

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  Мельник Ігор Сергійович  (*прізвище, ім’я, по батькові*)  варіант №8 |  | 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.

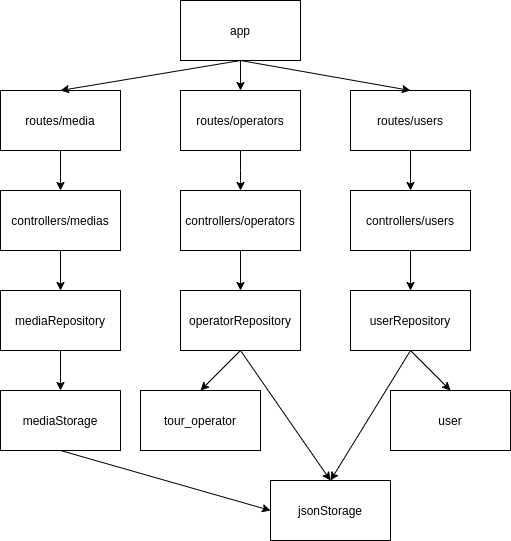
**Постановка завдання**

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 app = express();  const userRouter = require('./routes/users');  const operatorRouter = require('./routes/operators');  const mediaRouter = require('./routes/media.js');  const busboy = require('busboy-body-parser');  const busboyOptions = {  limit: '10mb',  multi: false,  };  app.use(busboy(busboyOptions));  app.use('/api/users', userRouter);  app.use('/api/operators', operatorRouter);  app.use('/api/media', mediaRouter);  const expressSwaggerGenerator = require('express-swagger-generator');  const expressSwagger = expressSwaggerGenerator(app);  const options = {  swaggerDefinition: {  info: {  description: 'documentation',  title: 'JSON HTTP API веб-сервер',  version: '1.0.0',  },  host: 'localhost:3001',  produces: ["application/json"],  },  basedir: \_\_dirname,  files: ['./routes/\*\*/\*.js', './models/\*\*/\*.js'],  };  expressSwagger(options);  app.listen(3001, function () { console.log('Server is ready'); }); |

|  |
| --- |
| **models/tour\_operator.js** |
| /\*\*  \* @typedef Tour\_operator  \* @property {integer} id  \* @property {string} name.required - name of operator  \* @property {string} countries - list of proposed countries  \* @property {integer} amount\_tours - amount of tours  \* @property {integer} amount\_departments - amount of departments  \* @property {string} foundation\_date.required - date of founfation  \*/  class Tour\_operator {  constructor(id, name, countries, amount\_tours, amount\_departments, foundation\_date) {  this.id = id;  this.name = name;  this.countries = countries ;  this.amount\_tours=amount\_tours || 0;  this.amount\_departments=amount\_departments || 0;  this.foundation\_date=foundation\_date || "1970-01-01";  }  };  module.exports = Tour\_operator; |

|  |
| --- |
| **models/user.js** |
| /\*\*  \* @typedef User  \* @property {integer} id  \* @property {string} login.required - unique username  \* @property {string} fullname - display name (defaults to login)  \* @property {integer} role - user role: 0-regular 1-admin (defaults to 0)  \* @property {string} registeredAt.required -date of registration  \* @property {string} avaUrl - URL adress of profile picture  \* @property {boolean} isEnabled - indicates if user is active(defaults to true)  \*/  class User {  constructor(id, login, fullname, role, registeredAt, avaUrl, isEnabled) {  this.id = id;  this.login = login;  this.fullname = fullname || login;  this.role=role || 0;  this.registeredAt=registeredAt;  this.avaUrl=avaUrl || null;  this.isEnabled=isEnabled || true;  }  };  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 router = express.Router();**  **const userController = require('../controllers/users');**  **/\*\***  **\* 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>} 200 - a page with users**  **\* @returns {Error} 404 - Page not found**  **\*/**  **router**  **.get('', userController.getUsers)**  **.get('/:id', userController.getUserById);**  **module.exports = router;** |

