

МІНІСТЕРСТВО ОСВІТИ 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  студент II курсу  групи КП-92  Гришаєв Дмитро Ігорович  (*прізвище, ім’я, по батькові*)  варіант №4 |  | 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.

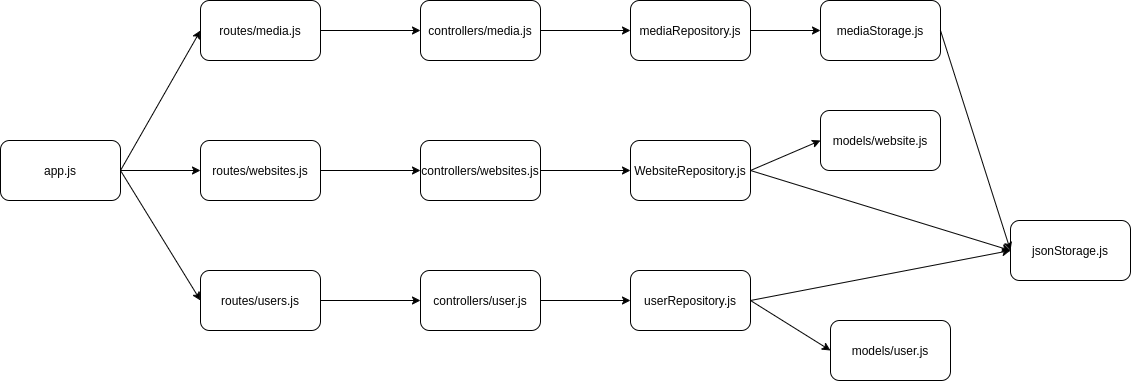
**Постановка завдання**

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**



**Код всіх всіх нових та змінених модулів**

|  |
| --- |
| **app.js** |
| const express = require('express');  const userRouter = require('./routes/users.js');  const websiteRouter = require('./routes/websites.js');  const mediaRouter = require('./routes/media.js');  const bodyParser = require("body-parser");  const busboy = require('busboy-body-parser');  const app = express();  const busboyOptions = {  limit: '10mb',  multi: false,  }  app.use(busboy(busboyOptions));  app.use(bodyParser.json());  app.use('/api/users', userRouter);  app.use('/api/websites', websiteRouter);  app.use('/api/media', mediaRouter);  const expressSwaggerGenerator = require('express-swagger-generator');  const expressSwagger = expressSwaggerGenerator(app);  const options = {  swaggerDefinition: {  info: {  description: 'JSON http api documentation',  title: 'documentation',  version: '1.0.0',  },  host: 'localhost:3000',  produces: ["application/json"],  },  basedir: \_\_dirname,  files: ['./routes//\*.js', './models//\*.js'],  };  expressSwagger(options);  app.listen(3000); |

|  |
| --- |
| **websites.js** |
| //models/website.js  /\*\*  \* @typedef Website  \* @property {integer} id  \* @property {string} name.required - unique name  \* @property {string} address.required - unique address  \* @property {integer} visits\_per\_day - amount of visits per day  \* @property {integer} count\_of\_ad\_bunners - amount of bunners on pages  \* @property {string} date\_of\_creating - the date of creating website  \*/  class Website {  constructor(id, name, address, visits\_per\_day, count\_of\_ad\_bunners, date\_of\_creating) {  this.id = id; // id  this.name = name; // string  this.address = address; // string  this.visits\_per\_day = visits\_per\_day; //number  this.count\_of\_ad\_bunners = count\_of\_ad\_bunners; //number  this.date\_of\_creating = date\_of\_creating; // Date ISO 8601  }  };  module.exports = Website; |

|  |
| --- |
| **user.js** |
| //models/user.js  /\*\*  \* @typedef User  \* @property {integer} id  \* @property {string} login.required - unique username  \* @property {string} fullname - user`s fullname  \* @property {integer} role - user`s role  \* @property {Date} registeredAt - the day when user registered in the system  \* @property {string} avaUrl - url of the picture  \* @property {bool} isEnabled - is user enabled  \*/  class User {  constructor(id, login, fullname, role, registeredAt, avaUrl, isEnabled) {  this.id = id; // number  this.login = login; // string  this.fullname = fullname; // string  this.role = role; //number 0 || 1  this.registeredAt = registeredAt; // Date ISO 8601  this.avaUrl = avaUrl; //strign (URL)  this.isEnabled = isEnabled; //bool  }  };  module.exports = User; |

