

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 2**

з дисципліни “Основи web-програмування”

тема “JSON HTTP API веб-сервер”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) 2 курсу  групи КП-91  Климчук Нікіта Олегович  (*прізвище, ім’я, по батькові*)  варіант № 9 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

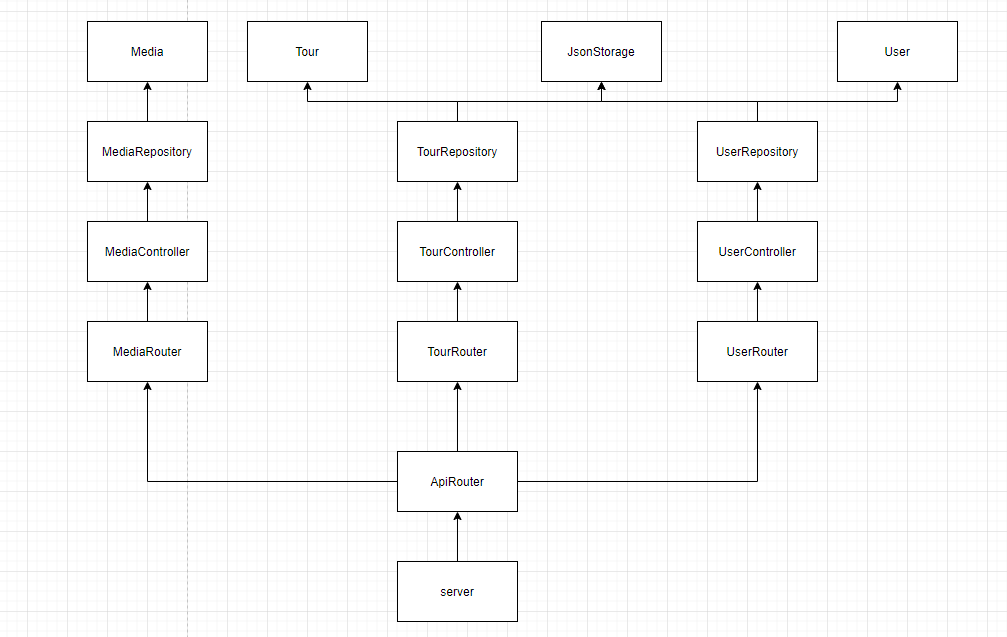
**Мета роботи**

Навчитись створювати простий JSON HTTP API веб-сервер, що надає доступ до ресурсів зі сховища.

### **Завдання**

Реалізувати JSON HTTP API веб-сервер зі сторінкою документації по цьому API.  
API має дозволяти отримувати дані про користувачів а також забезпечувати CRUD операції над сутностями за варіантом.  
Додатково реалізувати URI для завантаження та перегляду зображень.

**Діаграма залежності модулів**

****

**Тексти коду програм**

|  |
| --- |
| **Media.js** |
| class Media {  constructor(path) {  this.path = path;  }  }  module.exports = Media; |

|  |
| --- |
| **MediaRepository.js** |
| const fs = require('fs');  const path = require('path');  const Media = require('./../Models/Media');  class MediaRepository {  constructor(storage) {  this.storage = storage;  }  AddMedia(media) {  const file = JSON.parse(fs.readFileSync(`${this.storage}/NextId.json`));  const storagepath = path.join(this.storage, `${file.id}`)  fs.renameSync(media.path, storagepath);  file.id += 1;  fs.writeFileSync(`${this.storage}/NextId.json`, JSON.stringify(file));  }  GetMediaPathById(id) {  const filepath = path.join(this.storage, id);  if (fs.existsSync(filepath)) {  return new Media(filepath);  }  return null;  }  }  module.exports = MediaRepository; |

|  |
| --- |
| **MediaController.js** |
| const MediaRepository = require('./../Repositories/MediaRepository');  const Media = require('../Models/Media');  const mediaRepo = new MediaRepository('./Data/Media');  module.exports = {  getMediaById(req, res) {  const media = mediaRepo.GetMediaPathById(req.params.id);  if (media != null) {  res.download(media.path);  res.send(media);  res.end();  }  else {  res.sendStatus(404);  }      },  addMedia(req, res) {  try {  mediaRepo.AddMedia(new Media(req.file.path));  res.status(201);  res.send(req.files);  res.end();  }  catch{  res.sendStatus(400);  }  }  }; |

|  |
| --- |
| **TourController.js** |
| const TourRepository = require('./../Repositories/TourRepository');  const tourRepo = new TourRepository('./Data/Tours.json');  const tourProperty = Symbol('tour');  module.exports = {  GetTours(req, res) {  const page = req.query.page;  const per\_page = req.query.per\_page;  if (page && per\_page) {  res.send(tourRepo.GetTours().slice((page - 1) \* per\_page, page \* per\_page));  }  else {  res.send(tourRepo.GetTours());  }  res.end();  },  GetTourById(req, res, next) {  res.send(req[tourProperty]);  res.end();  },  DeleteTourById(req, res, next) {  tourRepo.DeleteTour(req[tourProperty].id);  res.send(req[tourProperty]);  },  AddTour(req, res, next) {  const id = tourRepo.AddTour(req.body);  res.status(201);  res.send(tourRepo.GetTourById(id));  res.end();  },  UpdateTour(req, res, next) {  const tour = tourRepo.GetTourById(req.body.id);  if (tour) {  tourRepo.UpdateTour(req.body);  res.send(tourRepo.GetTourById(req.body.id));  }  else {  res.sendStatus(404);  }  res.end();      },  GetTourHandler(req, res, next) {  const tour = tourRepo.GetTourById(parseInt(req.params.id));  if (tour) {  req[tourProperty] = tour;  next();  }  else {  res.sendStatus(404);  }  }  }; |

|  |
| --- |
| **UserController.js** |
| const UserRepositpory = require('./../Repositories/UserRepository')  const userRepo = new UserRepositpory('./Data/Users.json');  module.exports = {  getUsers(req, res) {  const page = req.query.page;  const per\_page = req.query.per\_page;  if (page && per\_page) {  res.send(userRepo.getUsers().slice((page - 1) \* per\_page, page \* per\_page));  }  else {  res.send(userRepo.getUsers());  }  res.end();  },  getUserById(req, res) {  const user = userRepo.getUserById(req.params.id);  if (user) {  res.send(user);  }  else {  res.sendStatus(404);  }  res.end();  },  }; |

|  |
| --- |
| **MediaRouter.js** |
| const express = require('express');  const mediaController = require('./../Controllers/MediaController');  const Router = express.Router();  const multer = require('multer');  const upload = multer({ dest: 'uploads/' });  /\*\*  \* return media by id  \* @route GET /api/media/{id}  \* @group Media - upload and get images  \* @param {integer} id.path.required - id of the media - eg: 1  \* @returns 200 - media object  \* @returns {Error} 404 - Media not found  \*/  Router.get('/:id', mediaController.getMediaById)  /\*\*  \* return media file  \* @route POST /api/media  \* @group Media - upload and get images  \* @consumes multipart/form-data  \* @param {file} image.formData.required - uploaded image  \* @returns {integer} 200 - added image id  \*/  .post('/', upload.single('image'), mediaController.addMedia);  module.exports = Router; |

|  |
| --- |
| **TourRouter.js** |
| const express = require('express');  const tourController = require('./../Controllers/TourController');  const Router = express.Router();  /\*\*  \* returns all tours  \* @route GET /api/tours/  \* @group Tours - tour operations  \* @param {integer} page.query - page number  \* @param {integer} per\_page.query - items per page  \* @returns {Array.<Tour>} Tours - all tours  \*/  Router.get('/', tourController.GetTours)  /\*\*  \* return tour by id  \* @route GET /api/tours/{id}  \* @group Tours - tour operations  \* @param {integer} id.path.required - id of the Tour - eg: 1  \* @returns {Tour.model} 200 - Tour object  \* @returns {Error} 404 - Tour not found  \*/  .get('/:id', tourController.GetTourHandler, tourController.GetTourById)  /\*\*  \* add tour  \* @route POST /api/tours/  \* @group Tours - tour operations  \* @param {Tour.model} id.body.required - new Tour object  \* @returns {Tour.model} 201 - added Tour object  \*/  .post('/', tourController.AddTour)  /\*\*  \* update tour  \* @route PUT /api/tours/  \* @group Tours - tour operations  \* @param {Tour.model} id.body.required - new Tour object  \* @returns {Tour.model} 200 - changed Tour object  \*/  .put('/', tourController.UpdateTour)  /\*\*  \* delete tour  \* @route DELETE /api/tours/{id}  \* @group Tours - tour operations  \* @param {integer} id.path.required - id of the Tour - eg: 1  \* @returns {Tour.model} 200 - deleted Tour object  \* @returns {Error} 404 - Tour not found  \*/  .delete('/:id', tourController.GetTourHandler, tourController.DeleteTourById);  module.exports = Router;  if (numMaxTouristsCount != NaN) {  tour.maxTouristsCount = numMaxTouristsCount;  const startDate = readlineSync.question('Enter tour startDate: ');  try {  const dateStartDate = new Date(startDate);  const strStartDate = dateStartDate.toISOString();  tour.startDate = strStartDate;  tourRepo.AddTour(tour);  console.log('the value is added');  }  catch (err) {  console.log('incorrect startDate format');  console.log('nothing changed...');  }  }  else {  console.log('incorrect maxTouristsCount format');  console.log('nothing changed...');  }  }  else {  console.log('incorrect price format');  console.log('nothing changed...');  }  }  module.exports = commandsHandler; |

