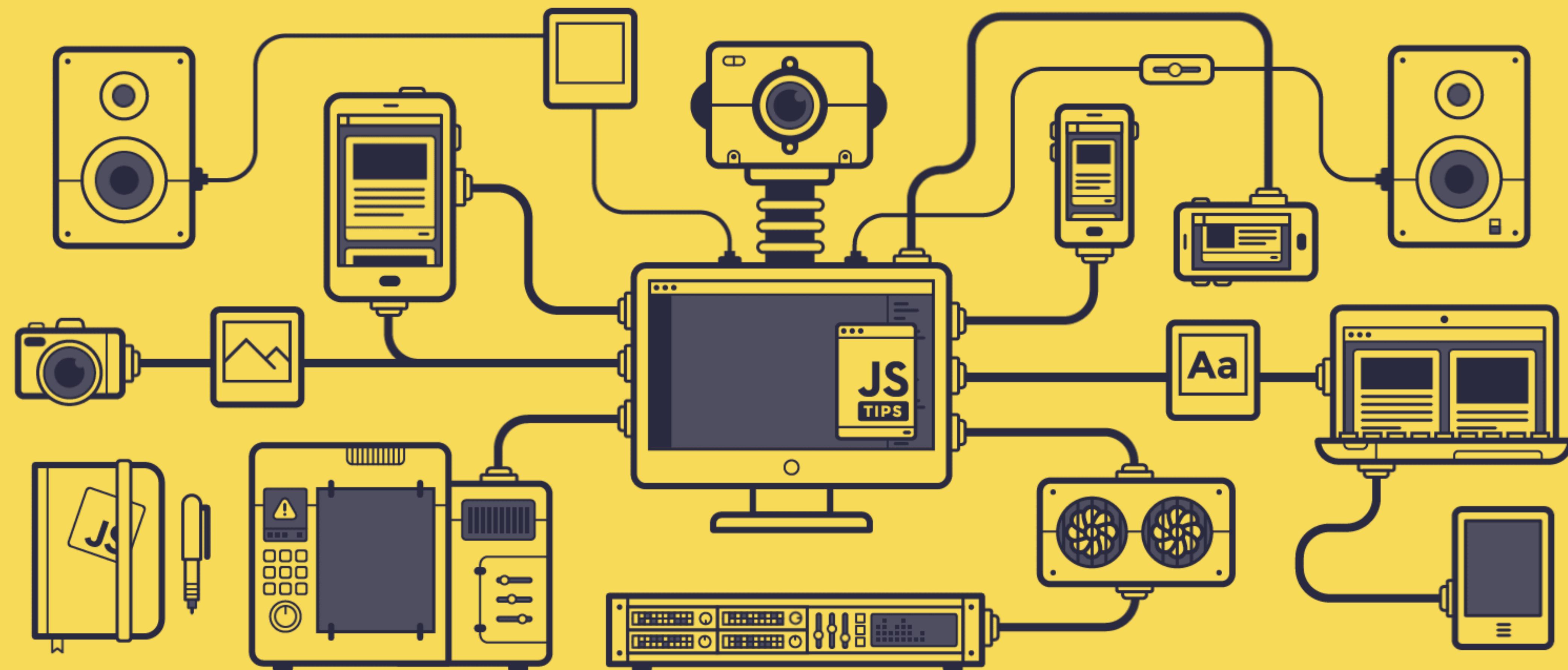
Disclaimer



Express JS

Please,
Do not try to
understand
everything!





Web frameworks and technologies



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

Web App

The screenshot shows a web application titled "CRUD App" running in a browser. The main interface displays a grid of six user profiles, each with a thumbnail, name, title, email, and two buttons: "UPDATE" and "DELETE". Below this grid is a modal window titled "Create a new User" containing four input fields: "Type your name", "Type your title", "Type your mail", and a file input field for "Paste image url".

	Name	Title	Email	Image URL
1	Peter Lind	Senior Lecturer	petl@kea.dk	https://
2	Rasmus Cederdorff	Senior Lecturer	race@dev.dk	https://
3	Lars Bogetoft	Head of Education	larb@eaaa.dk	https://
4	Edith Terte	Lecturer	edan@kea.dk	https://
5	Frederikke Bender	Head of Education	fbe@kea.dk	https://
6	Murat Kilic	Senior Lecturer	mki@eaaa.dk	https://
7	Anne Andersen	Head of Education	anki@mail.dk	https://

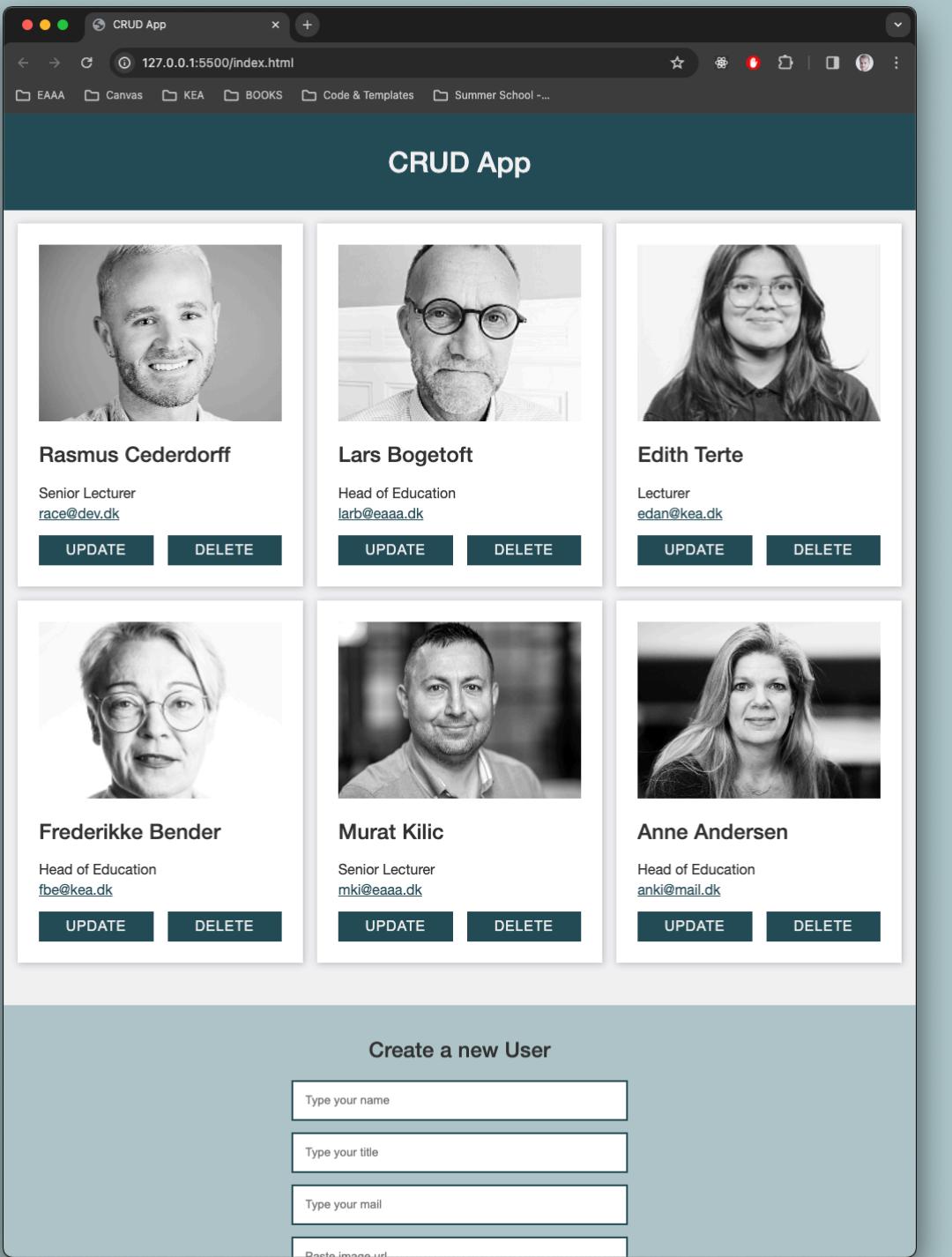
Database

A screenshot of a database table with columns: id, name, mail, title, and image. The data corresponds to the users listed in the web application.

	id	name	mail	title	image
1	Peter Lind	petl@kea.dk	Senior Lecturer	https://	
2	Rasmus Cederdorff	race@dev.dk	Senior Lecturer	https://	
3	Lars Bogetoft	larb@eaaa.dk	Head of Education	https://	
4	Edith Terte	edan@kea.dk	Lecturer	https://	
5	Frederikke Bender	fbe@kea.dk	Head of Education	https://	
6	Murat Kilic	mki@eaaa.dk	Senior Lecturer	https://	
7	Anne Kirketerp	anki@eaaa.dk	Head of Education	https://	

Hvordan “mapper” vi det her?

Frontend



Web App
(Client)



REST API
(Udveksler JSON)

REST API med Node.js

The screenshot shows a code editor with the file "app.js" open. The code defines a REST API for managing users. It includes endpoints for reading all users, reading one user by ID, creating a new user, updating an existing user, and deleting a user. The API interacts with a MySQL database using the "db" object.

```
15 // READ all users
16 app.get("/users", async (request, response) => {
17   const query = "SELECT * FROM users"; // SQL query
18   const [users] = await db.execute(query); // Execute the query
19   response.json(users); // Send the results as JSON
20 });
21
22 // READ one user
23 app.get("/users/:id", async (request, response) => {
24   const id = request.params.id; // grabs the id from the url
25   const query = "SELECT * FROM users WHERE id=?"; // sql query
26   const values = [id]; // values to insert into query
27   const [results] = await db.execute(query, values); // execute query
28   response.json(results[0]); // send response
29 });
30
31 // CREATE user
32 app.post("/users", async (request, response) => {
33   const user = request.body; // grab the user from the request
34   const query = "INSERT INTO users(name, mail, title, image)"
35   const values = [user.name, user.mail, user.title, user.image];
36   const [result] = await db.execute(query, values); // execute query
37   response.json(result); // send response
38 });
39
40 // UPDATE user
41 app.put("/users/:id", async (request, response) => {
42   const id = request.params.id; // grabs the id from the url
43   const user = request.body; // grab the user from the request
44   const query = "UPDATE users SET name=?, mail=?, title=?, image=?";
45   const values = [user.name, user.mail, user.title, user.image];
46   const [result] = await db.execute(query, values); // execute query
47   response.json(result); // send response
48 });
49
50 // DELETE user
```

Backend App
(Server)

Database

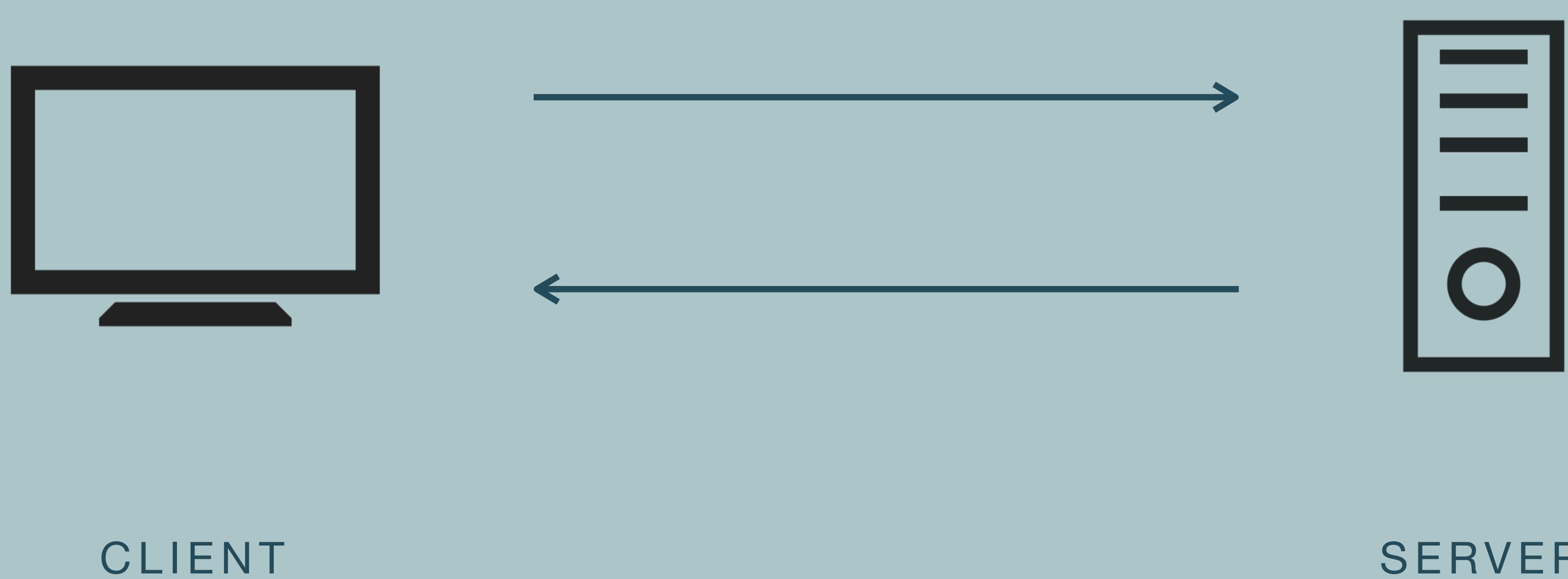
The screenshot shows a MySQL database table named "users". The table has columns for id, name, mail, and title. It contains 7 rows of data corresponding to the users listed in the REST API response.

	id	name	mail	title
1	Peter Lind	petl@kea.dk	Senior Lecturer	
2	Rasmus Cederdorff	race@dev.dk	Senior Lecturer	
3	Lars Bogetoft	larb@eaaa.dk	Head of Education	
4	Edith Terte	edan@kea.dk	Lecturer	
5	Frederikke Bender	fbe@kea.dk	Head of Education	
6	Murat Kilic	mki@eaaa.dk	Senior Lecturer	
7	Anne Kirketerp	anki@eaaa.dk	Head of Education	

SQL

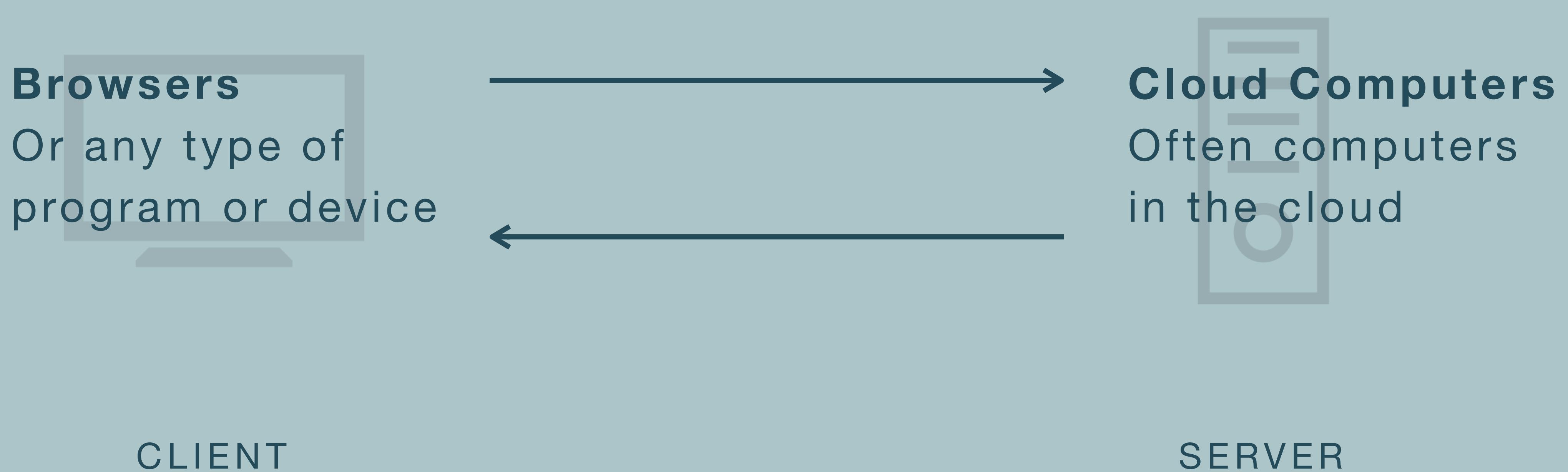
Client-Server Model

Communication between web **clients** and web **servers**.



