

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  студент 2 курсу  групи КП-93  Торговських Олександр  варіант № 21 |  | 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.

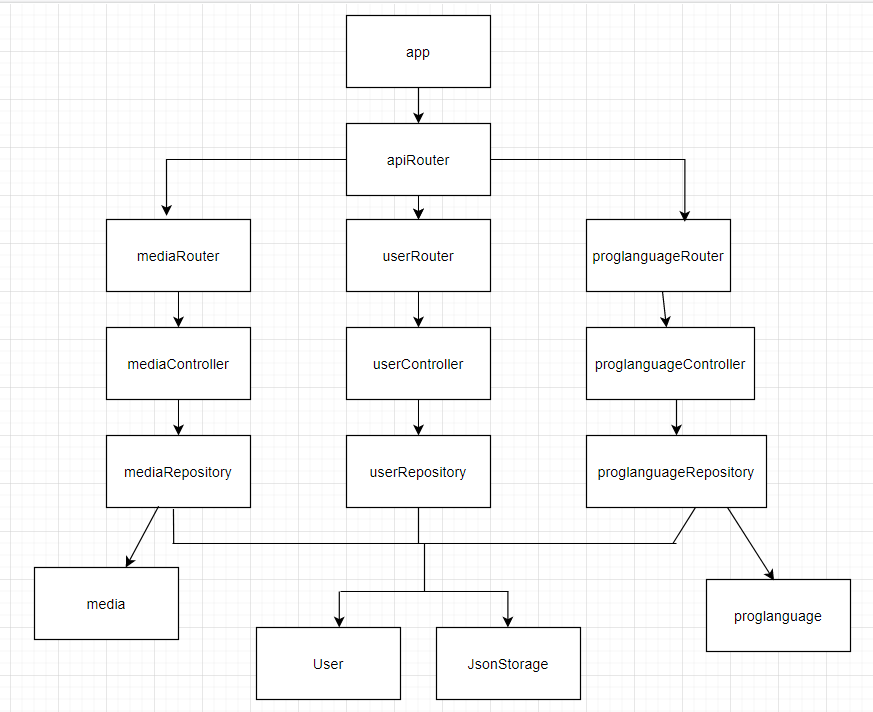
### Завдання

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 модулів**



**Код**

|  |
| --- |
| **app.js** |
| const express = require('express');  const app = express();  const apiRouter = require('./routes/apirouter');  port = process.env.port || 4444;    const expressSwaggerGenerator = require('express-swagger-generator');  const expressSwagger = expressSwaggerGenerator(app);  const options = {  swaggerDefinition: {  info: {  description: 'http api server',  title: 'lab2',  version: '1.0.0',  },  host: 'localhost:4444',  produces: [ "application/json" ],  },  basedir: \_\_dirname,  files: ['./routes/\*\*/\*.js', './models/\*\*/\*.js'],  };  expressSwagger(options);    app.use('/api' , apiRouter);    app.listen(port , err => {  if(err)  {  return console.log("ERROR" , err);  }  console.log('listening on port ${port}');  }); |

|  |
| --- |
| **models/media.js** |
| /\*\*  \* @typedef Media  \* @property {integer} id  \* @property {string} path - path to file  \*/    class Media{  constructor( id , path)  {  this.id = id; //number  this.path = path; //string  }  };    module.exports = Media |

|  |
| --- |
| **mediaRepository.js** |
| const JsonStorage = require('../jsonStorage');  const Media = require('../models/media');    class MediaRepository  {  constructor(filepath) {  this.storage = new JsonStorage(filepath);  }    addMedia(entityModel) {  let medias = this.getMedias();  const temp = this.storage.nextId;  const new\_media = new Media(temp , entityModel.path);  const next\_id = this.storage.incrementNextId();  if (medias[0] === null) {  medias[0] = new\_media;  } else {  medias[medias.length] = new\_media;  }  this.storage.writeItems(medias, next\_id);  return temp;  }    getMedias() {  let medias = [];  const items = this.storage.readItems();  if (items === null) {  return medias;  }  for (const item of items) {  medias.push(new Media(item.id, item.path));  }  return medias;  }    getMediaById(entityId) {  const items = this.storage.readItems();  if(items === null)  {  return null;  }  for(const item of items)  {  if(item.id == entityId)  {  return new Media(item.id , item.path);  }  }  return null;  }    };    module.exports = MediaRepository; |

