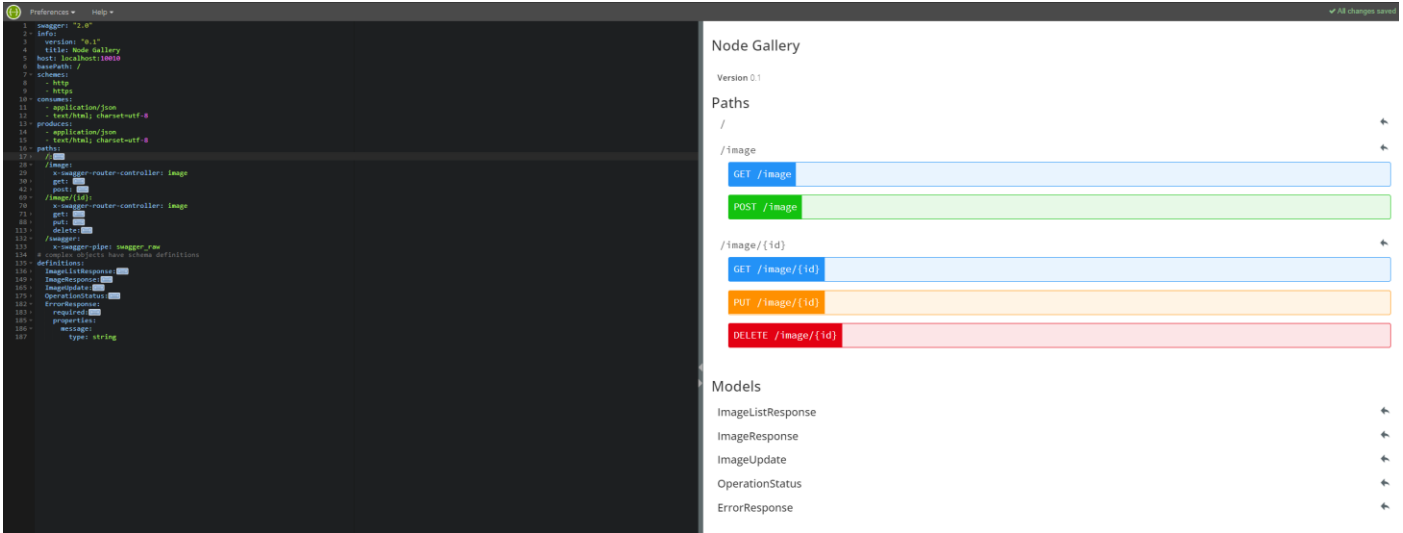


ZADANIE 5 Jakub Makaruk

1. Korzystając z nowej definicji pliku swagger.yml i edytora Swagger przetestuj definicję API dla wszystkich ścieżek i metod.



Wyniki testowanych usług za pomocą edytora Swagger:

- /

Paths

GET /

Responses

Code	Description	Schema
200	Success	string

Request

Scheme: http

Accept: text/html; charset=utf-8

GET http://localhost:10010/ HTTP/1.1

Host: localhost
Accept: text/html; charset=utf-8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,fa;q=0.6,sv;q=0.4
Cache-Control: no-cache
Connection: keep-alive
Origin: http://127.0.0.1:52128
Referer: http://127.0.0.1:52128/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36

This is a cross-origin call. Make sure the server at localhost:10010 accepts GET requests from 127.0.0.1:52128. Learn more

Send Request

Response

ERROR Internal Server Error

Rendered Pretty Raw

Headers

undefined

Body

W terminalu IDE prezentowany jest następujący komunikat z błędem:

```
Terminal: Local x Local (2) x + ↗
  at Runner.runBound (domain.js:314:12)
  at Runner.pipeline (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\pipeworks\pipeworks.js:72:17)
  at Runner.flow (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\pipeworks\pipeworks.js:223:19)
  at Pipeworks.flow (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\pipeworks\pipeworks.js:135:17)
  at Pipeworks.siphon (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\pipeworks\pipeworks.js:186:19)
  at Runner.<anonymous> (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\bagpipes\lib\bagpipes.js:98:22)
Error: Cannot resolve the configured swagger-router handler: homepage_homepage
  at swaggerRouter (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\middleware\swagger-router.js:414:18)
  at swagger_router (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-node-runner\typings\swagger_router.js:31:5)
  at Runner.<anonymous> (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\bagpipes\lib\bagpipes.js:171:7)
  at bound (domain.js:301:14)
  at Runner.runBound (domain.js:314:12)
  at Runner.pipeline (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\pipeworks\pipeworks.js:72:17)
  at Runner.flow (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\pipeworks\pipeworks.js:223:19)
  at Pipeworks.flow (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\pipeworks\pipeworks.js:135:17)
  at Pipeworks.siphon (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\pipeworks\pipeworks.js:186:19)
  at Runner.<anonymous> (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\bagpipes\lib\bagpipes.js:98:22)
```

Błąd spowodowany jest brakiem pliku w api/controllers o nazwie „homepage.js” oraz brakiem implementacji funkcji homepage().

