

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 №** 1

з дисципліни “Основи web-програмування”

тема “Основи JavaScript”

|  |  |  |
| --- | --- | --- |
| Виконав  студент II курсу  групи КП-93  Катрич Владислав Сергійович  (*прізвище, ім’я, по батькові*)  варіант №25 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2019

**Мета роботи**

Познайомитись з мовою програмування JavaScript та середовищем Node.js.  
Навчитись створювати модулі, синхронно працювати із файловими потоками та обробляти JSON-текст.

**Постановка завдання**

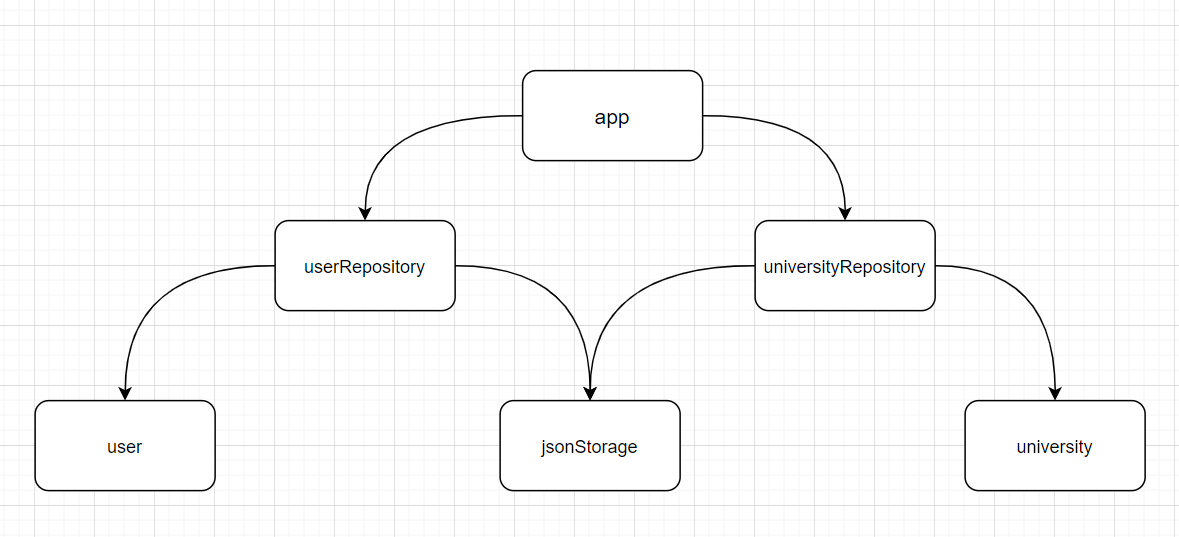
Реалізувати на мові JavaScript (або TypeScript) консольну програму для керування сутностями різних типів даних із файлового сховища у форматі JSON.

**Вказівки до виконання**

Кроки:

1. Підготувати Node.js проект
   * Встановити найновішу доступну LTS версію [**Node.js (node)**](https://nodejs.org/en/).
   * Встановити глобально пакет **ESLint** за допомогою команди:  
     npm install -g eslint (із sudo на Linux).
   * Створити у навчальному репозиторії **webprogbase** директорію проекту: labs/lab1 і перейти до неї у редакторі (IDE).
   * Створити у корені проекту файл app.js та .eslintrc.json (див. Додаток).
   * Додати у корінь проекту файл .gitignore ([вміст файлу](https://github.com/github/gitignore/blob/master/Node.gitignore))
   * Відкрити термінал у кореневій директорії завдання і виконати команду: npm init --yes, яка автоматично створить файл package.json.
   * Всі зміни закомітити у віддалений репозиторій.
2. Створити у корені проекту директорії data, models, repositories.
3. JSON Сховище об'єктів
   * Створити файл data/users.json з даними користувачів (див. [приклад](https://docs.google.com/document/d/1-Uz-Y-H6AtDygYpv_OhcOel6atb6clcGaTd04FD58WE/edit?usp=sharing)).
   * Обрати спосіб зберігання ідентифікатора останнього доданого об'єкту (nextId), і інкрементувати його при додаванні нових елементів у сховище для забезпечення унікальності ідентифікаторів колекції елементів.
   * Створити модуль jsonStorage.js (див. [приклад](https://docs.google.com/document/d/1-Uz-Y-H6AtDygYpv_OhcOel6atb6clcGaTd04FD58WE/edit?usp=sharing)), для зчитування об'єктів з JSON файлу data/users.json.
4. Модель користувача і репозиторій користувачів:
   * Створити модуль models/user.js (див. [приклад](https://docs.google.com/document/d/1-Uz-Y-H6AtDygYpv_OhcOel6atb6clcGaTd04FD58WE/edit?usp=sharing)):  
     Кожен користувач повинен містити такі поля:
     + id - унікальний числовий (або uuid) ідентифікатор
     + login - унікальний рядок символів
     + fullname - рядок повного імені користувача
     + role - ціле число (0 - простий користувач, 1 - адміністратор).
     + registeredAt - рядок із датою у форматі [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601).
     + avaUrl - рядок з URL зображення.
     + isEnabled - відмітка чи користувача було активовано/деактивовано.
   * Створити модуль repositories/userRepository.js (див. [приклад](https://docs.google.com/document/d/1-Uz-Y-H6AtDygYpv_OhcOel6atb6clcGaTd04FD58WE/edit?usp=sharing)) для отримання моделей всіх користувачів зі сховища та отримання моделі користувача за ідентифікатором.
5. Реалізувати у app.js командний інтерфейс користувача для виконання операцій з сутностями типу Користувач (формат команд можна змінити і придумати самостійно):
   * На команду get/users завантажити через репозиторій і показати користувачу список (або таблицю) з короткою інформацією про всі сутності.
   * На команду get/users/{id} (замість {id} вводиться ідентифікатор сутності) вивести у консоль детальну інформацію про обрану сутність.
6. Сутності за варіантом:
   * Створити модуль models/{entity}.js (замість {entity} використати назву сутності англійською мовою) і підключити його у app.js. Використати у модулі затверджений тип сутностей за варіантом для створення модуля-репозиторія колекції об'єктів цього типу.  
     Кожен об'єкт сутності за варіантом повинен містити як мінімум:
     + унікальний ідентифікатор
     + 2 рядкових значеня
     + 2 числових значення
     + 1 рядок із датою у форматі [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601).
   * Модуль репозиторія repository/{entity}Repository.js повинен містити такі синхронні функції:
     + add{Entity}(entityModel) - додати у сховище модель та повернути її новий ідентифікатор
     + get{Entities}() - отримати списком всі моделі зі сховища
     + get{Entity}ById(entityId) - отримати модель зі сховища за ідентифікатором
     + update{Entity}(entityModel) - оновити дані моделі у сховищі
     + delete{Entity}(entityId) - видалити модель зі сховища за ідентифікатором
   * Реалізувати репозиторій сутностей таким чином, щоби всі описані функції синхронно працювали зі сховищем з JSON файлом (data/{entities}.json, замість {entities} використати назву сутності англійською мовою у множині), що буде містити масив із об'єктами.  
     Кожна зміна вмісту сховища повинна перезаписувати файл. **Текст JSON зберігати із відступами (indented)**.
7. Реалізувати оброблення консольних команд, за допомогою яких користувач буде мати можливість використовувати всі функцій із модуля-репозиторія сутностей за варіантом, наприклад (формат команд можна змінювати):
   * get/{entities} - отримати список всіх сутностей за варіантом
   * get/{entities}/{id} - отримати детальну інформацію про обрану сутність
   * delete/{entities}/{id} - видалити з репозиторія обрану сутність
   * update/{entities}/{id} - ввести нові дані для обраної сутності і оновити її у репозиторії
   * post/{entities} - розпочати заповнення даних нової сутності, яку додати у репозиторій
8. Для всіх випадків неможливості виконання операції виводити про це повідомлення.

**Діаграма залежностей модулів:**



**Код розроблених модулів:**

|  |
| --- |
| **user** |
| class User {      constructor(          uuid,          login,          fullname,          role = 0,          registeredAt = new Date().toISOString(),          avaUrl = '#',          isEnabled = true      ) {          this.uuid = uuid;          this.login = login;          this.fullname = fullname;          this.role = role;          this.registeredAt = registeredAt;          this.avaUrl = avaUrl;          this.isEnabled = isEnabled;      }  }  module.exports = User; |

|  |
| --- |
| **university** |
| class University {      constructor(uuid, name, country, numOfStudents, campus, foundationDate) {          this.uuid = uuid;          this.name = name;          this.country = country;          this.numOfStudents = numOfStudents;          this.campus = campus;          this.foundationDate = foundationDate;      }  }  module.exports = University; |

|  |
| --- |
| **jsonStorage** |
| const fs = require('fs');  class JsonStorage {      constructor(filePath) {          this.filePath = filePath;      }      readItems() {          const jsonText = fs.readFileSync(this.filePath);          const jsonArray = JSON.parse(jsonText);          return jsonArray;      }      writeItems(items) {          fs.writeFileSync(this.filePath, JSON.stringify(items, null, 4));      }  }  module.exports = JsonStorage; |

|  |
| --- |
| **userRepository** |
| const User = require('../models/user');  const JsonStorage = require('../jsonStorage');  class UserRepository {      constructor(filePath) {          this.storage = new JsonStorage(filePath);      }      getUsers() {          return this.storage.readItems();          /\* //?if you need to know the type of array objects          const items = [];          for (const item of this.storage.readItems()) {              items.push(this.getUserByUuid(item.uuid));          }          return items;          \*/      }      getUserByUuid(uuid) {          for (const item of this.storage.readItems()) {              if (item.uuid === uuid) {                  return new User(                      item.uuid,                      item.login,                      item.fullname,                      item.role,                      item.registeredAt,                      item.avaUrl,                      item.isEnabled                  );              }          }          console.log('This index is missing');          return null;      }  }  module.exports = UserRepository; |