|  |
| --- |
| **routes/operators.js** |
| **const express = require('express');**  **const router = express.Router();**  **const operatorController = require('../controllers/operators.js');**  **const bodyParser = require('body-parser');**  **const jsonParser = bodyParser.json();**  **/\*\***  **\* @route GET /api/operators**  **\* @group Operators - operator operations**  **\* @param {integer} page.query - page number**  **\* @param {integer} per\_page.query - items per page**  **\* @returns {Array.<Tour\_operator>} 200 - a page with operators**  **\*/**  **/\*\***  **\* @route GET /api/operators/{id}**  **\* @group Operators - operator operations**  **\* @param {integer} id.path.required - id of operator**  **\* @returns {Tour\_operator.model} 200 - Operator object**  **\* @returns {Error} 404 - Operator not found**  **\*/**  **/\*\***  **\* @route POST /api/operators**  **\* @group Operators - operator operations**  **\* @param {Tour\_operator.model} id.body.required - new Operator Object**  **\* @returns {Tour\_operator.model} 201 - Created operator object**  **\* @returns {Error} 400 - Invalid field data**  **\*/**  **/\*\***  **\* @route PUT /api/operators**  **\* @group Operators - operator operations**  **\* @param {Tour\_operator.model} object.body.required - new Operator Object**  **\* @returns {Tour\_operator.model} 201 - Updated