PRZYGOTOWANIE KONTROLERA DLA SWAGGERA

Plik opisujący część związaną z obsługą obrazów we wstępnej wersji API galerii internetowej wygląda następująco:

```
swagger: "2.0"
info:
  version: "0.1"
  title: Node Gallery
host: localhost:10010
basePath: /
schemes:
  - http
  - https
consumes:
  - application/json
  - text/html; charset=utf-8
produces:
  - application/json
  - text/html; charset=utf-8
paths:
  /:
    x-swagger-router-controller: homepage
    get:
      operationId: homepage
      produces:
        - text/html; charset=utf-8
      responses:
        200:
          description: Success
          schema:
            type: string
  /image:
    x-swagger-router-controller: image
    get:
      operationId: listImages
      description: Get list of all images.
      responses:
        "200":
          description: Success
          schema:
            $ref: "#/definitions/ImageListResponse"
        default:
          description: Error
          schema:
            $ref: "#/definitions/ErrorResponse"
    post:
      operationId: createImage
      description: Add image to list with upload
      consumes:
        - multipart/form-data
      parameters:
        - name: title
          description: Image title.
          type: string
          in: formData
        - name: description
          description: Image description.
          type: string
          in: formData
```

```
      - name: upfile
        description: The file to upload.
        in: formData
        type: file
    responses:
      "200":
        description: Success
        schema:
          $ref: "#/definitions/ImageResponse"
      default:
        description: Error
        schema:
          $ref: "#/definitions/ErrorResponse"
/image/{id}:
  x-swagger-router-controller: image
  get:
    operationId: readImage
    description: Get image with selected id
    parameters:
      - name: id
        type: string
        in: path
        required: true
    responses:
      "200":
        description: Success
        schema:
          $ref: "#/definitions/ImageResponse"
      default:
        description: Error
        schema:
          $ref: "#/definitions/ErrorResponse"
  put:
    operationId: updateImage
    description: Update image with selected id.
    consumes:
      - application/json
    parameters:
      - name: id
        type: string
        in: path
        required: true
      - name: image
        description: Image properties.
        in: body
        required: true
        schema:
          $ref: "#/definitions/ImageUpdate"
    responses:
      "200":
        description: Success
        schema:
          $ref: "#/definitions/ImageResponse"
      default:
        description: Error
        schema:
          $ref: "#/definitions/ErrorResponse"
  delete:
    operationId: deleteImage
    description: Delete image with selected id.
    consumes:
```

```
        - application/json
      parameters:
        - name: id
          type: string
          in: path
          required: true
      responses:
        "200":
          description: Success
          schema:
            $ref: "#/definitions/OperationStatus"
        default:
          description: Error
          schema:
            $ref: "#/definitions/ErrorResponse"

  /swagger:
    x-swagger-pipe: swagger_raw
definitions:
  ImageListResponse:
    properties:
      images:
        type: array
        items:
          type: object
          properties:
            id:
              type: string
            title:
              type: string
            path:
              type: string
  ImageResponse:
      type: object
      properties:
        id:
          type: string
        title:
          type: string
        description:
          type: string
        date:
          type: string
          format: date-time
        path:
          type: string
        size:
          type: integer
  ImageUpdate:
      type: object
      properties:
        title:
          type: string
        description:
          type: string
        date:
          type: string
          format: date-time
  OperationStatus:
      type: object
      properties:
```

```
        id:
          type: string
        status:
          type: string
  ErrorResponse:
    required:
      - message
    properties:
      message:
        type: string
```

Zdefiniowane zostały w nim trzy ścieżki:

- / - prowadząca do strony głównej serwisu
- /image – obsługująca dwie metody, gdzie GET pozwala na pobranie listy obrazów, a POST na dodanie nowego obrazu
- /image/{id} – pozwalająca w oparciu o metodę GET na pobranie szczegółów pliku o wskazanym id, UPDATE na jego aktualizację i DELETE na jego usunięcie.

