

Raport z wykonania ćwiczenia REST, Mikroserwisy

Jakub Płotnikowski

Styczeń 2020

Spis treści

1	Kod z zadań 1, 2, 3	2
1.1	Zadanie 1	2
1.2	Zadanie 2	3
1.3	Zadanie 3	4
2	Zadanie 4	6
2.1	Treść zadania	6
2.2	Pobranie listy pacjentów	6
2.3	Dodanie przykładowego pacjenta	6
2.4	Zmiana imienia i nazwiska pacjenta	6
2.5	Ponowne pobranie listy pacjentów	6
2.6	Usunięcie pacjenta o id równym 1	6
2.7	Ponowne pobranie listy pacjentów	6

1 Kod z zadań 1, 2, 3

Jako, że zadania 1, 2 oraz 3 wykonałem oraz oddałem na zajęciach, wklejam tylko ich kod bez dodatkowego omówienia.

1.1 Zadanie 1

```
1 // File 01_HttpServer/app.js
2 // Import Express web framework
3 const express = require("express");
4 // Create main app
5 const app = express();
6
7 // function printReqSummary(request) {
8 // // Display handled HTTP method and link (path + queries)
9 // console.log('Handling ${request.method} ${request.originalUrl}');
10 //}
11
12 // Helper function -- print request summary
13 function printReqSummary(request) {
14 // Display handled HTTP method and link (path + queries)
15 console.log('Handling ${request.method} ${request.originalUrl} ${Date.now()}
16   ');
17 }
18
19 // GET / -- Show main page
20 app.get("/", function(request, response) {
21   printReqSummary(request);
22   // Send response to the request (here as a HTML markup)
23   response.send("<h1>HTTP Server</h1><p>Go to /hello subpage!</p>");
24 }
25 );
26
27 // GET /hello -- Show message
28 app.get("/hello", function(request, response) {
29   printReqSummary(request);
30   response.send("<p>Anonymous message: Oh, Hi Mark!</p>");
31 }
32 );
33
34 // GET /time -- Show current time
35 app.get("/time", function(request, response) {
36   printReqSummary(request);
37   response.send('<p>Printing current time: ${new Date()}</p>');
38 }
39 );
40
41 // Start HTTP server at port 3000
42 // (type in the browser or use cURL: http://localhost:3000/)
43 app.listen(3000);
```

1.2 Zadanie 2

```
1 // File 02_UrlParameters/app.js
2 const express = require("express");
3 const app = express();
4
5 function printReqSummary(request) {
6   console.log('Handling ${request.method} ${request.originalUrl}');
7 }
8
9 function getRandomInt(min, max) {
10   return Math.floor(Math.random() * (max - min + 1)) + min;
11 }
12
13 function getRandomParameter(parameters) {
14   return parameters["p" + getRandomInt(1, 3)];
15 }
16
17 // GET / -- Show main page
18 app.get("/", function(request, response) {
19   printReqSummary(request);
20   response.send(
21     '<h1>URL Parameters (and Queries)</h1><ul>
22       <li>Show normal message (GET /hello/segment1)</li>
23       <li>Show special message (GET /hello/segment1/segment2?age=NUMBER&height=
24         NUMBER)</li>
25       <li> where segment1, segment2 - any valid URL part</li>
26     </ul>';
27   );
28 });
29
30 // GET /hello/:name -- Show normal message for a named person
31 app.get("/hello/:name", function(request, response) {
32   printReqSummary(request);
33   // Grab URL parameters from 'request.params' object
34   response.send('<p>Normal message for: ${request.params.name}</p>');
35 });
36
37 // GET /hello/:name/:surname -- Show special message with plenty of parameters
38 app.get("/hello/:name/:surname", function(request, response) {
39   printReqSummary(request);
40   // Grab (optional) URL queries from 'request.query' object
41   const age = request.query.age !== null ? request.query.age : "unknown";
42   const height = request.query.height !== null ? request.query.height : "unknown";
43   response.send(
44     '<p>Special message for: ${request.params.name} ${request.params.surname}
45       (age: ${age} years, height: ${height} cm)</p>';
46   );
47 });
48
49 app.get("/random/:p1/:p2/:p3", function(request, response) {
50   printReqSummary(request);
51   response.send('<p>Random parameter: ${getRandomParameter(request.params)}</p>');
52 });
53
54
55 app.listen(3000);
```

