Node.js - Laboratorium 6

JSON-SERVER

Aplikacja pozwalająca na postawienie szybkiego serwera REST

- https://github.com/typicode/json-server

W tym laboratorium do każdego z zadań będziemy wykorzystywać lokalny serwer z przykładowymi danymi. W tym celu powinniśmy wejść do naszego katalogu `server` i wykonać następujące kroki:

1. zainstalować niezbędne zależności naszego projektu: `npm install`
2. uruchomić nasz serwer poleceniem:

```
json-server --port 4800 --watch db.json
```

> lub

`npm start` - jest to alias do naszego skryptu zapisanego w `package.json`

Więcej informacji odnośnie postawienia serwera i komunikacji znajdziesz na repozytorium paczki `json-server`

AXIOS

Biblioteka pozwalająca na komunikowanie się klienta z serwerem przy pomocy protokołu HTTP. Swoją implementację bazuje na `Promise`.

- https://github.com/axios/axios

Użycie różnych wariantów można znaleźć na repozytorium pakietu.

JSON-SERVER i AXIOS

Aby `json-server` był wstanie odpowiednio przetworzyć nasze żądania niezbędne jest dodanie dodatkowego nagłówku do naszych zapytań, np.:

```
axios.post('http://localhost:4800/users', newUserData, {
    headers: {
        'Content-Type': 'application/json'
    }
})
```

lub zgodnie z dokumentacją możemy stworzyć nową instancję naszego serwisu `axios`, która będzie zawierała w każdym żądaniu nasz nagłówek:

```
const myAxios = axios.create({
    headers: {
```

```
        'Content-Type': 'application/json'
    }
});

myAxios.post('http://localhost:4800/users', newUserData);
```

Przydatne linki

Podstawy HTTP: https://learn.onemonth.com/understanding-http-basics

HTTP Messages: https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages

Lista nagłówków HTTP: https://pl.wikipedia.org/wiki/Lista_nag%C5%82%C3%B3wk%C3%B3w_HTTP

Kody odpowiedzi HTTP: https://pl.wikipedia.org/wiki/Kod_odpowiedzi_HTTP

Zadania do wykonania na laboratorium

1. Stwórzmy aplikację która stworzy nowego użytkownika wysyłając odpowiednie zapytanie do naszego lokalnego serwera.

Model użytkownika:

```
{
    "name": "...",
    "username": "...",
    "email": "...",
}
```

Wyświetlmy informację czy poprawnie użytkownik został zapisany i jaki otrzymał `id`.

2. Dodajmy do zadania 1 funkcjonalność pozwalającą na pobranie użytkownika. Już na tym etapie powinniśmy podzielić naszą aplikację na odpowiednie moduły.
3. Dodajmy do zadania 2 możliwość modyfikowania naszego użytkownika i wysłania zmian na nasz serwer.
4. Dodajmy do naszej aplikacji usuwanie użytkownika z serwera. Sprawdźmy czy nasz użytkownik został usunięty wysyłając zapytanie o pobranie danych.
5. Wykorzystując wiedzę z poprzednich zajęć dodajmy możliwość dodawania oraz usuwania użytkownika poprzez argumenty uruchamiane naszą aplikację(`yargs`, itp...).

```
node app.js delete --id 2

node app.js add --name "..." ...

node app.js getUser ...
```

Zachęcam do zapoznania się z dokumentacją pakietu `yargs`. (commands, itp...)

6. Rozszerzmy naszą aplikację o dodawanie, usuwanie oraz modyfikowanie postów analogicznie jak zrobiliśmy z użytkowników.

> Pamiętajmy o zachowaniu czystości kodu i podzieleniu naszej aplikacji na odpowiednie moduły.

> Przed dodaniem nowego `posta` upewnijmy się że dany użytkownik istnieje na naszym serwerze.

7. Wykonajmy te same operacje analogicznie dla encji `albums`.