|  |
| --- |
| **routes/media.js** |
| **const express = require('express');**  **const mediaController = require('../controllers/medias.js');**  **const router = express.Router();**  **/**  **\* Posting media**  **\* @route POST /api/media**  **\* @group Media - upload and get images**  **\* @consumes multipart/form-data**  **\* @param {file} image.formData.required - uploaded image**  **\* @returns {integer} 201 - added image id**  **\*/**  **/**  **\* Description of getting media by id**  **\* @route GET /api/media/{id}**  **\* @group Media - media operations**  **\* @param {integer} id.path.required - id of the image**  **\* @returns {file} 200 - image object**  **\* @returns {Error} 404 - image not found**  **\*/**  **router**  **.post('/', mediaController.addMedia)**  **.get('/:id', mediaController.getMediaById);**  **module.exports = router;** |

|  |
| --- |
| **routes/users.js** |
| **const express = require('express');**  **const userController = require('../controllers/users.js');**  **const router = express.Router();**  **/**  **\* Description of getting user by id**  **\* @route GET /api/users/{id}**  **\* @group Users - user operations**  **\* @param {integer} id.path.required - id of the User**  **\* @returns {User.model} 200 - User object**  **\* @returns {Error} 404 - User not found**  **\*/**  **/**  **\* Description of getting 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 - a page with users**  **\*/**  **router**  **.get("/:id", userController.getUserById)**  **.get("/", userController.getUsers)**  **module.exports = router;** |

|  |
| --- |
| **routes/websites.js** |
| **const express = require('express');**  **const websiteController = require('../controllers/websites.js');**  **const router = express.Router();**  **/**  **\* Description of getting all websites**  **\* @route GET /api/websites**  **\* @group Websites - website operations**  **\* @param {integer} page.query - page number**  **\* @param {integer} per\_page.query - items per page**  **\* @returns {Array.<Website>} 200 - a page with websites**  **\*/**  **/**  **\* Description of getting website by id**  **\* @route GET /api/websites/{id}**  **\* @group Websites - website operations**  **\* @param {integer} id.path.required - id of the Website**  **\* @returns {Website.model} 200 - Website object**  **\* @returns {Error} 404 - Website not found**  **\*/**  **/**  **\* Description of getting all websites**  **\* @route POST /api/websites**  **\* @group Websites - website operations**  **\* @param {Website.model} id.body.required - new Website object**  **\* @returns {Website.model} 201 - added Website object**  **\* @returns {Error} 400 - Website wasnt added**  **\*/**  **/**  **\* Description of getting all websites**  **\* @route PUT /api/websites**  **\* @group Websites - website operations**  **\* @param {Website.model} id.body.required - new Website object**  **\* @returns {Website.model} 200 - changed Website object**  **\* @returns {Error} 404 - Website wasnt updated**  **\*/**  **/\*\***  **\* Description of getting website by id**  **\* @route DELETE /api/websites/{id}**  **\* @group Websites - website operations**  **\* @param {integer} id.path.required - id of the Website**  **\* @returns {Website.model} 200 - deleted Website object**  **\* @returns {Error} 404 - Website not found**  **\*/**  **router**  **.get("/", websiteController.getWebsites)**  **.get("/:id", websiteController.getWebsitesById)**  **.post("/", websiteController.addWebsite)**  **.put("/", websiteController.updateWebsite)**  **.delete("/:id", websiteController.deleteWebsites)**  **module.exports = router;** |