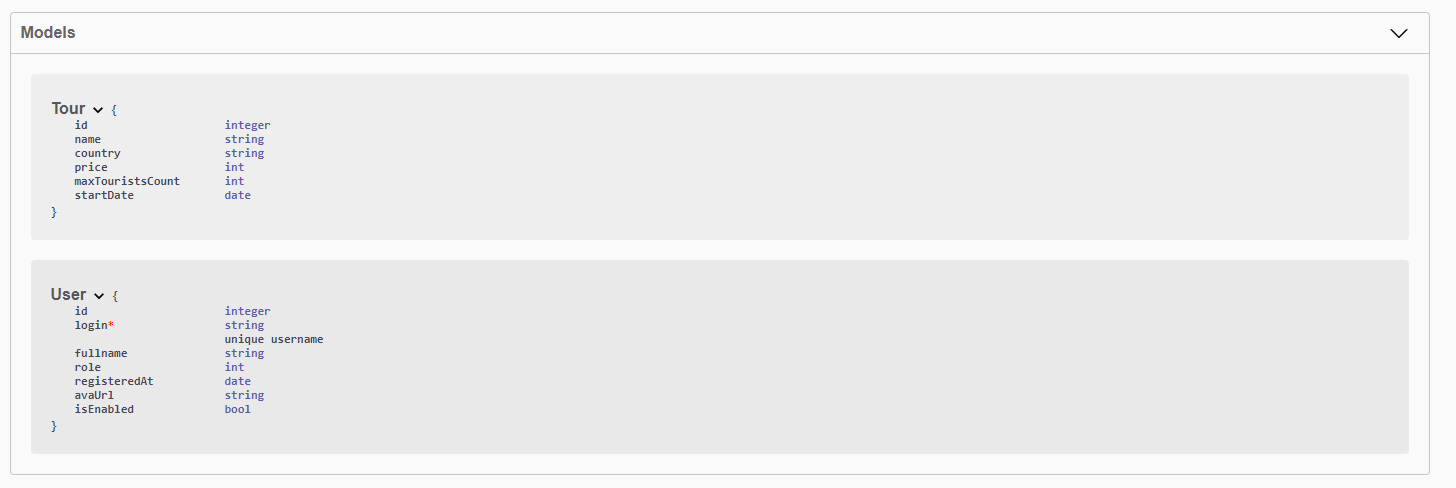
|  |
| --- |
| **UserRouter.js** |
| const express = require('express');  const userController = require('./../Controllers/UserController');  const User = require('../Models/User');  const Router = express.Router();  /\*\*  \* returns all users  \* @route GET /api/users/  \* @group Users - user operations  \* @param {integer} page.query - page number  \* @param {integer} per\_page.query - items per page  \* @returns {Array.<User>} User - all users  \*/  Router.get('/', userController.getUsers)  /\*\*  \* return user by id  \* @route GET /api/users/{id}  \* @group Users - user operations  \* @param {integer} id.path.required - id of the User - eg: 1  \* @returns {User.model} 200 - User object  \* @returns {Error} 404 - User not found  \*/  .get('/:id', userController.getUserById);  module.exports = Router; |

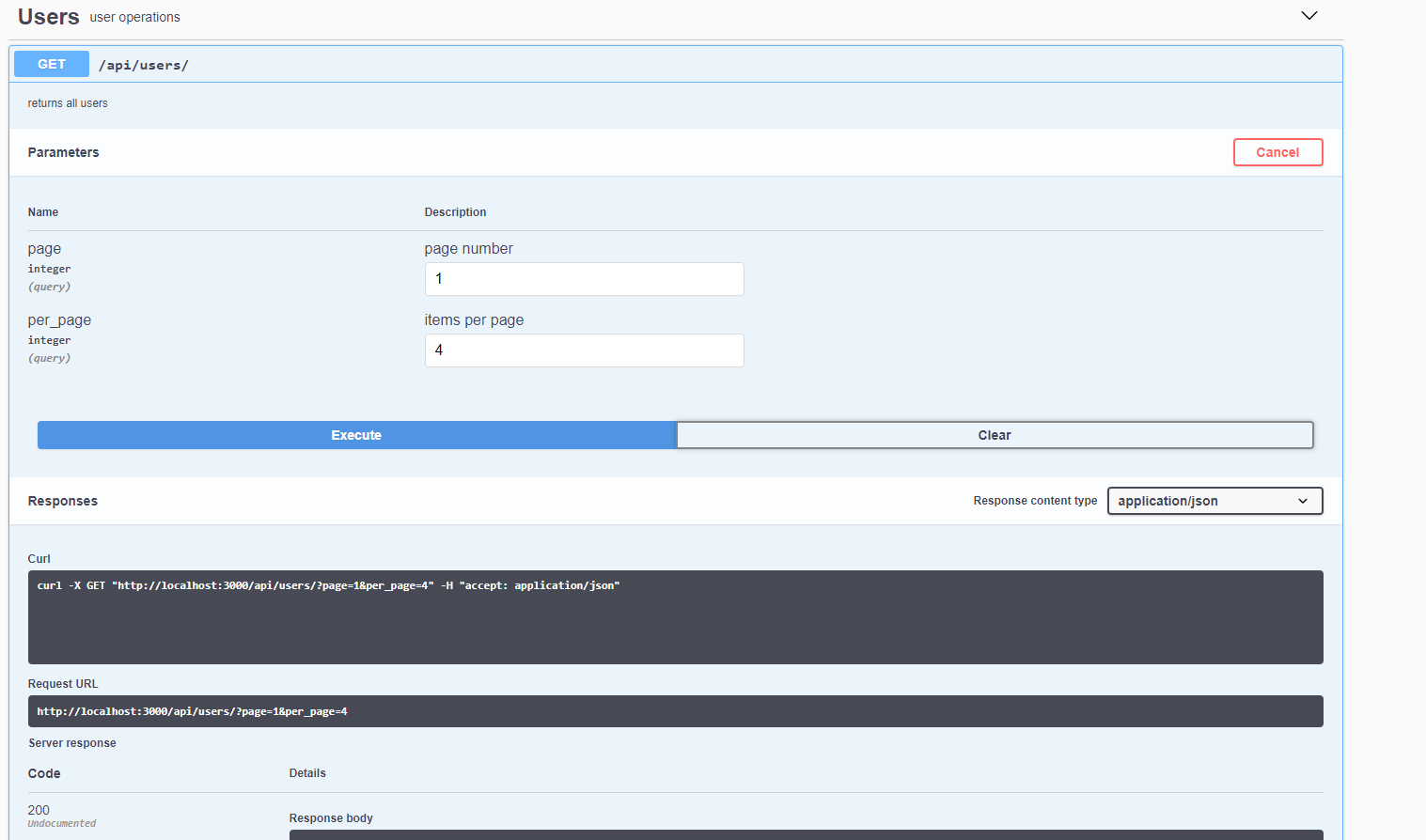
**Вміст json файлів**

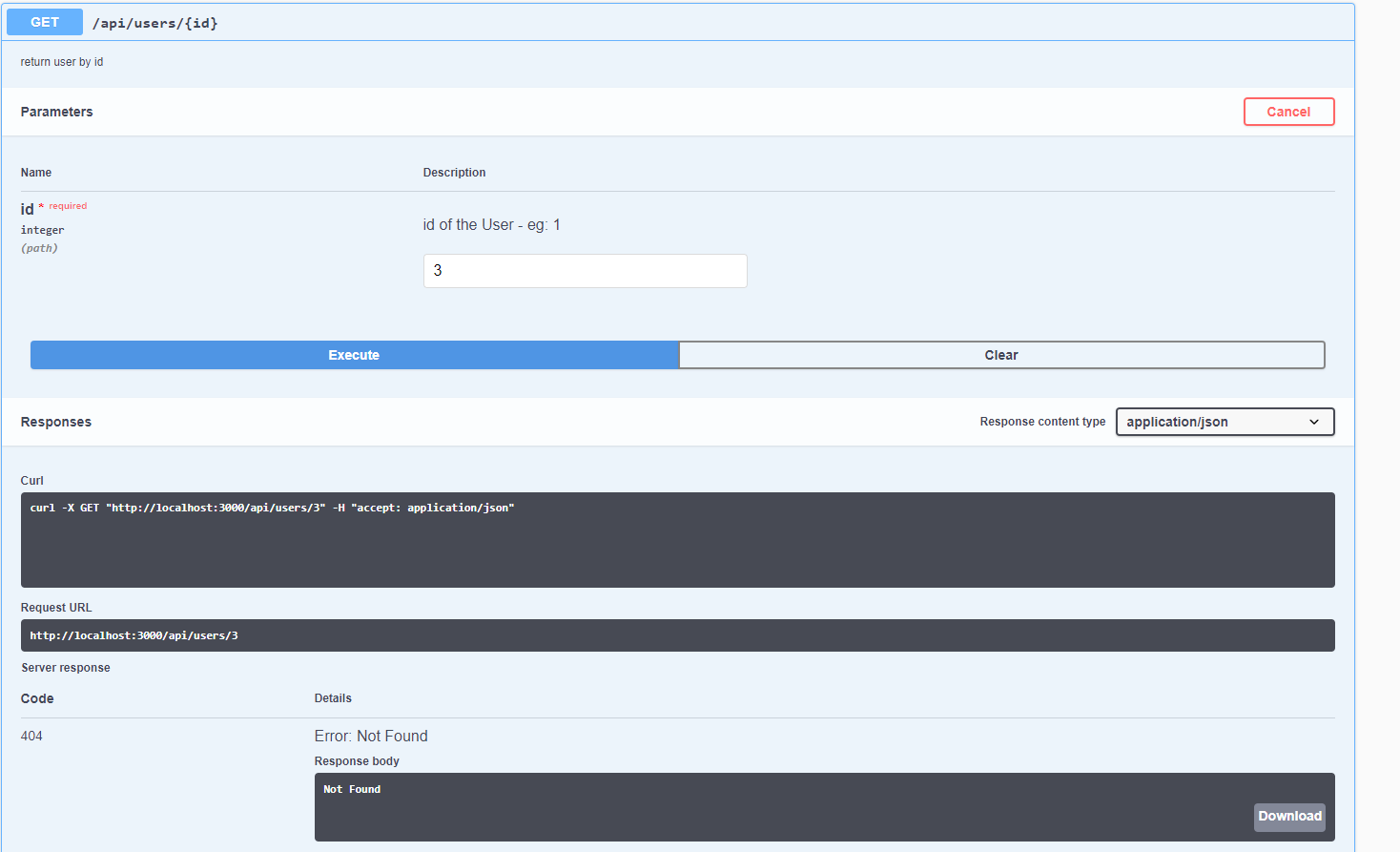
|  |
| --- |
| **ApiRouter.js** |
| const express = require('express');  const Router = express.Router();  const userRouter = require('./UserRouter');  const tourRouter = require('./TourRouter');  const mediaRouter = require('./MediaRouter');  Router.use('/users', userRouter);  Router.use('/tours', tourRouter);  Router.use('/media', mediaRouter);  module.exports = Router; |