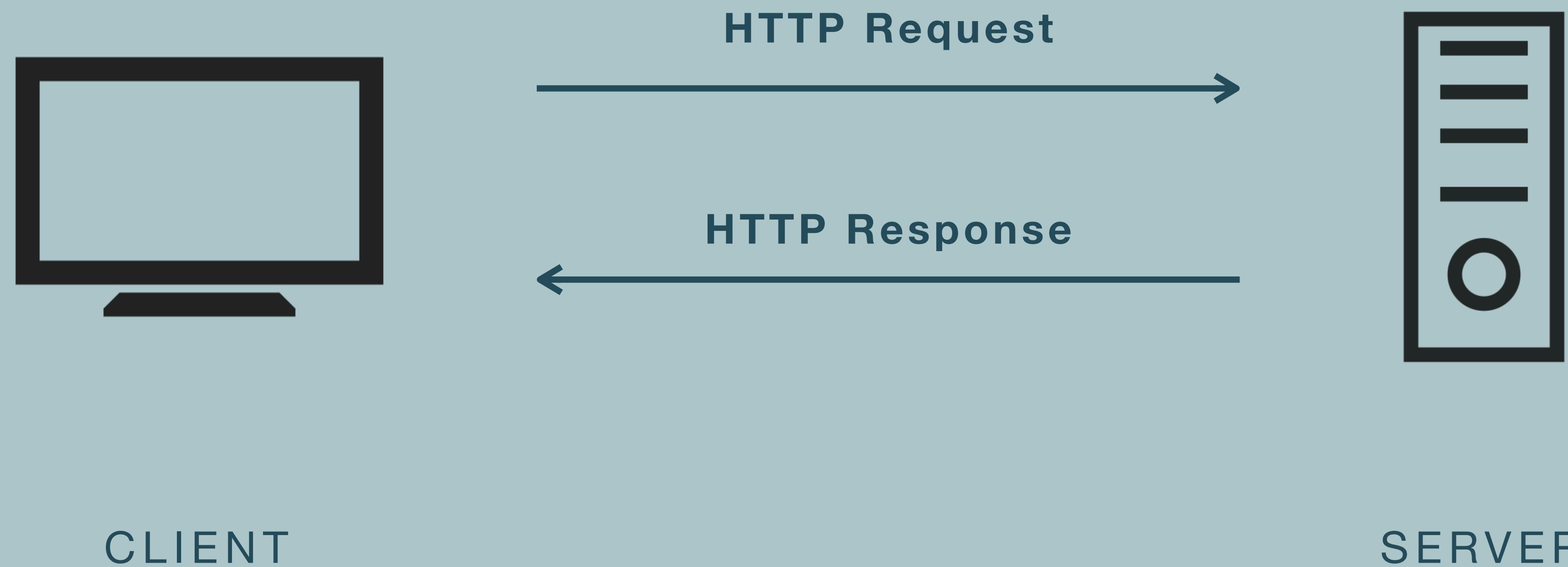
Client-Server Model

Communication between web **clients** and web **servers**.



Client-Server Model

Communication between web **clients** and web **servers**.



Hyper Text Transfer Protocol

- A protocol and standard for fetching data, HTML and other resources (text, images, videos, scripts, JSON).
- The foundation of the web.



What is HTTP

Not Secure | w3schools.com/whatis/whatis_http.asp

HTML CSS JAVASCRIPT SQL PYTHON

HTTP Request / Response

Communication between clients and servers is done by **requests** and **responses**:

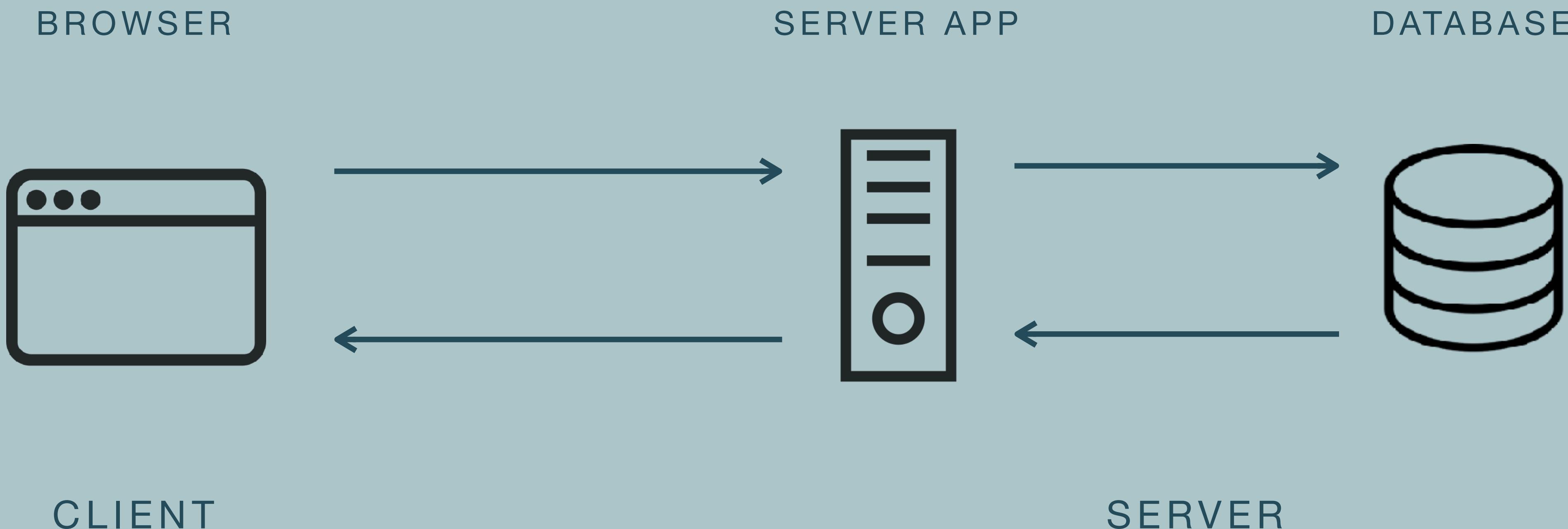
1. A client (a browser) sends an **HTTP request** to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an **HTTP response** (output) to the browser
5. The client (the browser) receives the response

The HTTP Request Circle

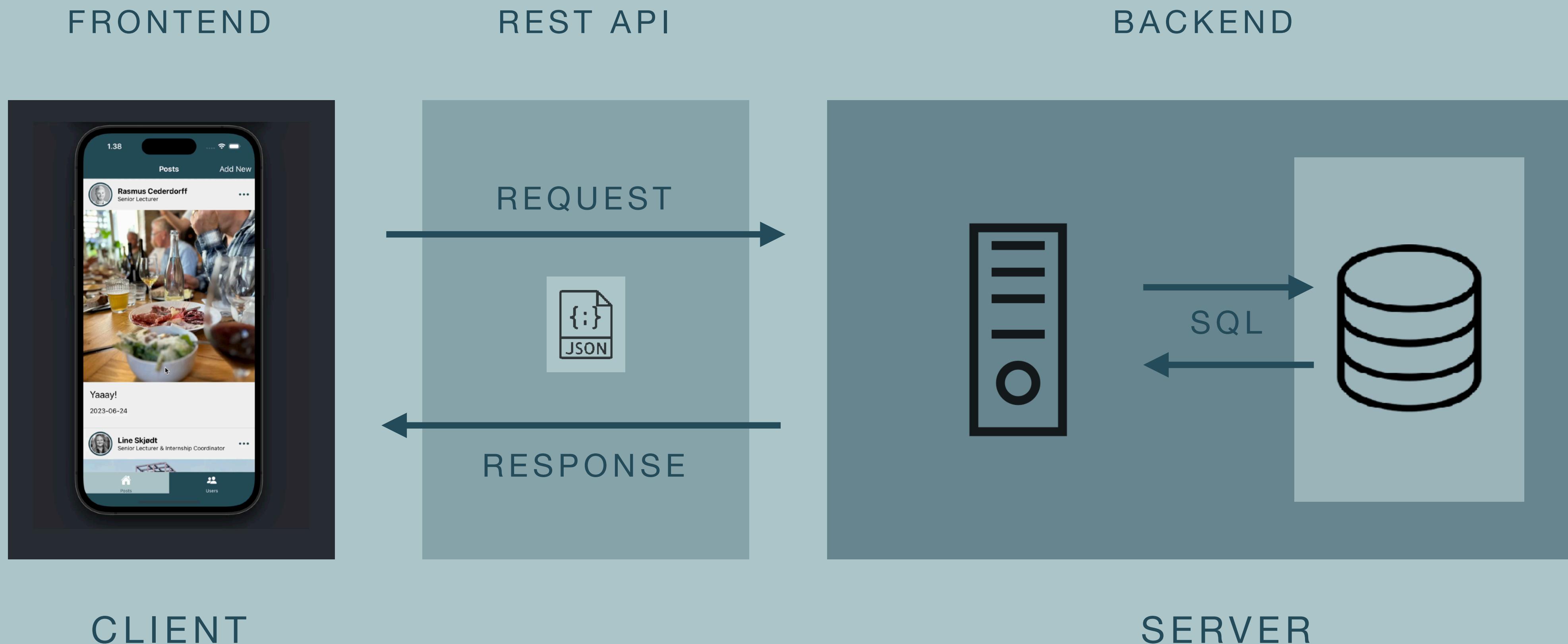
A typical HTTP request / response circle:

1. The browser requests an HTML page. The server returns an HTML file.
2. The browser requests a style sheet. The server returns a CSS file.
3. The browser requests an JPG image. The server returns a JPG file.
4. The browser requests JavaScript code. The server returns a JS file
5. The browser requests data. The server returns data (in XML or JSON).

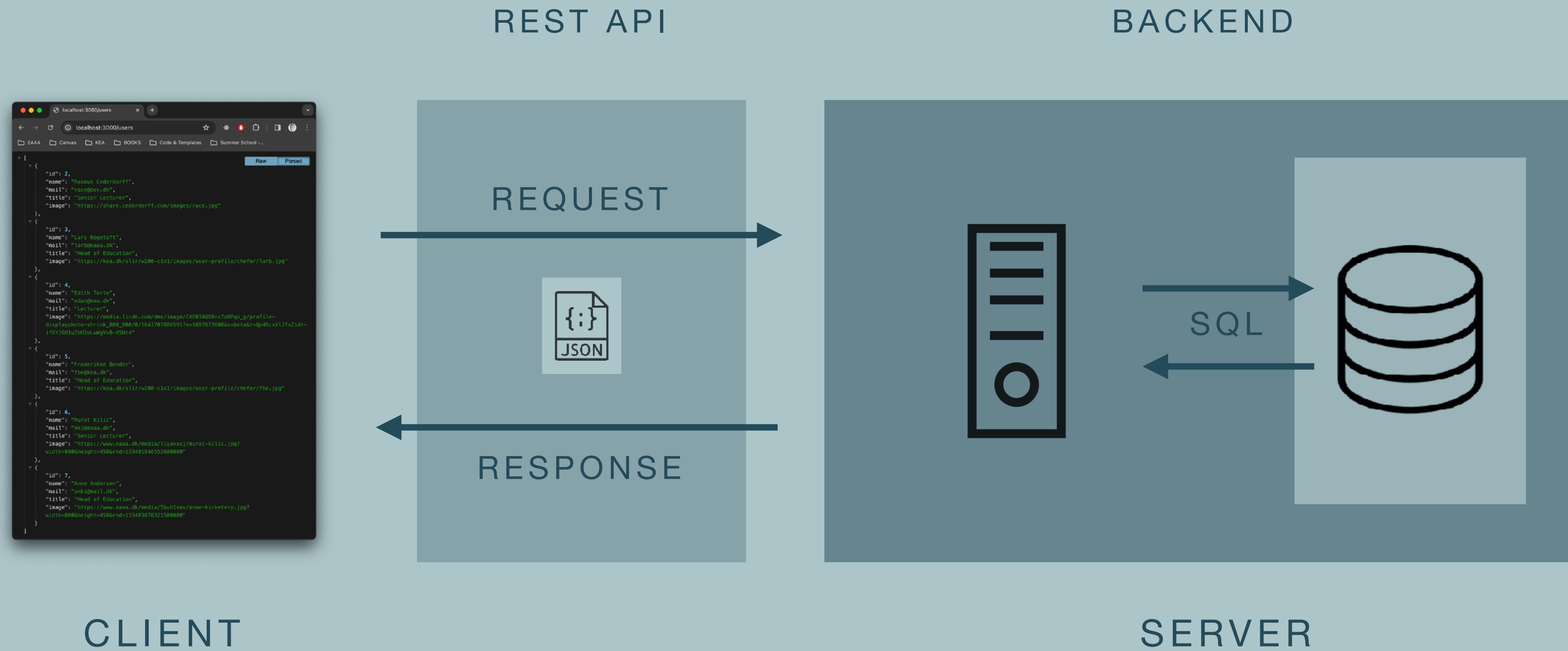
Web Dev Architecture



Web Dev Architecture



Web Dev Architecture



Web Dev Architecture

A screenshot of a web browser window titled "localhost:3000/users". The page displays a JSON array of user objects. Each user object contains fields: id, name, mail, title, and image. The "Parsed" tab is selected, showing the expanded JSON structure. The "Raw" tab shows the raw JSON code.

```
[{"id": 2, "name": "Rasmus Cederdorff", "mail": "race@dev.dk", "title": "Senior Lecturer", "image": "https://share.cederdorff.com/images/race.jpg"}, {"id": 3, "name": "Lars Bogetoft", "mail": "larb@eaaa.dk", "title": "Head of Education", "image": "https://kea.dk/slir/w200-c1x1/images/user-profile/chefer/larb.jpg"}, {"id": 4, "name": "Edith Terte", "mail": "edan@kea.dk", "title": "Lecturer", "image": "https://media.licdn.com/dms/image/C4E03AQE6nx7oUPqo_g/profile-displayphoto-shrink_800_800/0/1643707886591?e=1697673600&v=beta&t=Qp4GcxVLJfsZi4t-if6YJ601u7bH2oLwWgVxB-X5Nt4"}, {"id": 5, "name": "Frederikke Bender", "mail": "fbe@kea.dk", "title": "Head of Education", "image": "https://kea.dk/slir/w200-c1x1/images/user-profile/chefer/fbe.jpg"}, {"id": 6, "name": "Murat Kilic", "mail": "mki@eaaa.dk", "title": "Senior Lecturer", "image": "https://www.eaaa.dk/media/llyavasj/murat-kilic.jpg?width=800&height=450&rnd=133401946552600000"}]
```

