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 “Веб-програмування”

# тема “ JSON HTTP API веб-сервер”

|  |  |  |
| --- | --- | --- |
| Виконав  Студент II курсу  групи КП-91  Деркач СтаніславДмитрович  (*прізвище, ім’я, по батькові*)  варіант 22 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

### **Мета**

### 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 залежностей:**

### 

**Тексти коду програм**

|  |
| --- |
| **app.js** |
| const express = require("express");  const bodyParser = require("body-parser");  const router = require("./routes/api");  const app = express();  app.use(bodyParser.urlencoded({ extended: true }));  app.use(bodyParser.json());  app.use("/api/", router);  const expressSwaggerGenerator = require('express-swagger-generator');  const expressSwagger = expressSwaggerGenerator(app);    const options = {      swaggerDefinition: {          info: {              description: 'Working with Users and Wepons',              title: 'LABA 2',              version: '1.0.0',          },          host: 'localhost:3000',          produces: [ "application/json" ],      },      basedir: \_\_dirname,      files: ['./routes/\*\*/\*.js', '../models/\*\*/\*.js'],  };  expressSwagger(options);  app.listen(3000, ()=>{console.log("Server is ready");}); |

|  |
| --- |
| **mediaRepository.js** |
| const JsonStorage = require('../jsonStorage');  const fs = require('fs');  class MediaRepository {        constructor(filePath) {          this.storage = new JsonStorage(filePath+'.json');          this.folderPath = filePath;      }      get getPath()      {          return this.folderPath;      }      getPicturePath(id)      {          for (const format of this.getFormats())          {              const path = this.folderPath + '/' + id + '.' + format;              if (fs.existsSync(path))              {                  return path;              }          }          return undefined;      }      getFormats()      {          return this.storage.readItems();      }      getNextId()      {          return this.storage.nextId;      }      incrementNextId()      {          this.storage.incrementNextId();      }  };    module.exports = MediaRepository; |

|  |
| --- |
| **Сontrollers/media.js** |
| const MediaRepository = require('../../repositories/mediaRepository');  const mediaRepository = new MediaRepository('./data/media');  const multer = require('multer');  const storage = multer.diskStorage({      destination: function (req, file, cb) {          cb(null, mediaRepository.getPath)      },      filename: function (req, file, cb) {          const fileFormat = file.mimetype.split('/')[1];          cb(null, String(mediaRepository.getNextId())+"."+fileFormat);      }  });  const upload = multer({ storage: storage }).any();  module.exports = {      getPicture(req, res) {          try          {              const picturePath = mediaRepository.getPicturePath(req.params.id);              if (!picturePath && !isNaN(req.params.id))              {                  res.status(404).send({ message: 'Picture not found' });              }              else if (!picturePath)              {                  res.status(400).send({ message: 'Bad request' });              }              else              {                  res.status(200).sendFile(picturePath, { root: '.' });              }          }          catch(err)          {              console.log('err ', err.message);              res.status(500).send({ pictureId: null, message: 'Server error' });          }      },      addPicture(req, res) {          upload(req, res, (err) => {              if (err) {                  console.log('err ', err.message);                  res.status(500).send({ pictureId: null, message: 'Server error' });              } else if (req.files) {                  const fileFormat = req.files[0].mimetype.split('/')[1];                  if (!mediaRepository.getFormats().includes(fileFormat))                  {                      res.status(400).send({ message: 'Bad request' });                      return;                  }                  res.status(201).send({ pictureId: mediaRepository.getNextId(), message: 'Picture has been uploaded' })                  mediaRepository.incrementNextId();              } else {                  res.status(400).send({ message: 'Bad request' });              }          })      }  }; |