|  |
| --- |
| **mediaController.js** |
| const MediaRepository = require('../repositories/mediaRepository');  const Media = require('../models/media');  const mediaRepository = new MediaRepository('data/medias.json');  const fs = require('fs');  const path = require('path');  module.exports =  {  postMedia(req , res)  {  if(req.file == undefined)  {  res.status(400).send("400 Bad request");  }else{  const temp = new Media(0 , req.file.path);  temp.id = mediaRepository.addMedia(temp);  res.status(201);  res.json(temp.id);  }  },    getImage(req , res)  {  const imageId = parseInt(req.params.id);  const image = mediaRepository.getMediaById(imageId);  if(image == null)  {  res.status(404).send("404 Not found");  }  fs.createReadStream(path.resolve(\_\_dirname + '/..' , image.path)).pipe(res);  }  }; |

|  |
| --- |
| **userController.js** |
| const UserRepository = require('../repositories/userRepository');  const userRepository = new UserRepository("data/users.json");    module.exports =  {  getUser(req , res)  {  const userId = parseInt(req.params.id);  const user = userRepository.getUserById(userId);  if(user == null)  {  res.status(404).send("404 Not found");  }  res.json(user);  },    getUsers(req , res)  {  if(req.query.page == null || req.query.size == null)  {  res.status(400).send("400 Bad request");  }else{  const page = parseInt(req.query.page);  const amount = parseInt(req.query.size);  const offset = (page - 1)\*amount;    const users = userRepository.getUsers();  const new\_users = users.slice(offset , offset+amount);  res.json(new\_users);  }  },  }; |

|  |
| --- |
| **proglanguagesController.js** |
| const ProgLanguage = require('./../models/ProgLanguage.js');  const ProgLanguageRepository = require('./../repositories/ProgLanguageRepository');  const { json } = require('body-parser');  const proglanguageRepository = new ProgLanguageRepository('data/proglanguages.json')    module.exports =  {  getProglanguage(req , res)  {  const proglanguageId = parseInt(req.params.id);  const proglanguage = proglanguageRepository.getProgLanguagesById(proglanguageId);  if(proglanguage == null)  {  res.status(404).send("404 Not found");  }else{  res.json(proglanguage);  }  },    getProglanguages(req , res)  {  if(req.query.page == null || req.query.size == null)  {  res.status(400).send("400 Bad request");  }else{  const page = parseInt(req.query.page);  const amount = parseInt(req.query.size);  const offset = (page - 1)\*amount;  const proglanguages = proglanguageRepository.getProgLanguages();  const new\_proglanguageRepository = proglanguages.slice(offset , offset + amount);  res.json(new\_proglanguageRepository);  }  },    addProglanguage(req , res)  {  if(req.body.name == null || req.body.founderName == null || req.body.lastversion == null || req.body.countofusers == null|| req.body.lasteditData == null)  {  res.status(400).send("400 Bad request");  }else{  const x = new ProgLanguage(0 , req.body.name , req.body.founderName , req.body.lastversion , req.body.countofusers , req.body.lasteditData);  x.id = proglanguageRepository.addProgLanguage(x);  res.status(201);  res.json(x);  }  },    updateProglanguage(req , res)  {  if(req.body.name == null || req.body.founderName == null || req.body.lastversion == null || req.body.countofusers == null|| req.body.lasteditData == null)  {  res.status(400).send("400 Bad request");  }else{  const x = new ProgLanguage( req.body.id , req.body.name , req.body.founderName , req.body.lastversion , req.body.countofusers , req.body.lasteditData);  if(proglanguageRepository.updateProgLanguages(x) == null)  {  res.status(404).send("404 Not found");  }else{  res.json(x);  }  }  },    deleteProglanguage(req ,res)  {  const proglanguageId = parseInt(req.params.id);  const temp = proglanguageRepository.getProgLanguagesById(proglanguageId);  if(proglanguageRepository.deleteProgLanguages(proglanguageId) == null)  {  res.status(404).send("404 Not found");  }else{  res.json(temp);  }  }  }; |

|  |
| --- |
| **mediarouter.js** |
| const router = require('express').Router();  const mediaController = require('./../controllers/mediaController');    router  /\*\*  \* @route POST /api/media  \* @group Medias - media operations  \* @returns {integer} id.body.requires - media id  \*/  .post("/", mediaController.postMedia)  /\*\*  \* @route GET /api/medias  \* @group Medias - media operations  \* @consumes multipart/form-data  \* @returns {Media.model} Media - media image  \* @returns {Error} 404 - Not found  \*/  .get("/:id" , mediaController.getImage)    module.exports = router; |

