Rafał Gaweł

Bazy Danych

REST

Etap 1: Katalog 01_HttpServer

```
$ curl -sX GET http://localhost:3000/
<h1>HTTP Server</h1><p>Go to /hello subpage!</p>

$ curl -sX GET http://localhost:3000/hello
<p>Anonymous message: Oh, Hi Mark!</p>
```

Po stronie serwera wyświetlane zostają komunikaty:

```
Handling GET /
Handling GET /hello
```

Modyfikacja kodu raportującego

Funkcja printReqSummary przed modyfikacją i po.

```
// function printReqSummary(request) {
//   // Display handled HTTP method and link (path + queries)
//   console.log(`Handling ${request.method} ${request.originalUrl}`);
// }

function printReqSummary(request) {
  // Display handled HTTP method and link (path + queries)
  console.log(`[${Date.now()}] ${request.method} ${request.originalUrl}`);
}
```

Po zmianie wiadomości wyglądają następująco:

```
[1579813379040] GET /
[1579813389981] GET /hello
```

Endpoint /time

Nowy endpoint powinien zwracać obecny timestamp.

```
// GET /time -- Show current timestamp
app.get("/time", function(request, response) {
  printReqSummary(request);
  response.send(`<p>${Date.now()}</p>`);
});
```

Wysłanie zapytania:

```
$ curl -sX GET http://localhost:3000/time
<p>1579813539393</p>
```

Log serwera:

```
[1579813539393] GET /time
```

Etap 2: Katalog 02_HttpServer

Wyświetlona zostaje lista obsługiowanych endpointów wraz z informacją o parametrach

```
$ curl -sX GET http://localhost:3000
<h1>URL Parameters (and Queries)</h1><ul>
        <li>Show normal message (GET /hello/segment1)</li>
        <li>Show special message (GET
/hello/segment1/segment2?age=NUMBER&height=NUMBER)</li>
    <li>  where segment1, segment2 - any valid URL part</li>
    </ul>
```

Dla /hello/:name

To co podamy w miejscu :name zostanie użyte do budowania odpowiedzi. Jeżeli nie podamy nic, to Express uzna, że pytamy o ścieżkę /hello/, która nie jest zdefiniowana.

```
$ curl -sX GET http://localhost:3000/hello/Rafal
<p>Normal message for: Rafal</p>

$ curl -sX GET http://localhost:3000/hello/Michal
<p>Normal message for: Michal</p>

$ curl -sX GET http://localhost:3000/hello/
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET /hello/</pre>
</body>
</html>
```

Dla /hello/:name/:surname?age=_&height=_

Poniższe wyniki pokazują, że query-stringi są opcjonalne. W wypadku braku odpowiedniego klucza, pole w odpowiedzi ma wartość undefined.

```
$ curl -sX GET http://localhost:3000/hello/Rafal/Gawel
<p>Special message for: Rafal Gawel
  (age: undefined years, height: undefined cm)</p>

$ curl -sX GET http://localhost:3000/hello/Rafal/Gawel?age=22\&height=22
<p>Special message for: Rafal Gawel
  (age: 22 years, height: 22 cm)</p>
```

Endpoint zwracający losowe parametry

```
app.get("/rand/:a/:b/:c", function(req, response) {
    printReqSummary(req);
    function getRandomInt(min, max) { return Math.floor(Math.random() * (max - min + 1)) +
min; }
    let rand = [req.params.a, req.params.b, req.params.c][randint(0, 2)];
    response.send(`<p>${rand}</p>`);
});
```

Po wysłaniu wielu zapytań widać, że wyniki są losowe:

```
$ curl -sX GET http://localhost:3000/rand/a/b/c
<p>b</p>

$ curl -sX GET http://localhost:3000/rand/a/b/c
<p>c</p>[

url -sX GET http://localhost:3000/rand/a/b/c
<p>a</p>

 curl -sX GET http://localhost:3000/rand/a/b/c
<p>c</p>

$ curl -sX GET http://localhost:3000/rand/a/b/c
<p>a</p>

 $ curl -sX GET http://localhost:3000/rand/a/b/c
<p>a</p>

$ curl -sX GET http://localhost:3000/rand/a/b/c
<p>a</p>
```

Etap 3: Katalog 03_HttpServer

Testowanie endpointów

Wyświetlone informacje o endpointach
$ curl -sX GET http://localhost:3000
<h1>HTTP Methods</h1>
	Show items (GET /item)
	Add an item (PUT /item/:name)
	Remove an item (DELETE /item/:name)

Nic nie dodaliśmy, więc brak jakichkolwiek itemów.
$ curl -sX GET http://localhost:3000/item/
Available items:

Dodajemy nowy element
$ curl -sX PUT http://localhost:3000/item/laptop
Item "laptop" added successfully [

Próba dodania istniejącego wpisu się nie powiedzie.
$ curl -sX PUT http://localhost:3000/item/laptop
Item "laptop" already in collection [