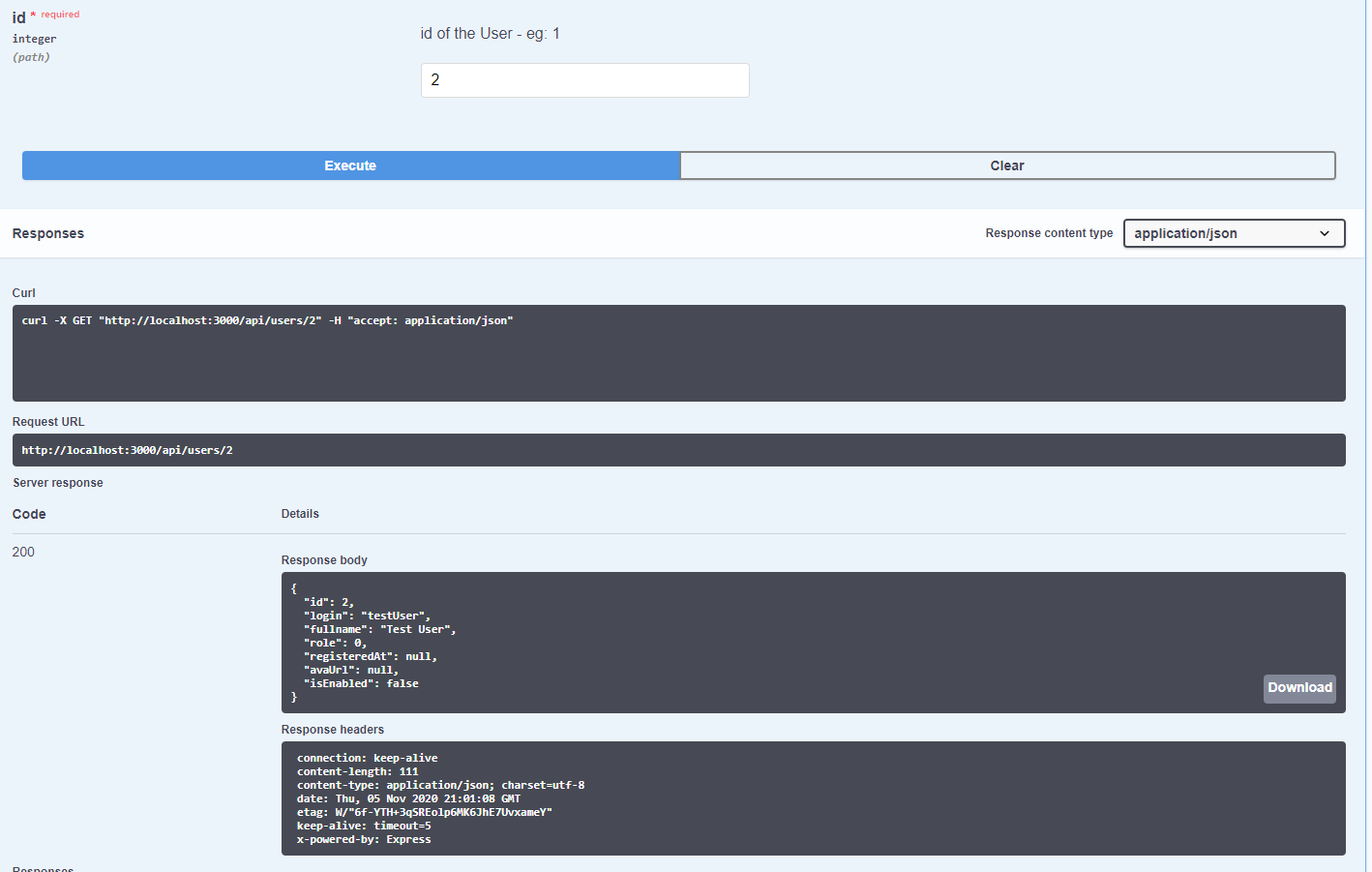
|  |
| --- |
| **Server.js** |
| 'use strict';  const bodyParser = require('body-parser');  const express = require('express');  const app = express();  const apiRouter = require('./Routes/ApiRouter');  app.use(bodyParser.json());  app.use(bodyParser.urlencoded({ extended: false }));  app.use('/api', apiRouter);  const expressSwaggerGenerator = require('express-swagger-generator');  const expressSwagger = expressSwaggerGenerator(app);  const options = {  swaggerDefinition: {  info: {  description: 'description',  title: 'title',  version: '1.0.0',  },  host: 'localhost:3000',  produces: ["application/json"],  },  basedir: \_\_dirname,  files: ['./routes/\*\*/\*.js', './models/\*\*/\*.js'],  };  expressSwagger(options);  app.listen(3000, () => {  }); |