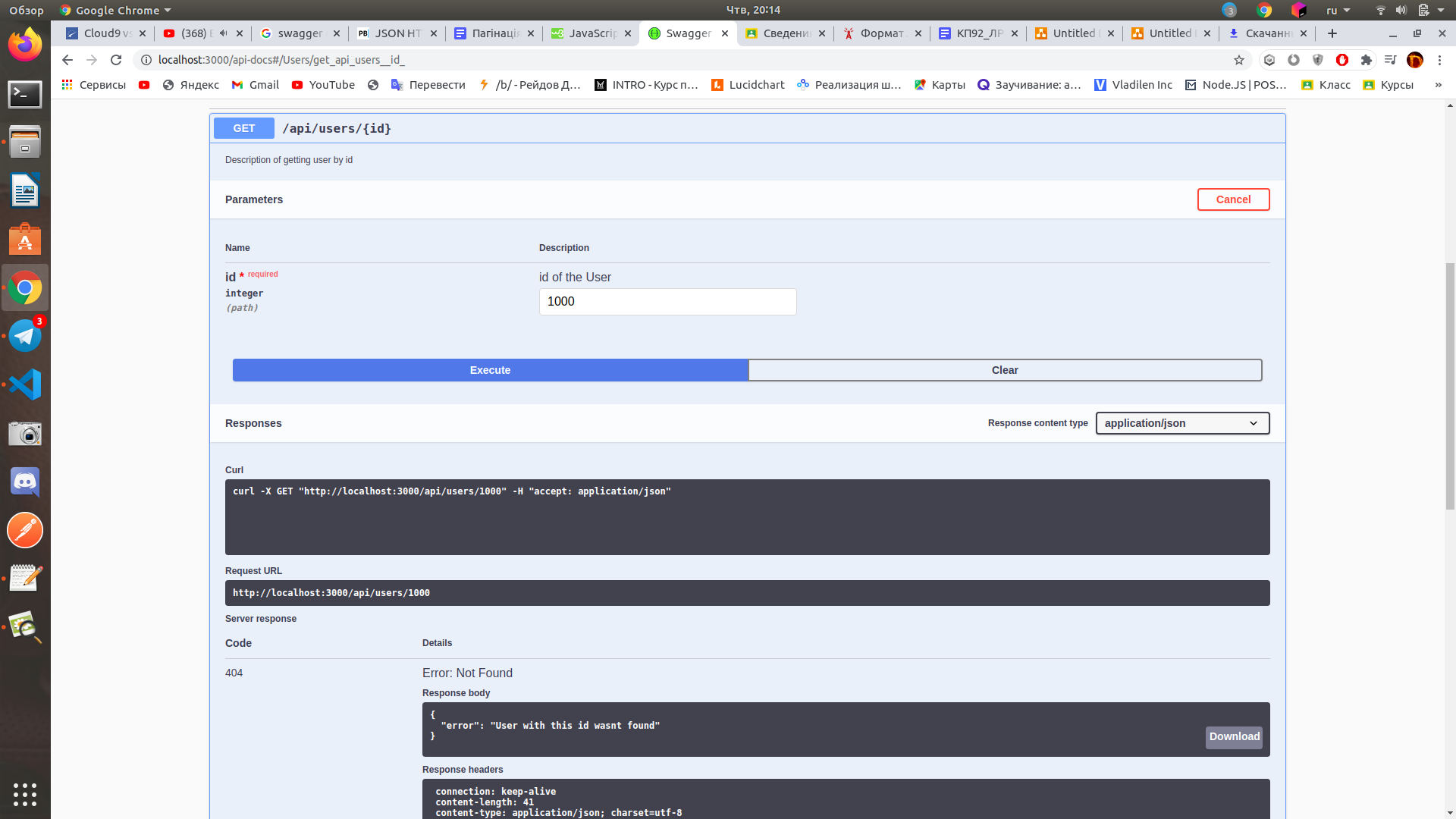
|  |
| --- |
| **controllers/media.js** |
| **const media\_rep = require('../repositories/mediaRepository.js');**  **const mediaRepository = new media\_rep('data/media/', 'data/media.json');**  **module.exports = {**  **getMediaById(req, res) {**  **res.set("Content-Type", "application/json");**  **try {**  **let media = mediaRepository.getMediaById(req.params.id);**  **res.send(JSON.stringify({ image: { data: media } }, null, 4));**  **} catch (err) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: 'Media with this id wasnt found' }));**  **}**  **},**  **addMedia(req, res) {**  **res.set("Content-Type", "application/json");**  **console.log(req.files);**  **res.statusCode = 201;**  **const id = mediaRepository.addMedia(req.files.image);**  **res.send(JSON.stringify({ id: id }));**  **}**  **};** |