- /image GET

/image

GET /image

Description

Get list of all images.

Responses

Code	Description	Schema
200	Success	<pre>ImageListResponse { images: >[] }</pre>
default	Error	<pre>ErrorResponse { message: string * }</pre>

Close

Request

Scheme

http

Accept

application/json

```
GET http://localhost:10010/image HTTP/1.1
Host: localhost
Accept: application/json
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,fa;q=0.6,sv;q=0.4
Cache-Control: no-cache
Connection: keep-alive
Origin: http://127.0.0.1:52128
Referer: http://127.0.0.1:52128/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
```

This is a cross-origin call. Make sure the server at localhost:10010 accepts GET requests from 127.0.0.1:52128 . Learn more

Send Request

Response

PARSERERROR SyntaxError: Unexpected end of JSON input

Rendered

Pretty

Raw

Headers

undefined

Body

Odpowiedź wynika z ciała funkcji, która jest wywoływana dla usługi GET /image, a konkretnie z braku parametru w funkcji json():

```
19 function listImages(req,res,next) {
20     res.json();
21 }
```

• /image POST

POST /image

Description

Add image to list with upload

Parameters

Name	Located in	Description	Required	Schema
title	formData	Image title.	No	string
description	formData	Image description.	No	string
upload	formData	The file to upload.	No	file

Responses

Code	Description	Schema
200	Success	<div>ImageResponse { id: string title: string description: string date: string path: string size: integer }</div>
default	Error	<div>ErrorResponse { message: string }</div>

Request

Scheme

http

Accept

application/json

Content-Type

multipart/form-data

Parameters

title

Zdjęcie 1

Image title.

description

Testowe zdjęcie JM

Image description.

upload

Wybierz plik

IMG_20190312_130824.jpg

The file to upload.

POST http://localhost:10010/image HTTP/1.1

Host: localhost
Accept: application/json
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,fa;q=0.6,sv;q=0.4
Cache-Control: no-cache
Connection: keep-alive
Origin: http://127.0.0.1:52128
Referer: http://127.0.0.1:52128/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Content-Length:
Content-Type: multipart/form-data

File

This is a cross-origin call. Make sure the server at localhost:10010 accepts POST requests from 127.0.0.1:52128. [Learn more](#)

Send Request

Response

ERROR Internal Server Error

```
Error: Response validation failed: failed schema validation
    at throwErrorWithCode (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\lib\validators.js:121:13)
    at Object.module.exports.validateAgainstSchema (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\lib\validators.js:176:7)
    at C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\middleware\swagger-validator.js:141:22
```

• /image/{id} GET

/image/{id}



GET /image/{id}

Description

Get image with selected id

Parameters

Name	Located in	Required	Schema
id	path	Yes	<code>string</code>

Responses

Code	Description	Schema
200	Success	<pre>ImageResponse { id: string title: string description: string date: string path: string size: integer }</pre>
default	Error	<pre>ErrorResponse { message: string }</pre>

Close

Request

Scheme
http

Accept
application/json

Parameters
id
0123456789abcd

GET http://localhost:10010/image/0123456789abcd HTTP/1.1
Host: localhost
Accept: application/json
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,fajq=0.6,sv;q=0.4
Cache-Control: no-cache
Connection: keep-alive
Origin: http://127.0.0.1:52128
Referer: http://127.0.0.1:52128/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36

⚠ This is a cross-origin call. Make sure the server at localhost:10010 accepts GET requests from 127.0.0.1:52128. [Learn more](#)

Send Request

Response

SUCCESS

Rendered Pretty Raw

Headers
Object
content-length: "153"
content-type: "application/json; charset=utf-8"

Body
Object
id: "0123456789abcd"
title: "testowy obrazek"
description: "Opis do obrazka"
date: "2017-11-09T10:20:00.214Z"
path: "/library/images/"
size: 1024

• /image/{id} PUT

PUT /image/{id}

Description

Update image with selected id.