game object**  **\* @returns {Error} 400 - Invalid field data**  **\* @returns {Error} 404 - Operator not found**  **\*/**  **/\*\***  **\* @route DELETE /api/operators/{id}**  **\* @group Operators - operator operations**  **\* @param {integer} id.path.required - id of the Operator**  **\* @returns {Tour\_operator.model} 201 - deleted Operator object**  **\* @returns {Error} 404 - Operator not found**  **\*/**  **router**  **.get("/:id", operatorController.getTour\_operatorById)**  **.get("", operatorController.getTour\_operators)**  **.post("/", jsonParser, operatorController.addTour\_operator)**  **.put("/", jsonParser, operatorController.updateTour\_operator)**  **.delete("/:id", operatorController.deleteTour\_operator);**  **module.exports = router;** |

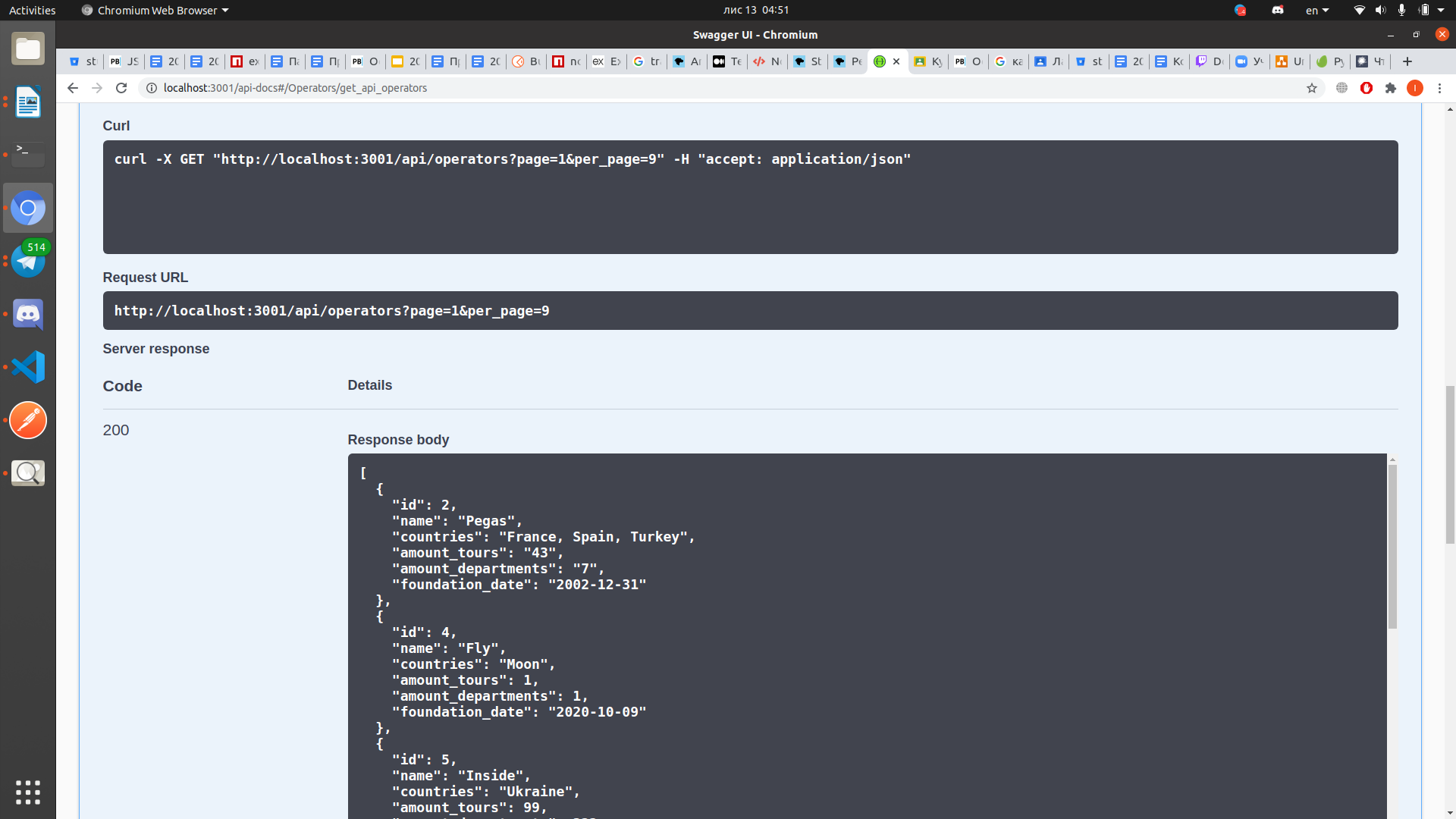
|  |
| --- |
| **controllers/medias.js** |
| **const media\_rep = require('../repositories/mediaRepository.js');**  **const mediaRepository = new media\_rep('data/media/', 'data/media.json');**  **module.exports = {**  **getMediaById(req, res) {**  **res.set("Content-Type", "application/json");**  **try {**  **let media = mediaRepository.getMediaById(req.params.id);**  **let responce = {**  **image: {data:media}**  **};**  **res.send(JSON.stringify(responce));**  **} 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}));**  **}**  **};** |