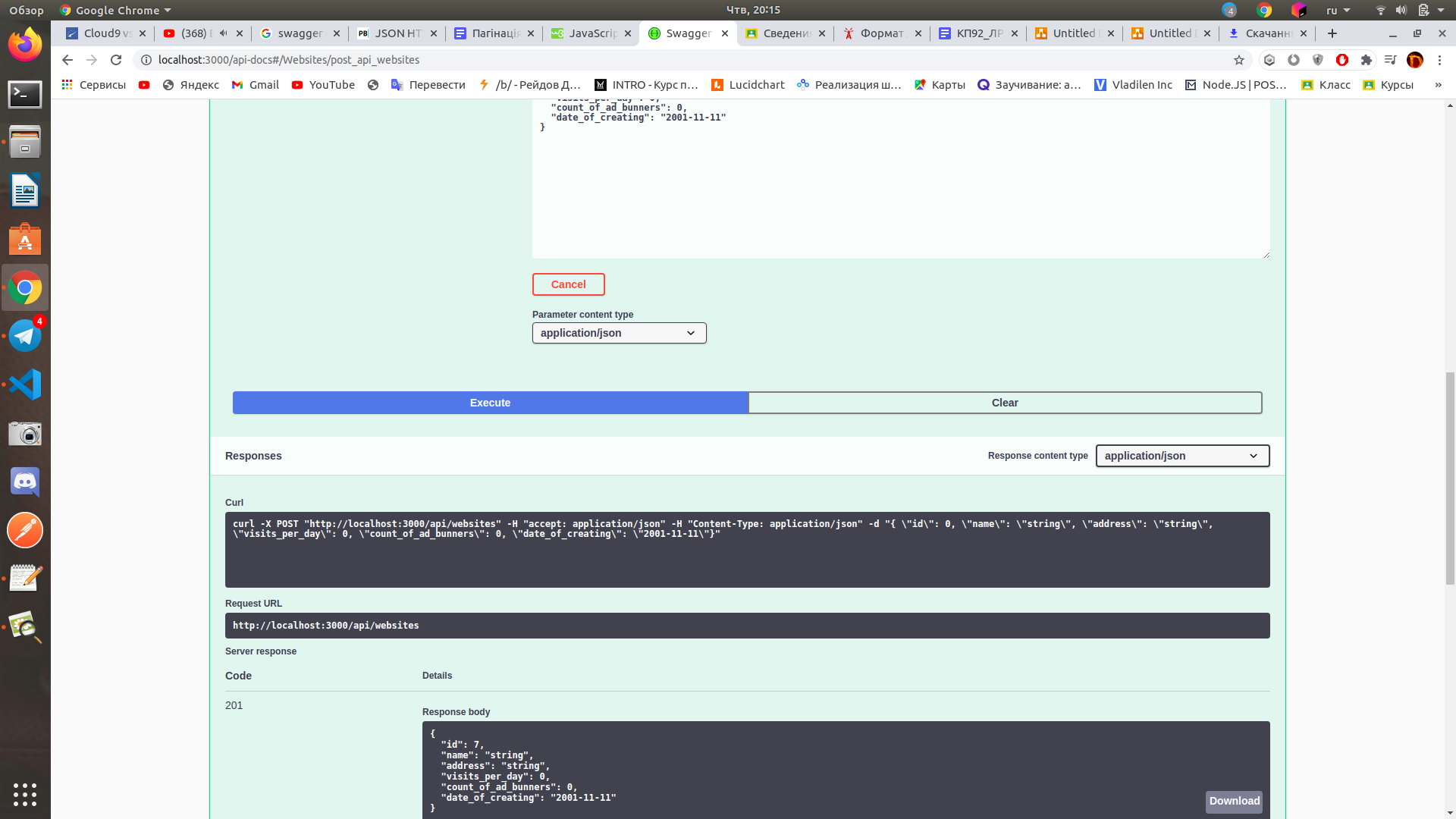
|  |
| --- |
| **controllers/users.js** |
| **const user\_rep = require('../repositories/userRepository.js');**  **const userRepository = new user\_rep("data/users.json");**  **module.exports = {**  **getUsers(req, res) {**  **res.set("Content-type", "application/json");**  **let page = Number.parseInt(req.param("page") || 1); //2**  **let records = Number.parseInt(req.param("per\_page") || 5); //3**  **let records = Number.parseInt(req.param("per\_page") || 5); //3**  **if (!Number.isInteger(page)) || !Number.isInteger(records))) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: "param should be an integer" }));**  **}**  **if (records > 100) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: "page size should be less than 100" }));**  **}**  **let arr = userRepository.getUsers();**  **let record\_count = arr.length; //6**  **let page\_count = Math.ceil(record\_count / records); //6/3 = 2**  **if (page > page\_count) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: "Page was not found" }));**  **} else {**  **let left;**  **let right;**  **if (page\_count < page) // 2<=1**  **{**  **left = (page\_count - 1) \* records;**  **right = record\_count;**  **} else { //2>1**  **left = (page - 1) \* records; // 0**  **right = left + records; //0+3**  **}**  **const newArr = arr.slice(left, right);**  **res.send(JSON.stringify(newArr, null, 4));**  **}**  **},**  **getUserById(req, res) {**  **res.set("Content-type", "application/json");**  **const user = userRepository.getUserById(req.params.id);**  **if (user === null) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: "User with this id wasnt found" }));**  **} else {**  **res.send(JSON.stringify(user, null, 4));**  **}**  **}**  **};** |