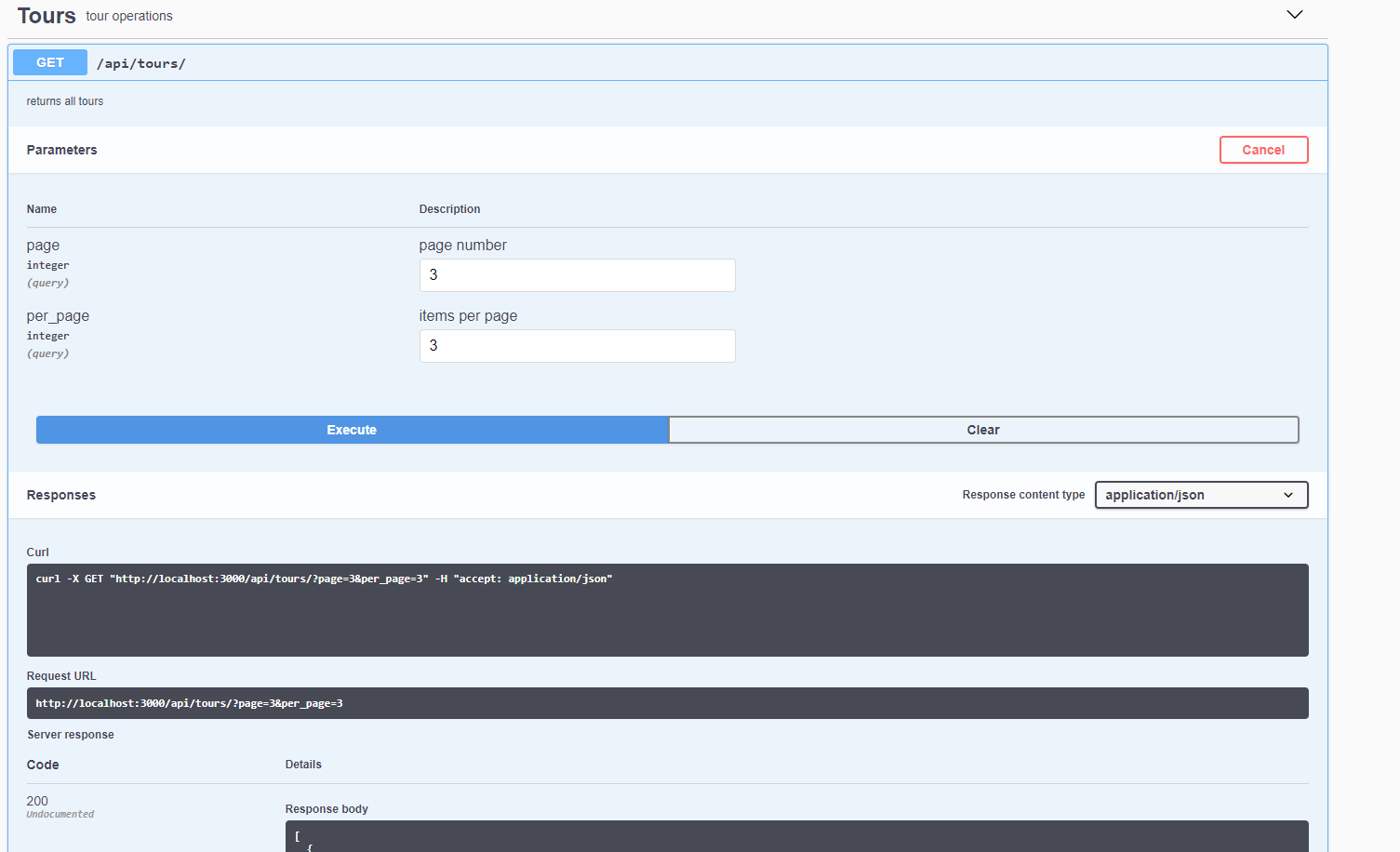
**Вивід роботи програми**

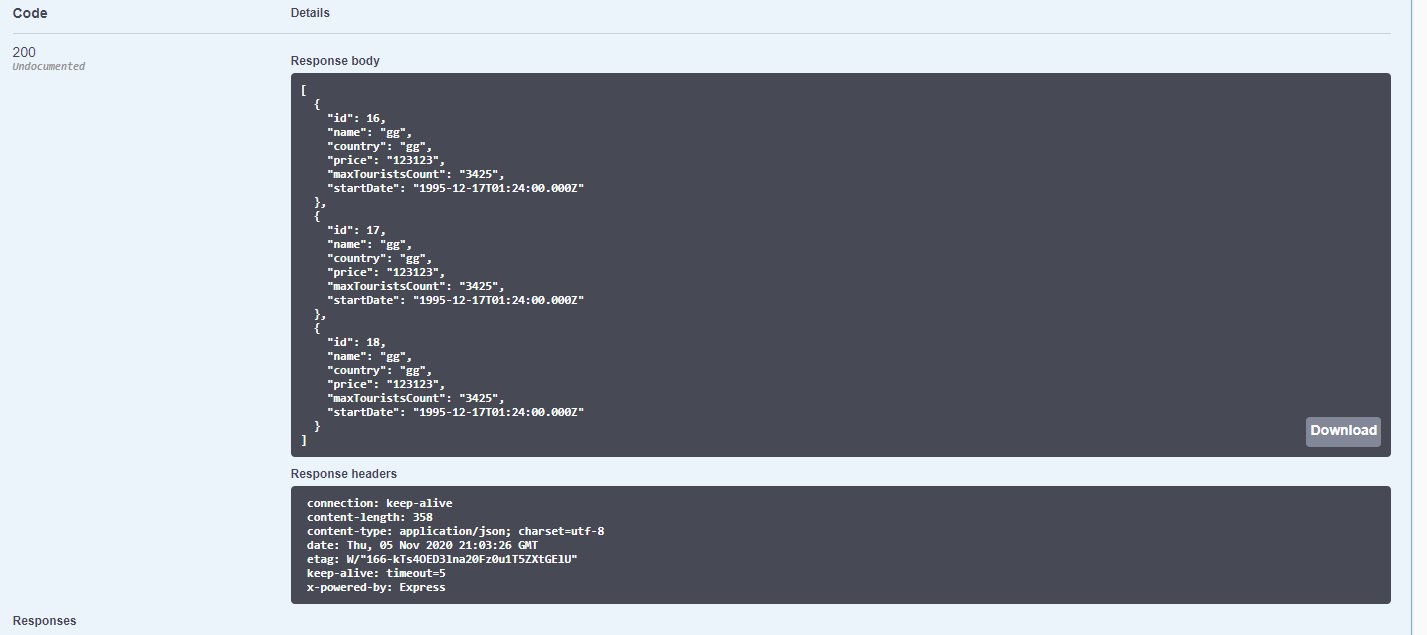


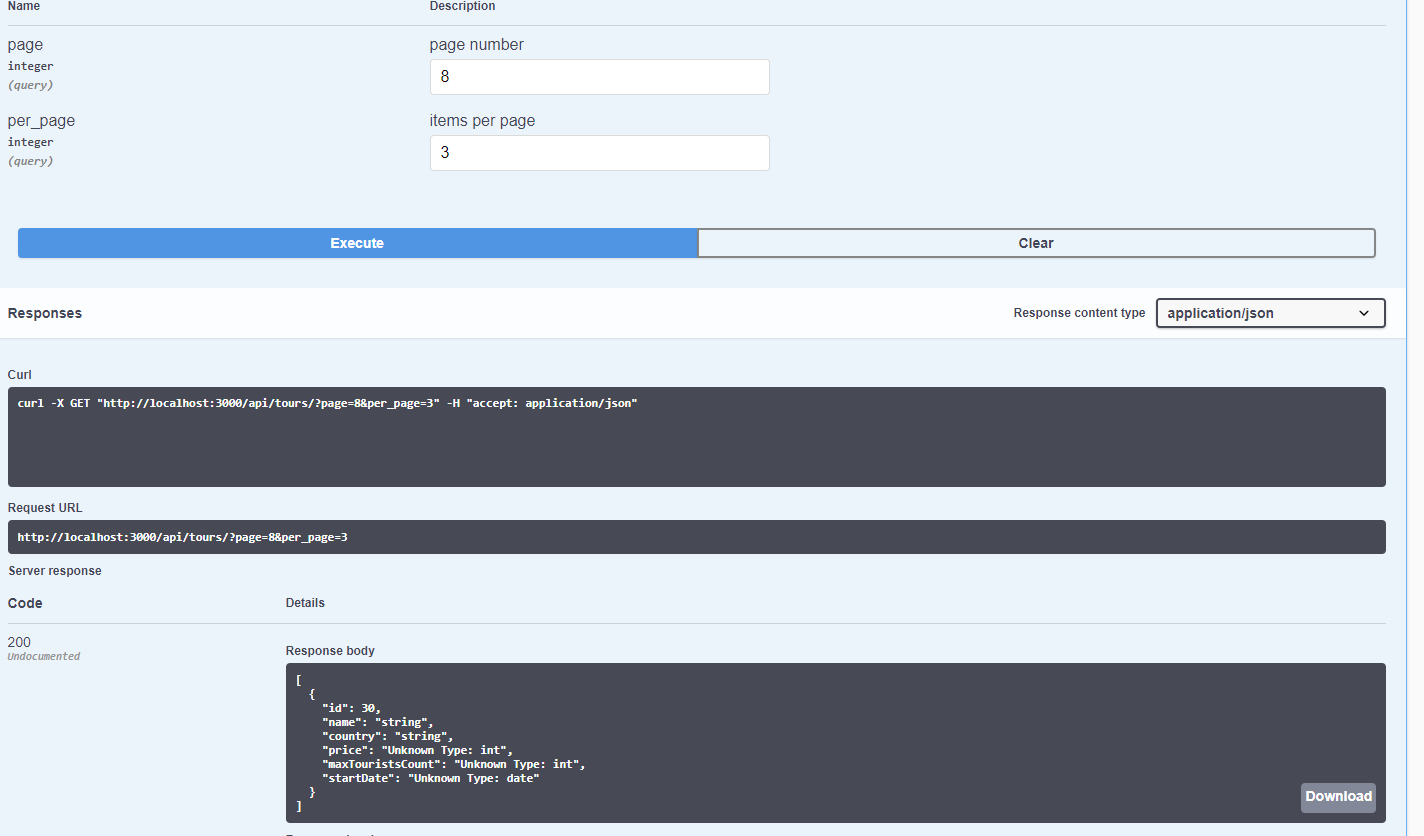


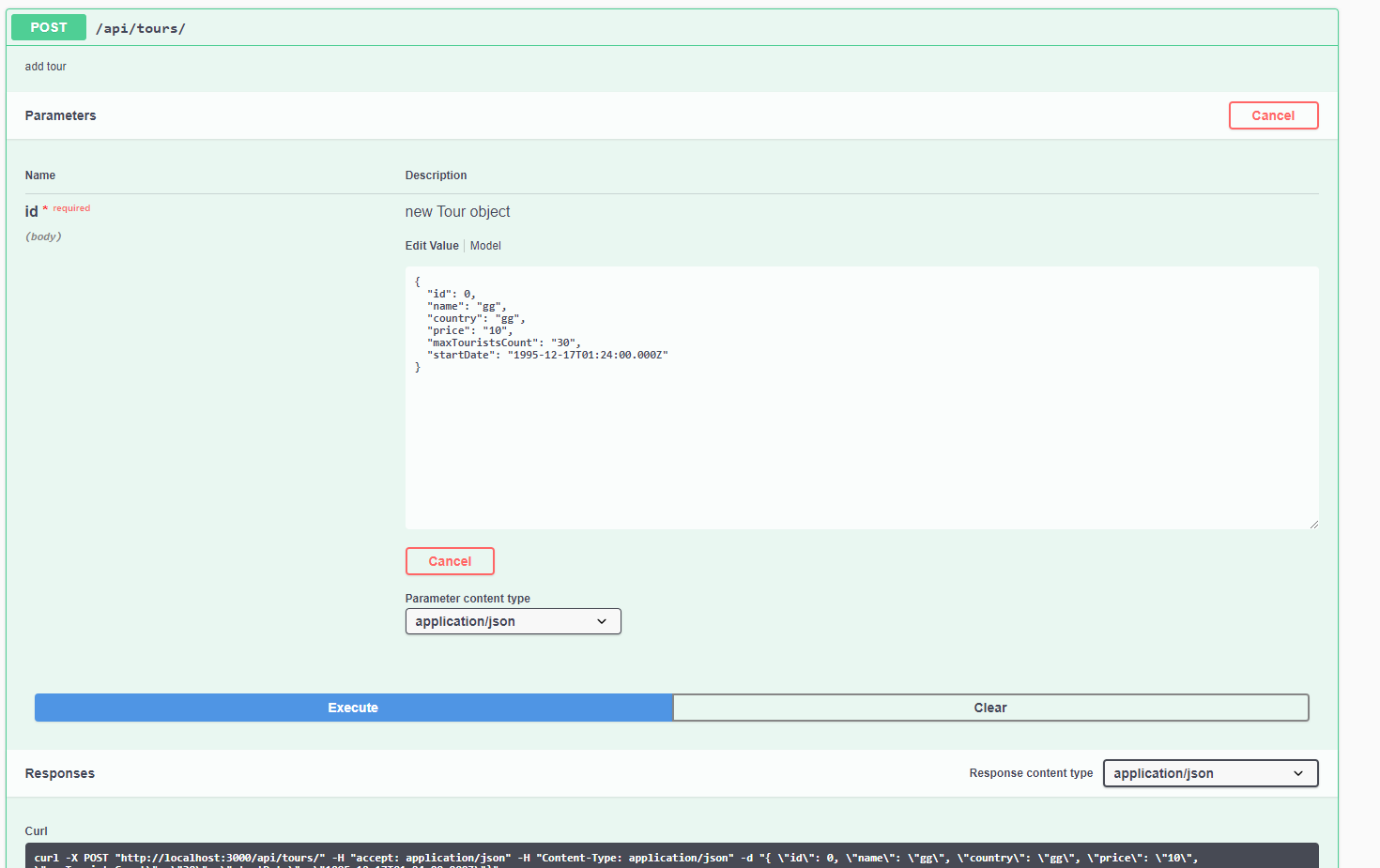