REST API

REQUEST



RESPONSE

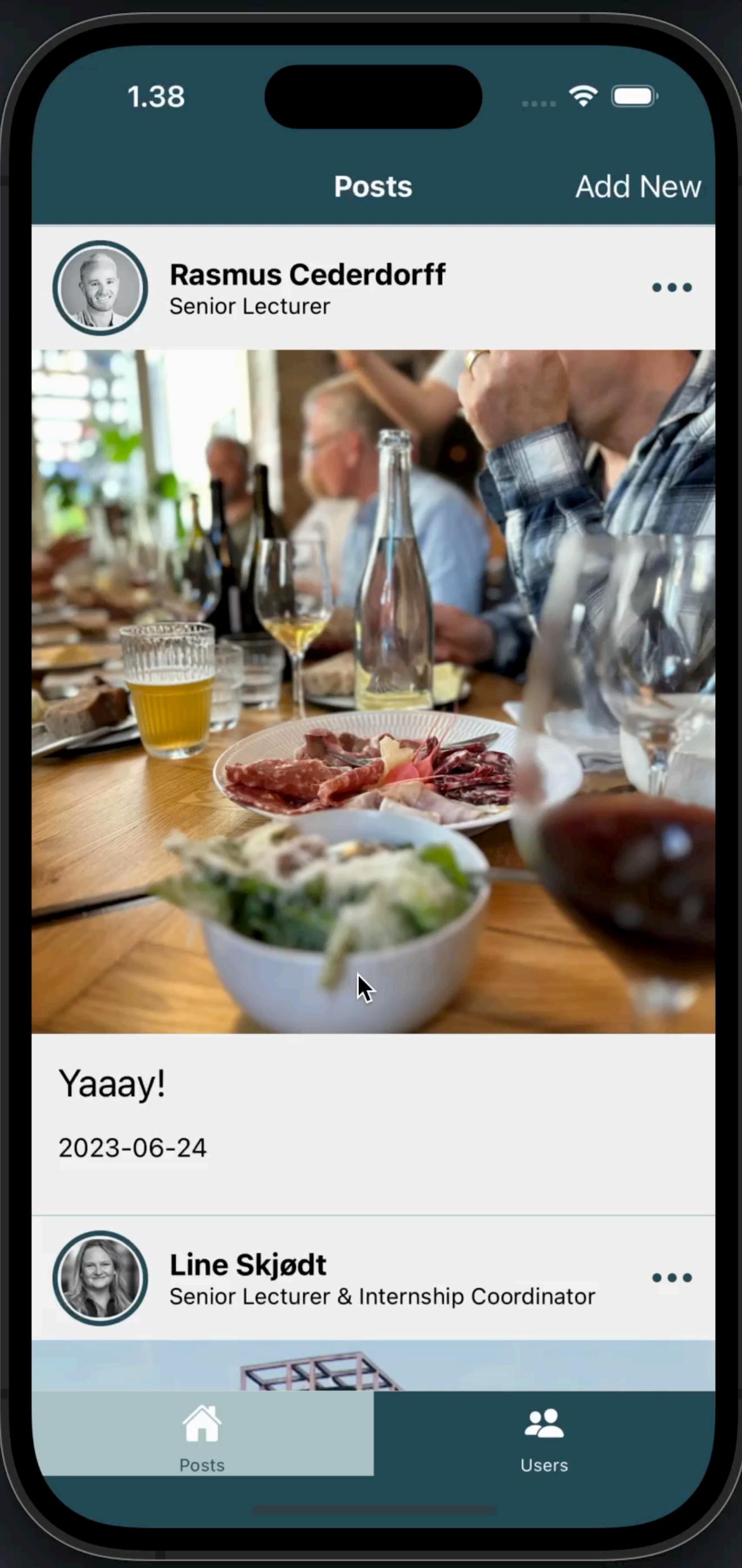
CLIENT

BACKEND

SQL



SERVER

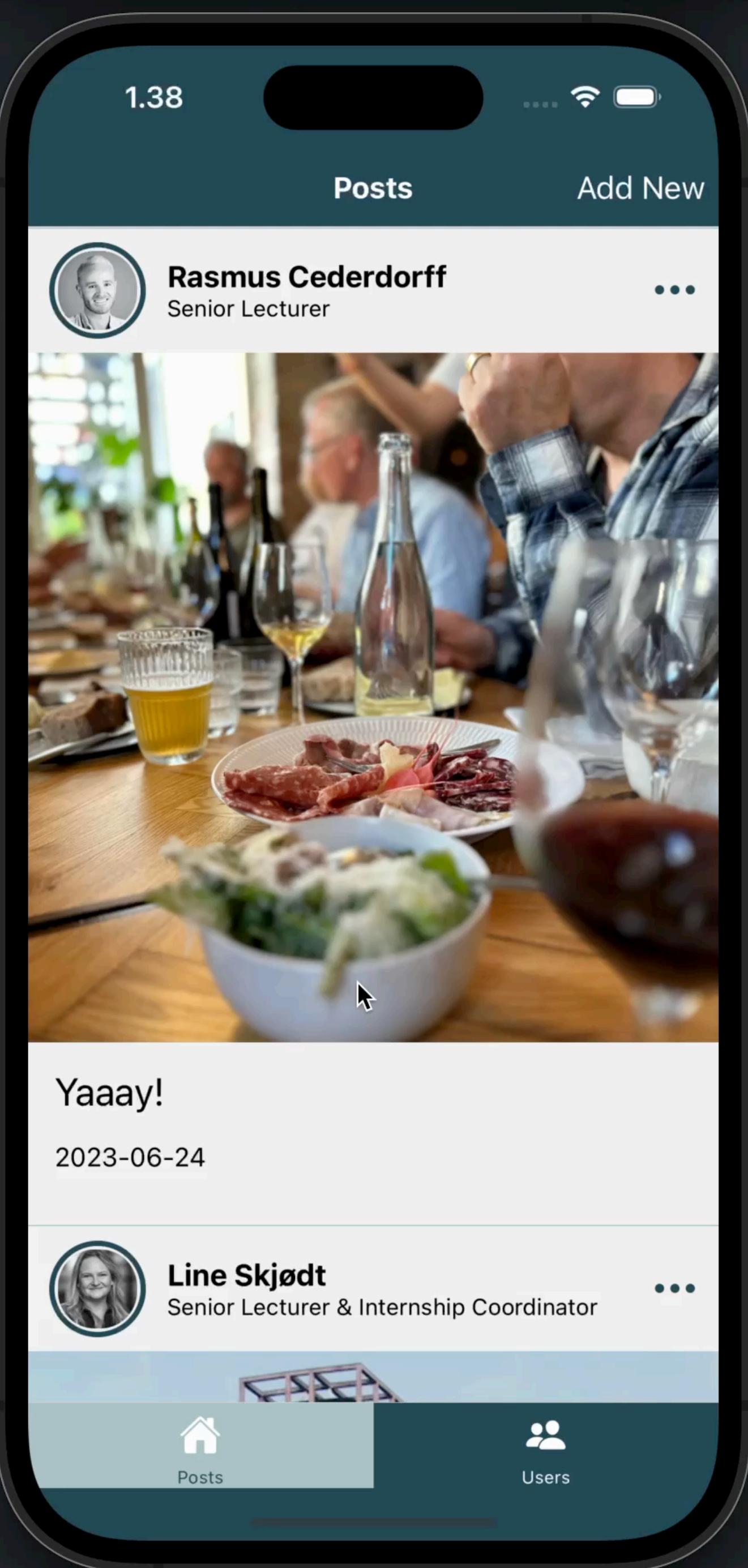


Posts & Users

Post	User	Title	Text Snippet
1	Morten Algy Bonderup Senior Lecturer	Qui est esse	Est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
2	Dan Okkels Brendstrup Lecturer	Consequuntur deleniti eos quia temporibus ab aliquid at	Voluptatem cumque tenetur consequatur expedita ipsum nemo quia explicabo aut eum minima consequatur tempore cumque quae est et et in consequuntur voluptatem voluptates aut
3	Kim Elkjær Marcher-Jepsen Senior Lecturer	At nam consequatur ea labore ea harum	Cupiditate quo est a modi nesciunt soluta ipsa voluptas error itaque dicta in autem qui minus magnam et distinctio eum accusamus ratione error aut
4	Birgitte Kirk Iversen Senior Lecturer		
5	Jes Arbov Lecturer		
6	Maria Louise Bendixen Senior Lecturer		

posts

users



Data from tables

Posts

#	id	caption	image	createdAt
1	1	Beautiful sunset at the beach	https://images.unsplash.com/photo-1566241832378-917a0f30db2c?ixlib=rb-4.0.3&ixid=	2023-09-13 16:51:07
2	2	Exploring the city streets of Aarhus	https://images.unsplash.com/photo-1559070169-a3077159ee16?ixlib=rb-4.0.3&ixid=	2023-09-13 16:51:07
3	3	Delicious food at the restaurant	https://images.unsplash.com/photo-1548940740-204726a19be3?ixlib=rb-4.0.3&ixid=	2023-09-13 16:51:07
4	4	Exploring the city center of Aarhus	https://images.unsplash.com/photo-1612624629424-ddde915d3dc5?ixlib=rb-4.0.3&ixid=	2023-09-13 16:51:07
5	5	A cozy morning with coffee	https://images.unsplash.com/photo-1545319261-f3760f9dd64d?ixlib=rb-4.0.3&ixid=	2023-09-13 16:51:07
6	6	Serenity of the forest	https://images.unsplash.com/photo-1661505216710-32316e7b5bb3?ixlib=rb-4.0.3&ixid=	2023-09-13 16:51:07
7	7	A beautiful morning in Aarhus	https://images.unsplash.com/photo-1573997953524-efed43db70a0?ixlib=rb-4.0.3&ixid=	2023-09-13 16:51:07
8	8	Rainbow reflections of the city of Aarhus	https://images.unsplash.com/photo-1558443336-dbb3de50b8b2?ixlib=rb-4.0.3&ixid=	2023-09-13 16:51:07

Users

#	id	name	mail	title	image
1	1	Maria Louise Bendixen	mlbe@eaaa.dk	Senior Lecturer	https://www.baaa.dk/media/b5ahrllra/maria-loui...
2	2	Rasmus Cederdorff	race@eaaa.dk	Senior Lecturer	https://share.cederdorff.com/images/race.jpg
3	3	Anne Kirketerp	anki@eaaa.dk	Head of Department	https://www.baaa.dk/media/5buh1xeo/anne-kirke...
4	4	Line Skjødt	lskj@eaaa.dk	Senior Lecturer & Internship Coord...	https://www.eaaa.dk/media/14qpfeq4/line-skjod...
5	5	Dan Okkels Brendstrup	dob@eaaa.dk	Lecturer	https://www.eaaa.dk/media/bdojel41/dan-okkels...

BACKEND



Objects

A set of named values

Objects are used to store keyed
collections of various data



Containers for named values
called properties. A property
is a “key: value” pair

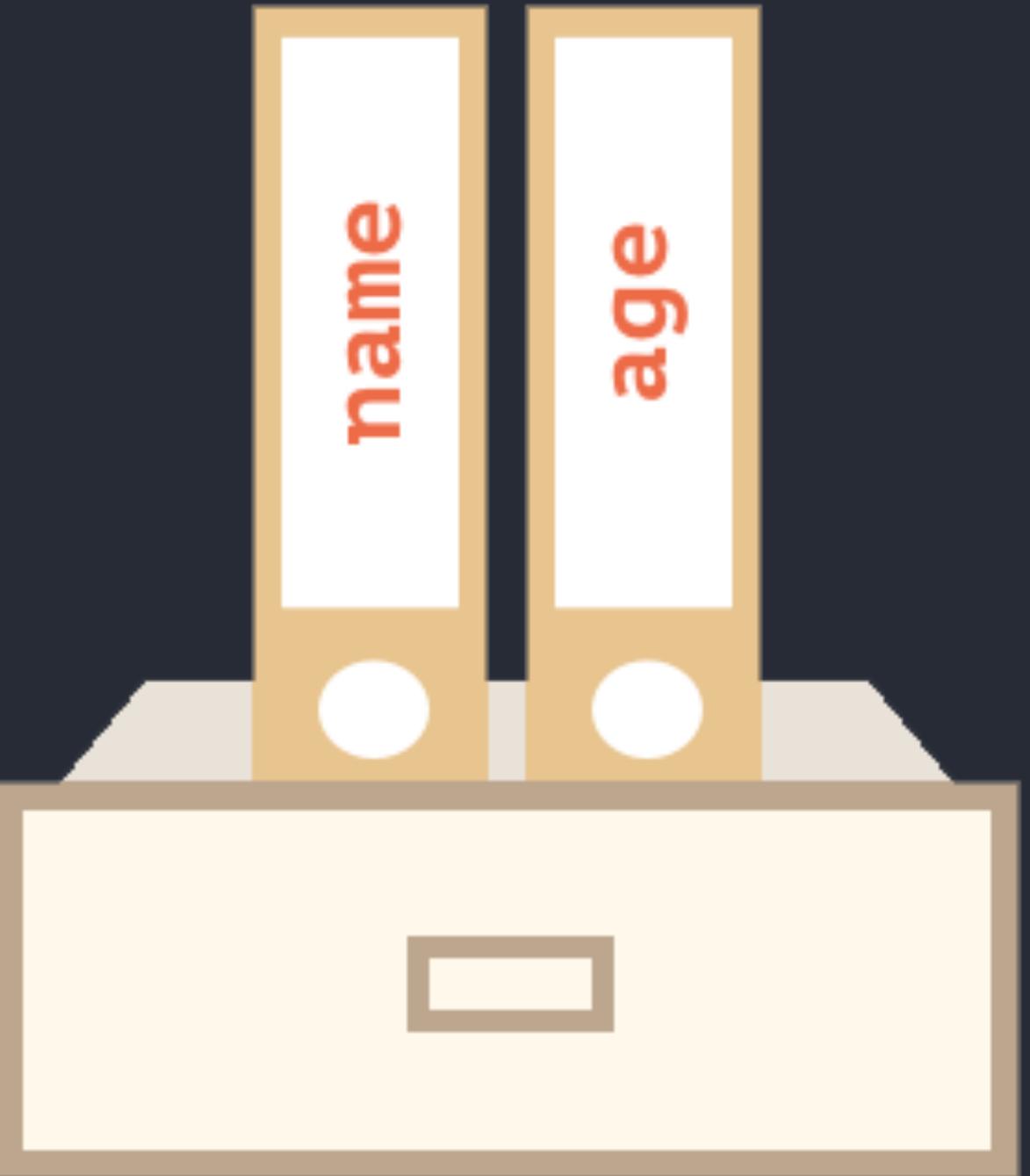
Objects

A set of named values

```
let user = {  
    name: 'Alicia',  
    age: 6  
};
```

```
console.log(user.name +  
    " is " + user.age +  
    " years old.");
```

user

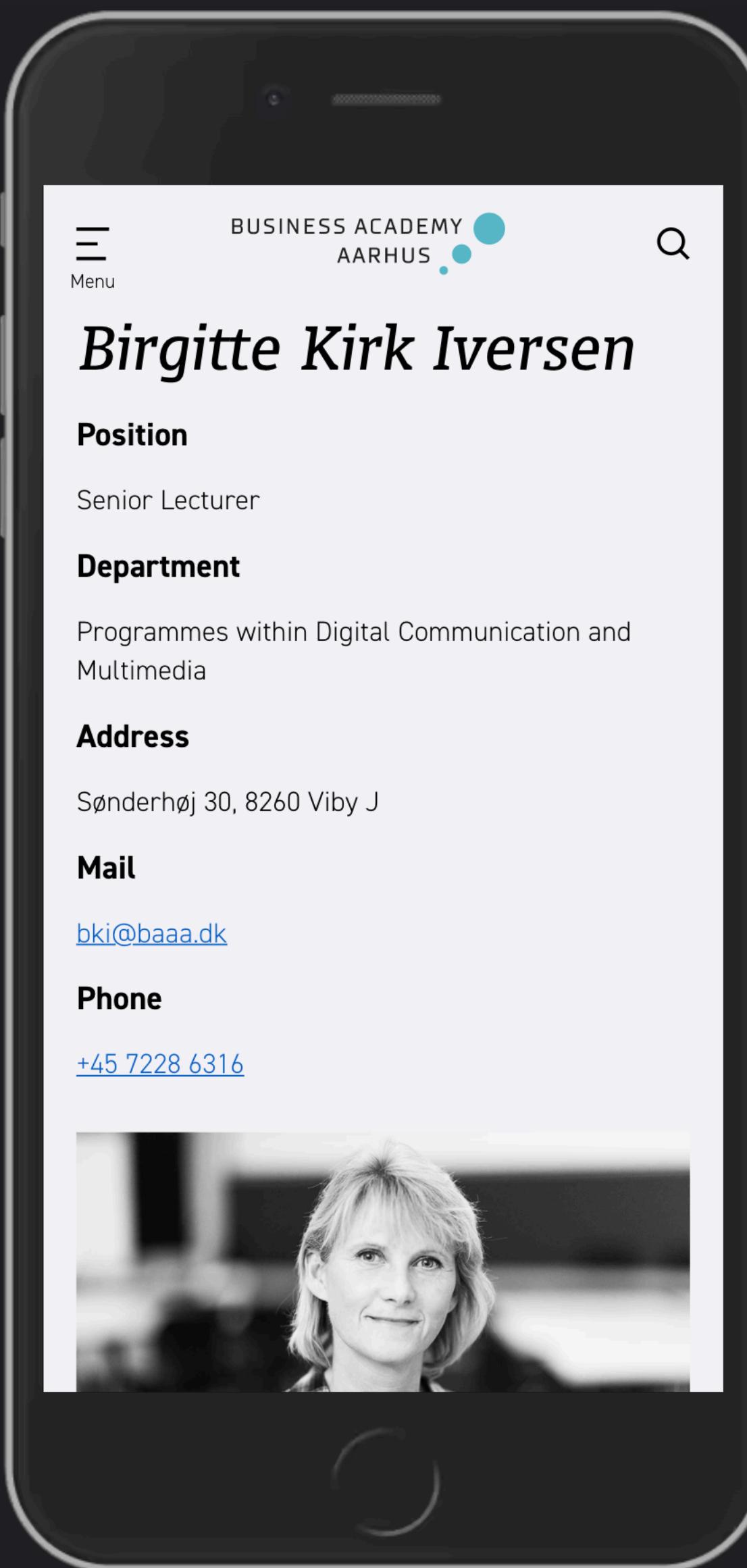


Alicia is 6 years old.

main.js:11

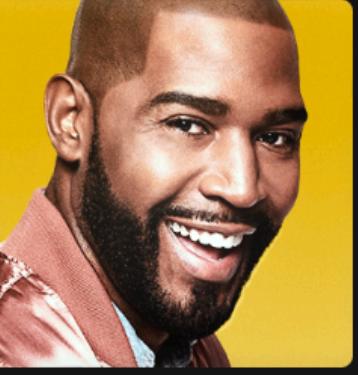
Objects

A set of named values



NETFLIX

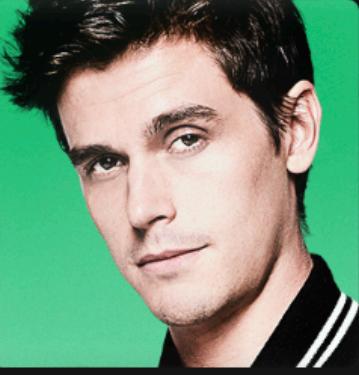
Hvem ser?



Personen der
rent faktisk
betaler for
profilen



Nasser 1



Nasser 2



Nasser 3



Nasser 4 Khader

[Administrer profiler](#)

● ○ ● | □ | < > ⌂ +

netflix.com

NETFLIX Start Serier Film Nyt og populært Min liste

N SERIE

TOO HOT TO HANDLE

TOP 10 Nr. 4 i Danmark i dag

På paradisets kyst mødes de lækkere singler og mingler. Men der er et tvist. For at vinde den attraktive pengepræmie, må de give afkald på at have sex.

Afspil Mere info

13+

Kun på Netflix

TOO HOT TO HANDLE NYE EPISODER

EMILY IN PARIS

QUEER EYE more than a makeover

The Woman in the House Across the Street From the Girl in the Window

BRIDGERTON NYE EPISODER

Se videre med profilen Nasser 1

the office

TIGER KING

Don't Look UP

JEFFREY EPSTEIN: FILTHY RICH

THE MIND explained

Frost II (2019) - IMDb

imdb.com/title/tt4520988/

IMDb Menu All Search IMDb

Frost II

Original title: Frozen II
2019 · 7 · 1h 43m

IMDb RATING YOUR RATING POPULARITY

★ 6.8/10 160K ★ Rate 896 ▲ 102

Cast & crew · User reviews · Trivia · IMDbPro 🔍 All topics | Share

Play trailer 0:16

Animation Adventure Comedy

+ Add to Watchlist

Anna, Elsa, Kristoff, Olaf and Sven leave Arendelle to travel to an ancient, autumn-bound forest of an enchanted land. They set out to find the origin of Elsa's powers in order to save their kingdom.