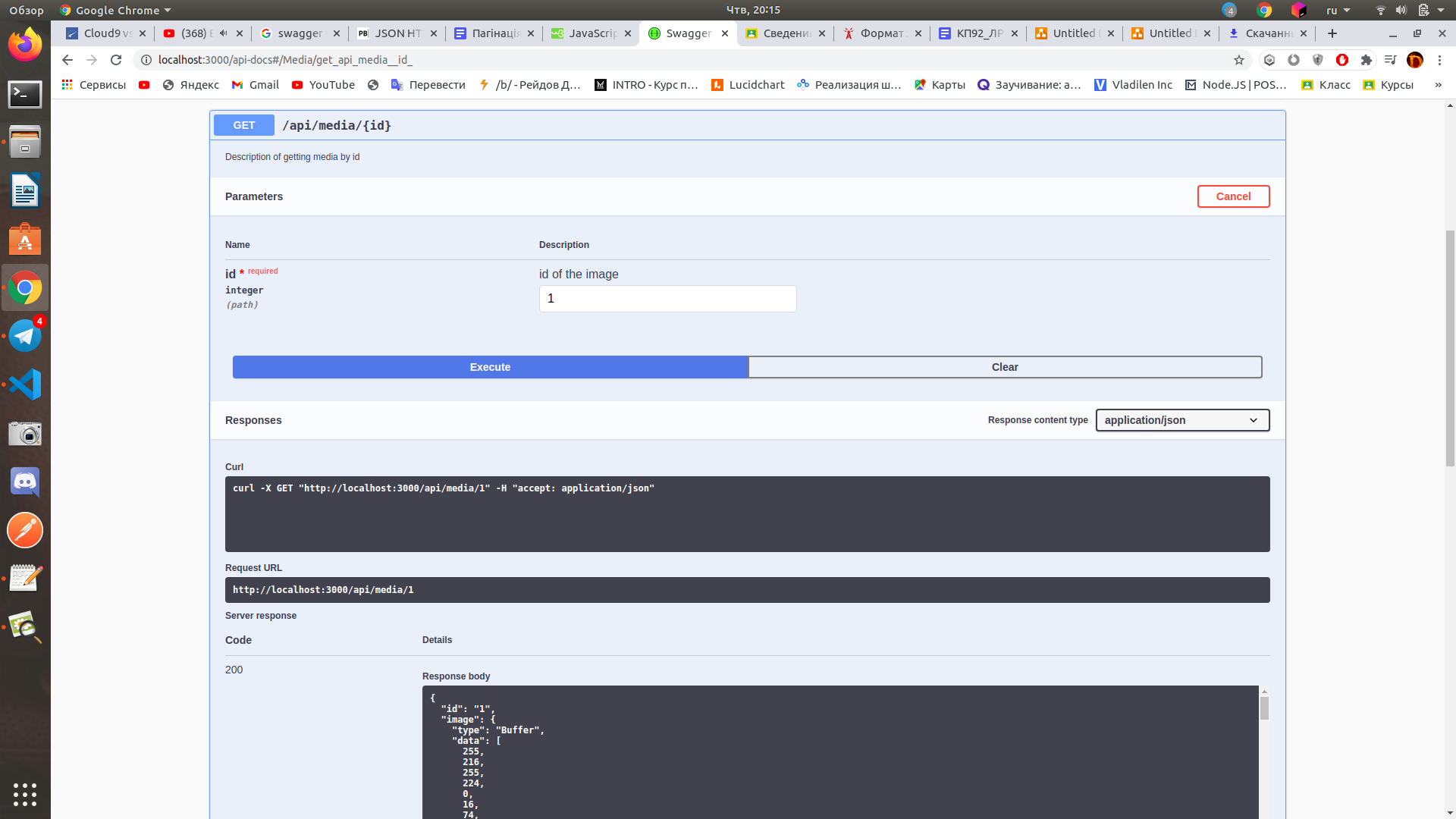
|  |
| --- |
| **controllers/websites.js** |
| **const site\_rep = require('../repositories/WebsiteRepository.js');**  **const websiteRepository = new site\_rep("data/Websites.json");**  **module.exports = {**  **getWebsites(req, res) {**  **res.set("Content-Type", "application/json");**  **let page = Number.parseInt(req.param("page") || 1); //2**  **let records = Number.parseInt(req.param("per\_page") || 5); //3**  **if (!Number.isInteger(page)) || !Number.isInteger(records))) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: "param should be an integer" }));**  **}**  **if (records > 100) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: "page size should be less than 100" }));**  **}**  **let arr = websiteRepository.getWebsites();**  **let record\_count = arr.length; //6**  **let page\_count = Math.ceil(record\_count / records); //6/3 = 2**  **if (page > page\_count) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: "Page was not found" }));**  **} else {**  **let left;**  **let right;**  **if (page\_count < page) // 2<=1**  **{**  **left = (page\_count - 1) \* records;**  **right = record\_count;**  **} else { //2>1**  **left = (page - 1) \* records; // 0**  **right = left + records; //0+3**  **}**  **const newArr = arr.slice(left, right);**  **res.send(JSON.stringify(newArr, null, 4));**  **}**  **},**  **getWebsitesById(req, res) {**  **res.set("Content-Type", "application/json");**  **const website = websiteRepository.getWebsitesById(req.params.id)**  **if (website === null) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: 'Website with this id wasnt found' }));**  **} else {**  **res.send(JSON.stringify(website, null, 4));**  **}**  **},**  **addWebsite(req, res) {**  **res.set("Content-Type", "application/json");**  **if (req.body.name === undefined || req.body.address === undefined) {**  **res.statusCode = 400;**  **res.send(JSON.stringify({ error: 'Bad request' }));**  **} else {**  **res.statusCode = 201;**  **res.send(JSON.stringify(websiteRepository.addWebsite(req.body), null, 4));**  **}**  **},**  **updateWebsite(req, res) {**  **res.set("Content-Type", "application/json");**  **const website = websiteRepository.updateWebsite(req.body)**  **if (website === null) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: 'Any website wasnt updated' }));**  **} else if (req.body.name === undefined || req.body.address === undefined) {**  **res.statusCode = 400;**  **res.send(JSON.stringify({ error: 'Bad request' }));**  **} else {**  **res.send(JSON.stringify(website, null, 4));**  **}**  **},**  **deleteWebsites(req, res) {**  **res.set("Content-Type", "application/json");**  **const website = websiteRepository.deleteWebsites(req.params.id)**  **if (website === null) {**  **res.statusCode = 404;**  **res.send(JSON.stringify({ error: 'Website with this id wasnt found' }));**  **} else {**  **res.send(JSON.stringify(website, null, 4));**  **}**  **}**  **};** |