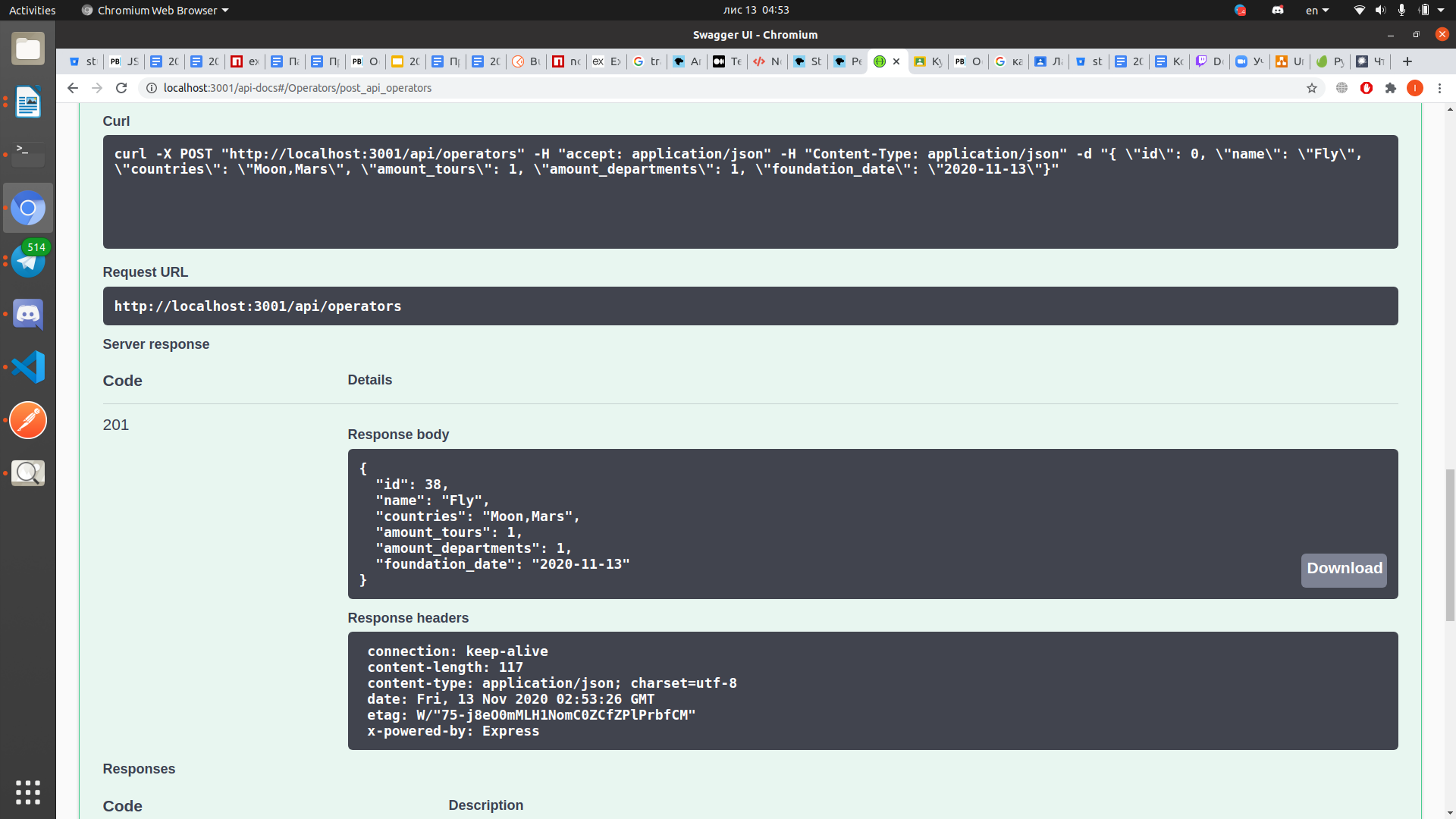
|  |
| --- |
| **controllers/users.js** |
| **const user\_rep = require('../repositories/userRepository.js');**  **const userRepository = new user\_rep("data/users.json");**  **module.exports = {**  **getUsers(req, res) {**  **res.set("Content-type", "application/json");**  **let page\_size = Number.parseInt(req.query.per\_page);**  **if(req.query.per\_page===undefined){**  **page\_size=2;**  **}**  **if(!Number.isInteger(page\_size) || page\_size<1 || page\_size>10 ){**  **return res.status(404).json({Error:"Page size should be natural number and <10"});**  **}**  **let page\_number = Number.parseInt(req.query.page);**  **if(req.query.page===undefined){**  **page\_number=1;**  **}**  **if(!Number.isInteger(page\_number) || page\_number<1){**  **return res.status(404).json({Error:"Page should be natural number"});**  **}**  **const users = userRepository.getUsers();**  **const items\_total = users.length;**  **if(page\_size \* (page\_number - 1)>=items\_total){**  **res.status(404).json({Error:"no such page exists"});**  **}**  **const page = users.slice(page\_size \* (page\_number - 1), page\_size \* (page\_number));**  **return res.status(200).json(page);**  **},**  **getUserById(req, res) {**  **res.set("Content-type", "application/json");**  **const userID=Number(req.params.id);**  **if(!Number.isInteger(userID) || userID<1){**  **return res.status(400).json({Error:"User id should be natural number"});**  **}**  **try{**  **const user=userRepository.getUserById(userID);**  **return res.status(200).json(user);**  **}**  **catch(error){**  **return res.status(404).json({Error:error.message});**  **}**  **}**  **};** |