****

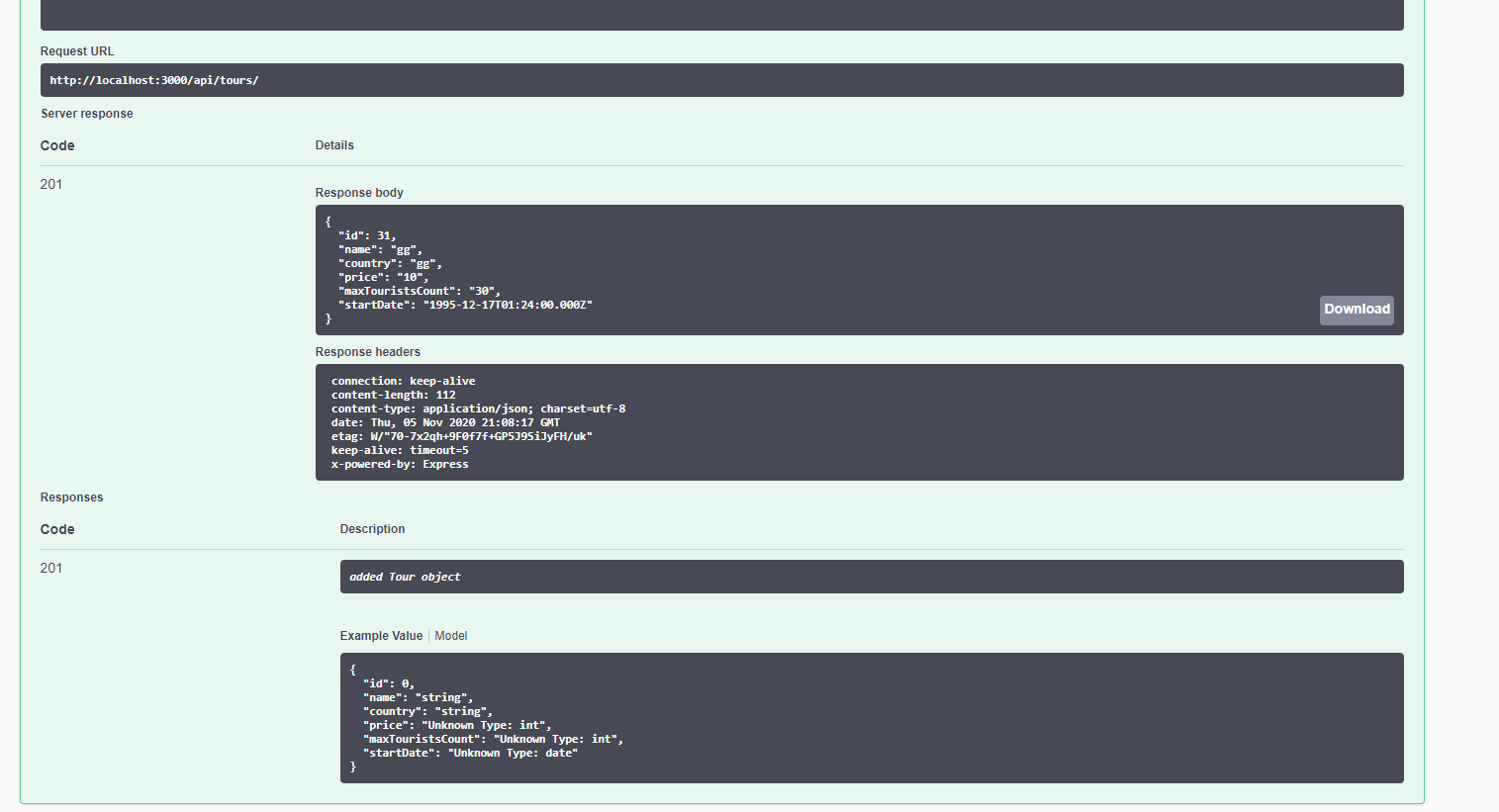
****

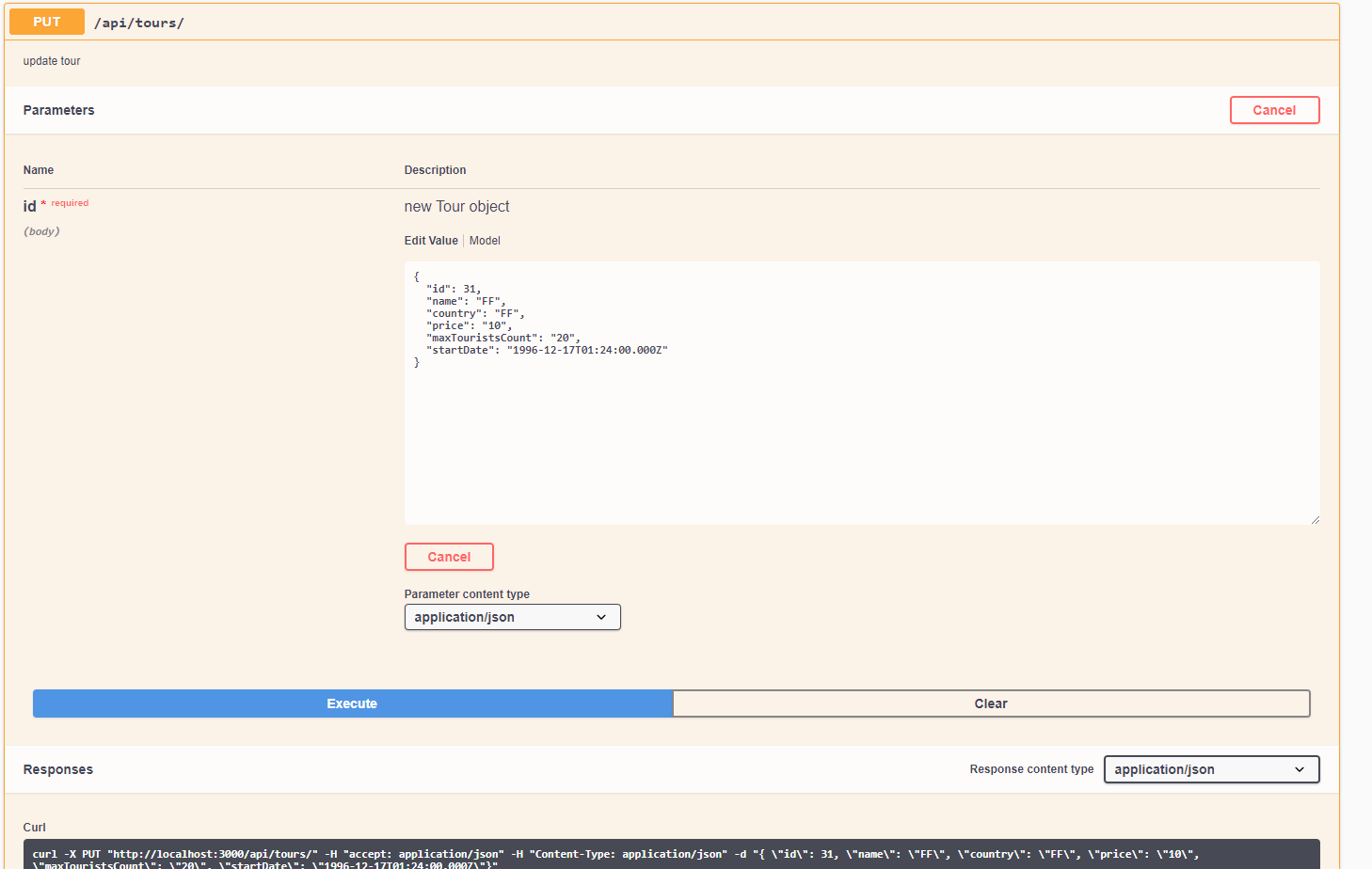
****

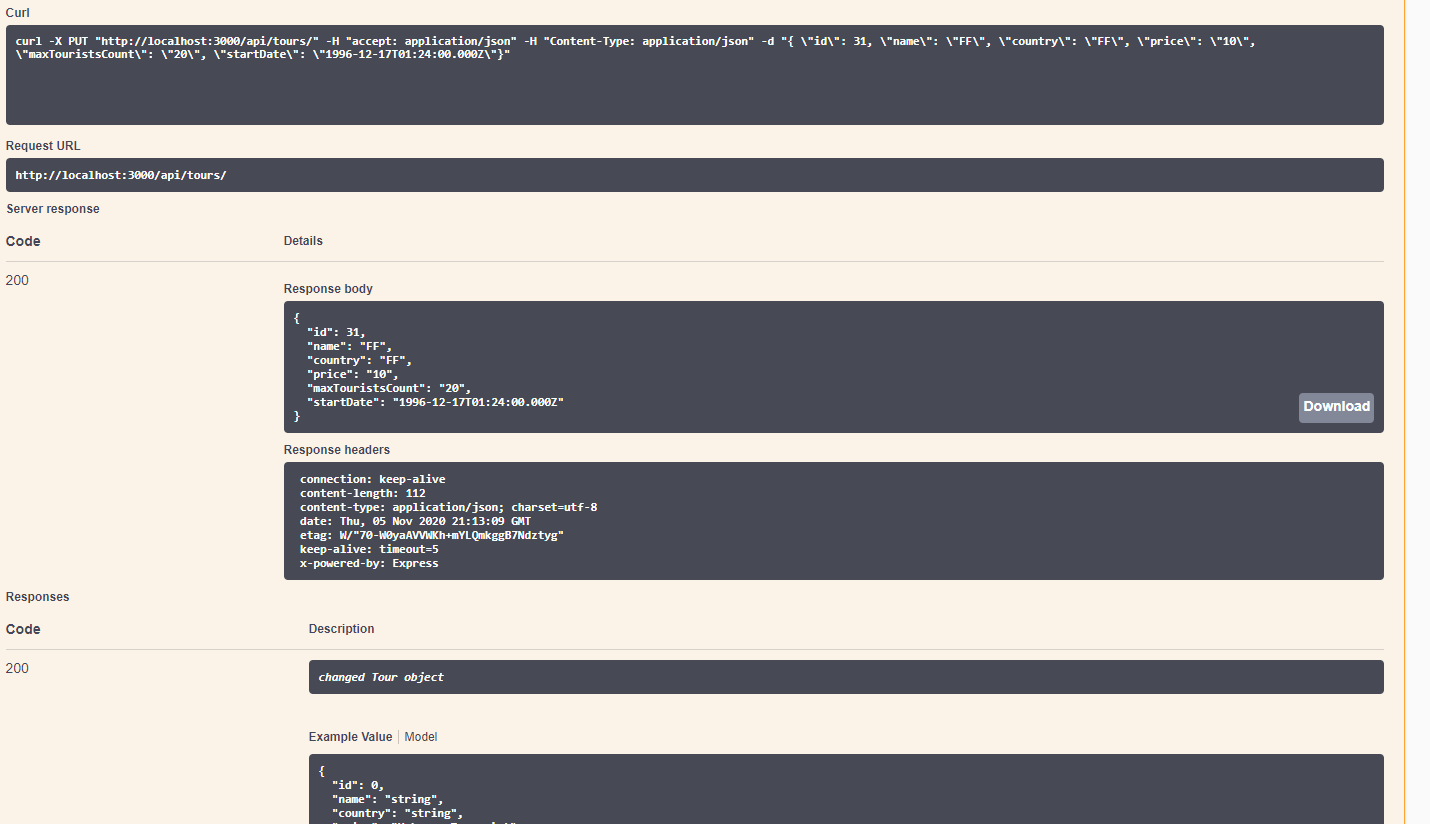
****

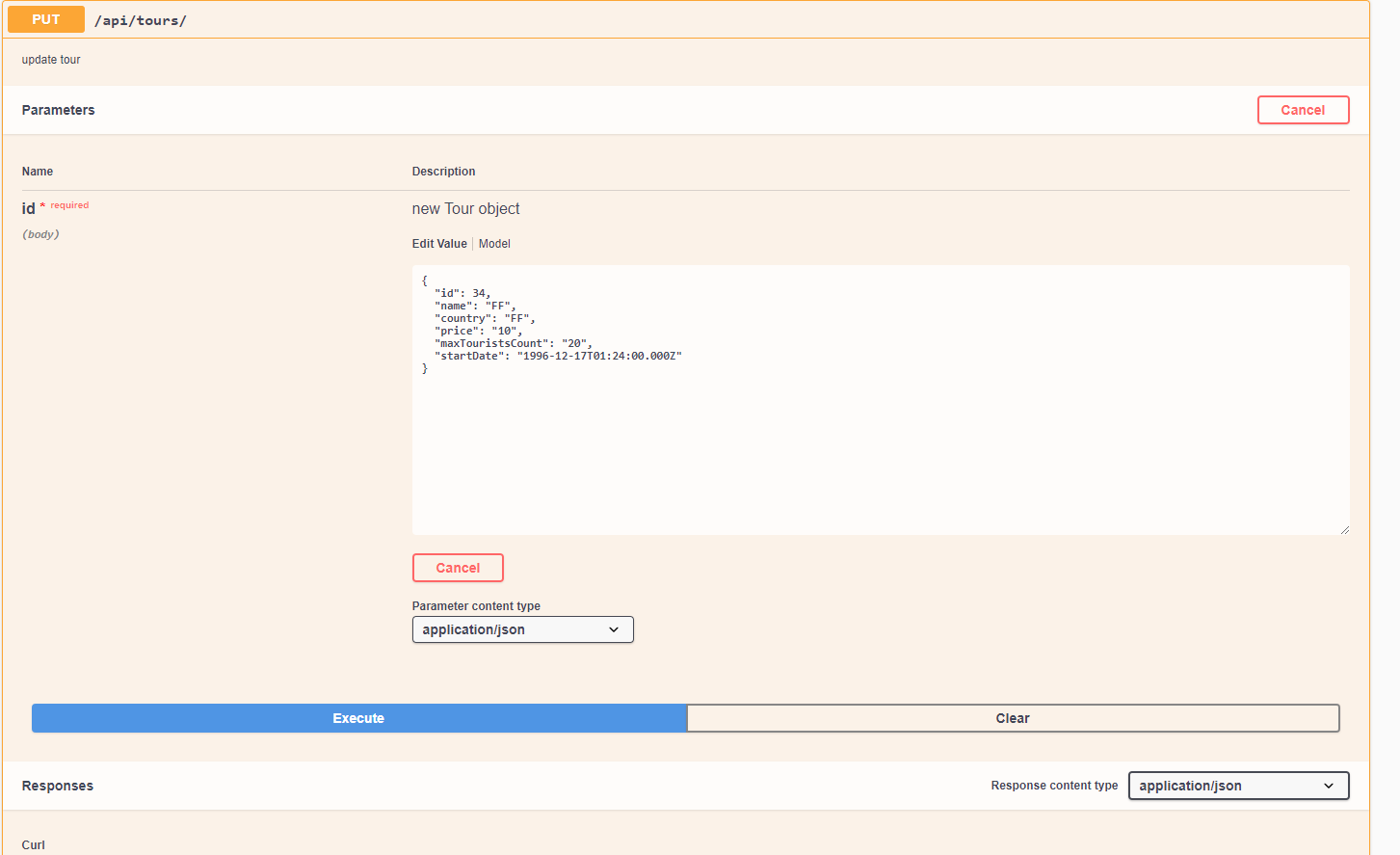