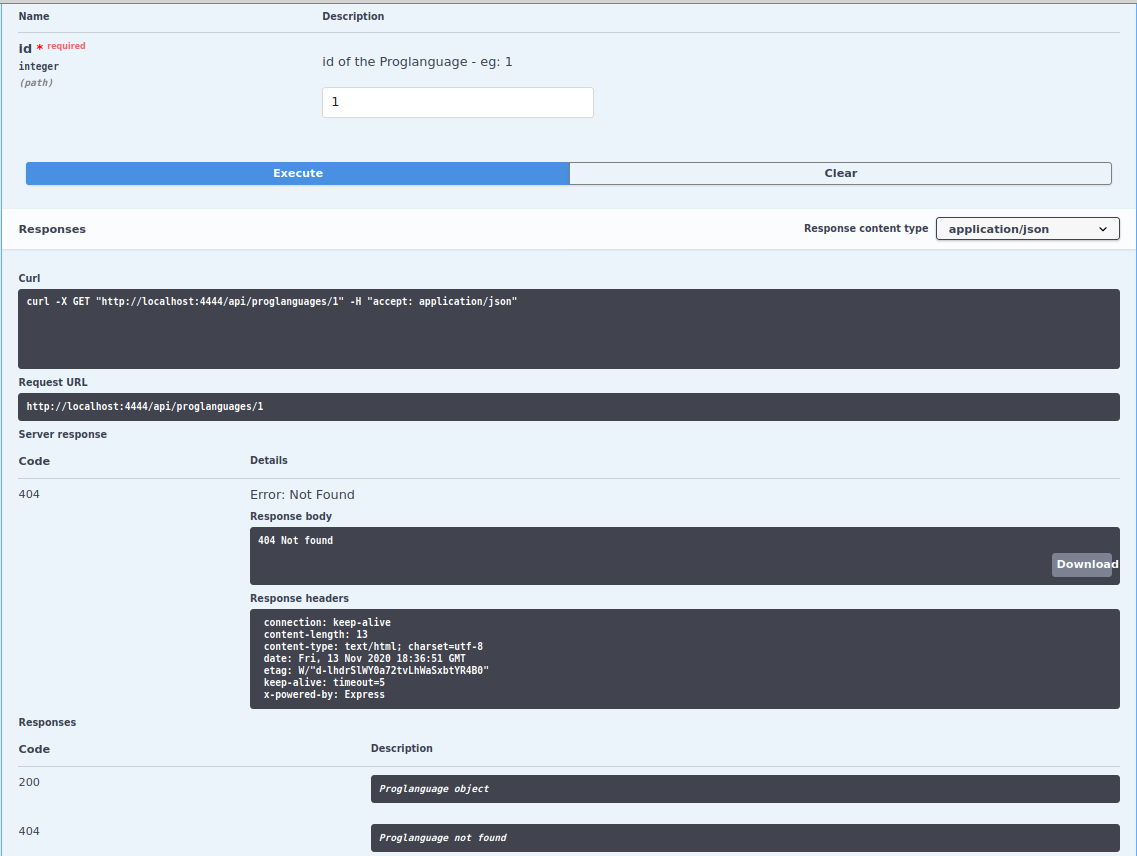
|  |
| --- |
| **apirouter.js** |
| const router = require('express').Router();  const userRoute = require('./users.js');  const proglanguageRoute = require('./ProgLanguageRoute.js');  const bodyparser = require('body-parser');  const mediaRouter = require('./mediaRouter');    const multer = require('multer');    router.use(multer({dest:"./data/media"}).single("image"))    router.use(bodyparser.json());  router.use(bodyparser.urlencoded({ extended: true }));    router.use('/media' , mediaRouter)  router.use('/users' , userRoute);  router.use('/proglanguages' , proglanguageRoute);    module.exports = router; |