Parameters

Name	Located in	Description	Required	Schema
id	path		Yes	string
image	body	Image properties.	Yes	<div>ImageUpdate { title: string description: string date: string</div>

Responses

Code	Description	Schema
200	Success	<div>ImageResponse { id: string title: string description: string date: string path: string size: integer</div>
default	Error	<div>ErrorResponse { message: string</div>

Request

Scheme

http

Accept

application/json

Content-Type

application/json

Parameters

id

0123456789abcd

ImageUpdate

Image properties.

title

Zmieniony tytuł obrazka

description

Zmieniony opis obrazka

date

2017-11-09T10:20:00.214Z

PUT http://localhost:10010/image/0123456789abcd HTTP/1.1

Host: localhost
Accept: application/json
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,fa;q=0.6,sv;q=0.4
Cache-Control: no-cache
Connection: keep-alive
Origin: http://127.0.0.1:52128/
Referer: http://127.0.0.1:52128/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Content-Length: 121
Content-Type: application/json

{
 "title": "Zmieniony tytuł obrazka",
 "description": "Zmieniony opis obrazka",
 "date": "2017-11-09T10:20:00.214Z"
}

This is a cross-origin call. Make sure the server at localhost:10010 accepts PUT requests from 127.0.0.1:52128. [Learn more](#)

Send Request

Response

ERROR Internal Server Error

Rendered

Pretty

Raw

Headers

Error: Response validation failed: failed schema validation

at throwErrorWithCode (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\lib\validators.js:121:13)

at Object.module.exports.validateAgainstSchema (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\lib\validators.js:176:7)

Error: Response validation failed: failed schema validation

at throwErrorWithCode (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\lib\validators.js:121:13)

at Object.module.exports.validateAgainstSchema (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\lib\validators.js:176:7)

• /image/{id} DELETE

DELETE /image/{id}

Description

Delete image with selected id.

Parameters

Name	Located in	Required	Schema
id	path	Yes	= string

Responses

Code	Description	Schema
200	Success	<div>▼</div> <div>OperationStatus { id: string status: string }</div>
default	Error	<div>▼</div> <div>ErrorResponse { message: string * }</div>

