

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 “**Основи JavaScript**”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) I курсу  групи КП-83  Бойчук В.А.  (*прізвище, ім’я, по батькові*)  варіант № 2 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2019

### Завдання

Реалізувати консольну програму для керування сутностями різних типів даних із файлового сховища у форматі JSON.

### Вказівки до виконання

Кроки:

1. Підготувати Node.js проект
   1. Встановити найновішу доступну LTS версію [**Node.js (node)**](https://nodejs.org/en/).
   2. Встановити глобально пакет **ESLint** за допомогою команди:  
      npm install -g eslint.
   3. Створити у навчальному репозиторії директорію проекту: labs/lab2 і перейти до неї у редакторі (IDE).
   4. Створити у корені проекту файл .eslintrc.json та app.js (вміст файлів взяти з Додатків даного завдання).
   5. Додати у корінь проекту файл .gitignore ([вміст файлу](https://github.com/github/gitignore/blob/master/Node.gitignore))
   6. Відкрити термінал у кореневій директорії завдання і виконати команду: npm init --yes, яка автоматично створить файл package.json.
   7. Всі зміни закомітити у віддалений репозиторій.
2. Спеціальні директорії проекту:
   1. Створити у корені директорію models.
   2. Створити у корені директорію data.
3. Сховище користувачів:
   1. Створити модуль models/user.js (див. [приклад](https://docs.google.com/document/d/1-Uz-Y-H6AtDygYpv_OhcOel6atb6clcGaTd04FD58WE/edit?usp=sharing)) і підключити його у app.js:

*// models/user.js*

module.exports = {};

*// app.js*

**const** user = require('./models/user');

*// ...*

* 1. Використати у модулі тип сутностей **Користувач** для створення модуля-сховища колекції об'єктів цього типу. Кожен користувач повинен містити такі поля:
     + id - унікальний числовий (або uuid) ідентифікатор
     + login - унікальний рядок символів
     + role - ціле число (0 - простий користувач, 1 - адміністратор).
     + fullname - рядок повного імені користувача
     + registeredAt - рядок із датою у форматі [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601).
     + avaUrl - рядок з URL зображення.
     + isDisabled - відмітка чи користувача було деактивовано.
  2. Модуль models/user.js повинен містити такі синхронні функції:
     + getAll() - отримати списком всі об'єкти зі сховища.
     + getById(id) - отримати елемент зі сховища за ідентифікатором. У випадку, якщо елемент не знайдено - повертати undefined.
  3. Реалізувати модуль таким чином, щоби всі описані функції синхронно працювали із JSON файлом (data/users.json), що буде містити масив із об'єктами користувачів. Рекомендована структура файлу (див. Додатки).

1. Реалізувати у app.js командний інтерфейс користувача для виконання операцій з сутностями типу Користувач (код для отримання вводу з консолі у Додатках, формат команд можна змінювати):
   1. На команду users завантажити і показати користувачу список (або таблицю) з короткою інформацією про всі сутності.
   2. На команду users/get/{id} (замість {id} вводиться ідентифікатор сутності) вивести у консоль детальну інформацію про обрану сутність.
2. Сховище сутностей за варіантом:
   1. Створити модуль models/{entity}.js (замість {entity} використати назву сутності англійською мовою) і підключити його у app.js. Використати у модулі тип сутностей із [ЛР№1](http://progbase.herokuapp.com/modules/webprogbase/labs/lab1) для створення модуля-сховища колекції об'єктів цього типу.  
      Кожен об'єкт у сховищі повинен містити як мінімум:
      * унікальний ідентифікатор, що присвоюється йому при внесенні у сховище
      * 2 рядкових значеня
      * 2 числових значення
      * 1 рядок із датою у форматі [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601).
   2. Модуль models/{entity}.js повинен містити такі синхронні функції:
      * insert(x) - додати у сховище новий елемент та повернути його новий ідентифікатор
      * getAll() - отримати списком всі об'єкти зі сховища
      * getById(id) - отримати елемент зі сховища за ідентифікатором
      * update(x) - оновити дані елемента у сховищі
      * deleteById(id) - видалити елемент зі сховища за ідентифікатором
   3. Реалізувати модуль таким чином, щоби всі описані функції синхронно працювали із JSON файлом (data/{entities}.json, замість {entities} використати назву сутності англійською мовою у множині), що буде містити масив із об'єктами (структура аналогічна data/users.json).  
      Також розмістити у цьому файлі ідентифікатор останнього доданого об'єкту (nextId), і інкрементувати його при додаванні нових елементів у сховище для забезпечення унікальності ідентифікаторів колекції елементів.  
      Кожна зміна вмісту сховища повинна перезаписувати файл. Текст JSON зберігати із відступами (indented).
3. Реалізувати оброблення консольних команд, за допомогою яких користувач буде мати можливість використовувати всі функцій із модуля-сховища сутностей за варіантом, наприклад (формат команд можна змінювати):
   1. {entities}/get/{id} - отримати детальну інформацію про обрану сутність
   2. {entities}/delete/{id} - видалити зі сховища обрану сутність
   3. {entities}/update/{id} - ввести нові дані для обраної сутності і оновити її у сховищі
   4. {entities}/insert - розпочати заповнення даних нової сутності, яку додати у сховище
4. На початку роботи програми (головне меню) дати користувачу моживість переглянути всі сутності обраного типу за допомогою команд:
   1. users - користувачі
   2. {entities} - сутності за варіантом