|  |
| --- |
| **Controllers/users.js** |
| const UserRepository = require('../../repositories/userRepository');  const userStorage = new UserRepository('./data/users.json');  function pagination(items, page=1, per\_page=2){      const start = (page-1)\*per\_page;      return items.slice(start, start+per\_page);  }  module.exports={       getUsers(req, res){          console.log(req.query);          const items = userStorage.getUsers();          let page = req.query.page;          let per = req.query.per\_page;          if (typeof page === 'undefined') page = 1, console.log("u1");              else page=Number(page);          if (typeof per === 'undefined') per = 2, console.log("u2");              else per=Number(per);          if (isNaN(page) || isNaN(per) || page <=0 || per <=0){              res.status(400).send({});              return;          }          const resItems = pagination(items, page, per);          res.set("Content-type", "application/json");          res.send(resItems);      },      getUserById(req, res){          const str\_id = req.params.id;          const id = Number(str\_id);          const user = userStorage.getUserById(id);          if (user===null) {              res.status(404).send({});              return;          }          res.send(user);      }  }; |

|  |
| --- |
| **controllers/wepons.js** |
| const WeponRepository = require('../../repositories/weponRepository');  const weponStorage = new WeponRepository('./data/wepons.json');  function pagination(items, page=1, per\_page=2){      const start = (page-1)\*per\_page;      return items.slice(start, start+per\_page);  }  module.exports={       getWepons(req, res){          console.log(req.query);          const items = weponStorage.getWepons();          let page = req.query.page;          let per = req.query.per\_page;          if (typeof page === 'undefined') page = 1, console.log("u1");              else page=Number(page);          if (typeof per === 'undefined') per = 2, console.log("u2");              else per=Number(per);          if (isNaN(page) || isNaN(per) || page <=0 || per <=0){              res.status(400).send({});              return;          }          const resItems = pagination(items, page, per);          res.set("Content-type", "application/json");          res.send(resItems);      },      getWeponById(req, res){          const str\_id = req.params.id;          const id = Number(str\_id);          const wepon = weponStorage.getWeponById(id);          if (wepon===null) {              res.status(404).send({});              return;          }          res.set("Content-type", "application/json");          res.send(wepon);      },      deleteWepon(req, res){          const str\_id = req.params.id;          const id = Number(str\_id);          const delItem = weponStorage.deleteWepon(id);          if (delItem === null) {              res.status(404).send({});              return;          }          res.set("Content-type", "application/json");          res.send(delItem);      },      updateWepon(req, res){          // console.log(req.body);          const str\_id=req.body.id;          const id = Number(str\_id);          if (isNaN(id)) {              res.status(400).send({});              return;          }          const wepon\_ = weponStorage.getWeponById(id);          // console.log(wepon\_);          if (wepon\_ ===null) {              res.status(404).send({});              return;          }          let wepon={              id: wepon\_.id,              name: wepon\_.name,              author: wepon\_.author,              damage: wepon\_.damage,              speed: wepon\_.speed,              createdAt: wepon\_.date          };          const name = req.body.name;          const author = req.body.author;          const str\_damage = req.body.damage;          const str\_speed = req.body.speed;          if (name!=="") wepon.name=name;          if (author!=="") wepon.author=author;          const damage = Number(str\_damage);          const speed = Number(str\_speed);          if (isNaN(damage)){              res.set("Content-type", "application/json");              res.status(400).send({});              return;          }          if (str\_damage!=="") wepon.damage=Number(damage);          if (isNaN(speed)){              res.set("Content-type", "application/json");              res.status(400).send({});              return;          }          if (str\_speed!=="") wepon.speed=Number(speed);          weponStorage.updateWepon(wepon);          res.set("Content-type", "application/json");          res.send(wepon\_);      },      addWepon(req,res){          const name = req.body.name;          const author = req.body.author;          const damage = req.body.damage;          const speed = req.body.speed;          const date = new Date();          let id;          if (name.length>0 && author.length>0 &&              !isNaN(damage) && damage>=0 && !isNaN(speed) && speed>0){              id = weponStorage.addWepon(name, author, Number(damage), Number(speed), date);              let wepon= {                  'id': id,                  'name': name,                  'author': author,                  'damage': Number(damage),                  'speed': Number(speed),                  'date': date              };              res.set("Content-type", "application/json");              res.send(wepon);          }          else {              res.set("Content-type", "application/json");              res.status(400).send({});          }      }  }; |