Request

Scheme

http

Accept

application/json

Parameters

id

0123456789abcd

Send Request

Response

ERROR Internal Server Error

Rendered

Pretty

Raw

Headers

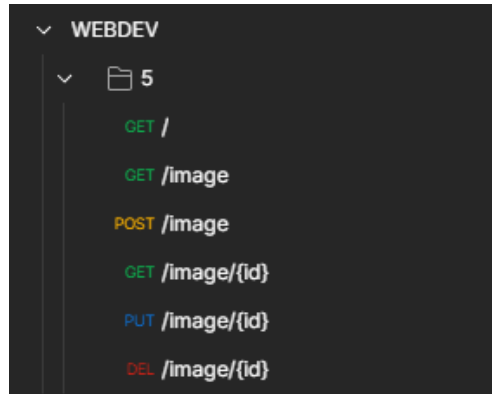
undefined

Body

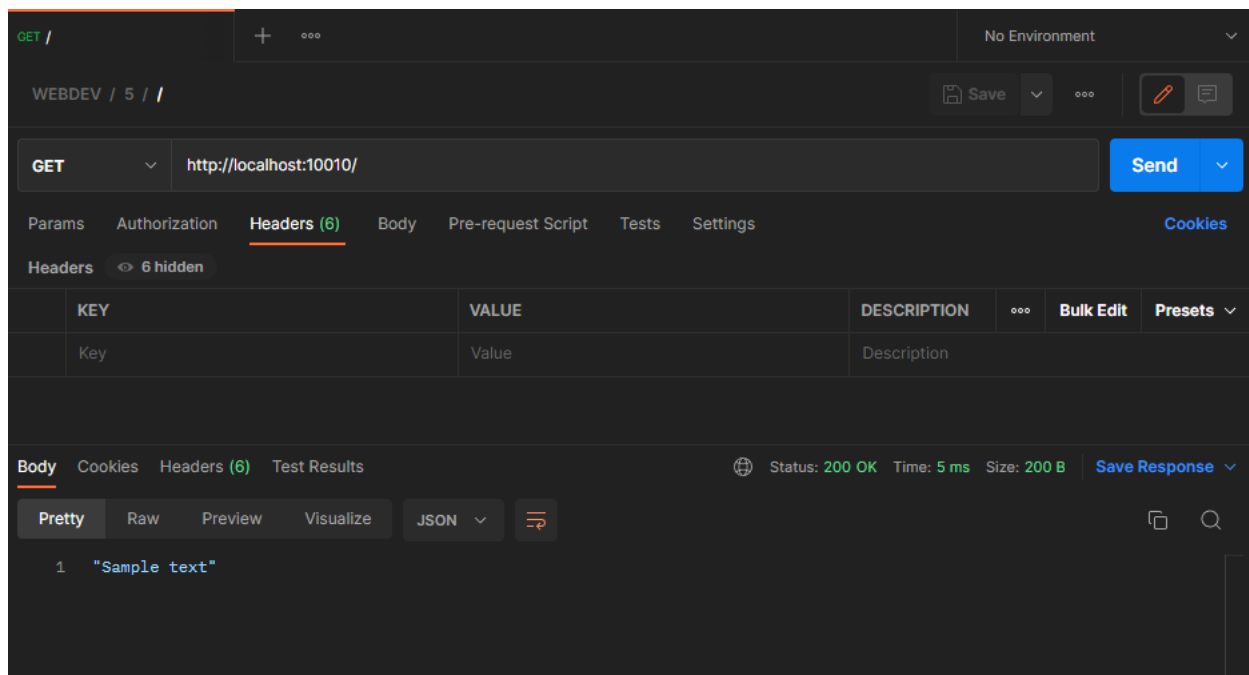
Error: Response validation failed: failed schema validation
at throwErrorWithCode (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\lib\validators.js:121:13)
at Object.module.exports.validateAgainstSchema (C:\Users\Jakub\Desktop\studia\node_gallery\node_modules\swagger-tools\lib\validators.js:176:7)

2. Uruchom Galerię w trybie mockup(przełącznik -m) i korzystając z Postmana zrób analogiczne testy dla serwisu. Które dodatkowo pola muszą być w Postmanie ustawione, żeby testy wykonały się poprawnie.

```
PS C:\Users\Jakub\Desktop\studia\node_gallery> swagger project start -m
Starting: C:\Users\Jakub\Desktop\studia\node_gallery\app.js...
project started here: http://localhost:10010/
project will restart on changes.
to restart at any time, enter `rs`
```



• /



- **/image GET**

GET /image

WEBDEV / 5 / /image

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 3 ms Size: 263 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "images": [
3      {
4        "id": "Sample text",
5        "title": "Sample text",
6        "path": "Sample text"
7      }
8    ]
9  }
```

- **/image POST (Musi być ustawione pole Content-Type w nagłówkach)**

POST /image

WEBDEV / 5 / /image

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Headers 7 hidden

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/> Content-Type	application/json				
Key	Value	Description			

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 11 ms Size: 322 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "id": "Sample text",
3    "title": "Sample text",
4    "description": "Sample text",
5    "date": "2022-12-29T14:15:17.106Z",
6    "path": "Sample text",
7    "size": 1
8  }
```


- **/image/{id} GET**

The screenshot shows a REST client interface with the following details:

- Request:** Method: GET, URL: http://localhost:10010/image/1234
- Params:** Query Params table with columns: KEY, VALUE, DESCRIPTION. A single row is present with Key, Value, and Description.
- Body:** Pretty view of a JSON response:

```
1 {
2   "id": "Sample text",
3   "title": "Sample text",
4   "description": "Sample text",
5   "date": "2022-12-29T14:16:50.600Z",
6   "path": "Sample text",
7   "size": 1
8 }
```
- Status:** 200 OK, Time: 4 ms, Size: 322 B