****

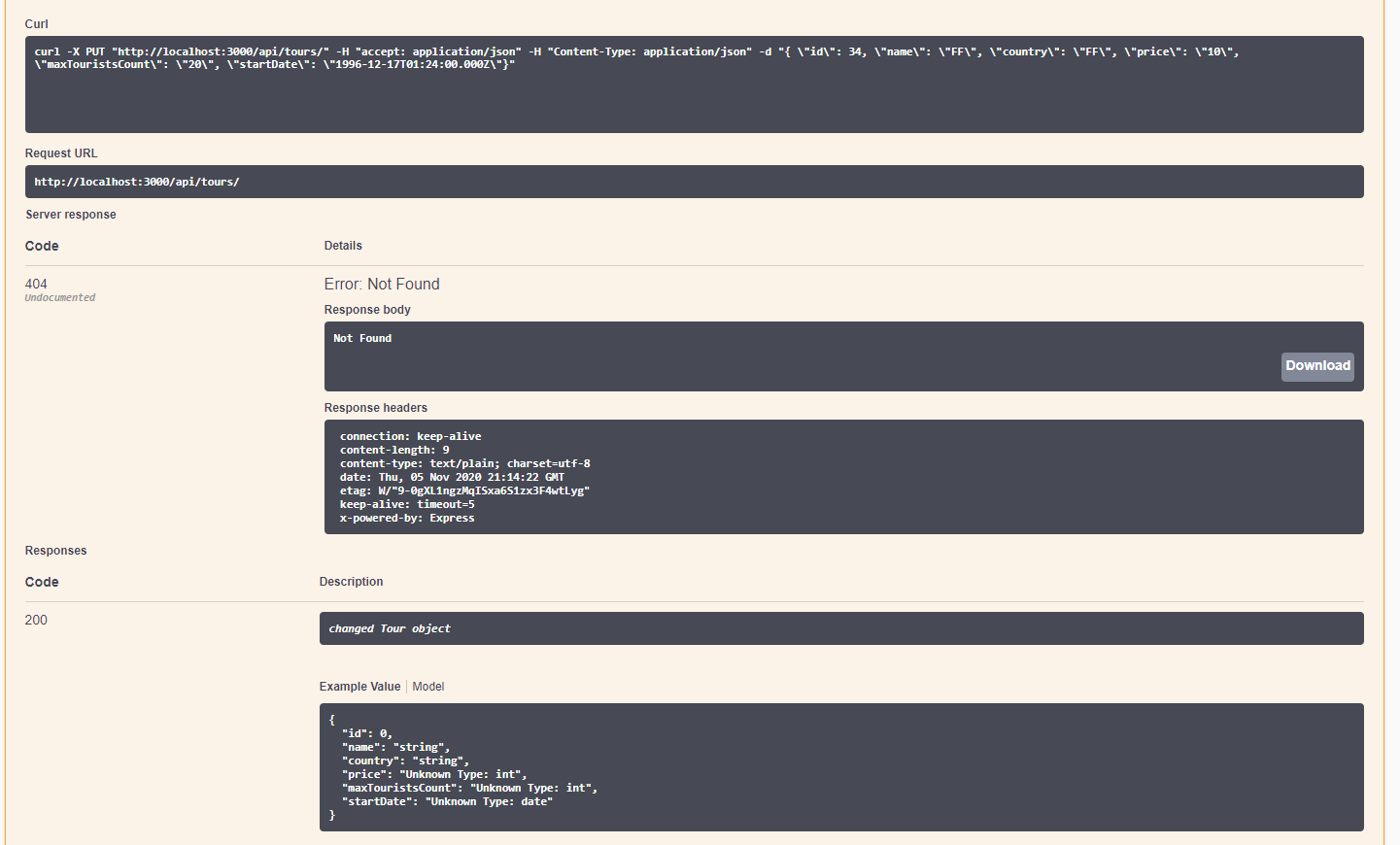
****

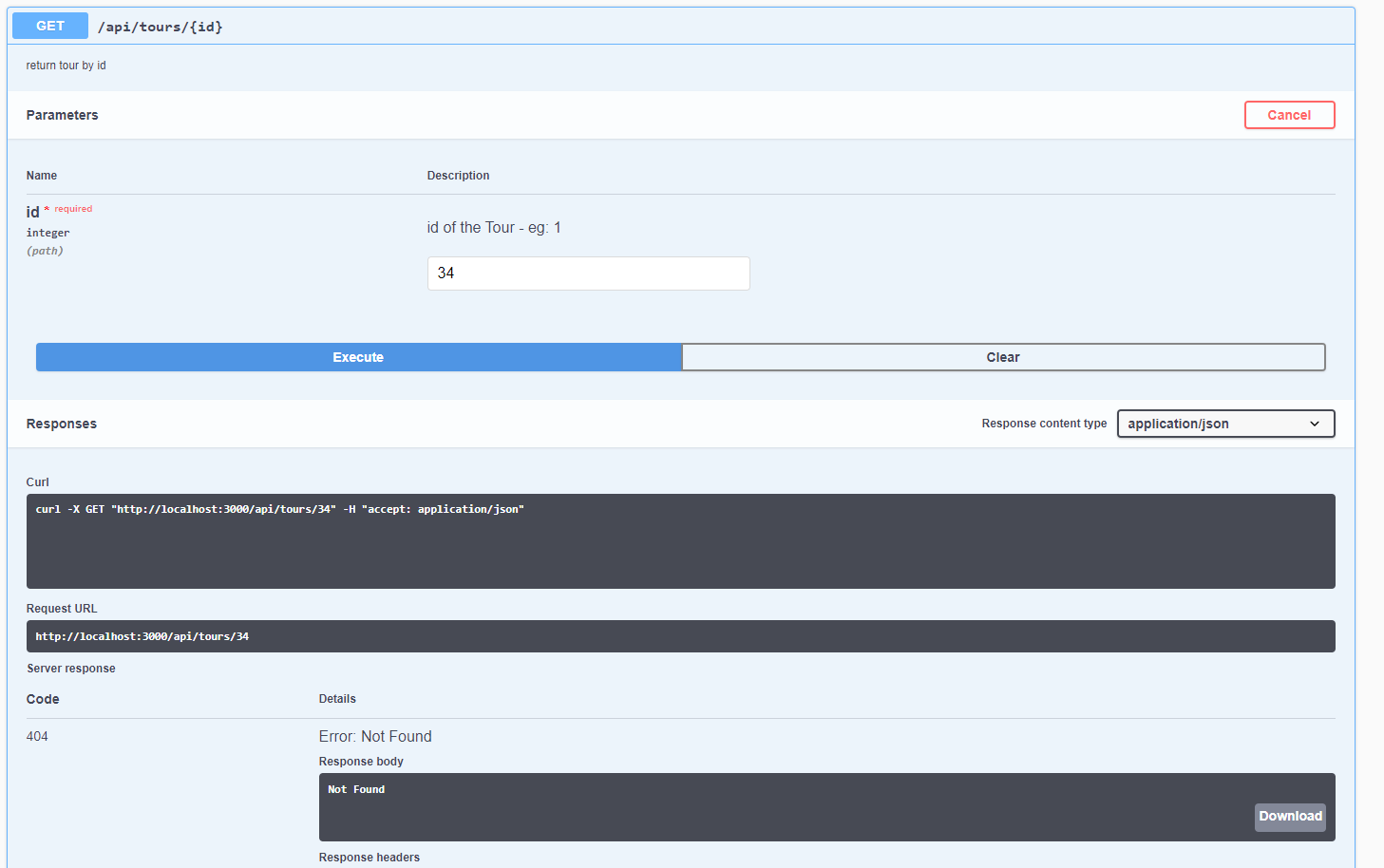
****

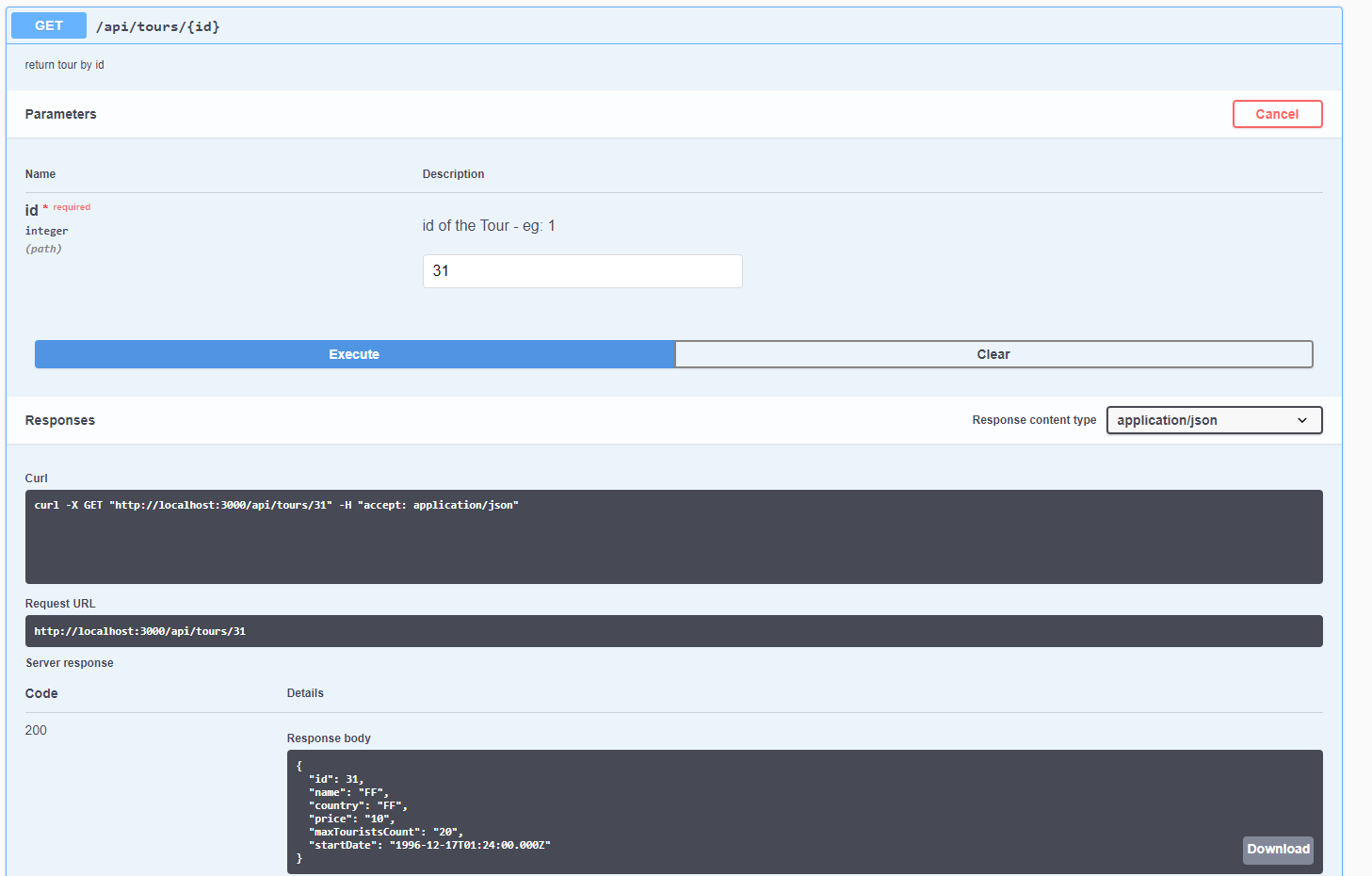