Element **`x-swagger-router-controller`** wskazuje na używany dla wskazanego routingu kontroler, który umieszczony jest w folderze *api/controllers*. To w nim znajdą się wszystkie funkcje obsługi API naszej aplikacji. Kojarzenie obsługi jest wykonywane w oparciu o element **`operationId`** i obiekt *module.exports* zdefiniowany w pliku kontrolera. Dla prezentowanego przykładu wyglądał on będzie następująco:

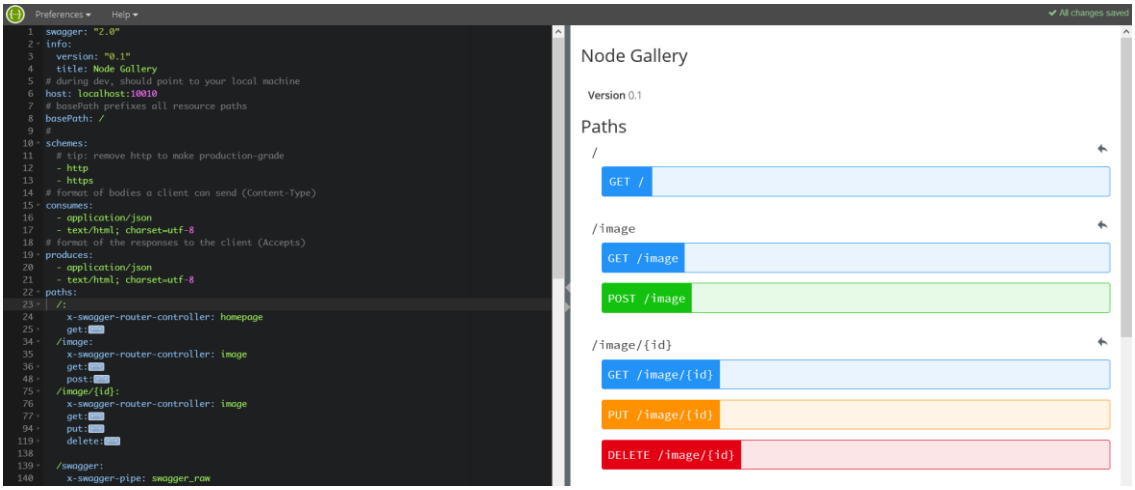
```
module.exports = {
  listImages,
  createImage,
  readImage,
  updateImage,
  deleteImage
};
```

Niezależnie od wykorzystanej metody wszystkie wartości do funkcji są przekazywane za pośrednictwem obiektu *req* i możliwe jest ich wyjęcie stamtąd. W przypadku korzystania z *swagger-express-mw*, tak jak to ma w naszym przypadku, musimy to zrobić uwzględniając obsługę Swaggera za pomocą: *req.swagger.params.name.value*, gdzie składowa *name* to nazwa przekazywanego pola (przykładowo może to być *id*).

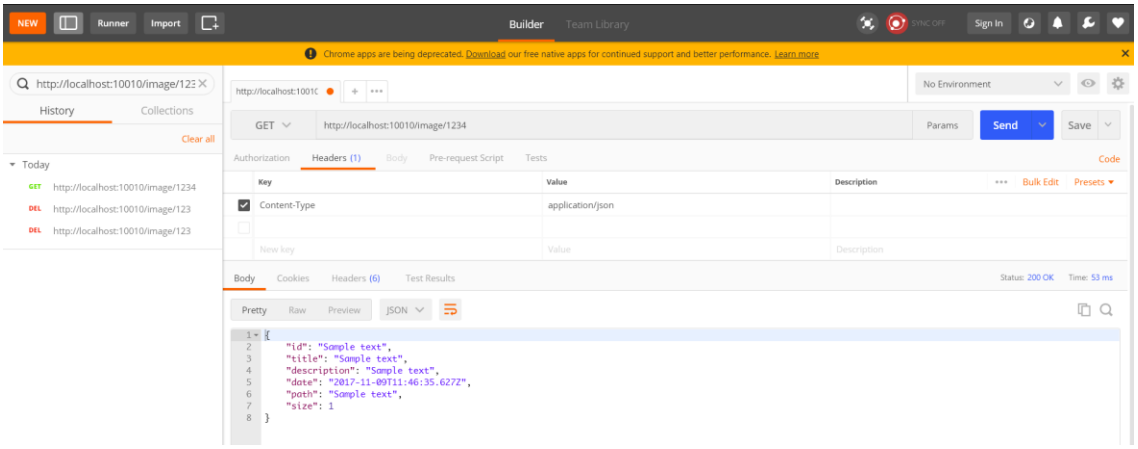
W celach testowych w pliku kontrolera została też umieszczona definicja obiektu Image (*testData*), która w sposób bezpośredni jest zwracana poprzez funkcję *testImage*.

ZADANIA

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



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.



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?
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).
5. Spróbuj odczytać parametry przekazane do funkcji *updateImage* i w oparciu o nie dokonać modyfikacji obiektu *testData.*
6. Przygotuj zmienne, które będą przechowywać wartości parametrów dla pozostałych funkcji.

STARTOWA POSTAĆ PLIKU IMAGE.JS

```
'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
};

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

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

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

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

function deleteImage(req, res, next) {
  res.json();
}
```