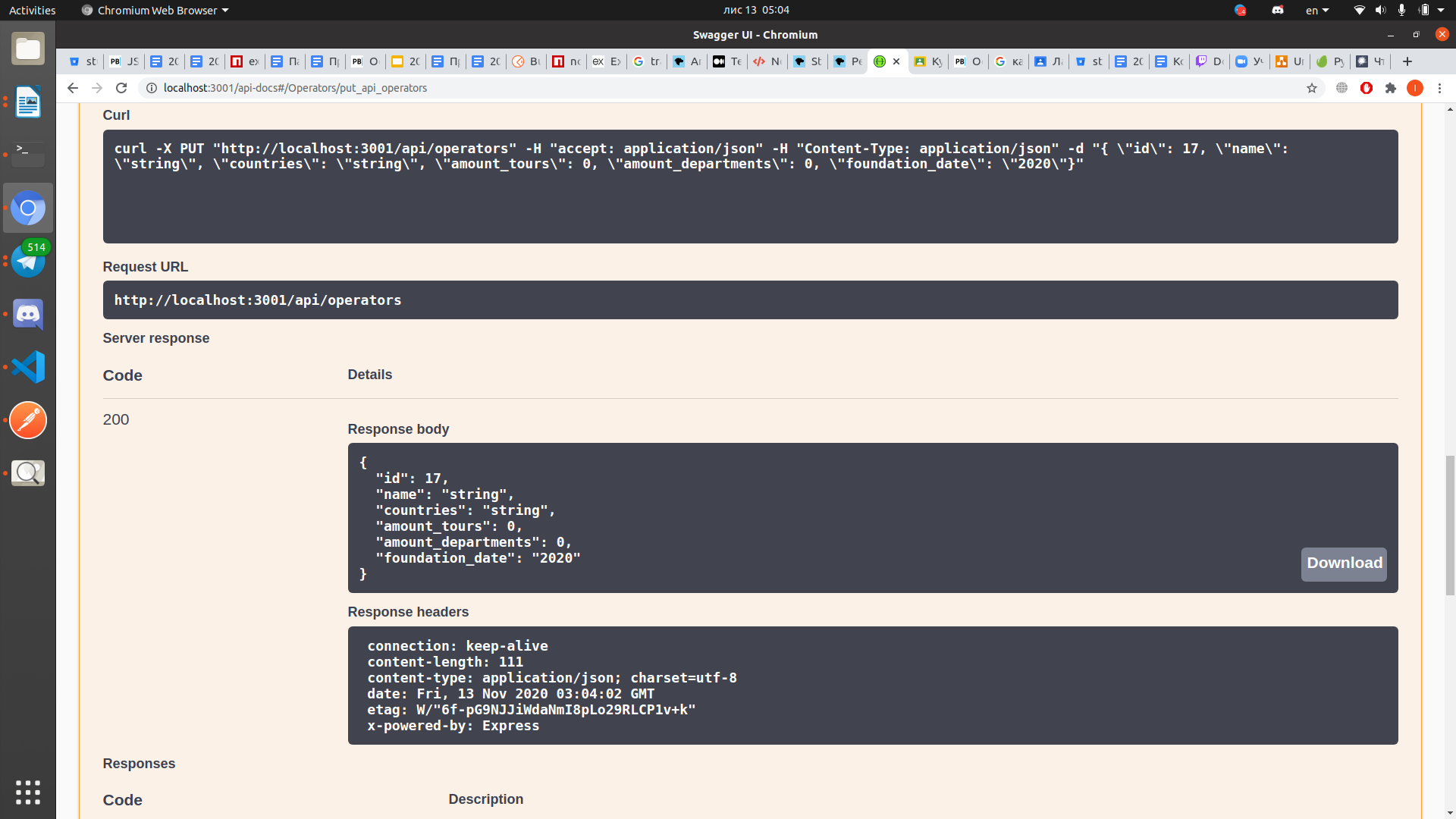
|  |
| --- |
| **controllers/operators.js** |
| **const op\_rep = require('../repositories/operatorRepository.js');**  **const operatorRepository = new op\_rep("data/tour\_operators.json");**  **const moment = require('moment');**  **const stringValidator = value => typeof (value) === "string";**  **const integerValidator = value => {**  **if (Number.isInteger(Number(value)) && Number(value) >= 0) {**  **return true;**  **}**  **return false;**  **};**  **const dateValidator = value => moment(value, moment.ISO\_8601).isValid();**  **const validators = {**  **id: integerValidator,**  **name: stringValidator,**  **countries: stringValidator,**  **amount\_tours: integerValidator,**  **amount\_departments: integerValidator,**  **foundation\_date: dateValidator**  **};**  **const errorMessages = {**  **id: "id should be inputed as integer",**  **name: "Name field should be string",**  **countries: "Countries field should be string",**  **amount\_tours: "Amount of yours should be inputed as integer",**  **amount\_departments: "Amount of departments should be integer",**  **foundation\_date: "Please enter date in ISO 8601 format "**  **};**  **module.exports = {**  **getTour\_operators(req, res) {**  **res.set("Content-type", "application/json");**  **let page\_size = Number.parseInt(req.query.per\_page);**  **if (req.query.per\_page === undefined) {**  **page\_size = 2;**  **}**  **if (!Number.isInteger(page\_size) || page\_size < 1 || page\_size > 10) {**  **return res.status(404).json({ Error: "Page size should be natural number and <10" });**  **}**  **let page\_number = Number.parseInt(req.query.page);**  **if (req.query.page === undefined) {**  **page\_number = 1;**  **}**  **if (!Number.isInteger(page\_number) || page\_number < 1) {**  **return res.status(404).json({ Error: "Page should be natural number" });**  **}**  **const operators = operatorRepository.getTour\_operators();**  **const items\_total = operators.length;**  **if (page\_size \* (page\_number - 1) >= items\_total) {**  **res.status(404).json({ Error: "no such page exists" });**  **}**  **const page = operators.slice(page\_size \* (page\_number - 1), page\_size \* (page\_number));**  **return res.status(200).json(page);**  **},**  **getTour\_operatorById(req, res) {**  **res.set("Content-type", "application/json");**  **const operatorID = Number(req.params.id);**  **if (!Number.isInteger(operatorID) || operatorID < 1) {**  **return res.status(400).json({ Error: "Operator id should be natural number" });**  **}**  **try {**  **const user = operatorRepository.getTour\_operatorById(operatorID);**  **return res.status(200).json(user);**  **}**  **catch (error) {**  **return res.status(404).json({ Error: error.message });**  **}**  **},**  **addTour\_operator(req, res) {**  **res.set("Content-type", "application/json");**  **const operator = req.body;**  **const properties = ["id", "name", "countries", "amount\_tours", "amount\_departments", "foundation\_date"];**  **if (!operator.name || !operator.countries) return res.status(400).json({ Error: "Operator must contain 'name' and 'countries' properties" });**  **for (const [key, value] of Object.entries(operator)) {**  **if (!properties.includes(key)) return res.status(400).json({ Error: `property "${key}" doesnt exist on operator entity` });**  **if (!validators[key](String(value))) return res.status(400).json({ message: errorMessages[key] });**  **}**  **const id = operatorRepository.addTour\_operator(operator);**  **const new\_op = operatorRepository.getTour\_operatorById(id);**  **return res.status(201).json(new\_op);**  **},**  **updateTour\_operator(req, res) {**  **res.set("Content-type", "application/json");**  **const operator = req.body;**  **const properties = ["id", "name", "countries", "amount\_tours", "amount\_departments", "foundation\_date"];**  **if (!operator.name || !operator.countries) return res.status(400).json({ Error: "Operator must contain 'name' and 'countries' properties" });**  **for (const [key, value] of Object.entries(operator)) {**  **if (!properties.includes(key)) return res.status(400).json({ Error: `property "${key}" doesnt exist on operator entity` });**  **if (!validators[key](String(value))) return res.status(400).json({ message: errorMessages[key] });**  **}**  **operatorRepository.updateTour\_operator(operator);**  **try {**  **const new\_op = operatorRepository.getTour\_operatorById(operator.id);**  **return res.status(200).json(new\_op);**  **}**  **catch (error) {**  **return res.status(404).json({ Error: error.message });**  **}**  **},**  **deleteTour\_operator(req, res) {**  **res.set("Content-type", "application/json");**  **let id = parseInt(req.params.id);**  **try {**  **const op = operatorRepository.getTour\_operatorById(id);**  **operatorRepository.deleteTour\_operator(op.id);**  **return res.status(200).json(op);**  **}**  **catch (error) {**  **return res.status(404).json({ Error: error.message });**  **}**  **}**  **};** |