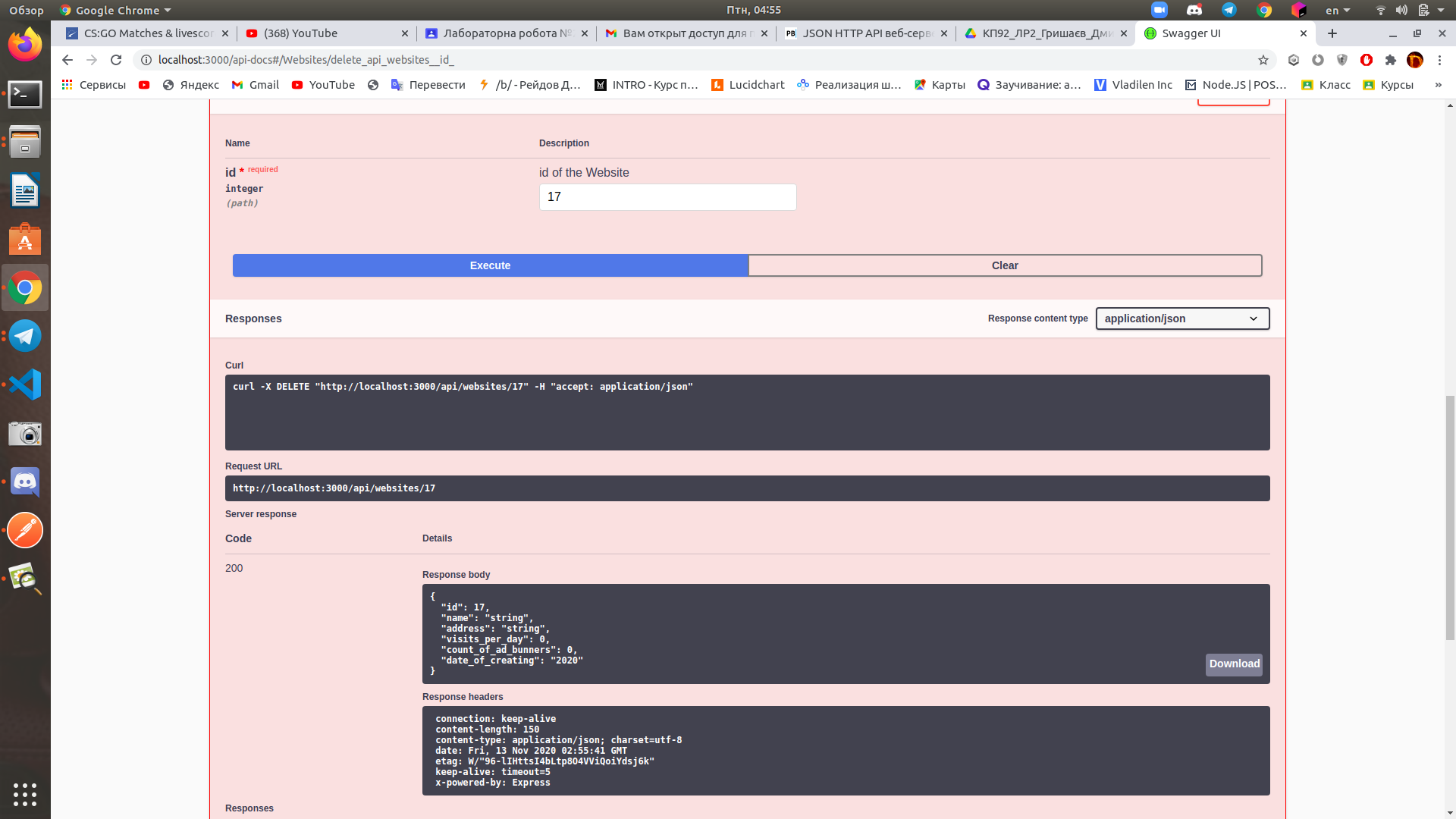
|  |
| --- |
| **mediaStorage.js** |
| **const fs = require('fs');**  **const JsonStorage = require('./jsonStorage.js');**  **class MediaStorage {**  **constructor(mediaPath, mediaInfoPath) {**  **this.mediaPath = mediaPath;**  **this.jsonstorage = new JsonStorage(mediaInfoPath);**  **}**  **readMedia(id) {**  **let mediaInfo = this.jsonstorage.readItems();**  **let path;**  **for (let i = 0; i < mediaInfo.items.length; i++) {**  **if (mediaInfo.items[i].id == id) {**  **path = mediaInfo.items[i].path;**  **break;**  **}**  **}**  **let mediaFile = fs.readFileSync(${path}, function(err, data) {**  **if (err) {**  **throw new Error(data);**  **}**  **});**  **return mediaFile;**  **}**  **writeMedia(media) {**  **let mediaInfo = this.jsonstorage.readItems();**  **mediaInfo.items.push({**  **id: mediaInfo.nextId,**  **path: ${this.mediaPath}${media.name}**  **});**  **mediaInfo.nextId++;**  **this.jsonstorage.writeItems(mediaInfo)**  **fs.writeFileSync(${this.mediaPath}${media.name}, media.data);**  **return --mediaInfo.nextId;**  **}**  **};**  **module.exports = MediaStorage;** |