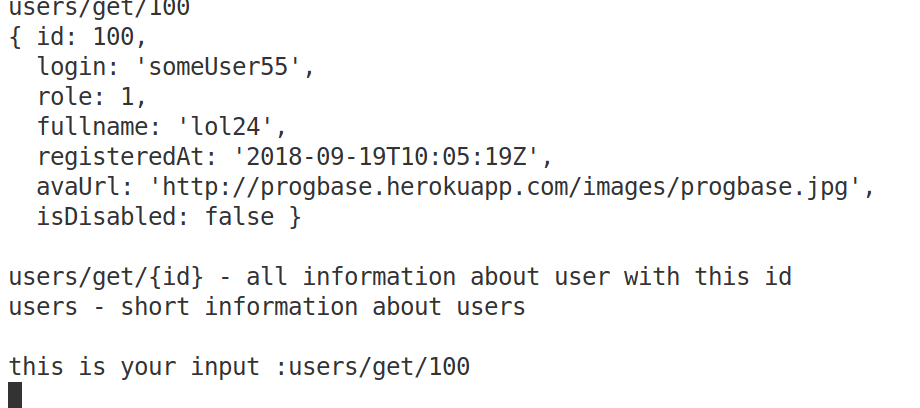
**Тексти коду програм**

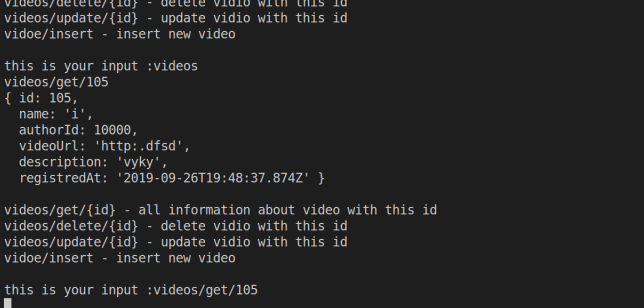
|  |
| --- |
| **app.js** |
| const user = require('./models/user');  const video = require('./models/videos');  const message = "This is lab2!";  console.log(message);  const stdin = process.openStdin();  stdin.addListener("data", onInput);  console.log("Input something:");  console.log("users\nvideos");  function printUserComands() {  console.log("\nusers/get/{id} - all information about user with this id");  console.log("users - short information about users\n");  }  function printVideosComands() {  console.log("\nvideos/get/{id} - all information about video with this id");  console.log("videos/delete/{id} - delete vidio with this id");  console.log("videos/update/{id} - update vidio with this id");  console.log("vidoe/insert - insert new video\n");  }  function onInput(dataObject) {  const inputString = dataObject.toString().trim();  let inputArray = inputString.split('/');  if (inputArray[0] === "users") {  if (!inputArray[1]) {  for (let i = 0; i < user.getAll().length; i++) {  console.log(` id :${user.getAll()[i].id}--- name:[${user.getAll()[i].fullname}]`);  }  printUserComands();  }  if (inputArray[1] === "get") {  let id = inputArray[2];  console.log(user.getById(Number(id)));  printUserComands();  }  }  if (inputArray[0] === "videos") {  if (!inputArray[1]) {  for (let i = 0; i < video.getAll().length; i++) {  console.log(` id :${video.getAll()[i].id}--- name:[${video.getAll()[i].name}]`);  }  printVideosComands();  }  if (inputArray[1] === "insert" && !inputArray[2]) {  const readline = require('readline-sync');  let insertElement = new video();  insertElement.authorId = Number(readline.question("author id :"));  insertElement.name = readline.question("name :");  insertElement.videoUrl = readline.question("url :");  insertElement.description = readline.question("descriptoin :");  insertElement.registredAt = new Date();  video.insert(insertElement);  printVideosComands();  }  if (inputArray[1] === "get") {  let id = inputArray[2];  console.log(video.getById(Number(id)));  printVideosComands();  }  if (inputArray[1] === "update") {  let id = inputArray[2];  if (video.getById(Number(id))) {  console.log("<--this is your video:\n");  console.log(video.getById(Number(id)));  const readline = require('readline-sync');  let newElement = new video();  newElement.id = Number(id);  newElement.authorId = Number(readline.question("new author id :"));  newElement.name = readline.question("new name :");  newElement.videoUrl = readline.question("new url :");  newElement.description = readline.question("new descriptoin :");  newElement.registredAt = new Date();  video.update(newElement);  console.log("<--Video has been updated-->");  printVideosComands();  }  } if (inputArray[1] === "delete") {  let id = inputArray[2];  if (video.getById(Number(id))) {  console.log("this is your video:\n");  console.log(video.getById(Number(id)));  const readline = require('readline-sync');  let answer = (readline.question("are you sure : 1 - yes , else - no\n"));  if (Number(answer) === 1) {  video.deleteById(Number(id));  console.log("<--Video has been deleted-->");  }  }  printVideosComands();  }  }  console.log(`this is your input :${inputString}`);  } |