|  |
| --- |
| **Models/user.js** |
| /\*\*   \* @typedef User   \* @property {integer} id.requested   \* @property {string} login.requested   \* @property {string} fullname.requested   \* @property {integer} role.requested - level of access   \* @property {string} registeredAt.requested   \* @property {string} avaUrl.requested - url adress if avatar   \* @property {boolean} isEnabled.requested   \*/  class User{      constructor (id, login, fullname, role, registeredAt, avaUrl, isEnabled){          this.id=id;          this.login=login;          this.fullname=fullname;          this.role=role;          this.registeredAt=registeredAt;          this.avaUrl=avaUrl;          this.isEnabled=isEnabled;      }  }  module.exports=User; |

|  |
| --- |
| **Models/wepons.js** |
| /\*\*   \* @typedef Wepon   \* @property {integer} id.requested   \* @property {string} name.requested   \* @property {string} author.requested   \* @property {integer} damage.requested   \* @property {integer} speed.requested - speed of using   \* @property {integer} createdAt.requested   \*/  class Wepon{      constructor(id, name, author, damage, speed, createdAt){          [this.id, this.name, this.author, this.damage, this.speed, this.createdAt]          = [id, name, author, damage, speed, createdAt];      }  }  module.exports=Wepon; |

|  |
| --- |
| **Routers/api.js** |
| const express = require("express");  const router = express.Router();  const uRouter = require("./users");  const wRouter = require("./wepons");  const mRouter = require("./media");  router.use("/users/", uRouter);  router.use("/wepons/", wRouter);  router.use("/media/", mRouter);  module.exports=router; |

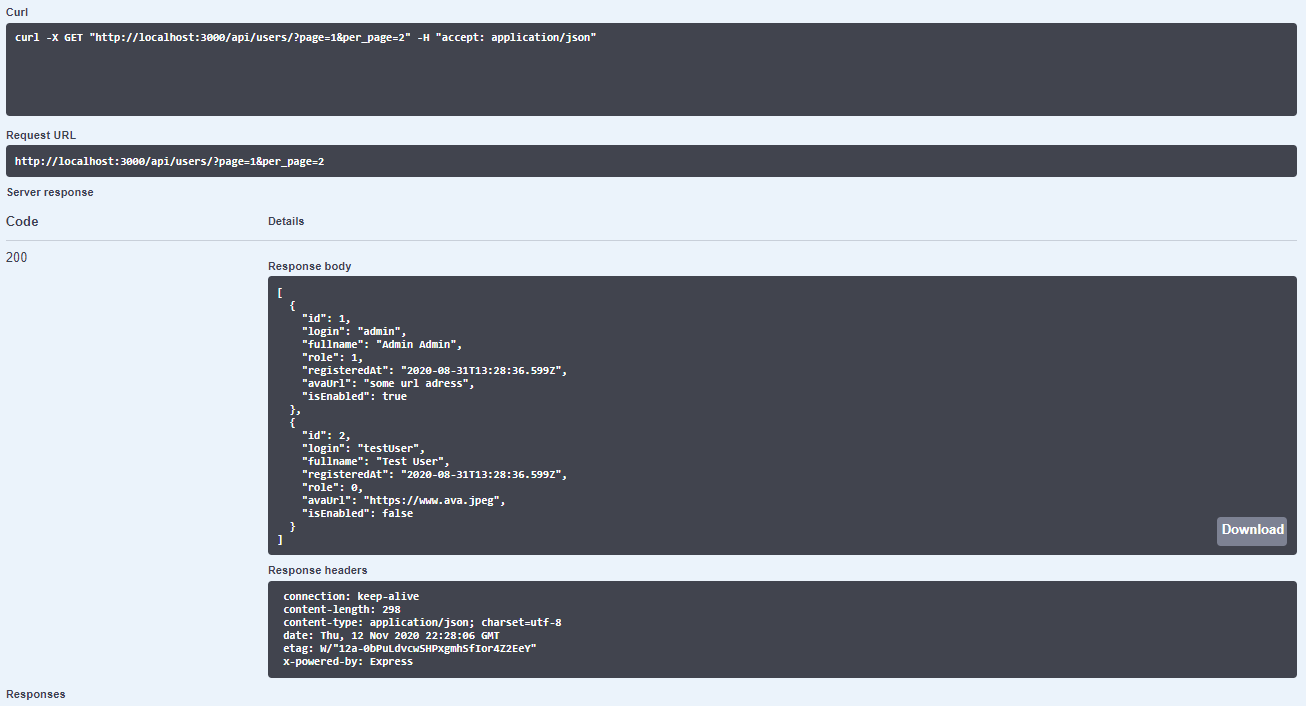
|  |
| --- |
| **Routers/media.js** |
| const mediaController = require('../controllers/media');  const router = require('express').Router();  /\*\*   \* Add a picture   \* @route POST /api/media   \* @group Media - upload images   \* @consumes multipart/form-data   \* @param {file} image.formData.required - uploaded image   \* @returns {Object} 201 - added image   \* @returns {Error} 400 - Bad request   \* @returns {Error} 500 - Server error   \*/  router.post("/", mediaController.addPicture);  /\*\*   \* Get a picture by id   \* @route GET /api/media/{id}   \* @group Media - get pictures   \* @param {integer} id.path.required - id of the picture - eg: 1   \* @returns {file} 200 - return image   \* @returns {Error} 400 - Bad request   \* @returns {Error} 404 - Picture not found   \* @returns {Error} 500 - Server error   \*/  router.get("/:id", mediaController.getPicture);  module.exports = router; |