1.4K User reviews 289 Critic reviews 64 Metascore

Directors Chris Buck · Jennifer Lee

Writers

```
let movie = {  
  title: "Frozen 2",  
  description: "Elsa the Snow Queen has a",  
  trailer: "https://www.youtube.com/embed",  
  length: "1h 43m",  
  year: "2019"  
}
```

Define yourself as an object

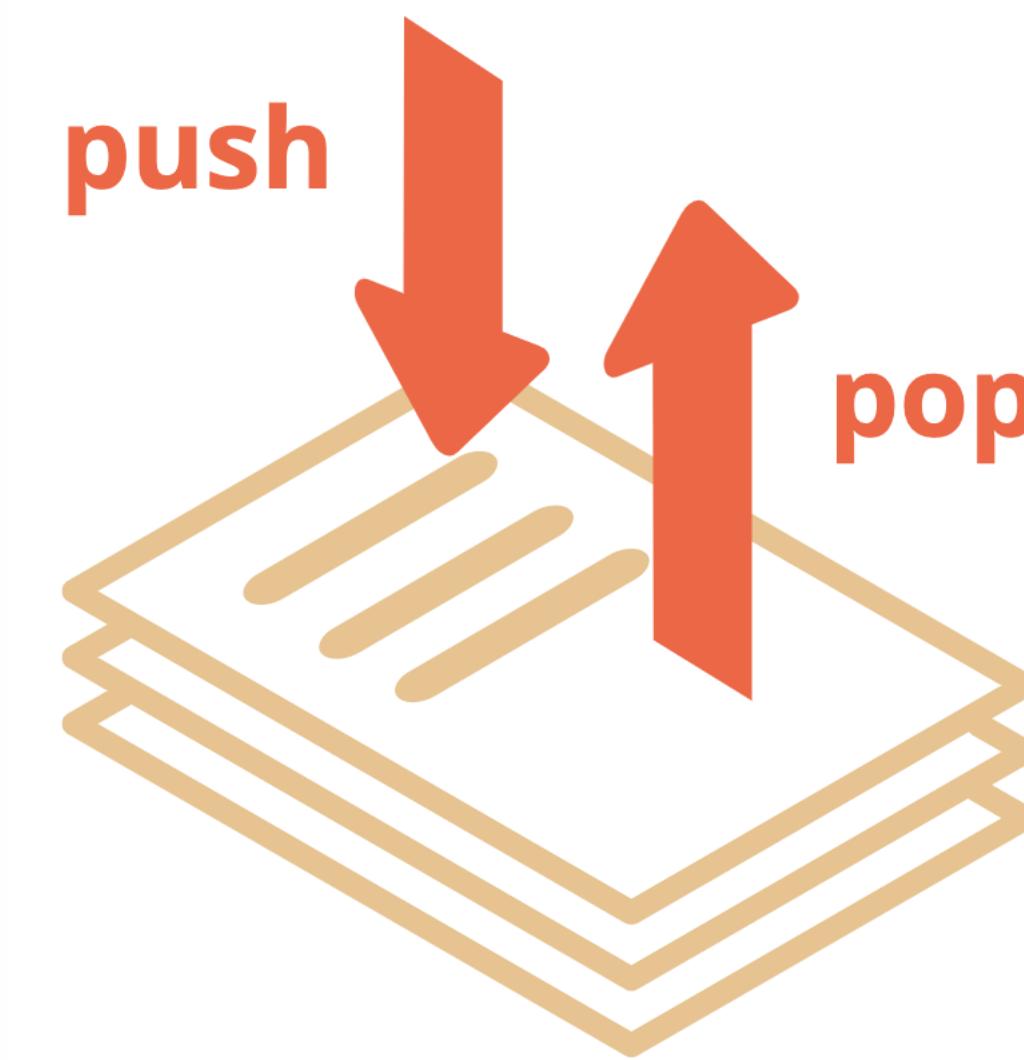
with the following properties

name, age, mail, phone, city, address

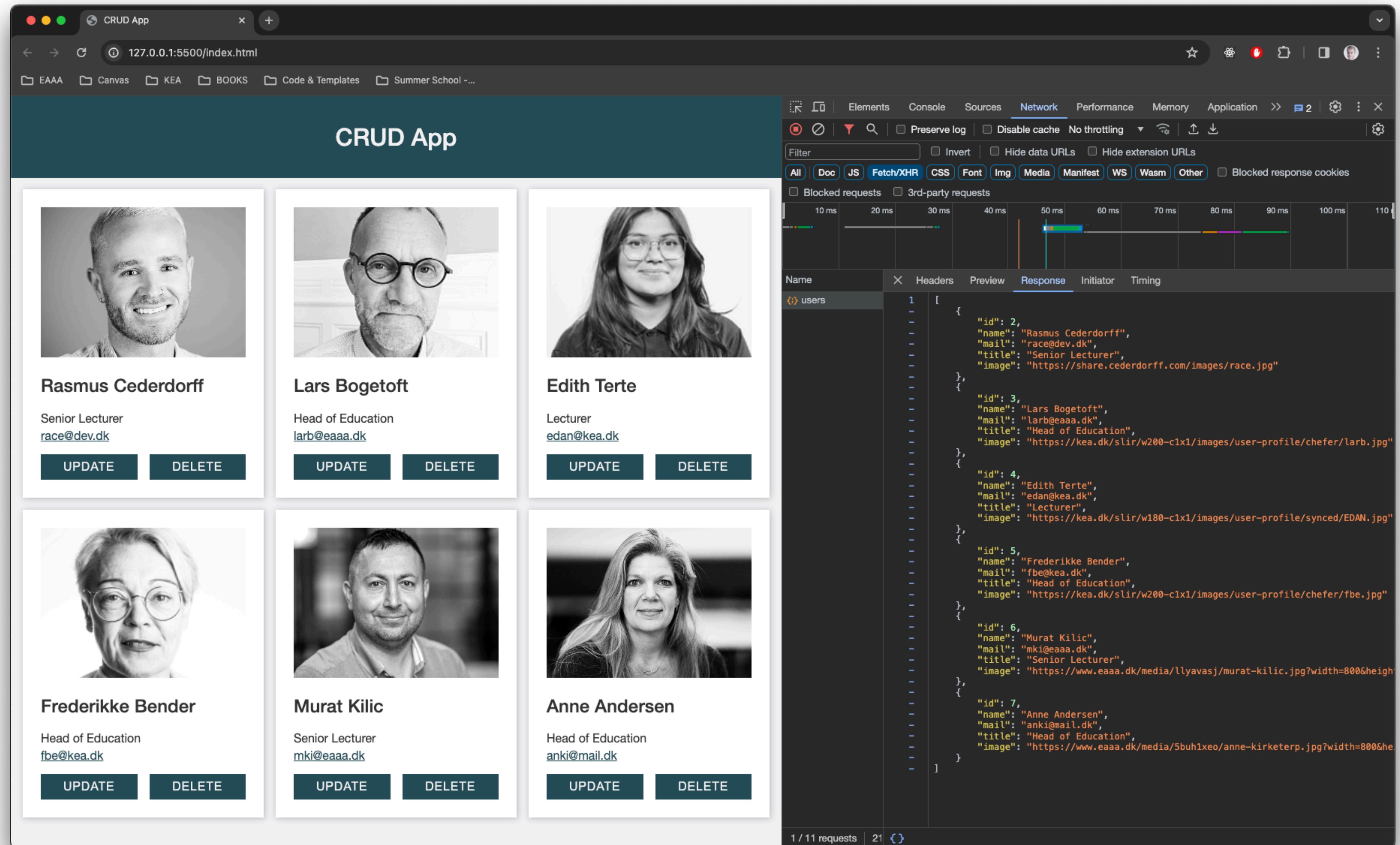
Arrays

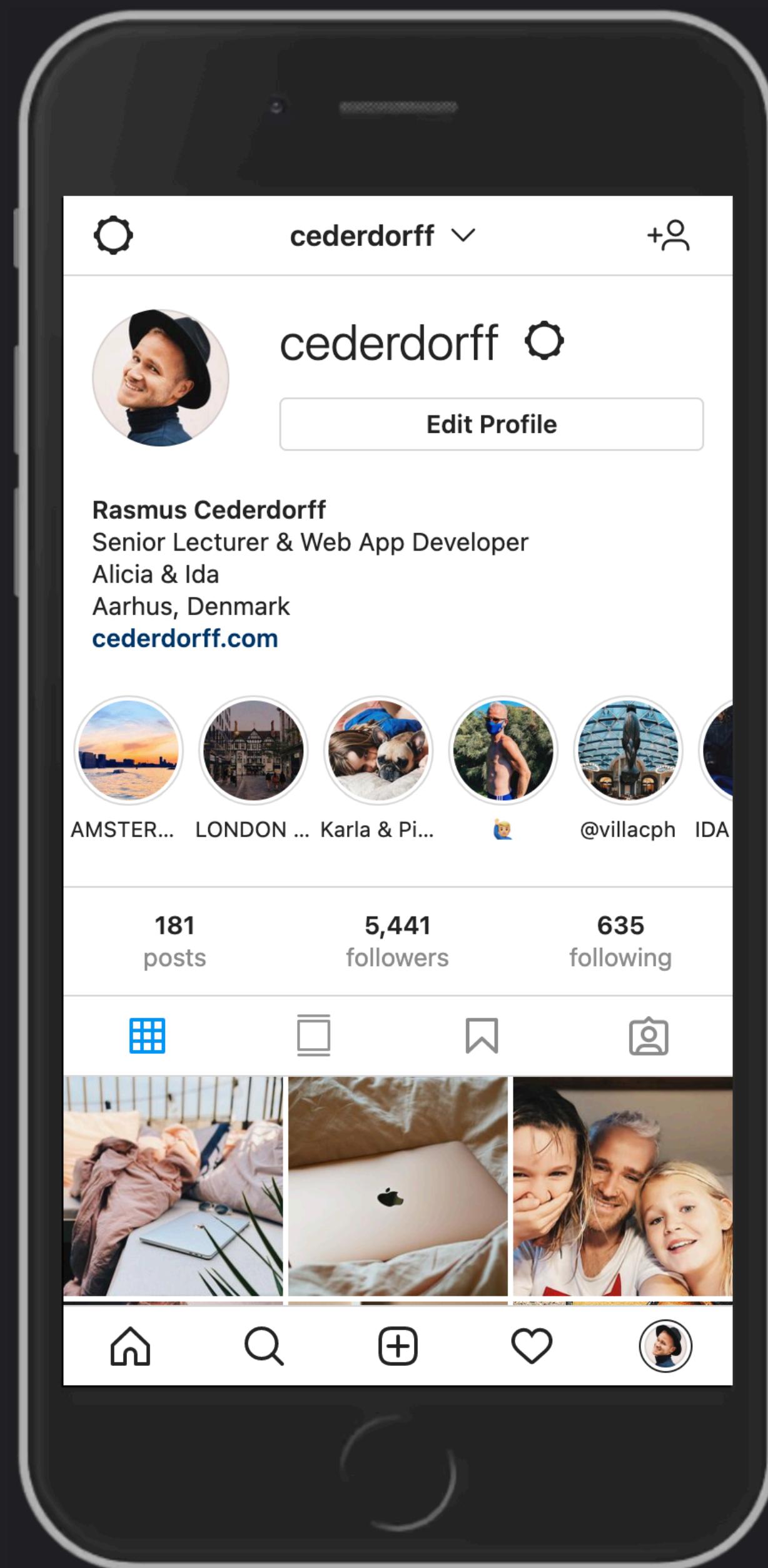
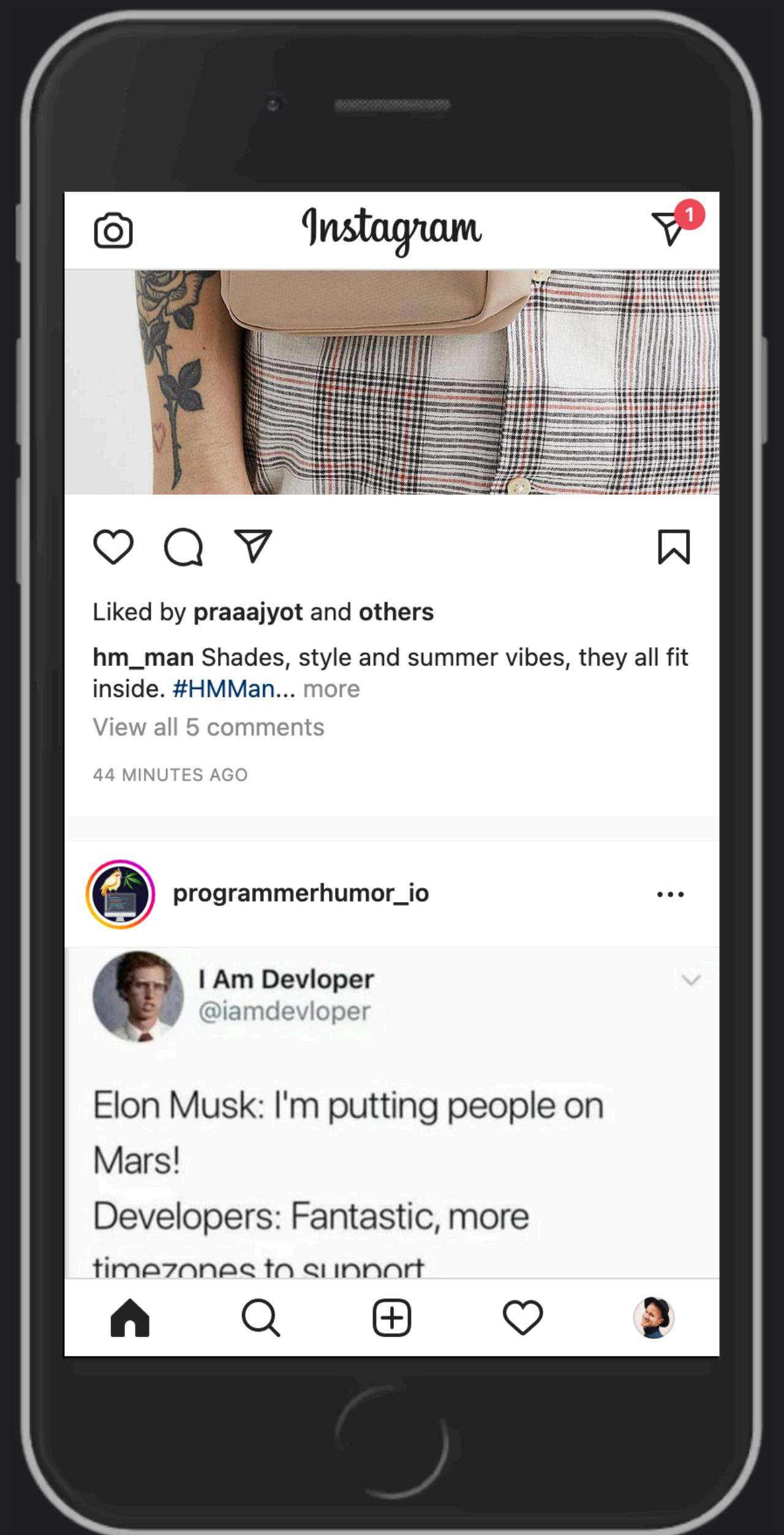
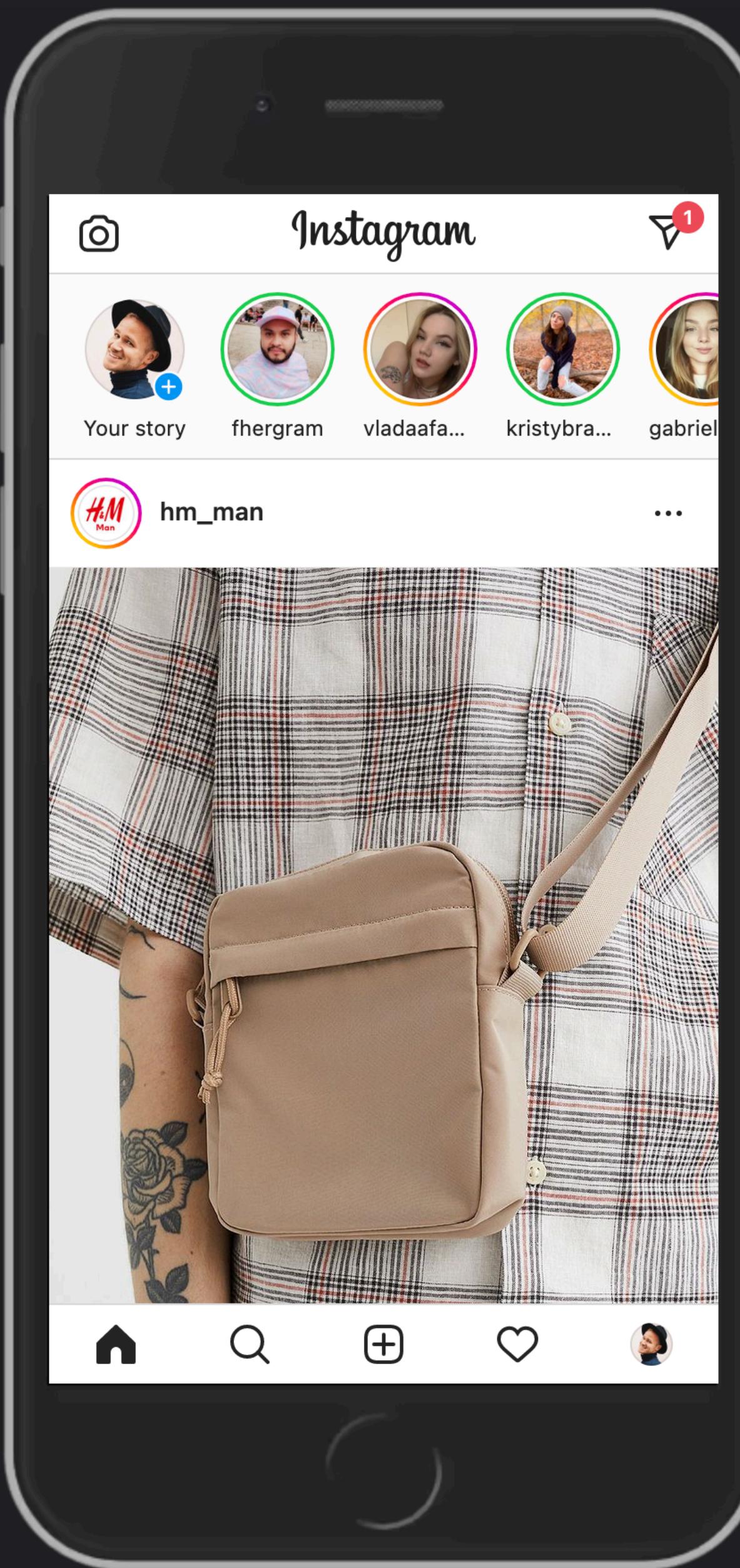
Collections