|  |
| --- |
| **videos.js** |
| class Video {  constructor(id, name, authorId) {  this.id = id;  this.name = name;  this.authorId = authorId;  this.registredAt;  this.videoUrl;  this.description;  }  static insert(element) {  let fs = require("fs");  let text = fs.readFileSync("./data/videos.json");  let jsonContent = JSON.parse(text);  let { nextId, videos } = { nextId: jsonContent.nextId, videos: jsonContent.videos };  element.id = nextId;  videos.push(element);  nextId++;  let newJsonContent = JSON.stringify({ nextId, videos }, null, ' ');  fs.writeFileSync("./data/videos.json", newJsonContent);  return element.id;  }  getAll() {  }  static getById(id) {  let fs = require("fs");  let text = fs.readFileSync("./data/videos.json");  let jsonContent = JSON.parse(text);  let videosArray = jsonContent.videos;  for (let element of videosArray) {  if (id === element.id) {  return element;  }  }  return;  }  static update(new\_element) {  let fs = require("fs");  let text = fs.readFileSync("./data/videos.json");  let jsonContent = JSON.parse(text);  let { nextId, videos } = { nextId: jsonContent.nextId, videos: jsonContent.videos };  for (let i = 0; i < videos.length; i++) {  if (videos[i].id === new\_element.id) {  videos[i] = new\_element;  }  }  console.log('vrodi norm \n');  let newJsonContent = JSON.stringify({ nextId, videos }, null, ' ');  fs.writeFileSync("./data/videos.json", newJsonContent);    }  static deleteById(id) {  let fs = require("fs");  let text = fs.readFileSync("./data/videos.json");  let jsonContent = JSON.parse(text);  let { nextId, videos } = { nextId: jsonContent.nextId, videos: jsonContent.videos };  for (let i = 0; i < videos.length; i++) {  if (videos[i].id === id) {  videos.splice(i, 1);  nextId = 100;  }  }  let newJsonContent = JSON.stringify({ nextId, videos }, null, ' ');  fs.writeFileSync("./data/videos.json", newJsonContent);  }  static getAll() {  let fs = require("fs");  let text = fs.readFileSync("./data/videos.json");  let jsonContent = JSON.parse(text);  return jsonContent.videos;  }  }  module.exports = Video; |

|  |
| --- |
| **user.js** |
| {  "nextId": 101,  "items": [  {  "id": 0,  "login": "someUser11",  "role": 1,  "fullname": "I Am User",  "registeredAt": "2018-09-19T10:05:19Z",  "avaUrl": "http://progbase.herokuapp.com/images/progbase.jpg",  "isDisabled": false  },  {  "id": 100,  "login": "someUser55",  "role": 1,  "fullname": "lol24",  "registeredAt": "2018-09-19T10:05:19Z",  "avaUrl": "http://progbase.herokuapp.com/images/progbase.jpg",  "isDisabled": false  }  ]  } |

**Приклади результатів**

  
Illustration 1: отримати данні по ай ді

  
Illustration 2: отримати данні додаткової сутностті по ай ді

**Висновки**

Виконуючи дану лабораторну роботу, я навчився працювати з мовою програмування javascript, а точніше з її серверним варіантом Node.js.

Програма надає користувачу можливості зберігання у сховищі певних сутностей, а також доступ до керування цим сховищем шляхом створенням, видаленням та оновленням сутностей

Для роботи користувача в консолі а також запису в файли використовуються стандартні модулі Node.js - fs та synchReadLine.

Ці пакети забезпечують роботу синхронний функцій. Встановлення цих пакетів відбулося за допомогою менеджера пакетів npm.