****

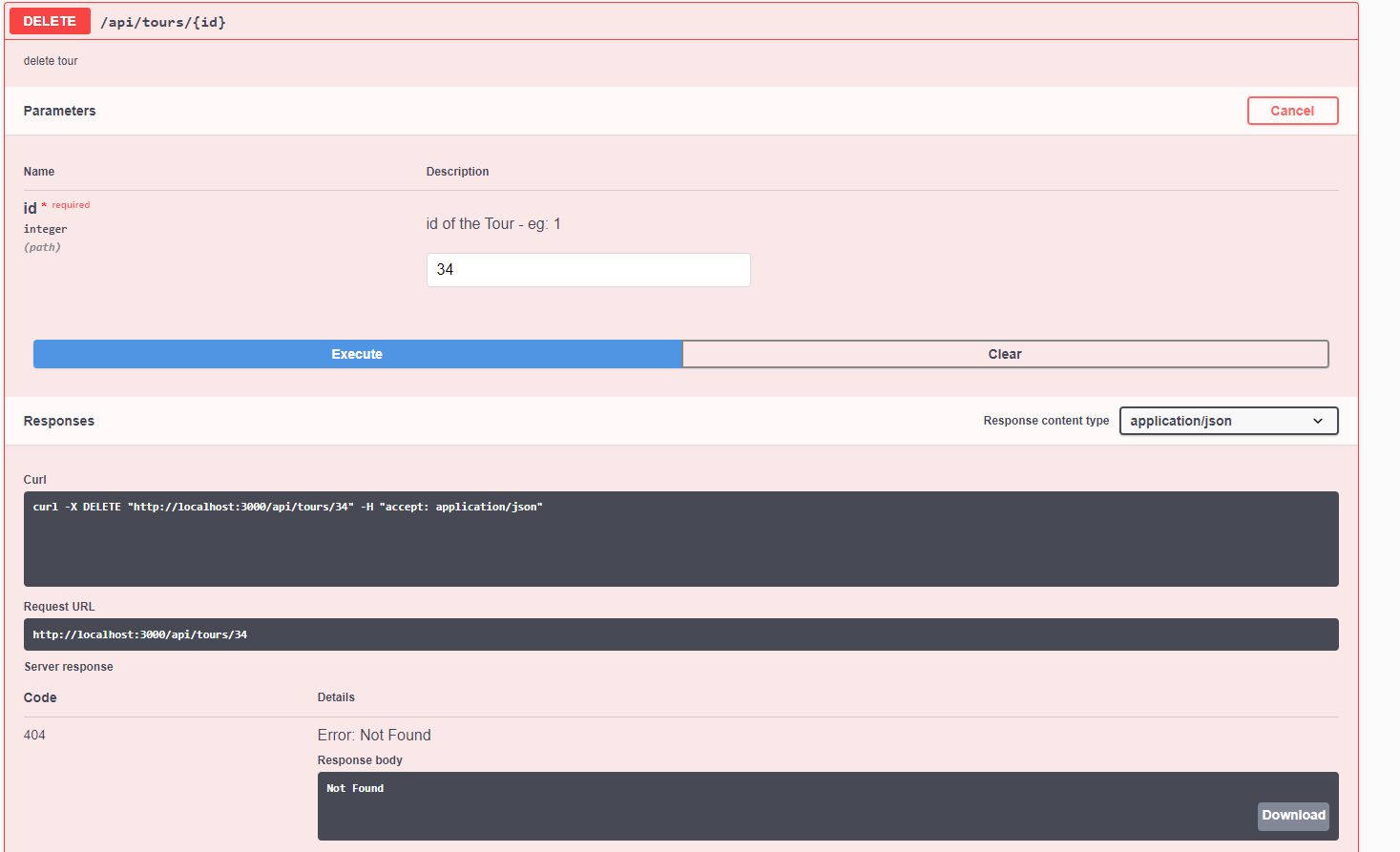
****

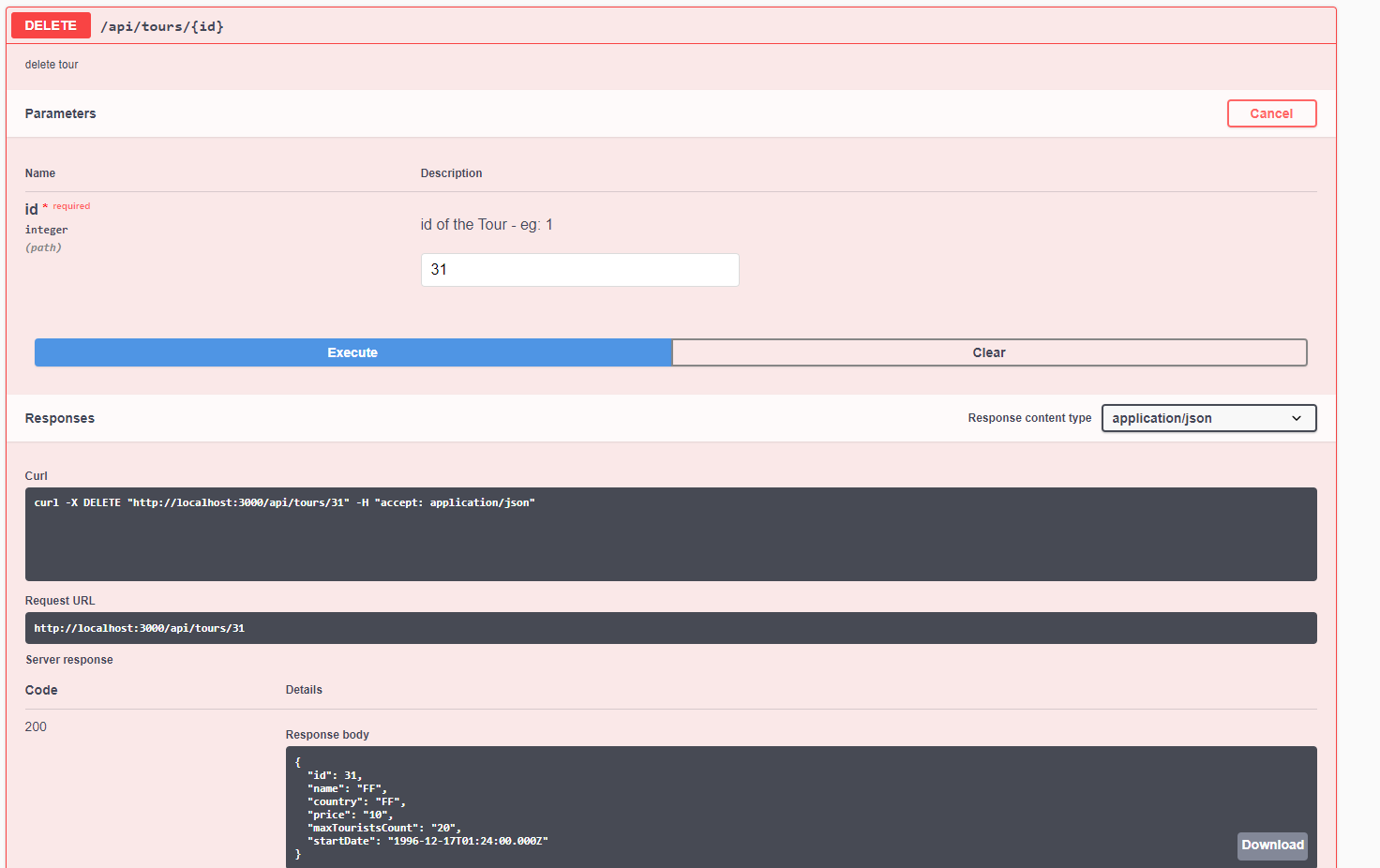
****

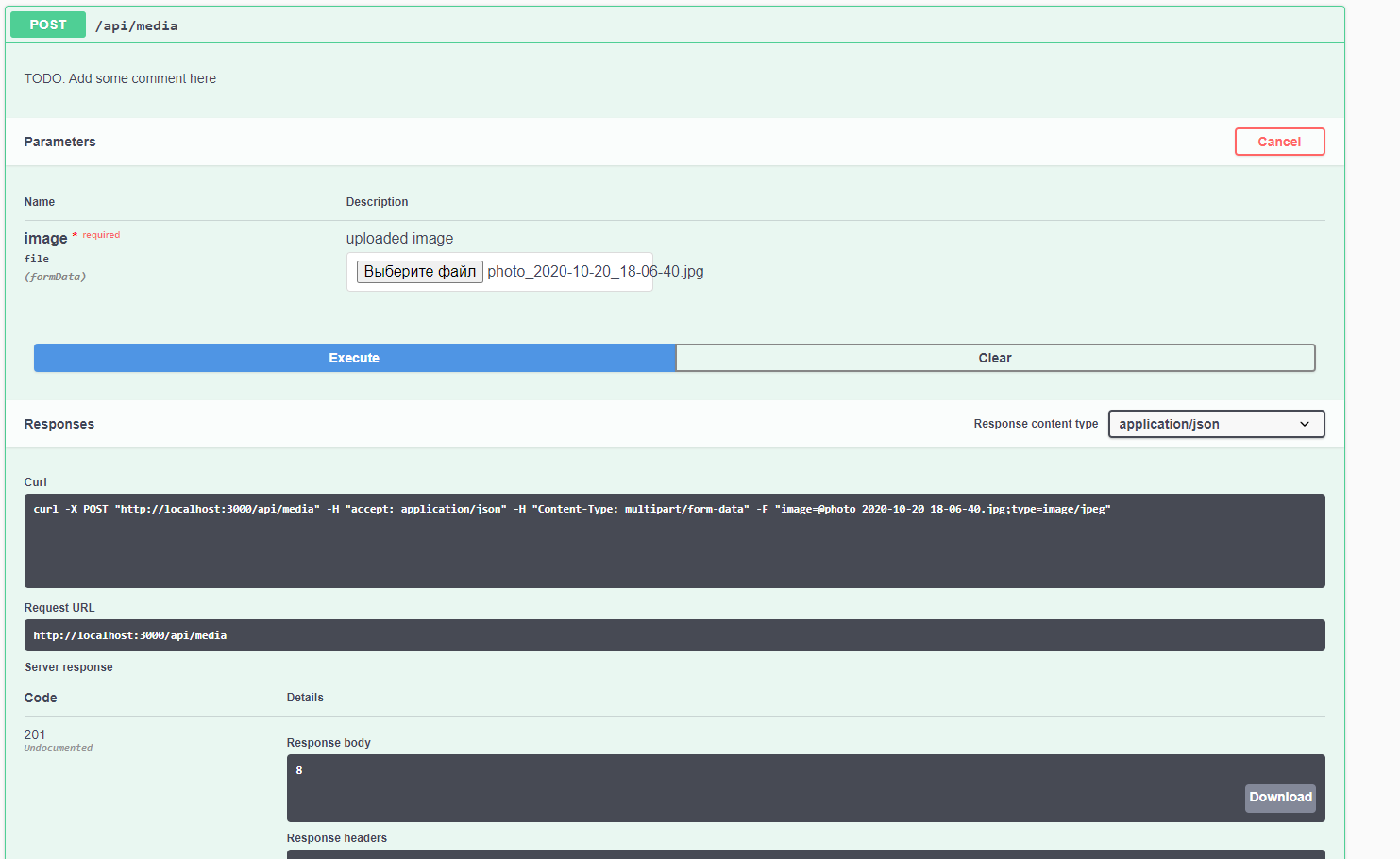
****

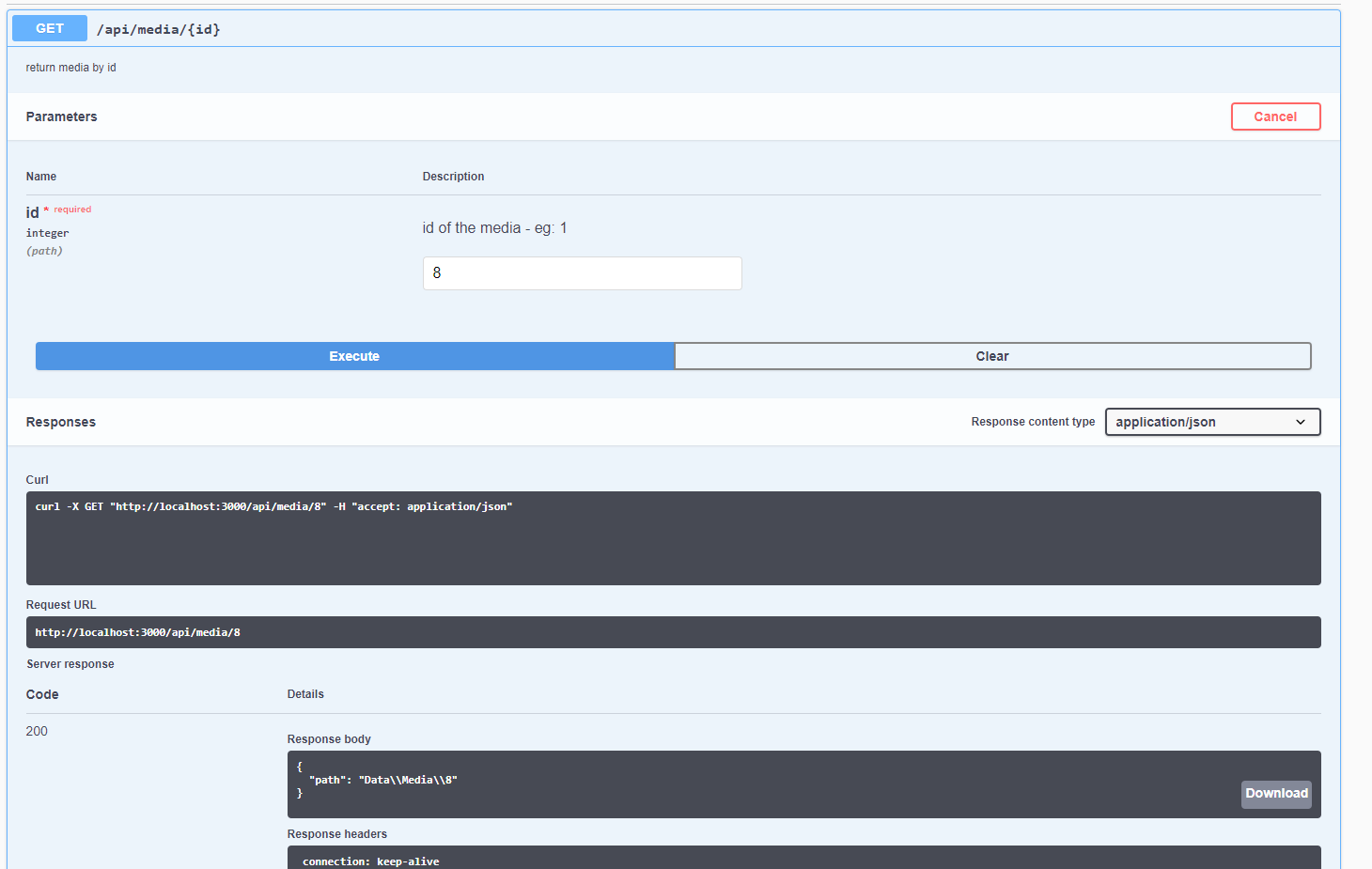