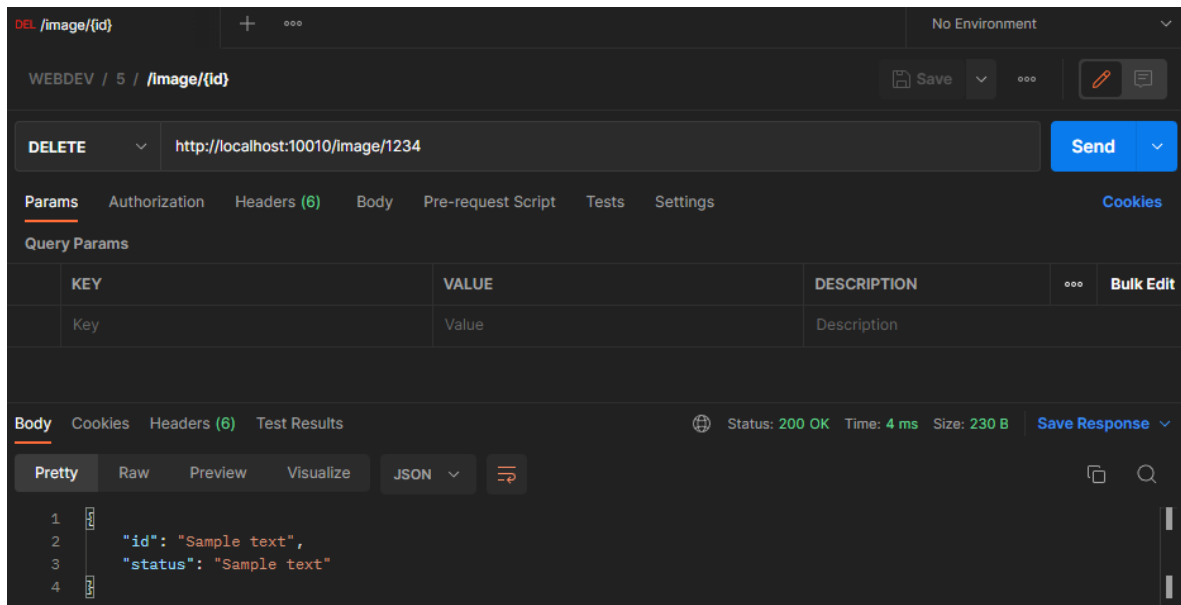
- **/image/{id} PUT (Musi być ustawione pole Content-Type w nagłówkach)**

The screenshot shows a REST client interface with the following details:

- Request:** Method: PUT, URL: http://localhost:10010/image/1234
- Headers:** Headers (8) section with 7 hidden. A table shows a checked checkbox for 'Content-Type' with the value 'application/json'.
- Body:** Pretty view of a JSON response:

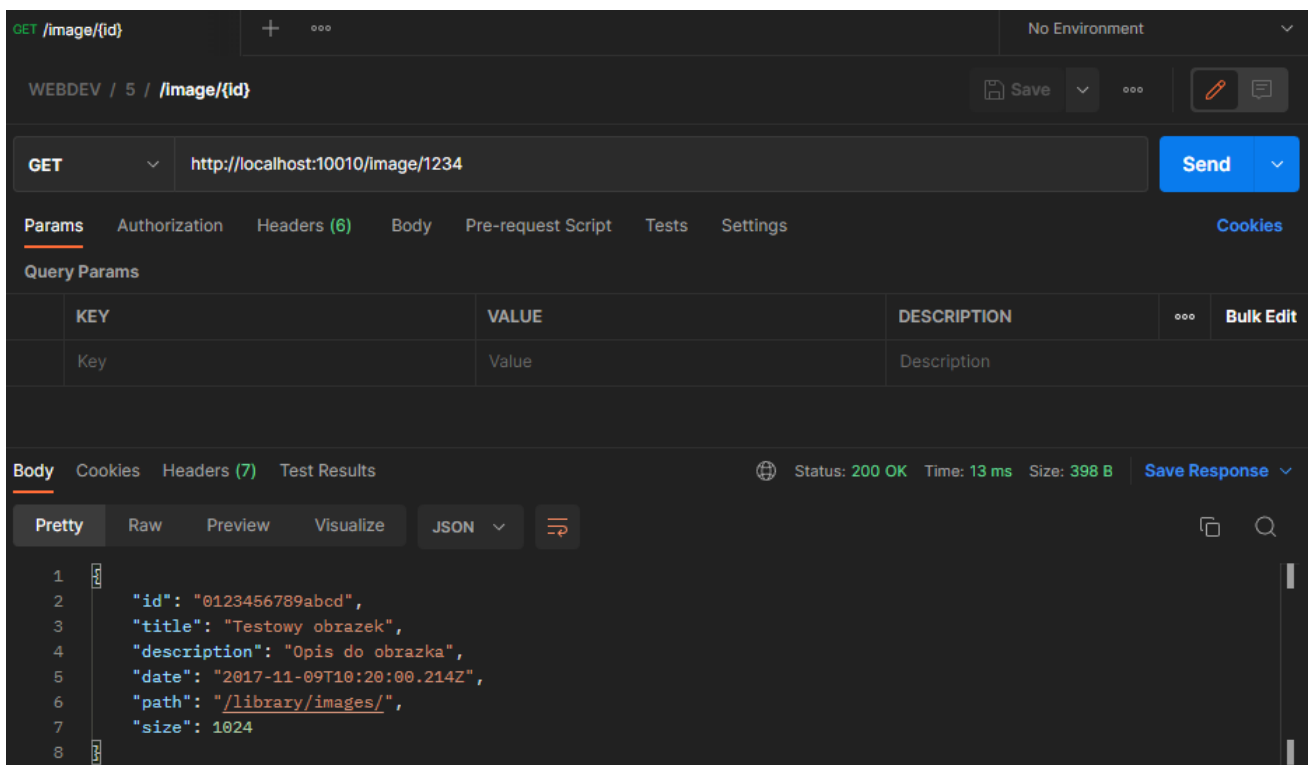
```
1 {
2   "id": "Sample text",
3   "title": "Sample text",
4   "description": "Sample text",
5   "date": "2022-12-29T14:17:51.533Z",
6   "path": "Sample text",
7   "size": 1
8 }
```
- Status:** 200 OK, Time: 4 ms, Size: 322 B