|  |
| --- |
| **proglanguagesRouter.js** |
| const router = require('express').Router();    const proglanguageController = require('./../controllers/proglanguageControllers');    router  /\*\*  \* @route GET /api/proglanguages/{id}  \* @group Proglanguages - proglanguage operations  \* @param {integer} id.path.required - id of the Proglanguage - eg: 1  \* @returns {Proglanguage.model} 200 - Proglanguage object  \* @returns {Error} 404 - Proglanguage not found  \*/  .get("/:id", proglanguageController.getProglanguage)  /\*\*  \* @route GET /api/proglanguages?page={page}&size={size}  \* @group Proglanguages - proglanguage operations  \* @param {integer} page.path.required - page  \* @param {integer} size.path.required - size  \* @returns {Array.<Proglanguage>} Proglanguage - all proglanguages  \*/  .get("/", proglanguageController.getProglanguages)  /\*\*  \* @route POST /api/proglanguages  \* @group Proglanguages - proglanguage operations  \* @param {Proglanguage.model} id.body.required - new Proglanguage object  \* @returns {Proglanguage.model} 201 - added Proglanguage object  \*/  .post("/" , proglanguageController.addProglanguage)  /\*\*  \* @route PUT /api/proglanguages  \* @group Proglanguages - proglanguage operations  \* @param {Proglanguage.model} id.body.required - new Proglanguage object  \* @returns {Proglanguage.model} 200 - changed Proglanguage object  \*/  .put("/" , proglanguageController.updateProglanguage)  /\*\*  \* @route DELETE /api/proglanguages/{id}  \* @group Proglanguages - proglanguage operations  \* @param {integer} id.path.required - id of the Proglanguage - eg: 1  \* @returns {Proglanguage.model} 200 - deleted Proglanguage object  \* @returns {Error} 404 - Proglanguage not found  \*/  .delete("/:id" , proglanguageController.deleteProglanguage)    module.exports = router; |

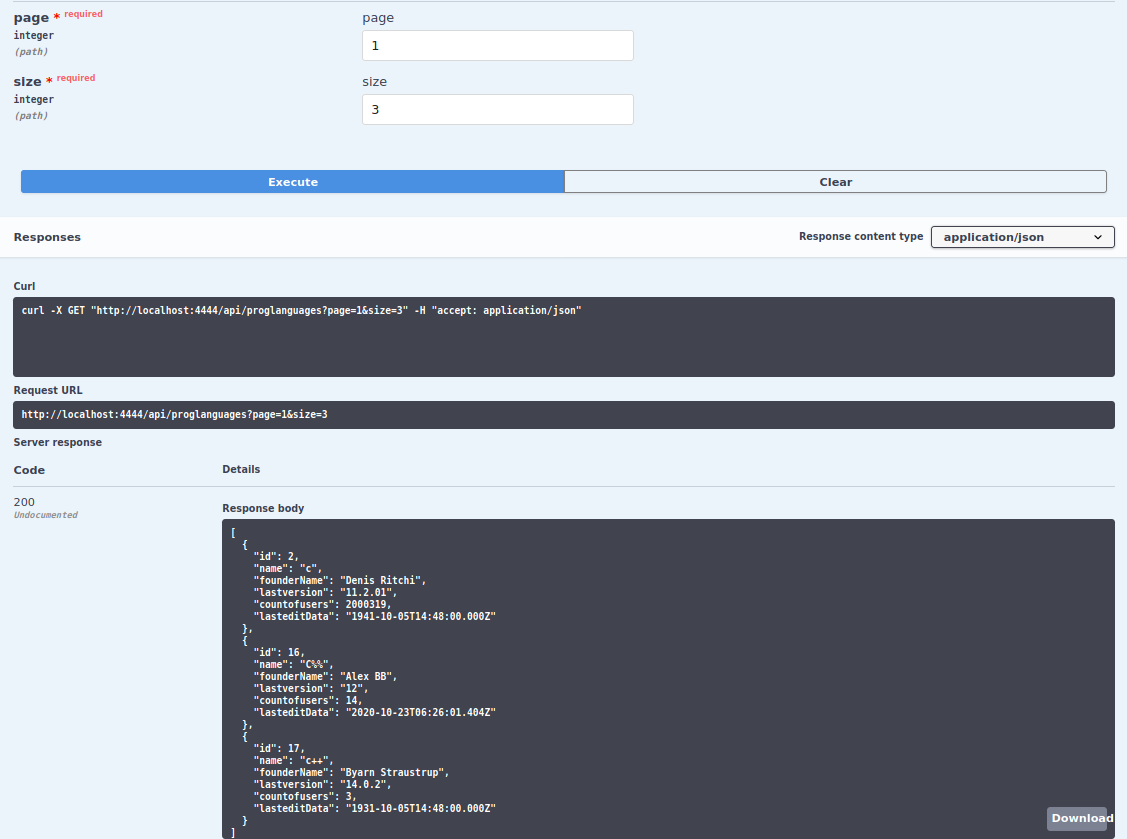
|  |
| --- |
| **usersRouter.js** |
| const router = require('express').Router();    const userController = require('./../controllers/userControllers');    router  /\*\*  \* @route GET /api/users/{id}  \* @group Users - user operations  \* @param {integer} id.path.required - id of the User - eg: 1  \* @returns {User.model} 200 - User object  \* @returns {Error} 404 - User not found  \*/  .get("/:id", userController.getUser)  /\*\*  \* @route GET /api/users?page={page}&size={size}  \* @group Users - user operations  \* @param {integer} page.path.required - page  \* @param {integer} size.path.required - size  \* @returns {Array.<User>} User - all users  \*/  .get("/", userController.getUsers)    module.exports = router; |

