



Web Technology 2

| Groep | Naam, Voornaam (student) | Studentennummer | Examencode |
|-------|--------------------------|-----------------|------------|
|-------|--------------------------|-----------------|------------|

| | | |
|-----------------------------------|-----------------------|-----------------------------|
| | Periode: 5 | TI Blended |
| Datum 04/06/2020 | Type examen Laptop | Examenduur 18-20u (120') |
| Vaktitularis(sen) Jan de Rijke | | Punten |

| |
|--|
| <p>Toegelaten hulpmiddelen</p> <p>Je mag informatie opzoeken op internet, maar het is niet toegelaten:</p> <ul style="list-style-type: none"> - Code over te nemen - Te communiceren met wie dan ook |
|--|

Starten

Download de examenbestanden en pak ze uit op je computer.

Je moet opdracht 2 uitvoeren vóór 3 en 4, verder kan je de opdrachten uitvoeren in de volgorde die je wenst. De gegeven volgorde is wel logisch. Hou je tijd in de gaten:

- Ga naar een andere opdracht als je vast zit en keer terug als je tijd overhebt
- Hou rekening met de puntenverdeling voor je tijdsindeling.

Procentuele puntenverdeling

- Opdracht 1: 33%
- Opdracht 2: 6%
- Opdracht 3: 40%
- Opdracht 4: 21%

Inleveren

Verwijder best eerst de node-modules mappen van de gemaakte opdrachten. Dit maakt het op te laden project veel kleiner. Zip dan je project en laad de zipfile op naar canvas.

SUCCES!

Algemene richtlijnen opdrachten

Elk examen gebruikt startbestanden met andere **entiteiten** (en subentiteiten, maar die gebruiken we niet). Je vind je **entiteiten** onder data\db.json. Vermeldingen van **entiteit** in deze opgave vervang je door de werkelijke naam ervan in jouw bestand. Als voorbeeld gebruiken we

- o **entiteit**: persons

Opdracht 1: Rest server

Je werkt in de map (directory) server. Je maakt een server met een GET en een PATCH request voor de **entiteit**. Voer daarvoor volgende stappen uit:



Web Technology 2

| Groep | Naam, Voornaam (student) | Studentennummer | Examencode |
|-------|--------------------------|-----------------|------------|
|-------|--------------------------|-----------------|------------|

0. Installeer npm dependencies

1. Delegeer naar router

De server vind je in app.js. Hij behandelt requests voor `http://localhost:4000`. Delegeer alle requests voor `http://localhost:4000/entiteit` (voorbeeld `http://localhost:4000/persons`) naar `routes/hoofd.js`.

2. GET

2.1 `hoofd.js` bevat in de constante data alle gegevens uit `db.json`. Je moet er wel je (hoofd) **entiteiten** nog uithalen.

2.2 Voeg een functie toe die een GET request afhandelt en een **entiteit** teruggeeft op basis van zijn. Antwoord volgens de REST conventies als de entiteit gevonden wordt en ook als ze niet gevonden wordt.

GET http://localhost:4000/entiteit/id_waarde

Voorbeeld request: GET <http://localhost:4000/persons/3>

Voorbeeld antwoord:

```
{
  "id": 3,
  "first_name": "Marie",
  "last_name": "Vertessen",
  "birth_day": "1988-12-23",
  "gender": "F",
  "image": "https://via.placeholder.com/200x80.png?text=Marie",
  "married": true,
  "yearsService": 21
}
```

Opgelet: test in je code best met `==` en niet met `===`, Dan slaagt je test ook als je een STRING uit een URL vergelijkt met een INT attribuut.

3. PATCH

Zorg ervoor dat je server een PATCH request afhandelt en een **entiteit** aanpast. De PATCH request bevat de `id_waarde` van de entiteit in de URL en een tweetal attributen met hun waarde in de body.

PATCH http://localhost:4000/entiteit/id_waarde

```
{
  "attribuut1": "waarde1",
  "attribuut2": "waarde2"
}
```

3.1 Zorg ervoor dat je router de JSON request body behandelt

3.2 Voeg een functie toe die bij een PATCH request de entiteit zoekt met de gegeven **id_waarde**. Pas de entiteit aan met de waarden voor de attributen in de request body, de andere attributen blijven onveranderd. Als de request slaagt geef je status 200 terug, als de entiteit niet gevonden wordt geef je status 404 terug.