- **/image/{id} DELETE**



3. Zatrzymaj serwis, po czym uruchom go w trybie normalnym (bez przełącznika -m). Sprawdź co tym razem zwraca metoda GET dla ścieżki /image/1234. Czy poprawnie działają pozostałe metody?

Metoda zwraca inne wartości (zgodnie z implementacją funkcji *readImage()* zwraca obiekt testData):



Pozostałe metody GET zwracają status 200 (działają poprawnie). Natomiast operacje POST, PUT, DELETE wymagają podania w *body* schematu danych.

4. Dodaj ciało do funkcji *listImages*, *createImage*, *updateImage* i *deleteImage* tak aby zwracało poprawne odpowiedzi na zapytania Postmana. Skup się na odpowiedziach i ich formacie (patrz definicje z sekcji definitions Swaggera).

```
'use strict';

module.exports = {
  listImages,
  createImage,
  readImage,
  updateImage,
  deleteImage
};

var testData = {
  id:"0123456789abcd",
  title:"Testowy obrazek",
  description:"Opis do obrazka",
  date:"2017-11-09T10:20:00.214Z",
  path:"/library/images/",
  size:1024
};

let images = [{
  id:"0123456789abcd",
  title:"Testowy obrazek",
  description:"Opis do obrazka",
  date:"2017-11-09T10:20:00.214Z",
  path:"/library/images/",
  size:1024
}];

function listImages(req,res,next) {
  res.json(images);
}

function createImage(req,res,next) {
  images.push(req.body);
  res.json(req.body);
}

function readImage(req,res,next) {
  res.json(testData);
}

function updateImage(req,res,next) {
  res.json({})
}

function deleteImage(req,res,next) {
  const imageToRemoveIndex = images.findIndex((image) => image.id ===
req.swagger.params.id.value);
  const result = imageToRemoveIndex !== -1 ? images[imageToRemoveIndex] : {};

  if (imageToRemoveIndex !== -1) {
    images.splice(imageToRemoveIndex, 1);
  }

  res.json(result);
}
```

5. Spróbuj odczytać parametry przekazane do funkcji `updateImage` i w oparciu o nie dokonać modyfikacji obiektu `testData`.

```
function updateImage(req, res, next) {
  const imageToUpdateIndex = images.findIndex((image) => image.id ===
req.swagger.params.id.value);

  if (imageToUpdateIndex !== -1) {
    images[imageToUpdateIndex].title = req.body.title;
    images[imageToUpdateIndex].description = req.body.description;
    images[imageToUpdateIndex].date = req.body.date;
  }

  res.json(images[imageToUpdateIndex]);
}
```

6. Przygotuj zmienne, które będą przechowywać wartości parametrów dla pozostałych funkcji.

- `/image` POST

```
{
  "title": "Nowy testowy obrazek 1",
  "description": "Nowy opis do nowego obrazka",
  "upfile": "/library/images/image1"
}
```

- `/image/{id}` PUT

```
{
  "title": "Nowy tytuł dla obrazka",
  "description": "Nowy opis dla obrazka",
  "date": "2017-11-09T10:20:00.214Z"
}
```