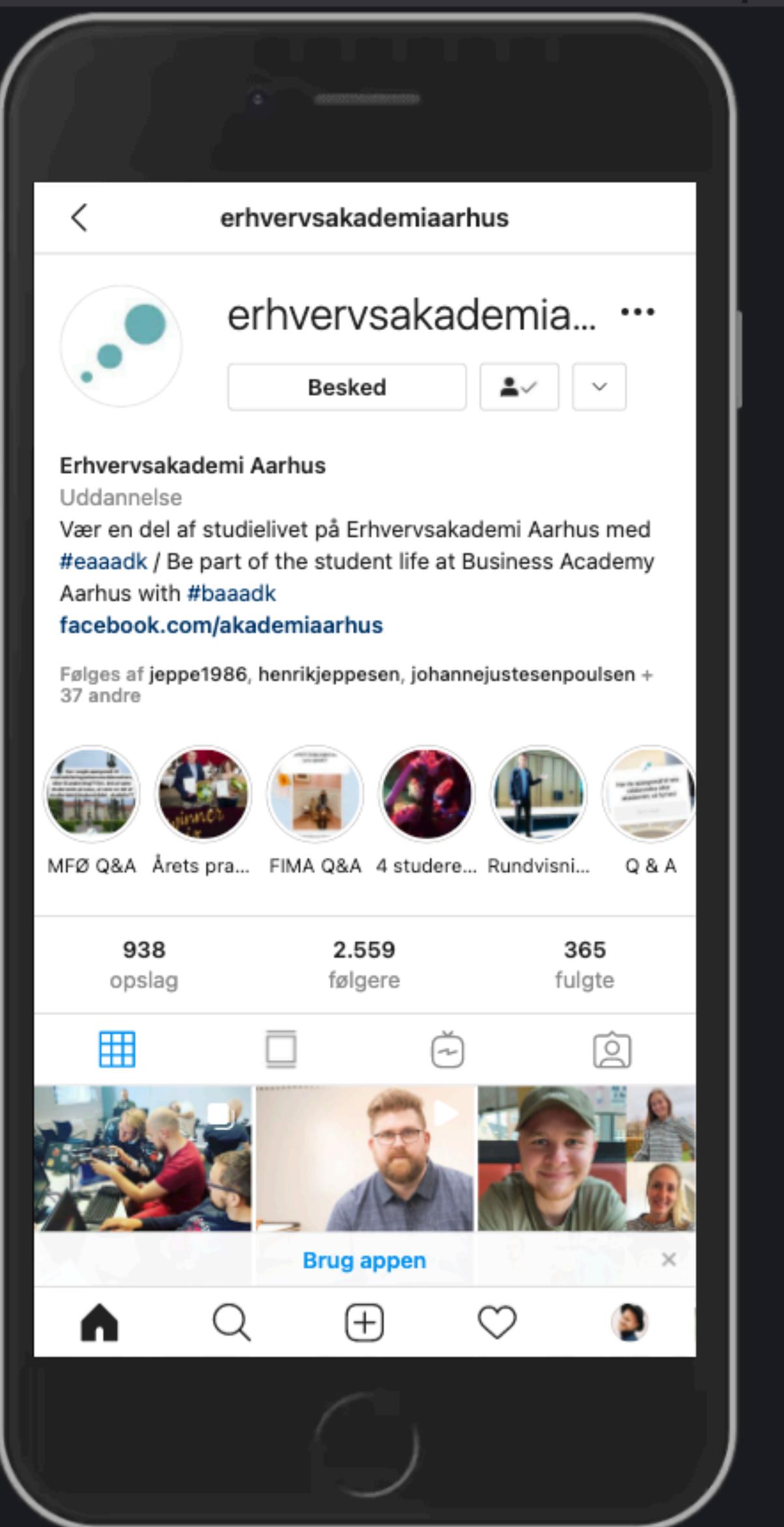
Ordered collection of values or
objects



An array is a way to hold more than one value at a time we have a 1st, a 2nd, a 3rd, a 4th element and so on.







The screenshot shows the Chrome DevTools Network tab. The timeline at the top indicates request times from 5000 ms to 35000 ms. Below the timeline, a list of requests is shown on the left, and a detailed preview of a selected request is on the right.

Selected Request Preview:

- Headers:** Headers for the selected request.
- Preview:** Detailed JSON preview of the response body. The response is a GraphQL query result for a video media type. It includes fields for user information, edge_web_feed_timeline (with edges), node (GraphVideo), and various edge_media_* fields.
- Response:** Raw JSON response body.
- Initiator:** Details about the initiator of the request.
- Timing:** Timing details for the request.
- Cookies:** Cookies associated with the request.

Request List:

- reels_tray/
- ?query_hash=db...
- ?query_hash=6ff...
- badge/
- logging_client_e...
- bz
- falco
- ?__a=1
- logging_client_e...
- falco
- batch_fetch_web/
- ?query_hash=d4...
- ?query_hash=8c...
- ?query_hash=8c...
- logging_client_e...
- falco

At the bottom, it says "16 / 97 requests | 17".

Course roster: WU-E22a - 1. se

<https://eaaa.instructure.com/courses/15482/users>

WU-E22a > People

60 Student view

Home Announcements Modules Assignments Discussions People BigBlueButton Grades Pages Files Syllabus Outcomes Rubrics Quizzes Collaborations Settings

Everyone Groups + Group set

Search people All roles + People

Name	Login ID	SIS ID	Section	Role	Last Activity	Total Activity
Clara Juul Birk	eaaclbi@students.eaaa.dk	WU-E22a - 1.	Student semester	Student	24 Aug at 13:16	01:04:21
Martin Rieper Boesen	eaamrbo@students.eaaa.dk	WU-E22a - 1.	Student semester	Student	24 Aug at 7:54	01:07:06
Dan Okkels Brendstrup	dob@eaaa.dk	WU-E22a - 1.	Teacher semester	Teacher	3 Aug at 8:55	
Rasmus Cederdorff	race@eaaa.dk	WU-E22a - 1.	Teacher semester	Teacher	25 Aug at 9:28	01:19:23
Jeffrey David Serio	jds@eaaa.dk	WU-E22a - 1.	Teacher semester	Teacher	17 Aug at 16:39	
Charlotte Meng Emanuel Dyrholm	eaacmed@students.eaaa.dk	WU-E22a - 1.	Student semester	Student	23 Aug at 16:59	22:24

property value

```
[{"id": "23974", "name": "Clara Juul Birk", "created_at": "2020-08-10T10:45:00+02:00", "email": "eaaclbi@students.eaaa.dk", "sis_user_id": null, "short_name": "Clara Juul Birk", "sortable_name": "Birk, Clara Juul", "integration_id": null, "login_id": "eaaclbi@students.eaaa.dk", "unread_count": 0, "group_categories": []}, {"id": "36267", "name": "Martin Rieper Boesen", "created_at": "2020-08-10T10:45:00+02:00", "email": "eaamrbo@students.eaaa.dk", "sis_user_id": null, "short_name": "Martin Rieper Boesen", "sortable_name": "Boesen, Martin Rieper", "integration_id": null, "login_id": "eaamrbo@students.eaaa.dk", "unread_count": 1, "group_categories": []}, {"id": "29923", "name": "Dan Okkels Brendstrup", "created_at": "2021-07-30T00:46:05+02:00", "email": "dob@eaaa.dk", "sis_user_id": null, "short_name": "Dan Okkels Brendstrup (adjunkt – dob@eaaa.dk)", "sortable_name": "Brendstrup, Dan Okkels", "integration_id": null, "login_id": "dob@eaaa.dk", "unread_count": 0, "group_categories": []}, {"id": "14427", "name": "Rasmus Cederdorff", "created_at": "2020-08-10T10:45:00+02:00", "email": "race@eaaa.dk", "sis_user_id": null, "short_name": "Rasmus Cederdorff", "sortable_name": "Cederdorff, Rasmus", "integration_id": null, "login_id": "race@eaaa.dk", "unread_count": 0, "group_categories": []}, {"id": "41", "name": "Jeffrey David Serio", "created_at": "2020-08-10T10:45:00+02:00", "email": "jds@eaaa.dk", "sis_user_id": null, "short_name": "Jeffrey David Serio", "sortable_name": "Serio, Jeffrey David", "integration_id": null, "login_id": "jds@eaaa.dk", "unread_count": 0, "group_categories": []}, {"id": "24043", "name": "Charlotte Meng Emanuel Dyrholm", "created_at": "2020-08-10T10:45:00+02:00", "email": "eaacmed@students.eaaa.dk", "sis_user_id": null, "short_name": "Charlotte Meng Emanuel Dyrholm", "sortable_name": "Dyrholm, Charlotte Meng Emanuel", "integration_id": null, "login_id": "eaacmed@students.eaaa.dk", "unread_count": 0, "group_categories": []}, {"id": "23978", "name": "Jeppe Frik", "created_at": "2020-08-07T10:45:00+02:00", "email": null, "sis_user_id": null, "short_name": "Jeppe Frik", "sortable_name": "Frik, Jeppe", "integration_id": null, "login_id": null, "unread_count": 0, "group_categories": []}, {"id": "23963", "name": "Daniel Tjerrild Gamborg", "created_at": "2020-08-07T10:45:00+02:00", "email": null, "sis_user_id": null, "short_name": "Daniel Tjerrild Gamborg", "sortable_name": "Gamborg, Daniel Tjerrild", "integration_id": null, "login_id": null, "unread_count": 0, "group_categories": []}, {"id": "23992", "name": "Casper Hedegaard Hansen", "created_at": "2020-08-07T10:45:00+02:00", "email": null, "sis_user_id": null, "short_name": "Casper Hedegaard Hansen", "sortable_name": "Hansen, Casper Hedegaard", "integration_id": null, "login_id": null, "unread_count": 0, "group_categories": []}, {"id": "36266", "name": "Morten Gedsted Hansen", "created_at": "2020-08-07T10:45:00+02:00", "email": null, "sis_user_id": null, "short_name": "Morten Gedsted Hansen", "sortable_name": "Hansen, Morten Gedsted", "integration_id": null, "login_id": null, "unread_count": 0, "group_categories": []}, {"id": "23980", "name": "Anders Husted", "created_at": "2020-08-07T10:45:00+02:00", "email": null, "sis_user_id": null, "short_name": "Anders Husted", "sortable_name": "Husted, Anders", "integration_id": null, "login_id": null, "unread_count": 0, "group_categories": []}, {"id": "23531", "name": "Søren Bo Jørgensen", "created_at": "2020-08-07T10:45:00+02:00", "email": null, "sis_user_id": null, "short_name": "Søren Bo Jørgensen", "sortable_name": "Jørgensen, Søren Bo", "integration_id": null, "login_id": null, "unread_count": 0, "group_categories": []}]
```

6 / 157 requests

Objects? Arrays?

The screenshot shows the homepage of DR Nyheder. At the top, there are navigation links for NYHEDER, DRTV, and DR LYD. Below the navigation, there are six thumbnail cards for TV shows: DR1: Løvens Hule, DR3: Nationens stærkeste, P1: LSD kælderen, DR LYD: Annas Margrethe, DR3: Du fucker med de forkerte, and A Very British Scandal. Under these, a section titled "Seneste nyt" (Latest news) displays three news items: "EU klager over Kinas hårde kurs over for Litauen" (5 MIN. SIDEN), "Børn og skoleelever opfordres stadig til to ugentlige coronatest" (13 MIN. SIDEN), and "England skrætter størstedelen af coronarestriktionerne fra i dag" (25 MIN. SIDEN). The main content area features a large image of medical supplies (a mask, a thermometer, a syringe, and a bottle of hand sanitizer) against a blue background, with the text "15 lande bakker Danmark op: Danske soldater skal blive i Mali" overlaid. At the bottom, a red banner reads "Regeringen har meldt genåbning - men ikke".

The screenshot shows the "ALLE ERHVERVSAKADEMI-UDDANNELSER" (All Business Academy Programs) page. At the top, there are two navigation links: "ALLE UDDANNELSER" and "UDDANNELSER UD FRA INTERESSE". Below this, a grid of 12 program profiles, each featuring a student's face and the program name. The programs are: AUTOMATIONSTEKNOLOG (with BYGGEKOORDINATOR), BYGGETEKNIKER, DATAMATIKER, DESIGNTEKNOLOG (with ENTREPRENØRSKAB OG DESIGN), EL-INSTALLATOR, ENERGITEKNOLOG, IT-TEKNOLOG (with KORT- OG LANDMÅLING), KORT- OG LANDMÅLING, MULTIMEDIEDESIGNER, PRODUKTIONSTEKNOLOG, and VVS-INSTALLATOR.

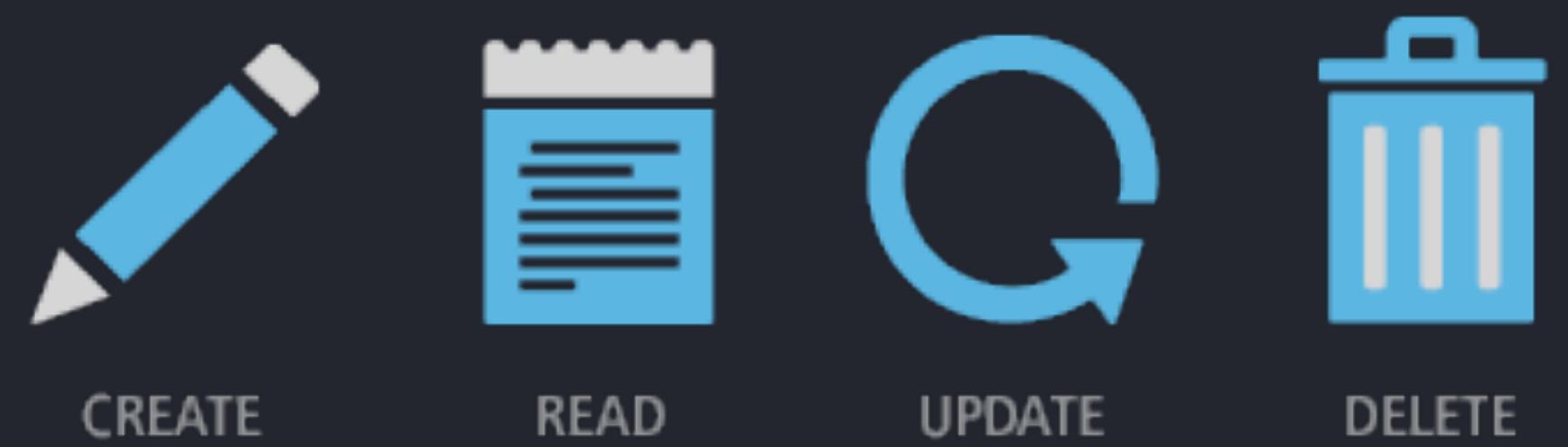
Objects with properties in arrays

The screenshot shows a web browser window for the Business Academy Aarhus website (baaa.dk/programmes/). The page displays various study programs:

- Programmes at Business Academy Aarhus**:
 - Study start in August**:
 - Multimedia Design**: AP degree - 2 years. For those who would like to work with digital communication and interactive design. The programme is the first part of a Bachelor's programme.
 - Digital Concept Development**: Bachelor's top-up degree - 1½ years. Get additional qualifications to develop concepts for digital platforms - at both the strategic and the practical level.
 - Study start in January**:
 - IT Technology**: AP degree (Final intake with study start in January 2022). Would you like to work with computers, server and network technology? The programme is the first part of a Bachelor's programme.
 - Chemical and Biotechnical Technology and Food Technology**: Bachelor's top-up degree (Final intake with study start in January 2022). Be successful in both national and international laboratory environments, and get updated on the
 - Web Development**: Bachelor's top-up degree (Final intake with study start in January 2022). Focus on the development of web technologies within several application fields and distribution platforms.
 - Programmes that no longer accept new applicants**:
 - Chemical and Biotechnical Science**: AP degree (We no longer accept new applicants for this programme).
 - Marketing Management**: AP degree (We no longer accept new applicants for this programme).