****

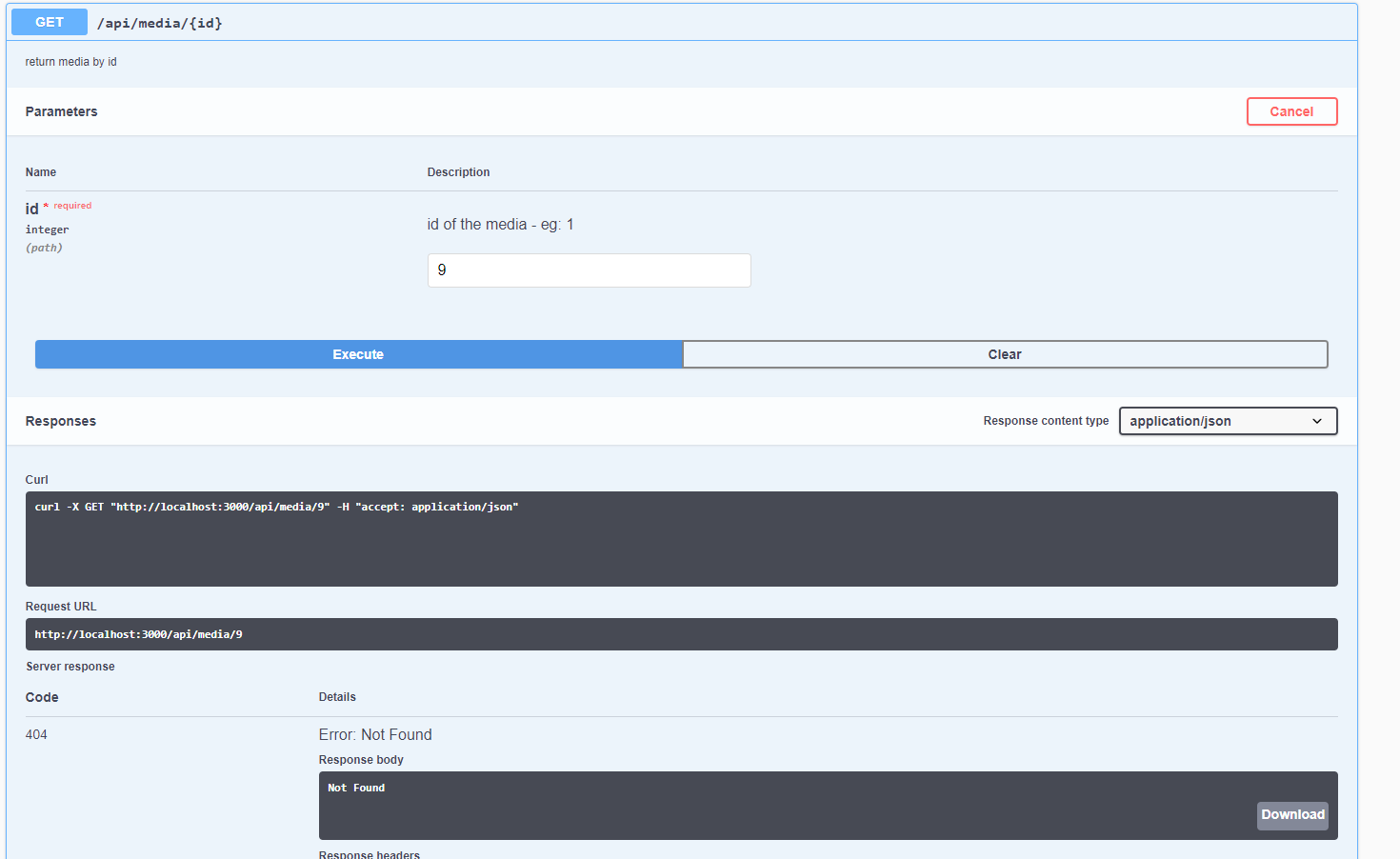
****

****

****

****

****

****

**Висновок**

Я навчився створювати простий JSON HTTP API веб-сервер, що надає доступ до ресурсів зі сховища.