**Приклади результатів з Swagger сторінки /api-docs**

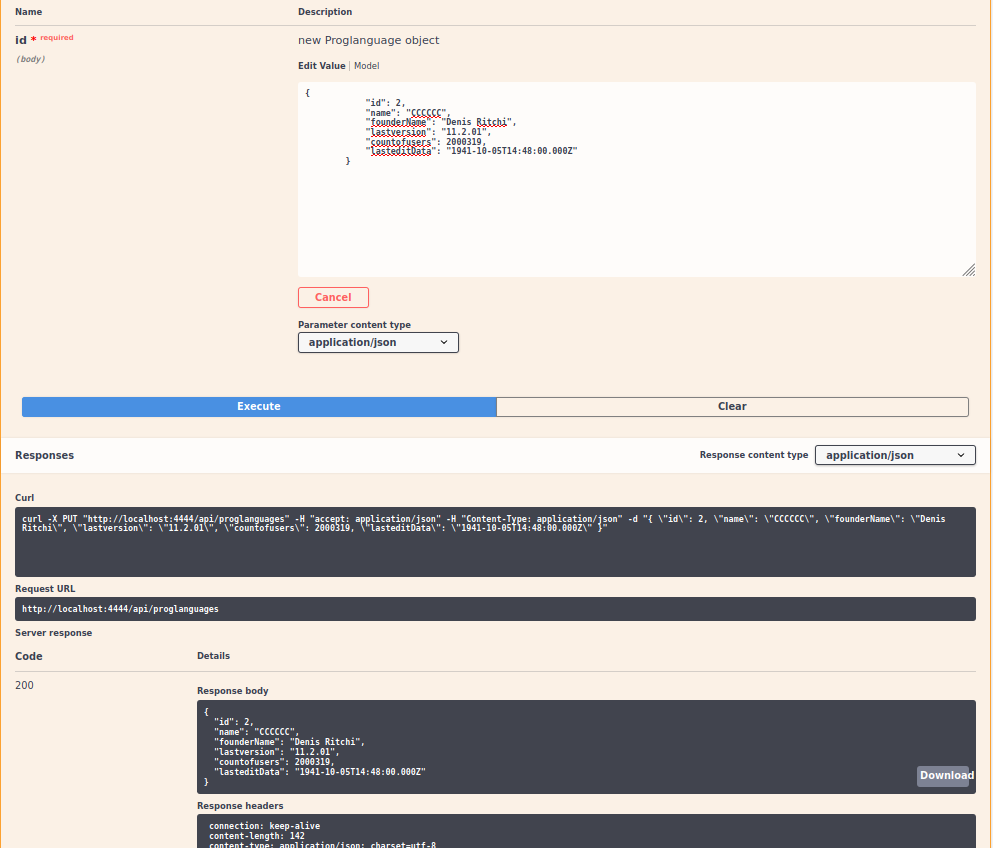
find entity by id

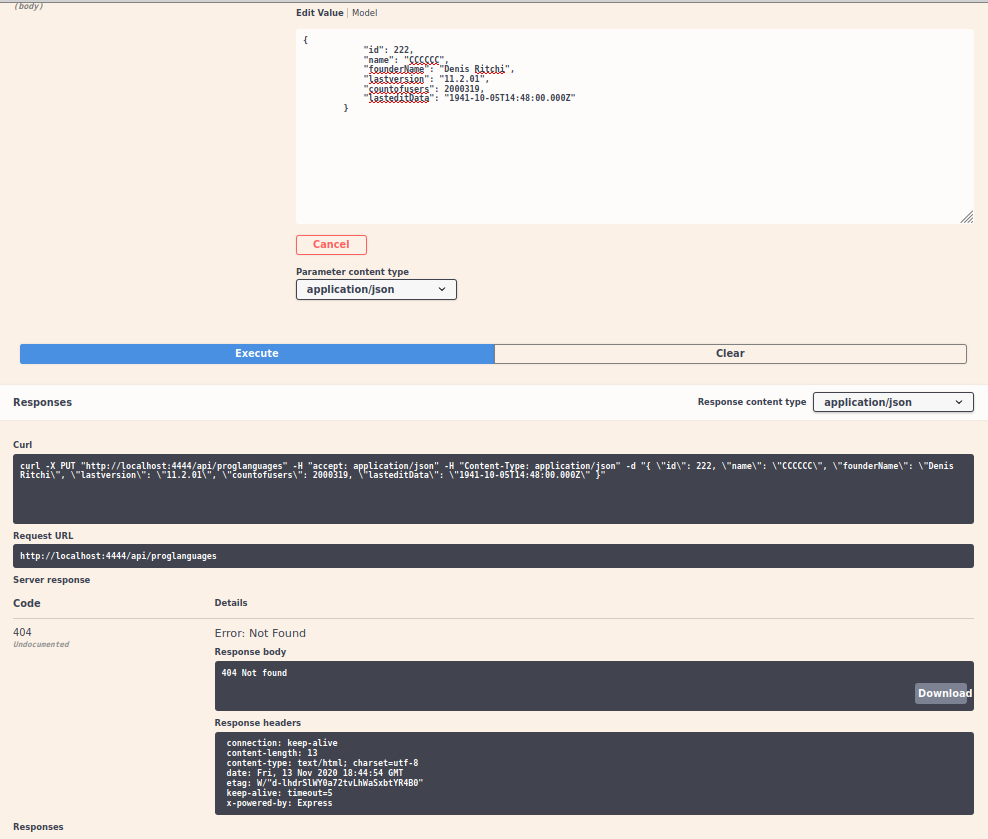