A "Chat now" button is located in the bottom right corner.

It's all objects &
arrays!



C R U D

What's CRUD?

- CREATE objects like a post, user, movie, product, etc.
- READ objects like an array (or object) of objects (posts, users, movies, products, etc)
- UPDATE an object, often given by a unique id.
- DELETE an object, often given by a unique id.

What's REST?

GET

POST

PUT

DELETE

- REpresentational State Transfer
- A standard for systems (client & server) to communicate over HTTP in order to retrieve or modify (data) resources.
- Stateless, meaning the two systems doesn't need to know anything about the state.
- The client makes the requests using the 4 basic HTTP verbs to define the operation.

100 *SECONDS OF* 



CRUD vs REST & HTTP Verbs

CREATE -> POST: create a new resource (object)

READ -> GET: retrieve a specific resource or a collection

UPDATE -> PUT: update a specific resource (by id)

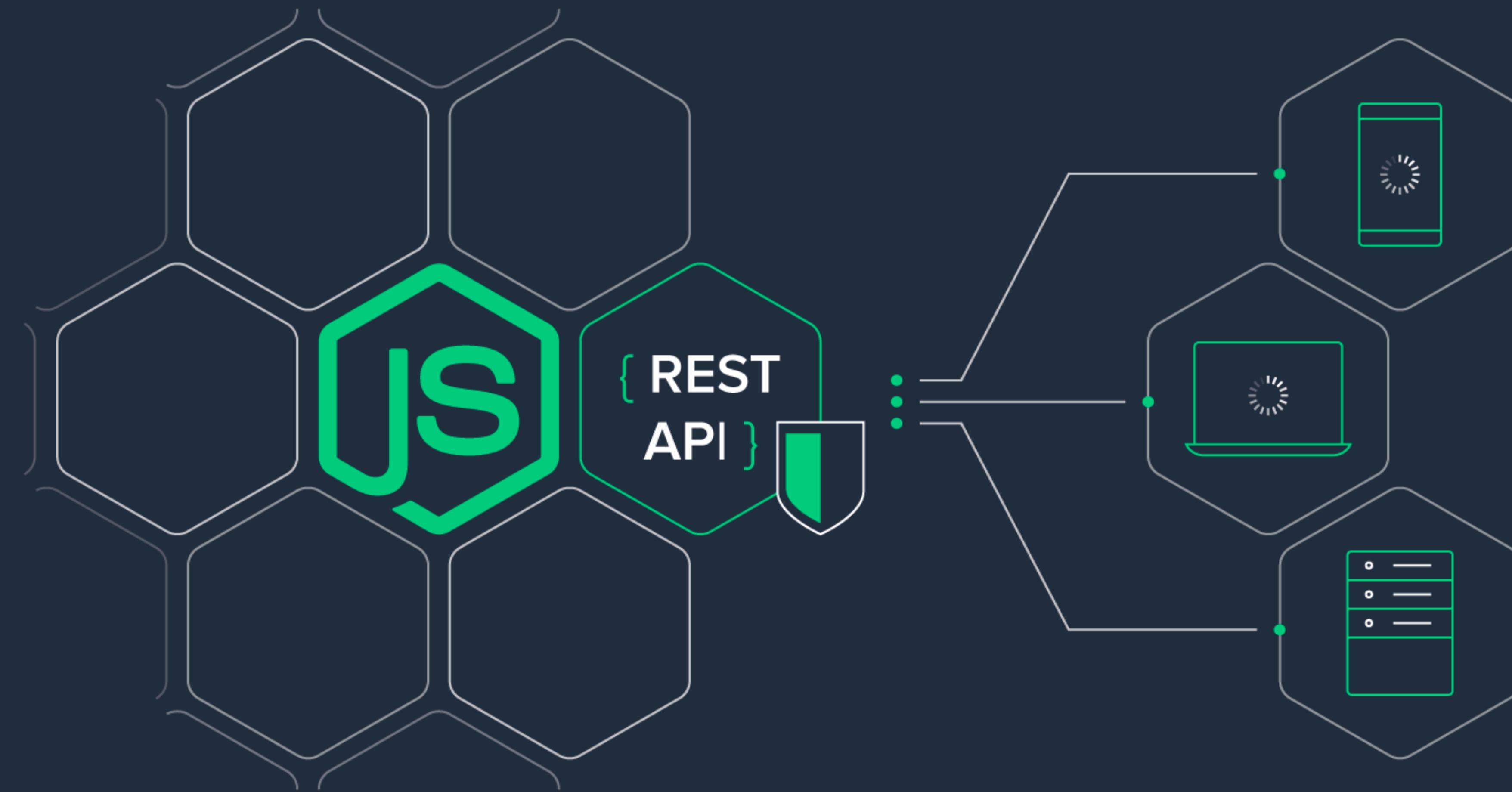
DELETE -> DELETE: remove a specific resource by id



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

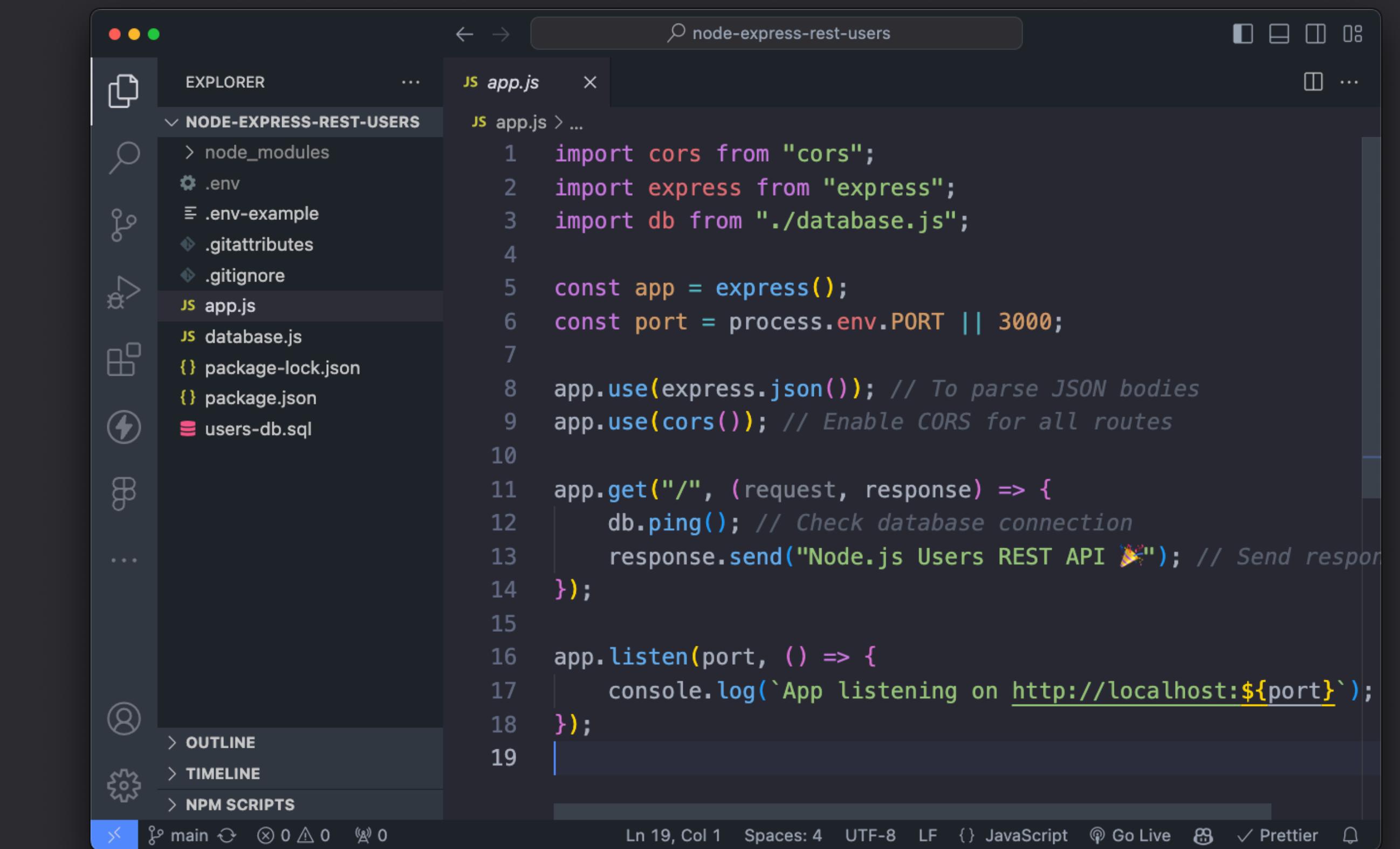


- Imagine Node.js as a powerful engine that lets you run JavaScript code outside of a web browser.
- Instead of just using JavaScript for web pages, Node.js lets you use it on your computer or on a server.
- It's like having a super-fast computer that can understand and do all the smart things you do in a web browser with JavaScript, but now it can do it on your own computer or a server.



Øvelse

Node.js & MySQL



The screenshot shows a dark-themed code editor interface. On the left is the Explorer sidebar with project files: node_modules, .env, .env-example, .gitattributes, .gitignore, app.js (selected), database.js, package-lock.json, package.json, and users-db.sql. The main editor area displays the contents of app.js:

```
JS app.js
1 import cors from "cors";
2 import express from "express";
3 import db from "./database.js";
4
5 const app = express();
6 const port = process.env.PORT || 3000;
7
8 app.use(express.json()); // To parse JSON bodies
9 app.use(cors()); // Enable CORS for all routes
10
11 app.get("/", (request, response) => {
12   db.ping(); // Check database connection
13   response.send("Node.js Users REST API 🎉"); // Send response
14 });
15
16 app.listen(port, () => {
17   console.log(`App listening on http://localhost:${port}`);
18 });
19
```

At the bottom, status bar items include: Ln 19, Col 1, Spaces: 4, UTF-8, LF, {}, JavaScript, Go Live, Prettier, and a file icon.

- **JavaScript Beyond Browser**

Instead of being limited to doing things only on the internet, Node.js lets you use JavaScript to build programs and applications on your computer or server. You get to use the same language you know from web pages.

- **Fast and Efficient**

Node.js is built to be really fast and efficient. It can handle many things at once without getting slow. This is really useful when you're working with apps that need to do lots of things at the same time, like chat apps or real-time updates.

- **Don't Wait for Things**

Usually, when a computer does something, it has to wait until that task is finished before moving on to the next one. Node.js does things differently. It can start a task and then keep working on other things while it waits for the first task to finish.

- **Customize with Modules**

Think of modules as little building blocks of code. Node.js gives you lots of ready-made code that you can use in your programs. You don't have to invent everything from scratch. It makes it easy to add new features to your projects.

- **Used for Many Things**

People use Node.js to build all sorts of things, from simple web servers to complex applications. It's popular in modern web development because it allows fast and efficient communication between clients and servers.

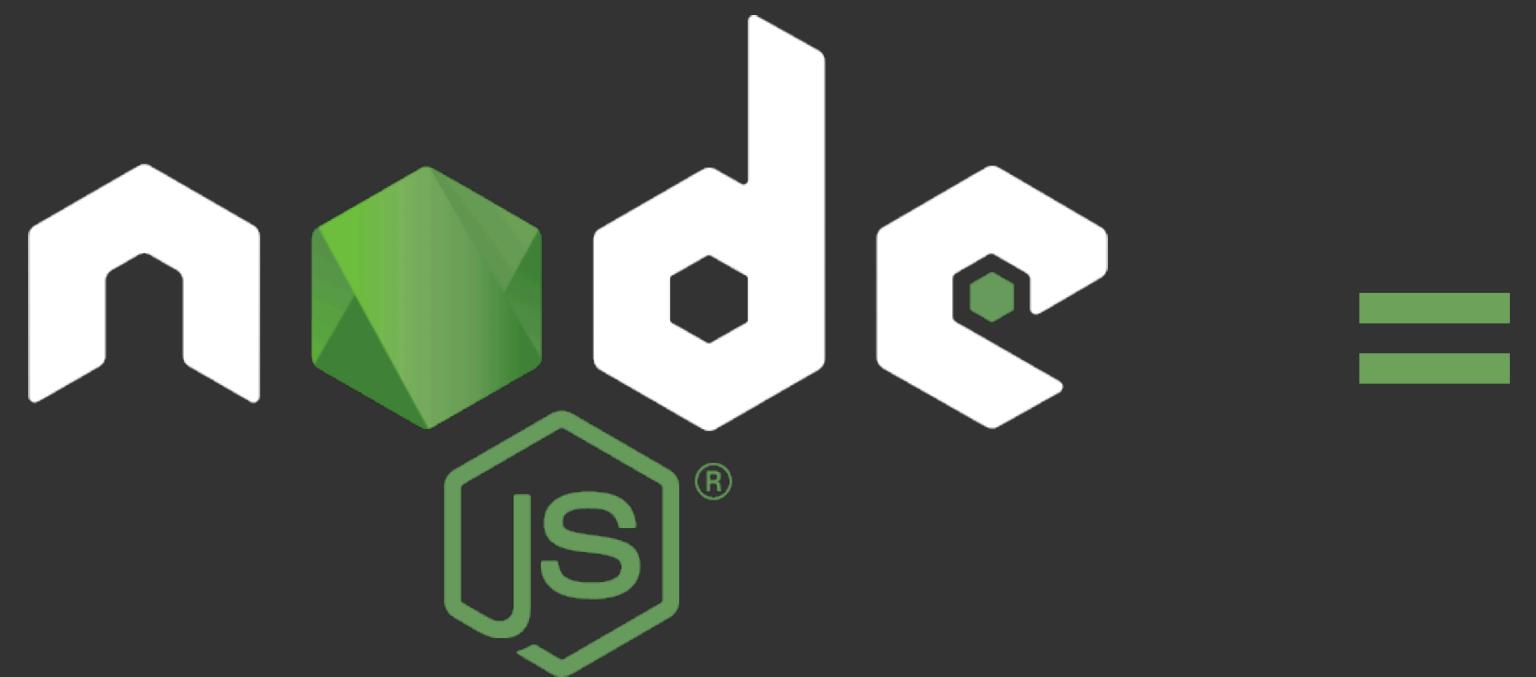


Key Points



Runtime Environment

- Node.js provides a runtime environment that allows developers to use JavaScript beyond the browser, enabling them to create high-performance, scalable, and real-time applications for various use cases.