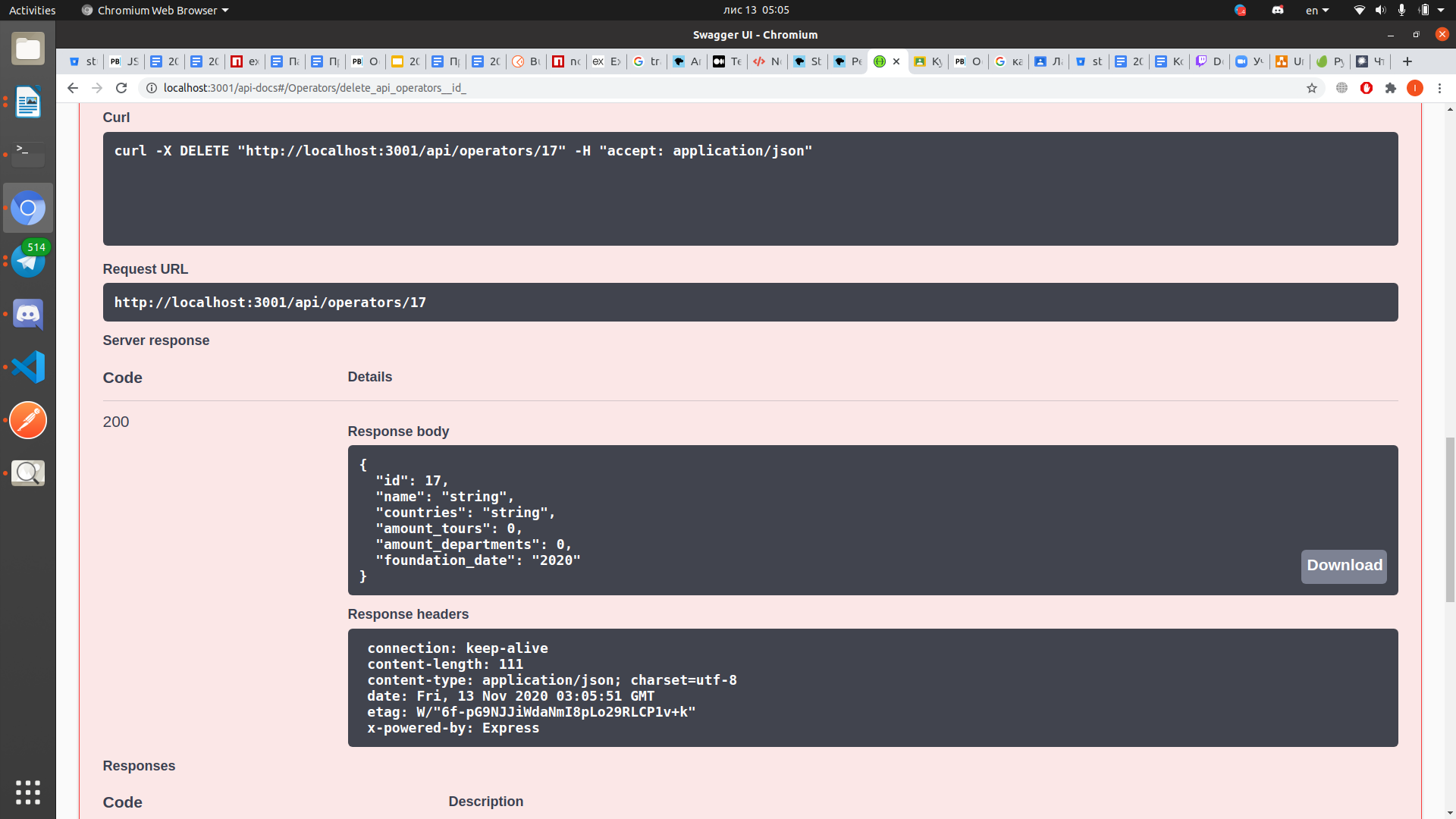
|  |
| --- |
| **mediaStorage.js** |
| **const fs = require('fs');**  **const JsonStorage = require('./jsonStorage.js');**  **class MediaStorage {**  **constructor(mediaPath, mediaInfoPath) {**  **this.mediaPath = mediaPath;**  **this.jsonstorage = new JsonStorage(mediaInfoPath);**  **}**  **readMedia(id) {**  **let mediaInfo = this.jsonstorage.readItems();**  **let path;**  **for (let i = 0; i < mediaInfo.length; i++) {**  **console.log(mediaInfo[i].id);**  **if (mediaInfo[i].id === Number(id)) {**  **path = mediaInfo[i].path;**  **break;**  **}**  **}**  **let mediaFile = fs.readFileSync(`${path}`, function(err, data) {**  **if (err) {**  **throw new Error(data);**  **}**  **});**  **return mediaFile;**  **}**  **writeMedia(media) {**  **let mediaInfo = this.jsonstorage.readItems();**  **mediaInfo.push({**  **id: this.jsonstorage.getnextId(),**  **path: `${this.mediaPath}${media.name}`**  **});**  **this.jsonstorage.incrementNextId();**  **this.jsonstorage.writeItems(mediaInfo);**  **fs.writeFileSync(`${this.mediaPath}${media.name}`, media.data);**  **return this.jsonstorage.getnextId()-1;**  **}**  **};**  **module.exports = MediaStorage;** |