show all entities with pagination

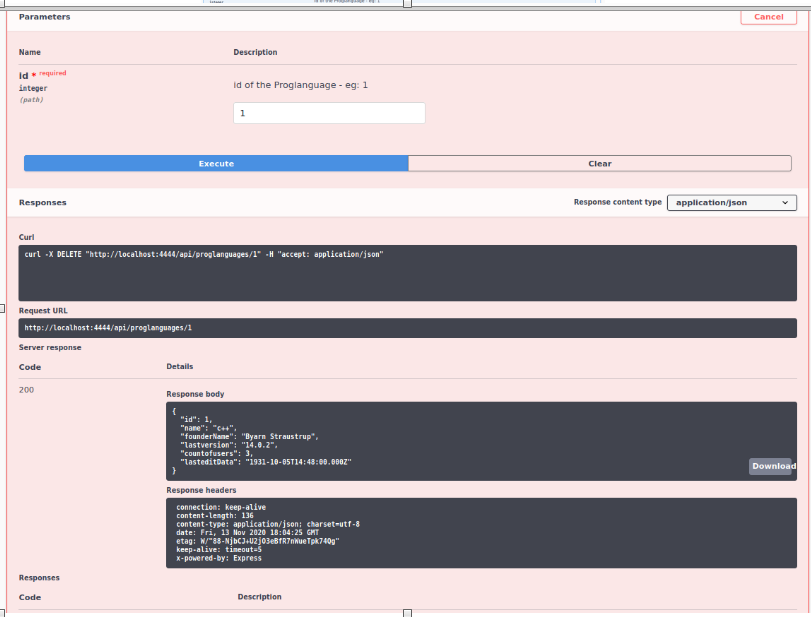


Edit entity

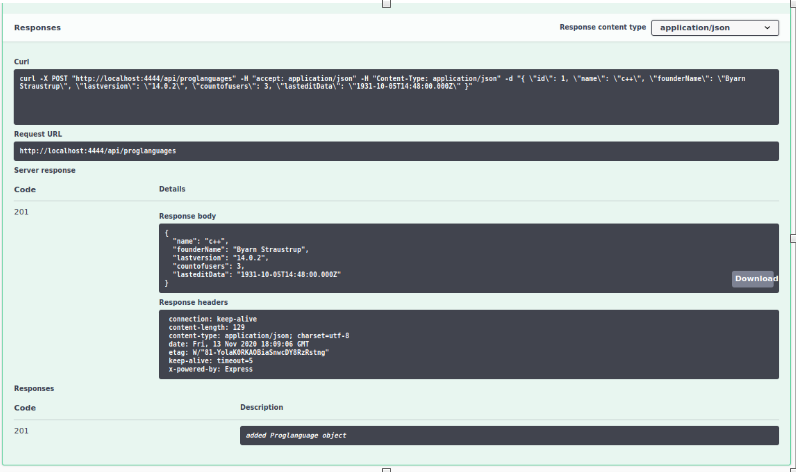
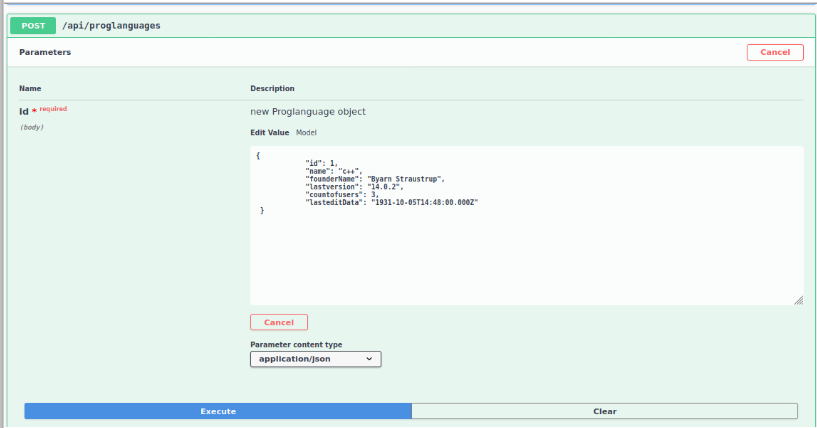