De merge functie, die de entiteit aanpast met de attribuutwaarden uit de body, zit reeds klaar in `routes/hoofd.js`. Ze voegt ook een uniek uuid nummer aan de entiteit toe voor elke transactie. Roep ze aan van uit jouw functie en geef volgende parameters mee:

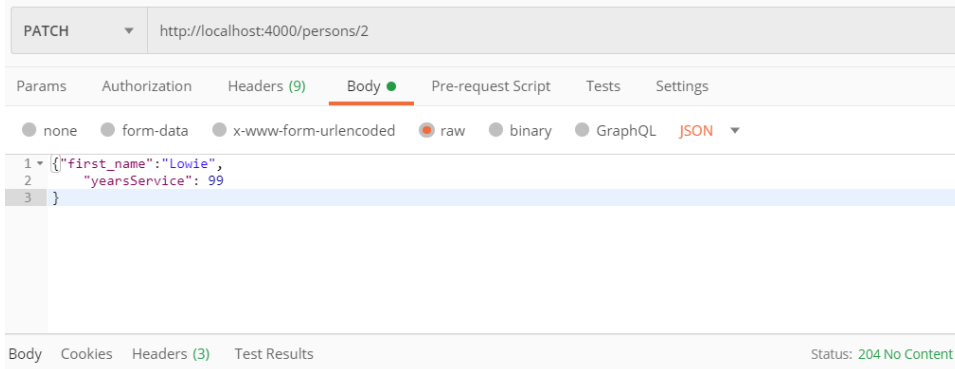
- Original: de entiteit die je opzoekt op basis van de **id_waarde**
- Modifier: het object dat je uit de body van de PATCH request haaldet.



Web Technology 2

| | | | |
|-------|--------------------------|-----------------|------------|
| Groep | Naam, Voornaam (student) | Studentennummer | Examencode |
|-------|--------------------------|-----------------|------------|

Voorbeeld test PATCH request in postman, let op de quotes in de meegegeven JSON body!



Je kan testen of je PATCH request geslaagd is door daarna (in postman of in de browser) dezelfde entiteit op te vragen.

voorbeeld: GET <http://localhost:4000/persons/2>

```
{
  "id": 2,
  "first_name": "Lowie",
  "last_name": "Beton",
  "birth_day": "1989-06-12",
  "gender": "M",
  "married": false,
  "image": "https://via.placeholder.com/200x80.png?text=Jan",
  "yearsService": 99,
  "uuid": "79234cf8-dbfd-501c-9a09-802f695db20f"
}
```

Opdracht 2: Invoer scherm entiteit

Werk in de map client.

Maak een invoer scherm in client\src\html\index.html met daarin

- De id
- **drie andere** attributen van de entiteit met bijpassend HTML invoer veld
- het image attribuut. Dit laatste attribuut moet je alleen tonen. Je moet het niet kunnen invoeren. Initieel (als het scherm nog leeg is) toon je `"https://via.placeholder.com/200x80.png?text=testbeeld"`

Onderaan het scherm voeg je een *Toon* en een *Pas aan* knop toe.

Je hebt dit scherm nodig in de volgende opdrachten. Verlies nu geen tijd met de opmaak, dat komt later.



Web Technology 2

| Groep | Naam, Voornaam (student) | Studentennummer | Examencode |
|-------|--------------------------|-----------------|------------|
| | | | |

Voorbeeld:

Opmerking: het client project is geconfigureerd met npm, webpack en bootstrap. Package.json en webpack.config.js zijn volledig in orde (je mag ze aanpassen, maar het is niet nodig).

Bovenstaand voorbeeld toont src\html\index.html rechtstreeks in de browser. Als je het project opstart met webpack/bootstrap ziet het scherm er beter uit (maar dat is nu nog onbelangrijk).

Opdracht 3: Ajax

1. Start json-server.

In de map data

- a. installeer je alle npm afhankelijkheden.
- b. Run je
`npm start`

Je roept deze server aan in de AJAX code die je in deze opdracht schrijft. Deze server voert dezelfde functies uit als degene die je schreef in opdracht 1, maar werkt op een andere poort (3000)

2. Gebruik webpack

Werk in de map client. Installeer alle npm afhankelijkheden. Start de webpack-dev-server met

`npm start`

3. Opzoeken entiteit op basis van id

Plaats de javascript code voor deze opdracht in src\js\ajax.js. Zorg ervoor dat de code gebruikt wordt via src/index.js.