|  |
| --- |
| **Routers/userRouter.js** |
| const express = require('express');  const router = express.Router();  const userController = require('../controllers/users');  /\*\*   \* Get all users with pagination   \* @route GET /api/users/   \* @group Users - user operations   \* @param {integer} page.query - Number of page   \* @param {integer} per\_page.query - Count of items on one page   \* @returns {Array.<User>} 200 - User objects   \* @returns {Error} 400 - Bad request   \*/  router.get('/', userController.getUsers);  /\*\*   \* Get user by id   \* @route GET /api/users/{id}   \* @group Users - user operations   \* @param {integer} id.params - user's id   \* @returns {User.model} 200 - User object   \* @returns {Error} 404 - user not found   \*/  router.get('/:id(\\d+)', userController.getUserById);  module.exports = router; |

|  |
| --- |
| **Routers/wepons.js** |
| const express = require('express');  const weponController = require('../controllers/wepons');  const router = express.Router();  /\*\*   \* Get all wepons with pagination   \* @route GET /api/wepons/   \* @group Wepons - wepon operations   \* @param {integer} page.query - Number of page   \* @param {integer} per\_page.query - Count of items on one page   \* @returns {Array.<Wepon>} 200 - Wepon objects   \* @returns {Error} 400 - Bad request   \*/  router.get('/', weponController.getWepons);  /\*\*   \* Get wepon by id   \* @route GET /api/wepons/{id}   \* @group Wepons - wepon operations   \* @param {integer} id.params.request - wepon's id   \* @returns {Wepon.model} 200 - Wepon object   \* @returns {Error} 404 - wepon not found   \*/  router.get('/:id(\\d+)', weponController.getWeponById);  /\*\*   \* Add new wepon   \* @route post /api/wepons/{id}   \* @group Wepons - wepon operations   \* @param {string} name.body.request - future wepon's name   \* @param {string} author.body.request - future wepon's author   \* @param {integer} damage.body.request - future wepon's damage   \* @param {integer} speed.body.request - future wepon's name   \* @returns {Wepon.model} 200 - Wepon object   \* @returns {Error} 400 - Bad request   \*/  router.post('/', weponController.addWepon);  /\*\*   \* Update wepon (leave parametr empty to leave the same value)   \* @route put /api/wepons/{id}   \* @group Wepons - wepon operations   \* @param {integer} id.params.request - wepon's id   \* @param {string} name.params.request - new wepon's name   \* @param {string} author.params.request - new wepon's author   \* @param {integer} damage.params.request - new wepon's damage   \* @param {integer} speed.params.request - new wepon's name   \* @returns {Wepon.model} 200 - Wepon object   \* @returns {Error} 400 - Bad request   \* @returns {Error} 404 - wepon not found   \*/  router.put('/',weponController.updateWepon);  /\*\*   \* Delete wepon by id   \* @route DELETE /api/wepons/{id}   \* @group Wepons - wepon operations   \* @param {integer} id.params.request - wepon's id   \* @returns {Wepon.model} 200 - Wepon object   \* @returns {Error} 404 - wepon not found   \*/  router.delete('/:id(\\d+)', weponController.deleteWepon);  module.exports = router; |