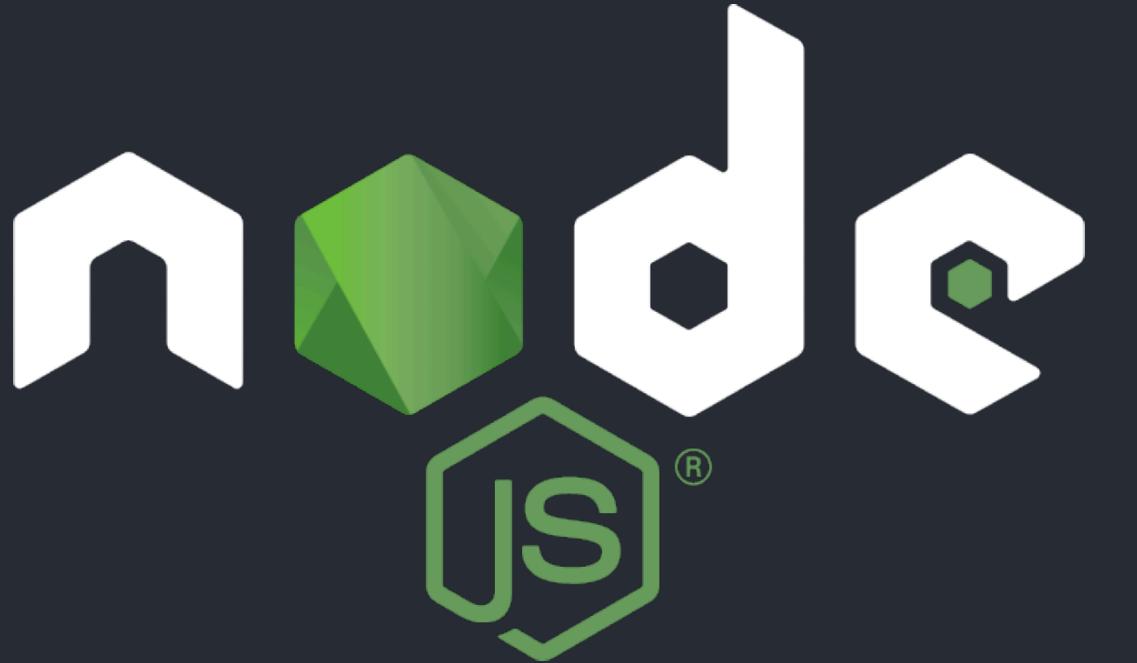


=

Runtime
Environment

+

JavaScript
Library

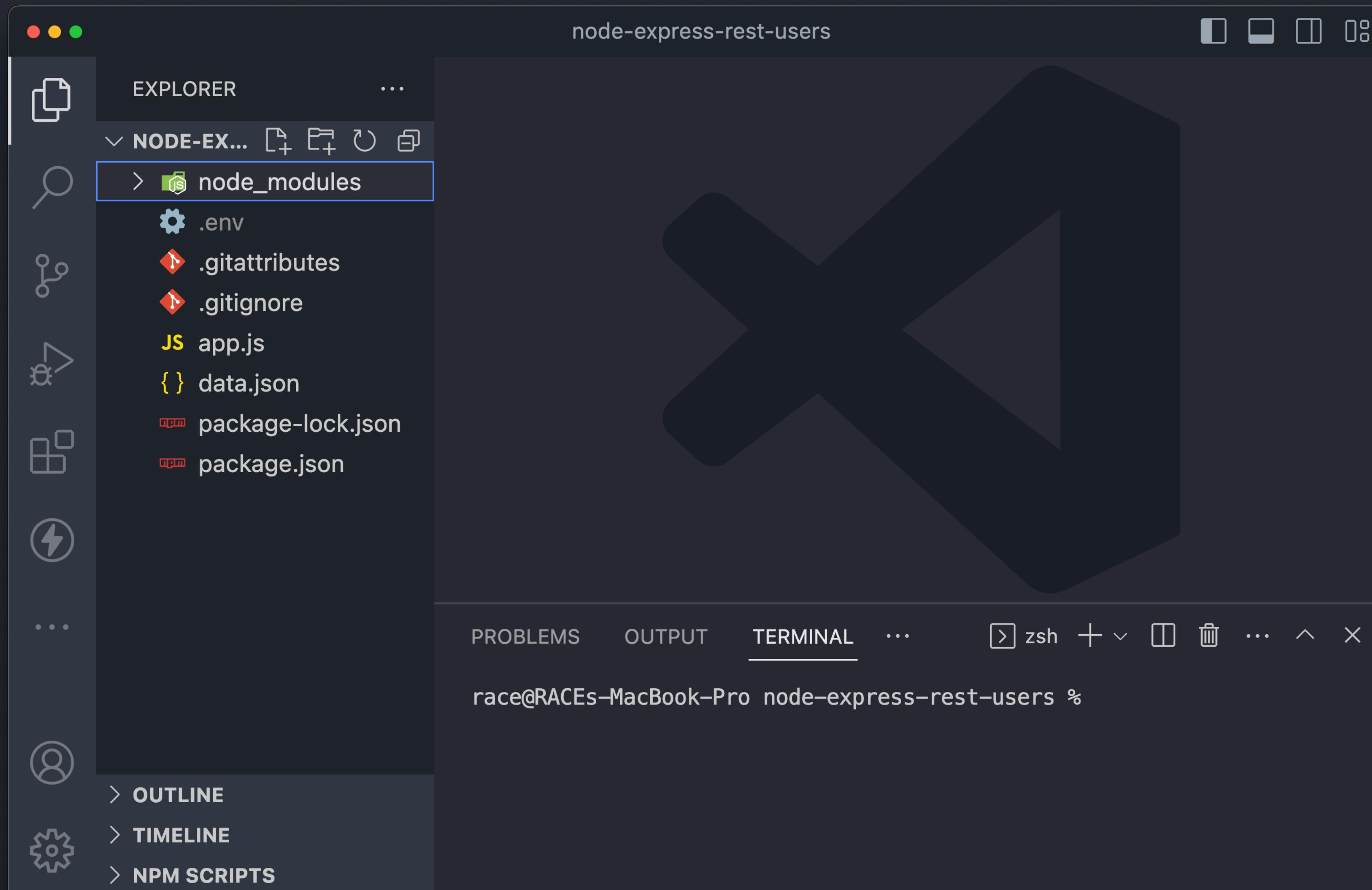


Node.js also comes with a toolbox called NPM, which has all sorts of ready-made tools (like Lego blocks) that developers can use to build their programs. Instead of building everything from scratch, you can use these tools to save time and make your app even better.

npm is a package manager for Node.js packages (JS modules)

- Node.js packages contains all the files you need for a module.
- Modules are JavaScript libraries you can include in your project.

node_modules

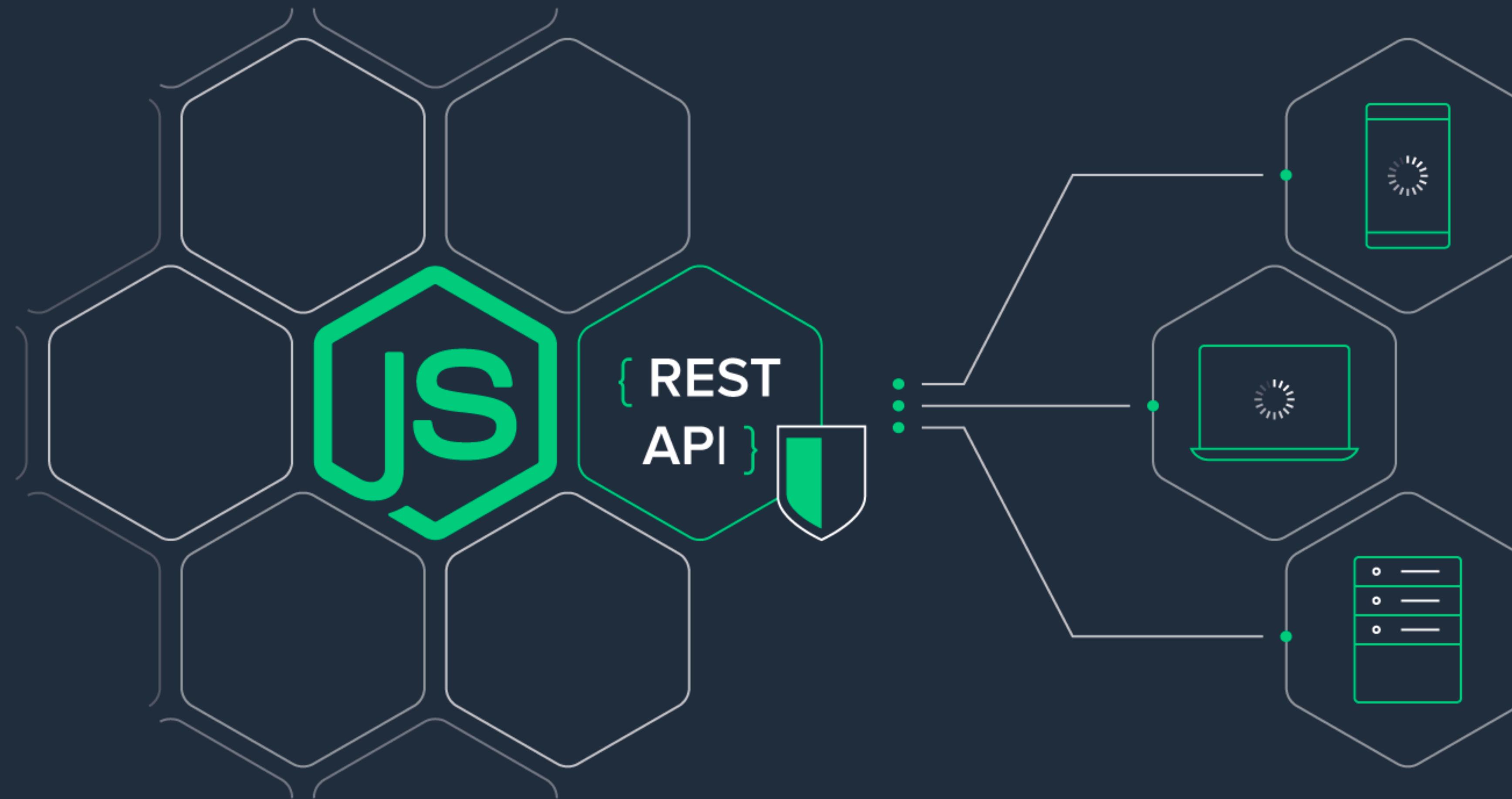


Express

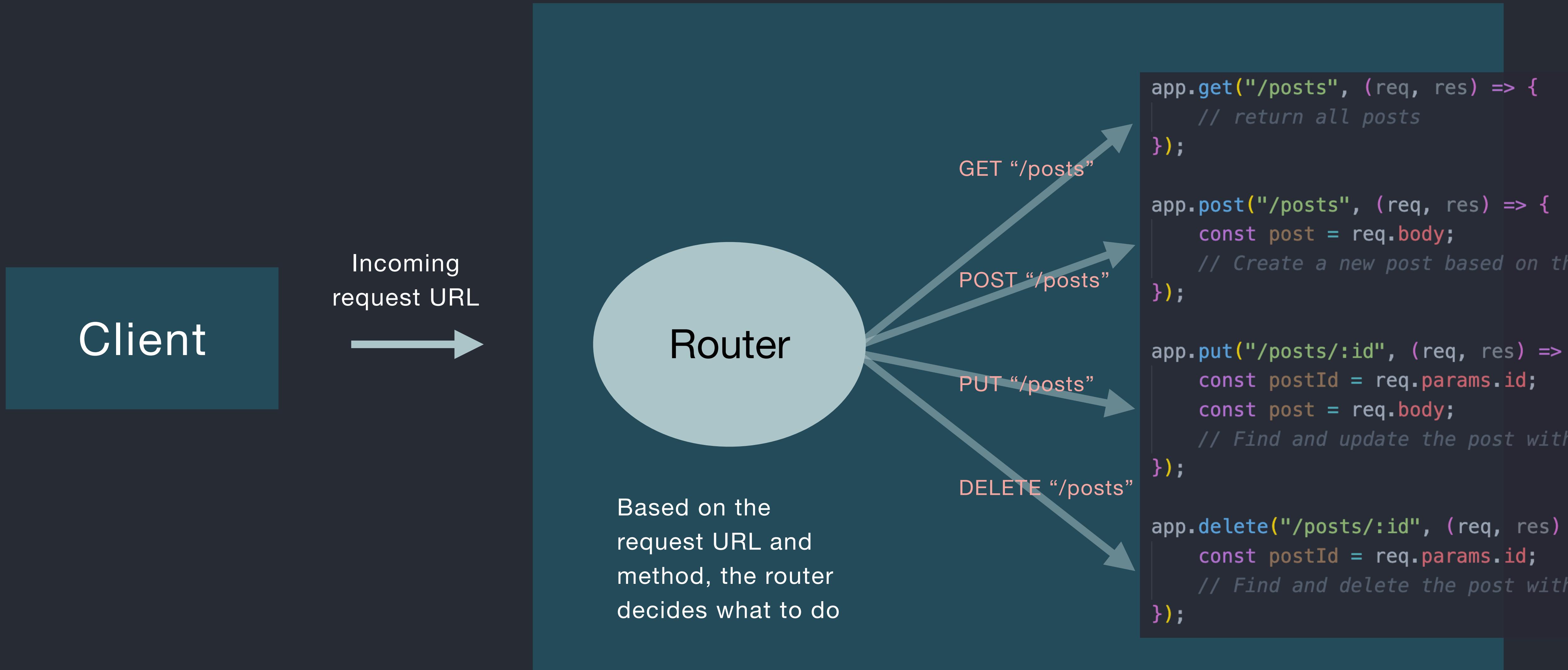


- A popular, minimalist web application framework for building APIs and web applications using Node.js.
- It simplifies the process of building web applications and APIs by providing a set of tools and utilities.

Streamlines and simplifies the implementation of REST API



Server



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

- The Express routing is built upon Node.js HTTP Module and it **simplifies the process of handling different URL paths and HTTP methods** in your web application.
- Express routing enables you to define routes for different endpoints, **simplifying request handling**.
- This enhances and streamlines code organization, and readability, and lets you focus on building functionality rather than intricate routing logic.

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");
});
```

Object Relational Mapping

ORM

id	name	mail	title	image
1	Peter Lind	petl@kea.dk	Senior Lecturer	https://share.cederdorff.com/images/pe...
2	Rasmus Cederdorff	race@dev.dk	Senior Lecturer	https://share.cederdorff.com/images/ra...
3	Lars Bogetoft	larb@eaaa.dk	Head of Education	https://kea.dk/slir/w200-c1x1/images/u...
4	Edith Terte	edan@kea.dk	Lecturer	https://media.lidcn.com/dms/image/C4E0...
5	Frederikke Bender	fbe@kea.dk	Head of Education	https://kea.dk/slir/w200-c1x1/images/u...
6	Murat Kilic	mki@eaaa.dk	Senior Lecturer	https://www.eaaa.dk/media/llyavasj/mur...
7	Anne Kirketerp	anki@eaaa.dk	Head of Education	https://www.eaaa.dk/media/5buh1xeo/ann...

From database to object

```
const user = {  
  id: 6,  
  name: "Murat Kilic",  
  mail: "mki@eaaa.dk",  
  title: "Senior Lecturer",  
  image: "https://www.eaaa.dk/media/llyavasj/mur...";  
};
```

Object Relational Mapping

ORM

id	name	mail	title	image
1	Peter Lind	petl@kea.dk	Senior Lecturer	https://share.cederdorff.com/images/pe...
2	Rasmus Cederdorff	race@dev.dk	Senior Lecturer	https://share.cederdorff.com/images/ra...
3	Lars Bogetoft	larb@eaaa.dk	Head of Education	https://kea.dk/slir/w200-c1x1/images/u...
4	Edith Terte	edan@kea.dk	Lecturer	https://media.licdn.com/dms/image/C4E0...
5	Frederikke Bender	fbe@kea.dk	Head of Education	https://kea.dk/slir/w200-c1x1/images/u...
6	Murat Kilic	mki@eaaa.dk	Senior Lecturer	https://www.eaaa.dk/media/llyavasj/mur...
7	Anne Kirketerp	anki@eaaa.dk	Head of Education	https://www.eaaa.dk/media/5buh1xeo/ann...

From database to array of objects