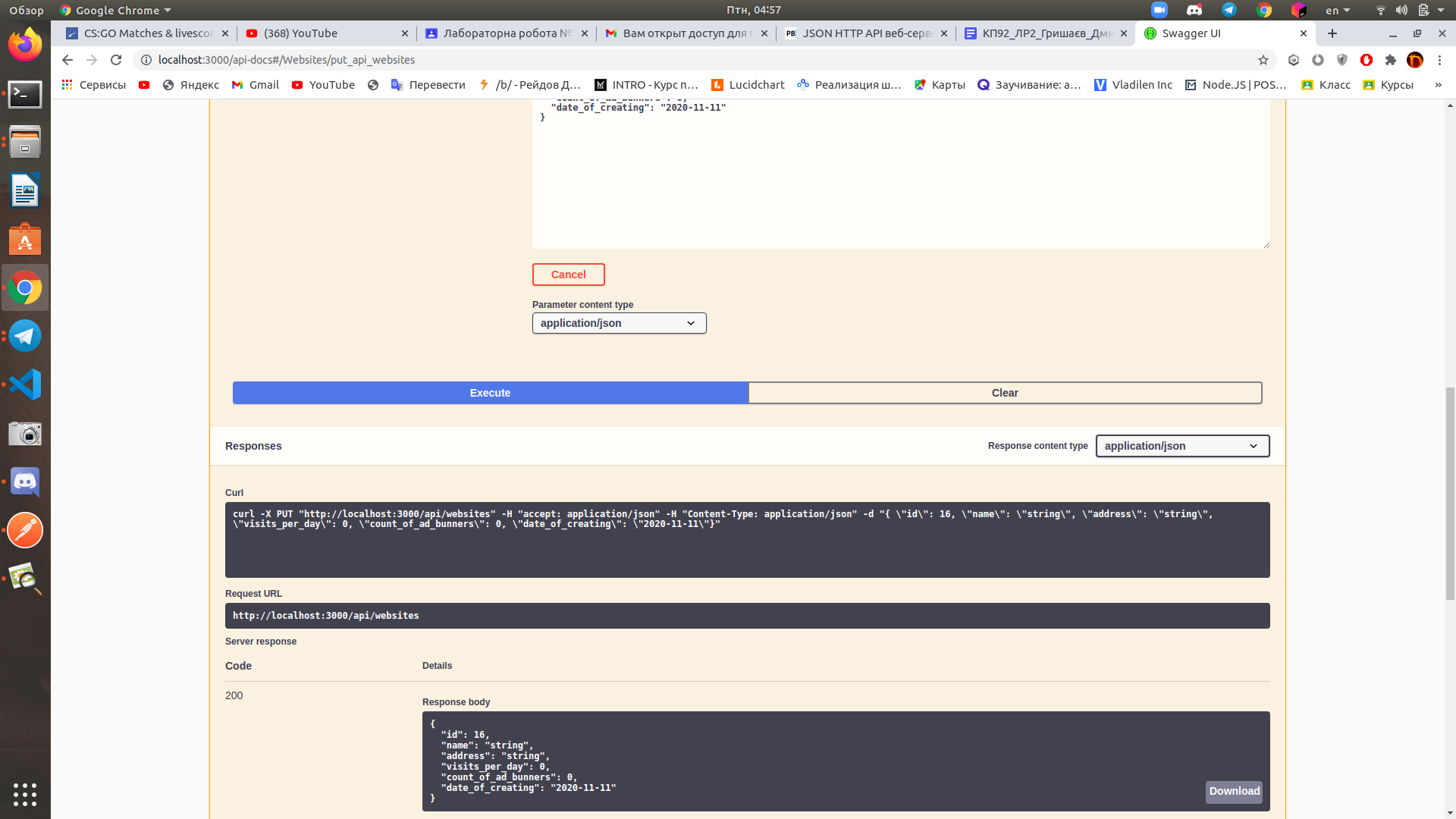
**Зображення Swagger сторінки /api-docs із прикладами взаємодії з JSON REST API**











**Висновки**

Виконавши дану лабораторну роботу було навчено створювати простий JSON HTTP API веб-сервер, що надає доступ до ресурсів зі сховища.  
 Лабораторна робота була виконана на мові JavaScript за допомогою середовища виконання Node.js у редакторі VS code.