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

Отримання першої сторінки користувачів на 2 елементи



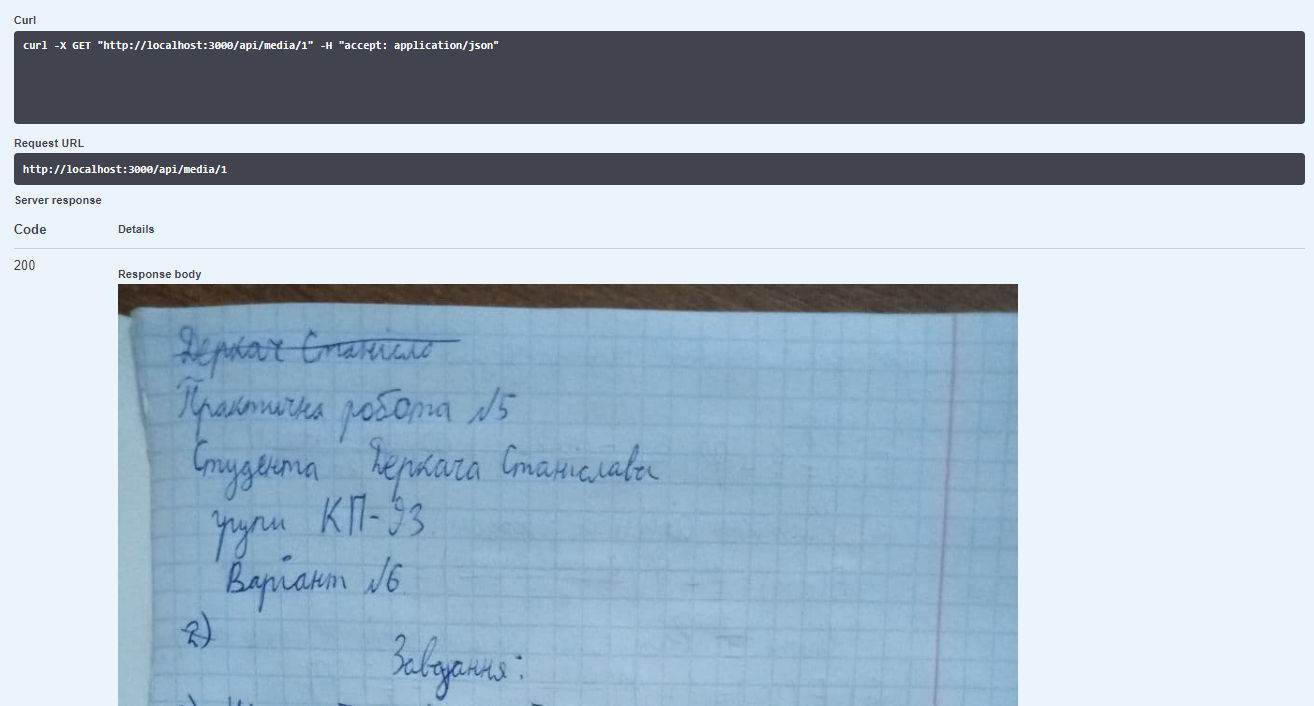
Додавання нової сутності



Видалення сутності з id 7



Отримання зображення за id



**Висновки**

Виконуючи цю лабораторну роботу я навчився створювати простий JSON HTTP API веб-сервер, що надає доступ до ресурсів зі сховища, документувати API за допомогою Swagger, вивчив коди статутів помилок серверів