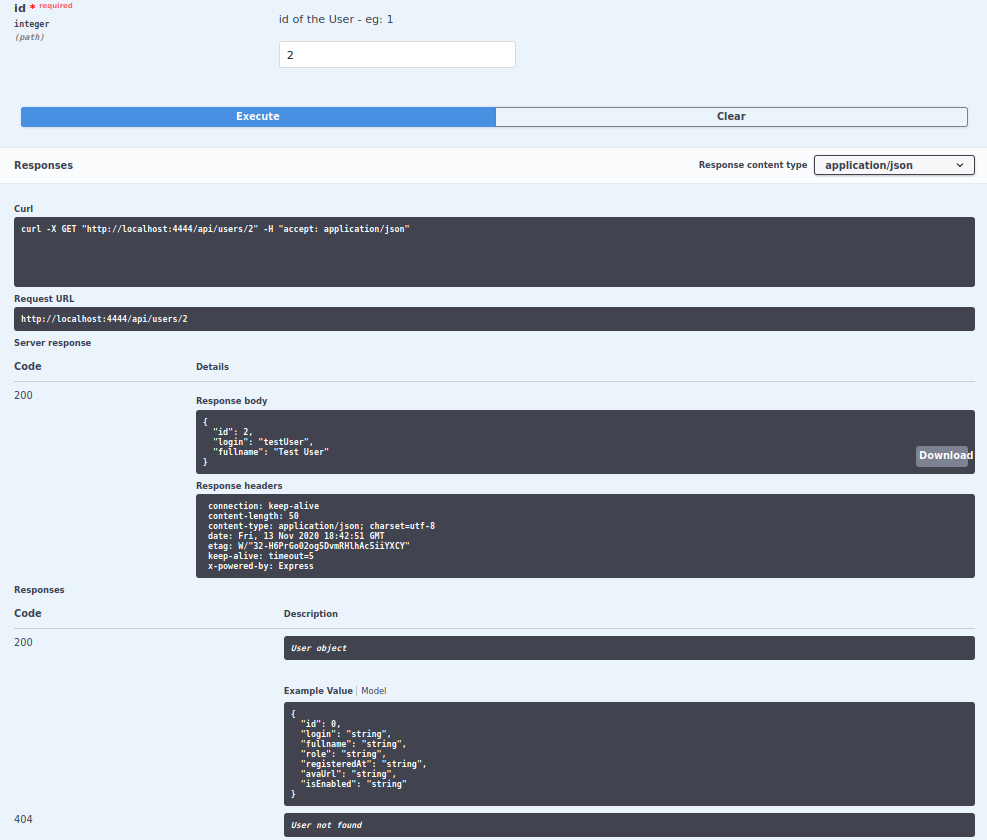
Delete entity



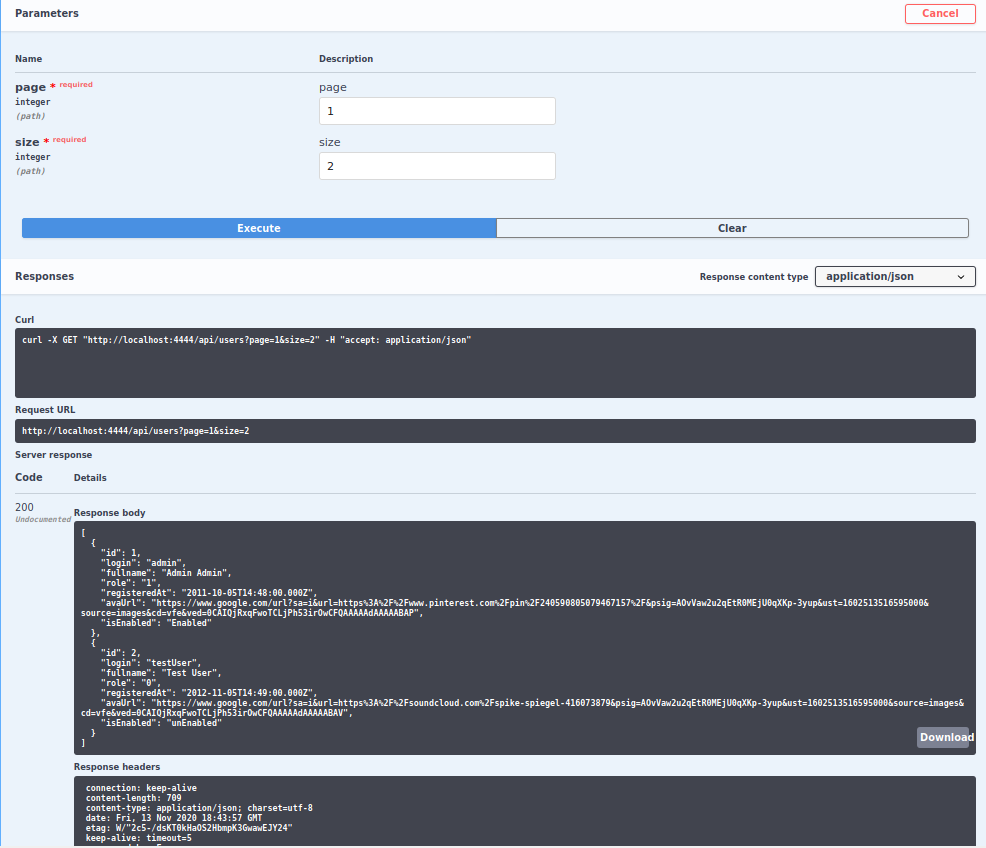
Add entity