1.3 Zadanie 3

```
1 // File 03_HttpMethods/app.js
2 //
3 const express = require("express");
4 const app = express();
5
6 function printReqSummary(request) {
7   console.log('Handling ${request.method} ${request.originalUrl}');
8 }
9
10 /* Store items collection in this array */
11 let items = [];
12
13 /* GET / -- Show main page */
14 app.get("/", function(request, response) {
15   printReqSummary(request);
16   response.send(
17     '<h1>HTTP Methods</h1><ul>
18       <li>Show items (GET /item)</li>
19       <li>Add an item (PUT /item/:name)</li>
20       <li>Remove an item (DELETE /item/:name)</li></ul>'
21   );
22 });
23
24 /* GET /item -- Show all items from the collection */
25 app.get("/item", function(request, response) {
26   printReqSummary(request);
27   response.send('<p>Available items: ${items.toString()}</p>');
28 });
29
30 // adding new item
31 app.post("/item/", function(request, response) {
32   printReqSummary(request);
33   const name = request.query.name !== null ? request.query.name : "undefined";
34   /* Is the item in collection? */
35   if (items.includes(name)) {
36     response.send('<p>Item "${name}" already in collection</p>');
37   } else {
38     items.push(name);
39     response.send('<p>Item "${name}" added successfully</p>');
40   }
41 });
42
43 // modification of an item
44 app.put("/item/:name", function(request, response) {
45   printReqSummary(request);
46   const itemName = request.params.name;
47   const newItemName = request.query.newItemName !== null ? request.query.
48     newItemName : "undefined";
49   /* Is the item in collection? */
50   if (items.includes(itemName)) {
51     items[items.indexOf(itemName)] = newItemName;
52     response.send('<p>Item "${itemName}" changed to "${newItemName}"
53       successfully</p>');
54   } else {
55     response.send('<p>Item "${itemName}" not found</p>');
56   }
57 });
```

```

57  /* DELETE /item/:name -- remove a given item from the collection */
58  app.delete("/item/:name", function(request, response) {
59      printReqSummary(request);
60      const itemName = request.params.name;
61      /* Is the item in collection? */
62      if (items.includes(itemName)) {
63          items = items.filter(item => item !== itemName);
64          response.send('<p>Item "${itemName}" removed successfully</p>');
65      } else {
66          response.send('<p>Item "${itemName}" doesn't exists</p>');
67      }
68  });
69
70  app.listen(3000);

```

2 Zadanie 4

2.1 Treść zadania

Wyprowadzić i przeanalizować wynik metody GET dla bazowego URLa. Zaobserwować rezultaty dla URLa zawierającego numer (id) pacjenta w zależności od użytej metody HTTP. Dla testowania wygodnie jest użyć wywołania curl -X Przeanalizować różnice w logice poszczególnych implementacji kodu dla obsługi tych metod. Uwaga: Zwrócić uwagę na fragment kodu pomiędzy db i średnikiem, który odwołuje się do kodu pakietu lowdb, zainicjowanego na początku. Zamieścić w raporcie przykładowe wyniki (i komentarz do nich).

2.2 Pobranie listy pacjentów

```
1 curl -X GET localhost:3000/patient
2 {"error":"No patients are registered"}
```

2.3 Dodanie przykładowego pacjenta

```
1 curl -X POST "localhost:3000/patient?name=Jakub&surname=Plotnikowski"
2 {"id":1,"name":"Jakub","surname":"Plotnikowski"}
```

2.4 Zmiana imienia i nazwiska pacjenta

2.5 Ponowne pobranie listy pacjentów

```
1 curl -X GET localhost:3000/patient
2 {"error":"No patients are registered"}
```

2.6 Usunięcie pacjenta o id równym 1

```
1 curl -X DELETE "localhost:3000/patient/1"
2 {"message":"Patient removed successfully"}
```

2.7 Ponowne pobranie listy pacjentów

```
1 curl -X GET localhost:3000/patient
2 {"error":"No patients are registered"}
```