Usuwamy istniejący wpis
$ curl -sX DELETE http://localhost:3000/item/laptop
Item "laptop" removed successfully [

Próbujemy usunąć już nieistniejący wpis
$ curl -sX DELETE http://localhost:3000/item/laptop
Item "laptop" doesn't exists [

Potwierdzamy, że wpis został usunięty
$ curl -sX GET http://localhost:3000/item
Available items: [

Dodawanie POSTem

Implementujemy dodawanie wpisów za pomocą metody POST oraz z przekazywaniem parametrów przez query stringi.

```
/* POST /item -- add new item to the collection */
app.post("/item/:name", function(request, response) {
  printReqSummary(request);
  const itemName = request.params.name;
```

```
  /* Is the item in collection? */
  if (items.includes(itemName)) {
        response.send(`<p>Item "${itemName}" already in collection</p>`);
  } else {
        items.push(itemName);
        response.send(`<p>Item "${itemName}" added successfully</p>`);
  }
});
```

Updatowanie PUTem

Modyfikujemy starą metodę PUT tak aby jako query string przyjmowała także nową nazwę.

```
/* PUT /item/:name -- add (put) new item to the collection */
// app.put("/item/:name", function(request, response) {
//   printReqSummary(request);
//   const itemName = request.params.name;
//   /* Is the item in collection? */
//   if (items.includes(itemName)) {
//     response.send(`<p>Item "${itemName}" already in collection</p>`);
//   } else {
//     items.push(itemName);
//     response.send(`<p>Item "${itemName}" added successfully</p>`);
//  }
// });

/* PUT /item/:old_name -- updates item with old_name with name */
app.put("/item/:old_name/:new_name", function(request, response) {
        printReqSummary(request);

        let oldName = request.params.old_name,
        newName = request.params.new_name,
        id = items.indexOf(oldName);

        if(newName === null || newName === undefined) {
        response.send("<p>Missing 'name' parameter.</p>");
        return;
        }

        if(id === -1) {
        response.send(`<p>Item "${oldName}" doesn't exist.</p>`);
        return;
        }

        if(items.includes(newName)) {
        response.send(`<p>Name "${newName}" is taken.</p>`);
```

```
        return;
        }

        items[id] = newName;
        response.send(`<p>Item "${oldName}" changed name to "${newName}."</p>`);
});
```

Test nowych endpointów

```
# Dodajemy nowy element
$ curl -sX POST http://localhost:3000/item/laptop
<p>Item "laptop" added successfully</p>[

# Nie możemy dodać duplikatów.
$ curl -sX POST http://localhost:3000/item/laptop
<p>Item "laptop" already in collection</p>

# Wyświetlamy zwartość
$ curl -sX GET http://localhost:3000/item/
<p>Available items: laptop</p>

# Próbujemy zmodyfikować nieistniejący i dostajemy błąd.
$ curl -sX PUT http://localhost:3000/item/komputer/telefon
<p>Item "komputer" doesn't exist.</p>[

# Zmieniamy nazwę istniejącego.
$ curl -sX PUT http://localhost:3000/item/laptop/telefon
<p>Item "laptop" changed name to "telefon."</p>[

# Potwierdzamy, że faktycznie została zmieniona.
$ curl -sX GET http://localhost:3000/item/
<p>Available items: telefon</p>[
```

Etap 4: Katalog 04_HttpServer

GET na /

Zostaje zwrócony opis endpointów w API:

- pod jaką ścieżkę powinniśmy kierować zapytanie
- jakiej metody HTTP powinniśmy użyć
- jak przekazywane są parametry

```
$ curl -sX GET http://localhost:3000/
<h1>REST + Database</h1><ul>
        <li>Show all patients (GET /patient )</li>
        <li>Show specific patient (GET /patient/:id)</li>
        <li>Add new patient (POST /patient?name=:NAME&surname=:SURNAME)</li>
        <li>Modify existing patient (PUT
/patient/:id?name=:NAME&surname=:SURNAME)</li>
        <li>Remove patient (DELETE /patient/:id)</li></ul>[
```

Dodawanie pacjentów - POST /patient

Na początku dodajemy kilku pacjentów, aby mieć jakieś dane testowe.

Dodajemy pacjenta Rafał Gaweł

```
$ curl -sX POST http://localhost:3000/patient\?name=Rafal\&surname=Gawel
{"id":1,"name":"Rafal","surname":"Gawel"}[
```

Dodajemy pacjenta Jan Kowalski

```
$ curl -sX POST http://localhost:3000/patient\?name=Jan\&surname=Kowalski
{"id":2,"name":"Jan","surname":"Kowalski"}[
```

Sprawdzenie

```
$ curl -sX GET http://localhost:3000/patient
[{"id":1,"name":"Rafal","surname":"Gawel"},{"id":2,"name":"Jan","surname":"Kowalski"}][
```

Po podglądnięciu pliku db.json:

```
{
  "patients": [
        {
        "id": 1,
        "name": "Rafal",
```

```
        "surname": "Gawel"
        },
        {
        "id": 2,
        "name": "Jan",
        "surname": "Kowalski"
        }
 ]
}
```