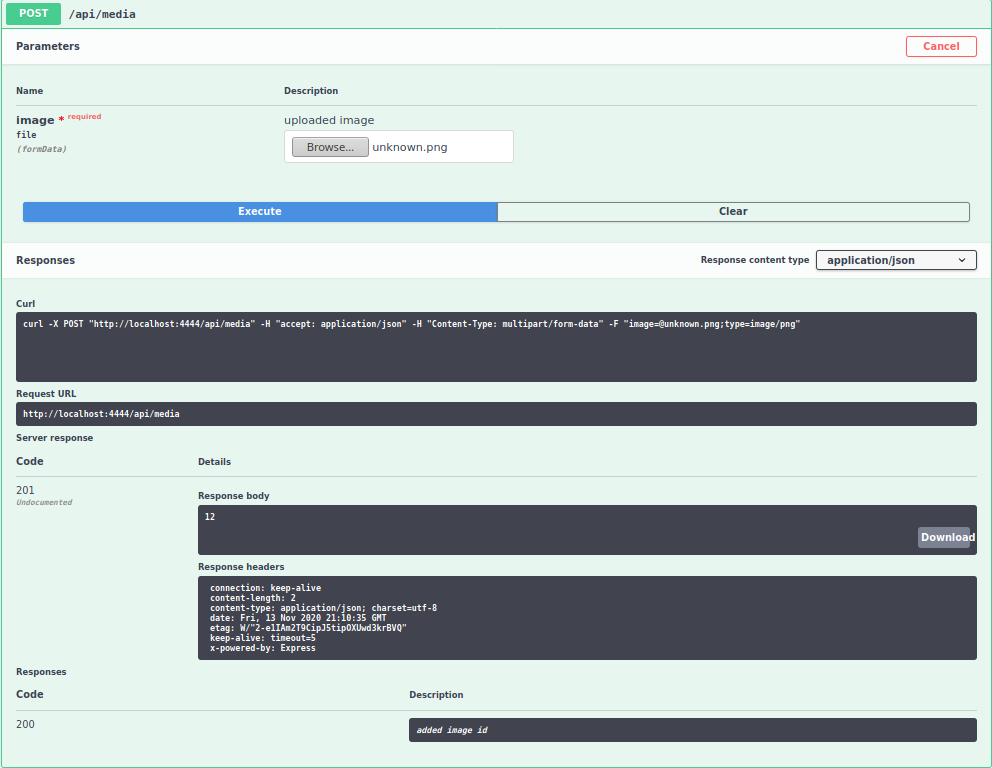
Find user by id



Show all users with pagination



add media



find media by id