```
const users = [
  {
    id: 2,
    name: "Rasmus Cederdorff",
    mail: "race@dev.dk",
    title: "Senior Lecturer",
    image: "https://share.cederdorff.com/images/race.jpg"
  },
  {
    id: 3,
    name: "Lars Bogetoft",
    mail: "larb@eaaa.dk",
    title: "Head of Education",
    image: "https://kea.dk/slir/w200-c1x1/images/user-profile/chefer/larb.jpg"
  },
  {
    id: 4,
    name: "Edith Terte",
    mail: "edan@kea.dk",
    title: "Lecturer",
    image: "https://media.licdn.com/dms/image/C4E03AQE6nx7oUPqo\_g/profile-displayphoto-shrink\_800\_800"
  },
  {
    id: 5,
    name: "Frederikke Bender",
    mail: "fbe@kea.dk",
    title: "Head of Education",
    image: "https://kea.dk/slir/w200-c1x1/images/user-profile/fbe.jpg"
  },
  {
    id: 6,
    name: "Murat Kilic",
    mail: "mki@eaaa.dk",
    title: "Senior Lecturer",
    image: "https://www.eaaa.dk/media/llyavasj/murat-kilic.jpg?width=800&height=450&rnd=13340194655260"
  },
  {
    id: 7,
    name: "Anne Andersen",
    mail: "anki@mail.dk",
    title: "Head of Education",
    image: "https://www.eaaa.dk/media/5buh1xeo/anne-kirketerp.jpg?width=800&height=450&rnd=133403878321"
  }
];
```

Object Relational Mapping

ORM

- **Simplifies Database Interaction:** Allows using objects in code to interact with a relational database.
- **Maps Objects to Tables:** Links object-oriented language structures to relational database tables.
- **Abstraction Layer:** Abstracts database operations, reducing manual SQL coding.
- **Automates Queries:** Generates and executes SQL queries based on object interactions.
- **Saves Development Time:** Reduces the need for low-level database code, streamlining development.



Sequelize

v6 - stable ▾

API References ▾

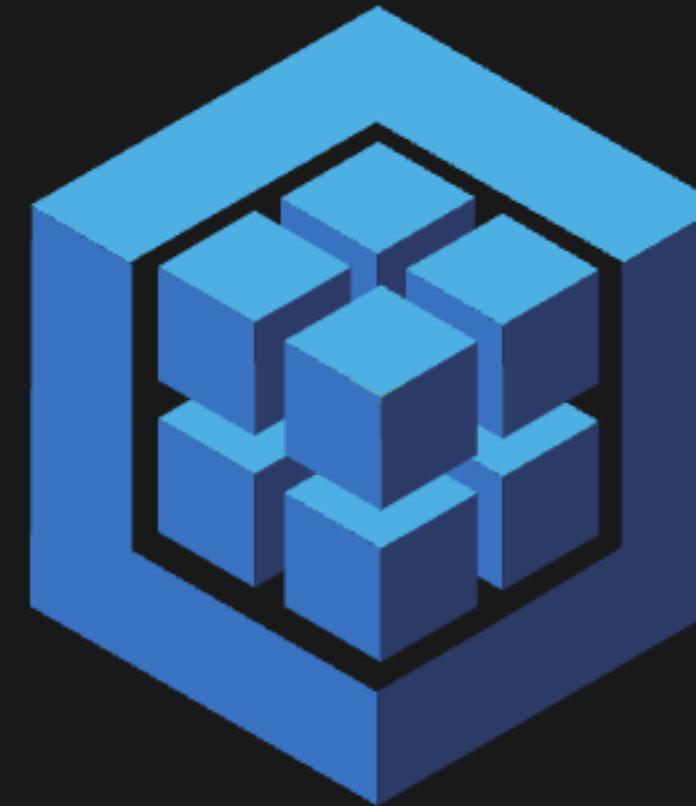
Slack

GitHub

Security



Search



Sequelize

Sequelize is a modern TypeScript and Node.js ORM for Oracle, Postgres, MySQL, MariaDB, SQLite and SQL Server, and more. Featuring solid transaction support, relations, eager and lazy loading, read replication and more.

Getting Started

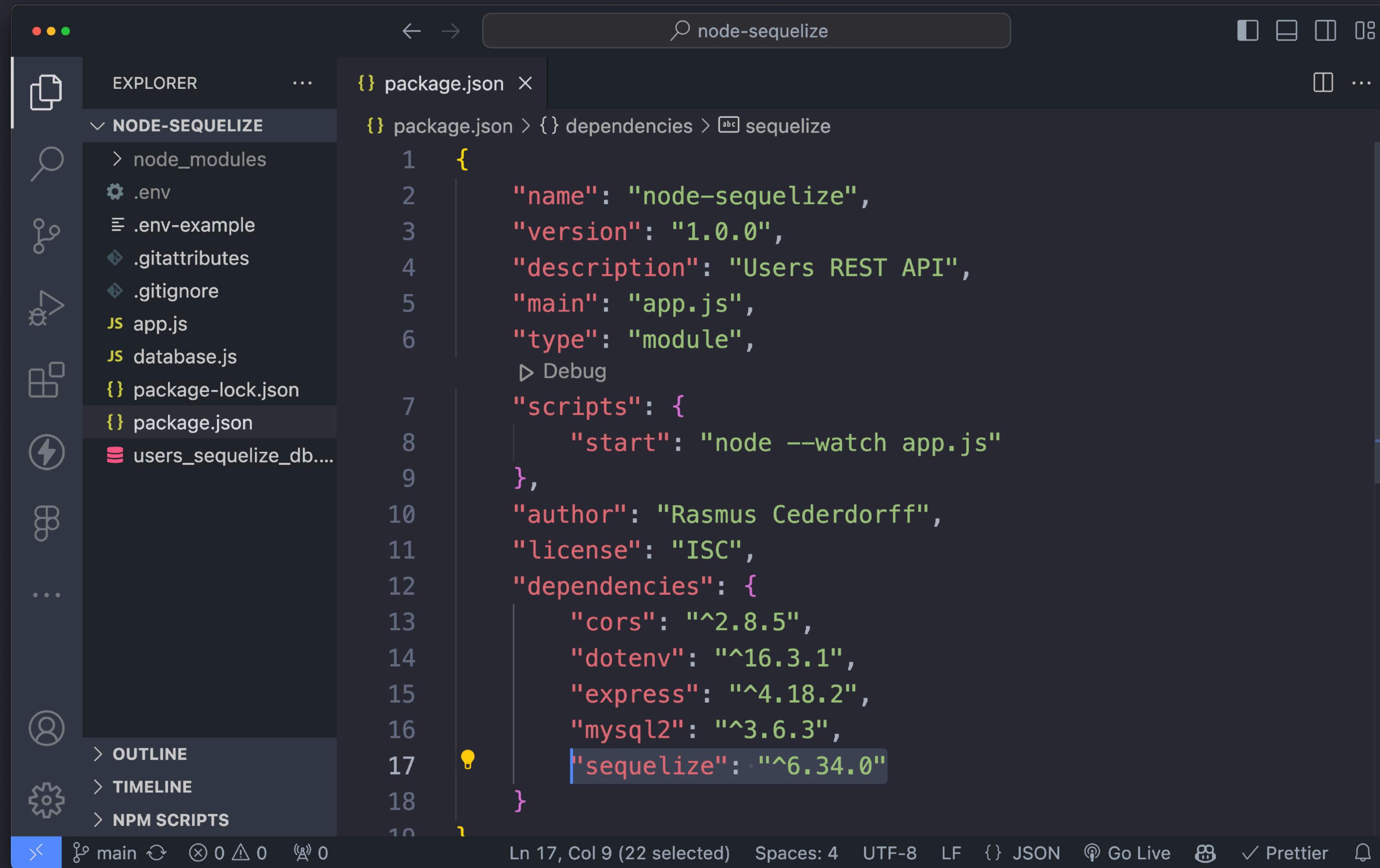
API Reference

Upgrade to v6

Support us

... npm package

Sequelize is a dependency (node module)



The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with the following details:

- File Explorer:** On the left, the "NODE-SEQUELIZE" folder is expanded, showing files like `node_modules`, `.env`, `.env-example`, `.gitattributes`, `.gitignore`, `app.js`, `database.js`, `package-lock.json`, `package.json`, and `users_sequelize_db....`.
- Search Bar:** At the top center, the search bar contains the text `node-sequelize`.
- Code Editor:** The main editor area displays the `package.json` file content. The `dependencies` section includes the entry `"sequelize": "^6.34.0"`.
- Status Bar:** At the bottom, the status bar shows the following information:
 - File name: `main`
 - Line and Column: `Ln 17, Col 9 (22 selected)`
 - Text Encoding: `UTF-8`
 - Character Set: `LF`
 - File Type: `JSON`
 - Live Server Status: `Go Live`
 - Prettier Status: `✓ Prettier`
 - Notification Bell: A small bell icon with a number `1` indicating one notification.

Define models to handle the DB interaction

```
// ===== Define Models ===== //
// Define user model
const User = sequelize.define("user", {
  // User model attributes
  name: {
    type: DataTypes.STRING,
    allowNull: false // Name is required
  },
  title: {
    type: DataTypes.STRING
  },
  mail: {
    type: DataTypes.STRING,
    allowNull: false // Email is required
  },
  image: {
    type: DataTypes.TEXT // URL to image
  }
});

// Sample users
// Sample user 1
const user1 = await User.create({
  name: "Rasmus Cederdorff",
  title: "Senior Lecturer",
  mail: "race@eaaa.dk",
  image: "https://share.cederdorff.com/images/race.jpg"
});

// Sample user 2
const user2 = await User.create({
  name: "Anne Kirketerp",
  title: "Head of Department",
  mail: "anki@eaaa.dk",
  image: "https://www.eaaa.dk/media/5buh1xeo/anne-kirketerp.jpg?width=1000"
});

// Sample user 3
const user3 = await User.create({
  name: "Murat Kilic",
  title: "Senior Lecturer",
  mail: "mki@eaaa.dk",
  image: "https://www.eaaa.dk/media/llyavasj/murat-kilic.jpg?width=1000"
});
```

Creates users table with entities

```
// ===== Define Models ===== //
// Define user model
const User = sequelize.define("user", {
  // User model attributes
  name: {
    type: DataTypes.STRING,
    allowNull: false // Name is required
  },
  title: {
    type: DataTypes.STRING
  },
  mail: {
    type: DataTypes.STRING,
    allowNull: false // Email is required
  },
  image: {
    type: DataTypes.TEXT // URL to image
  }
});
```

id	name	title	mail	image	createdAt	updatedAt
1	Rasmus Ced...	Senior Lect...	race@eaaa.dk	https://share.cederdo...	2023-11-12 19:0...	2023-11-12 19:0...
2	Anne Kirke...	Head of Dep...	anki@eaaa.dk	https://www.eaaa.dk/m...	2023-11-12 19:0...	2023-11-12 19:0...
3	Murat Kilic	Senior Lect...	mki@eaaa.dk	https://www.eaaa.dk/m...	2023-11-12 19:0...	2023-11-12 19:0...

Methods (SQL Queries)

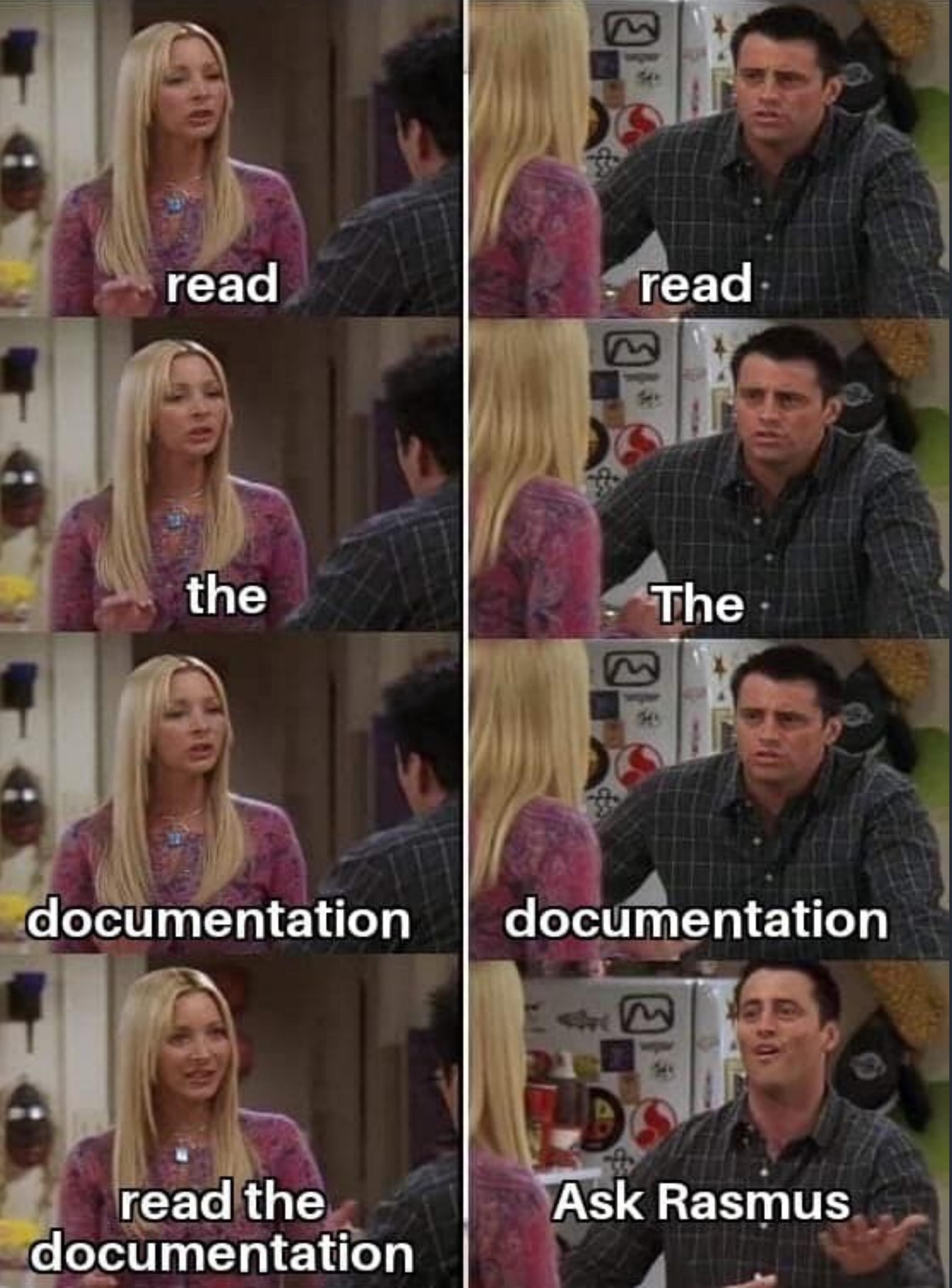
```
const users = await User.findAll(); // SELECT * FROM users;
```

```
const users = await User.findByPk(id); // SELECT * FROM users WHERE id = id;
```

```
const newUser = await User.create(user); // INSERT INTO users (name, title, mail, image) VALUES
```

```
const [result] = await User.update(user, { where: { id: id } }); // UPDATE users SET name =
```

```
const result = await User.destroy({ where: { id: id } }); // DELETE FROM users WHERE id = id
```

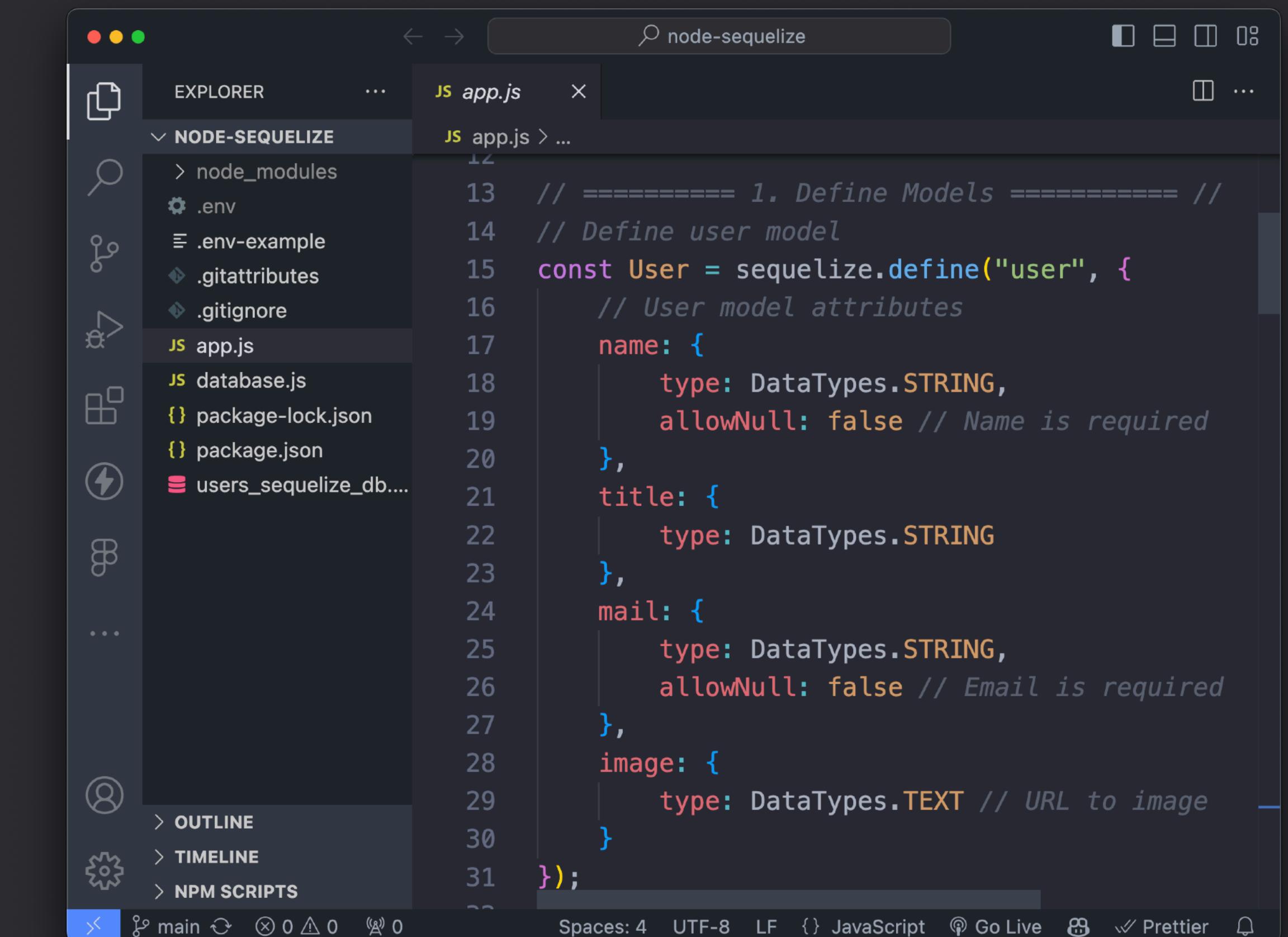


Useful Methods (see the docs)

- Core Concepts of Sequelize: <https://sequelize.org/docs/v6/category/core-concepts/>
- Model Basics: <https://sequelize.org/docs/v6/core-concepts/model-basics/>
- Model Querying - Basics: <https://sequelize.org/docs/v6/core-concepts/model-querying-basics/>
- Model Querying - Finders: <https://sequelize.org/docs/v6/core-concepts/model-querying-finders/>
- Associations: <https://sequelize.org/docs/v6/core-concepts/assocs/>

Node.js & Sequelize

Øvelse



The screenshot shows a dark-themed Node.js development environment. The Explorer pane on the left lists files and folders related to a Sequelize application, including `node_modules`, `.env`, `.env-example`, `.gitattributes`, `.gitignore`, `app.js` (which is selected), `database.js`, `package-lock.json`, `package.json`, and `users_sequelize_db....`. The Editor pane on the right displays the `app.js` file, which contains code for defining a User model using Sequelize. The code includes attributes for name, title, mail, and image, each with specific data types and allowNull settings.

```
// ===== 1. Define Models ===== //
// Define user model
const User = sequelize.define("user", {
  // User model attributes
  name: {
    type: DataTypes.STRING,
    allowNull: false // Name is required
  },
  title: {
    type: DataTypes.STRING
  },
  mail: {
    type: DataTypes.STRING,
    allowNull: false // Email is required
  },
  image: {
    type: DataTypes.TEXT // URL to image
  }
});
```

Code every Day