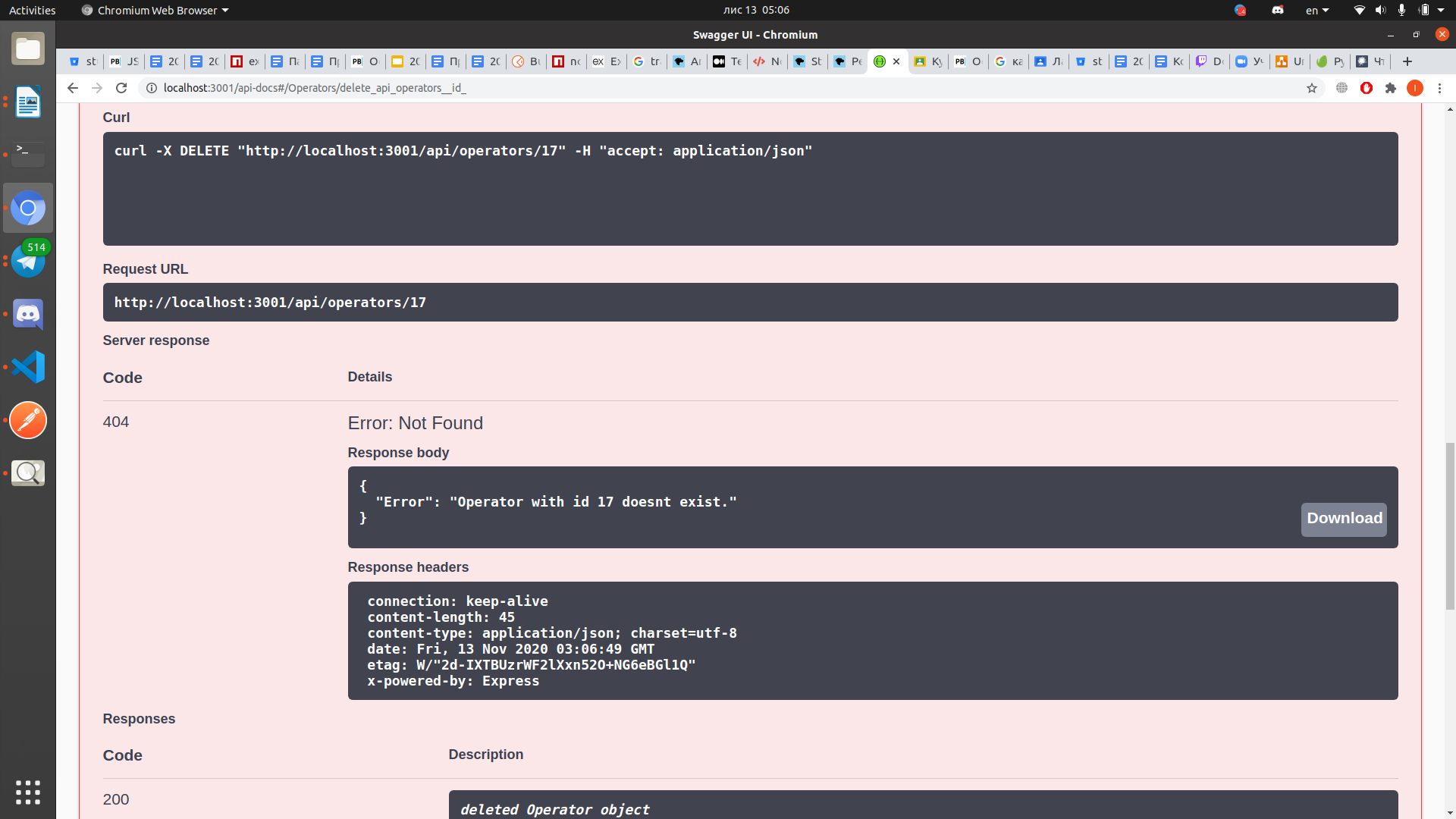
**Зображення Swagger сторінки /api-docs із прикладами взаємодії з JSON REST API**

****

****

****

****

****

**Висновки**

Виконавши дану лабораторну роботу я навчився 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.