Testowanie /patient/:id w zależności od metody HTTP

GET /patient/:id

Metoda GET pobiera informacje o pacjencie ze wskazanym identyfikatorem. Odbywa się to za pomocą funkcji getPatient. Kod odpowiedz HTTP to 200 OK lub 404 Not Found w zależności czy udalo się znaleźć pacjenta. Za tą logikę odpowiada następujący kod:

```
const patient = getPatient(id);
if (patient !== undefined) {
  response.status(200).send(JSON.stringify(patient));
} else {
  response.status(404).send({ error: "No patient with given id" });
}
```

Zwrócony zostaje JSON z informacją o konkretnym pacjencie.
Kod odpowiedzi 200

```
$ curl -siX GET http://localhost:3000/patient/2
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 42
ETag: W/"2a-P0OCauGZvbAiBbnJSTqlAGXfbDI"
Date: Thu, 23 Jan 2020 21:58:18 GMT
Connection: keep-alive
{"id":2,"name":"Jan","surname":"Kowalski"}[
```

Zwrócony zostanie błąd mówiący o tym, że brak pacjenta o id 3.
Kod odpowiedzi 404

```
$ curl -siX GET http://localhost:3000/patient/3
HTTP/1.1 404 Not Found
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 36
ETag: W/"24-289afur68CcXcCWSsu9/ylO8Xlc"
Date: Thu, 23 Jan 2020 21:59:12 GMT
```

Connection: keep-alive
{"error":"No patient with given id"}

DELETE /patient/:id

Najpierw sprawdzane jest czy pacjent z przekazanym identyfikatorem istnieje.

```
if (patient === undefined) {
  response.status(404).send({ error: "No patient with given id" });
} else { ... }
```

Jeśli nie to zwracana odpowiedź z odpowiednim komunikatem i kodem 404 Not Found. Jeżeli pacjent istnieje to wywoływana jest metoda na .remove({ id: id }), ktora skutuje usunięciem pacjentów z danym identyfikatorem.

```
db
.get("patients")
.remove({ id: id })
.write();
response.status(200).send({ message: "Patient removed successfully" });
```

Próba usunięcia niestniejącego da nam kod odpowiedzi 404
$ curl -siX DELETE http://localhost:3000/patient/3
HTTP/1.1 404 Not Found
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 36
ETag: W/"24-289afur68CcXcCWSsu9/yIO8Xlc"
Date: Thu, 23 Jan 2020 22:00:28 GMT
Connection: keep-alive
{"error":"No patient with given id"}[

Usuwamy pacjenta 2
$ curl -siX DELETE http://localhost:3000/patient/2
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 42
ETag: W/"2a-7pvd9tYJ+7Obbv+8ssaSkG5jOqg"
Date: Thu, 23 Jan 2020 22:00:58 GMT
Connection: keep-alive
{"message":"Patient removed successfully"}

PUT /patient/:id?name=_&surname=_

Zadaniem PUT jest zaktualizowanie informacji o istniejącym pacjencie. Więc najpierw to sprawdzamy

```
// Sprawdzenie czy pacjent istnieje.
const patient = getPatient(id);
if (patient === undefined) {
  response.status(404).send({ error: "No patient with given id" });

} else { ... }
```

Po znalezieniu pacjenta ze wskazanym ID sprawdzane jest czy podane zostały parametry name i surname.

```
const name = request.query.name;
const surname = request.query.surname;
if (name === undefined || surname === undefined) {
  response.status(400).send({
    error: "Invalid request - missing queries (name and/or surname)"
  });

} else { ... }
```

Jeżeli pacjent istnieje i parametry zostały podane, dochodzi do *zastąpienia* starego pacjenta nowym ze zaktualizowanymi danymi.

```
const updatedPatient = { id: patient.id, name: name, surname: surname };

db
.get("patients")
.find(patient)
.assign(updatedPatient)
.write();
response.status(200).send(updatedPatient);
```

Warstwa persystencji nadpisuje pacjenta za pomocą metody .assign().

Kiedy próbujemy aktualizować nieistniejącego

```
$  curl -siX PUT http://localhost:3000/patient/3
HTTP/1.1 404 Not Found
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 36
ETag: W/"24-289afur68CcXcCWSsu9/ylO8Xlc"
Date: Thu, 23 Jan 2020 22:04:22 GMT
Connection: keep-alive
```

{"error":"No patient with given id"}

Kiedy uda nam się w końcu zaktualizować

```
$ curl -siX PUT http://localhost:3000/patient/1\?name=Ferdynand\&surname=Nowak
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 45
ETag: W/"2d-Qw7VFidWL4nnXc3u8iBTF8QAq5M"
Date: Thu, 23 Jan 2020 22:05:13 GMT
Connection: keep-alive

{"id":1,"name":"Ferdynand","surname":"Nowak"}[
```