Als de gebruiker in het invoerscherm de **id_waarde** invult en op de **Toon** knop klikt, roep je de functie `toonentiteit(id_waarde)` aan en toon je de entiteit in het scherm. Schrijf deze functie.

In de functie voer je volgende call uit via AJAX. Maak verplicht gebruik van `async:await`.

GET http://localhost:3000/entiteit/id_waarde

Voorbeeld: GET <http://localhost:3000/persons/2>

Als de GET request een persoon teruggeeft vul je alle in het scherm aanwezige attributen (inclusief het image) in. Afhankelijk van je dataset kan het image van je json-server komen of van elders op het internet.

Opmerking: Als er een fout in het image zit (bestand niet aanwezig in data\public; internet link is niet juist...) mag



Web Technology 2

| Groep | Naam, Voornaam (student) | Studentennummer | Examencode |
|-------|--------------------------|-----------------|------------|
|-------|--------------------------|-----------------|------------|

je een image ophalen van `"https://via.placeholder.com/200x80.png?text=xxx"`, waarbij je xxx vervangt door de waarde van het attribuut dat de entiteit best beschrijft.

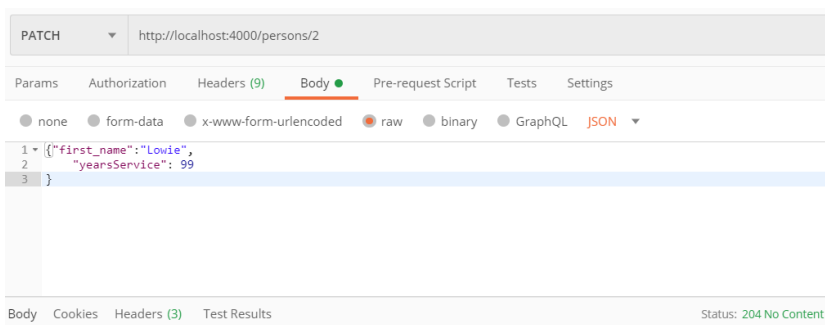
4. Aanpassen entiteit

Schrijf een functie voor volgend gedrag:

Als de gebruiker de id van een entiteit en minimum twee andere attributen invult en op de *Pas aan* knop klikt, wordt een PATCH request uitgevoerd. De aangepaste entiteit wordt daarna op het scherm getoond.

De functie verzendt een PATCH request met de waarde van het identificerende attribuut in de URL en de twee andere gekozen waarden in een JSON body.

Voorbeeld (in postman, maar jij moet de aanroep programmeren in javascript):



Indien het patch request een correcte status (200) teruggeeft, roep je de eerder geschreven functie toonentiteit (ID) aan, om het aangepaste object te tonen.

In deze functie handel je de resultaten van de asynchrone functies die je aanroept af met de Promises API. Je gebruikt deze keer **geen** `async/await`.

Verifieer in het scherm beide attributen aangepast zijn en de rest onveranderd is.

Opdracht 4: layout

Aanpassen layout invoerscherm.

Bij deze opdrachten is enkel de plaatsing van de componenten belangrijk. Andere aspecten zijn niet belangrijk (grootte image, velden... / kleur / labels moeten bij hun invoerveld staan maar of die er boven of er naast staan maakt niet uit).

1. Knoppen staan steeds onderaan
2. Op een extra small display is de image niet zichtbaar Voorbeeld:



Web Technology 2

| Groep | Naam, Voornaam (student) | Studentennummer | Examencode |
|-------|--------------------------|-----------------|------------|
|-------|--------------------------|-----------------|------------|

id

geboortedatum

voornaam

achternaam

[Toon](#) [Pas aan](#)

3. Op een small display staat de image onder de invoervelden

id

geboortedatum

voornaam

achternaam



[Toon](#) [Pas aan](#)


4. Op een medium display staat de image rechts

id

geboortedatum

voornaam

achternaam



[Toon](#) [Pas aan](#)

5. Op een grotere display staat de image links



EX/2018-19/4/20846/3

Web Technology 2

| Groep | Naam, Voornaam (student) | Studentennummer | Examencode |
|-------|--------------------------|-----------------|------------|
|-------|--------------------------|-----------------|------------|



id

geboortedatum

voornaam

achternaam

Toon

Pas aan