|  |
| --- |
| **universityRepository** |
| const University = require('../models/university');  const JsonStorage = require('../jsonStorage');  class UniversityRepository {      constructor(filePath) {          this.storage = new JsonStorage(filePath);      }      getUniversities() {          return this.storage.readItems();          /\* //? If it is necessary to know the type of array objects          const items = [];          for (const item of this.storage.readItems()) {              items.push(this.getUserByUuid(item.uuid));          }          return items;          \*/      }      getUniversityByUuid(uuid) {          for (const item of this.storage.readItems()) {              if (item.uuid === uuid) {                  return new University(                      item.uuid,                      item.name,                      item.country,                      item.numOfStudents,                      item.campus,                      item.foundationDate                  );              }          }          console.log('This uuid is missing');          return null;      }      addUniversity(universityModel) {          this.storage.writeItems(universityModel);      }      updateUniversity(universityModel) {          this.storage.writeItems(universityModel);      }      deleteUniversity(universityUuid) {          const items = this.getUniversities();          items.splice(universityUuid, 1);          this.storage.writeItems(items);      }  }  module.exports = UniversityRepository; |

|  |
| --- |
| **app** |
| console.clear();  const uuidv4 = require('uuid-v4');  const readline = require('readline');  const UserRepository = require('./repositories/userRepository');  const UniversityRepository = require('./repositories/universityRepository');  const readlineInterface = readline.createInterface(      process.stdin,      process.stdout  );  const userRepository = new UserRepository('./data/users.json');  const universityRepository = new UniversityRepository(      './data/universities.json'  );  function ask(questionText) {      return new Promise((resolve, reject) => {          readlineInterface.question(questionText, resolve);      });  }  function inputCommand() {      return new Promise((resolve, reject) => {          readlineInterface.question(`\nEnter your command: `, resolve);      });  }  onInput();  async function onInput() {      const text = await inputCommand();      const parts = text.split('/');      const command = parts[0];      const entity = parts[1];      const entityUuid = parts[2];      if (command === 'get') {          if (entity === 'user') {              if (entityUuid) {                  const user = userRepository.getUserByUuid(entityUuid);                  if (user) console.log(user);              } else {                  const users = userRepository.getUsers();                  users.forEach((user) => {                      console.log(`--------------------  UUID: ${user.uuid}  name: ${user.login}`);                  });                  console.log(`--------------------`);              }          } else if (entity === 'university') {              if (entityUuid) {                  const university = universityRepository.getUniversityByUuid(                      entityUuid                  );                  if (university) console.log(university);              } else {                  const universities = universityRepository.getUniversities();                  universities.forEach((university) => {                      console.log(`--------------------  UUID: ${university.uuid}  name: ${university.name}  country: ${university.country}`);                  });                  console.log(`--------------------`);              }          } else if (!entity) {              console.log(`You haven't selected an entity`);          } else {              console.log(`This entity doesn't exist`);          }      } else if (command === 'delete') {          if (entity === 'university') {              if (entityUuid) {                  const items = universityRepository.getUniversities();                  const requiredUuid = items.findIndex(                      (item) => item.uuid === entityUuid                  );                  if (requiredUuid === -1) console.log('This index is missing');                  else {                      universityRepository.deleteUniversity(requiredUuid);                  }              } else console.log(`You haven't selected an index`);          } else if (!entity) {              console.log(`You haven't selected an entity`);          } else {              console.log(`This entity doesn't exist`);          }      } else if (command === 'update') {          if (entity === 'university') {              if (entityUuid) {                  const items = universityRepository.getUniversities();                  const requiredUuid = items.findIndex(                      (item) => item.uuid === entityUuid                  );                  if (requiredUuid === -1) console.log('This index is missing');                  else {                      console.log(                          `Input new value(or press 'ENTER' to keep the original)`                      );                      items[requiredUuid].name =                          (await ask(`name: `)) || items[requiredUuid].name;                      items[requiredUuid].country =                          (await ask(`country: `)) || items[requiredUuid].country;                      let numOfStudents;                      do {                          numOfStudents = +(                              (await ask(`num of students: `)) ||                              items[requiredUuid].numOfStudents                          );                      } while (!numOfStudents);                      items[requiredUuid].numOfStudents = numOfStudents;                      let campus;                      do {                          campus = +(                              (await ask(`campus: `)) ||                              items[requiredUuid].campus                          );                      } while (!campus);                      items[requiredUuid].campus = campus;                      let foundationDate;                      do {                          foundationDate =                              (await ask(`foundation date: `)) ||                              items[requiredUuid].foundationDate;                      } while (!isIsoDate(foundationDate));                      items[requiredUuid].foundationDate = foundationDate;                      universityRepository.updateUniversity(items);                  }              } else console.log(`You haven't selected an index`);          } else if (!entity) {              console.log(`You haven't selected an entity`);          } else {              console.log(`This entity doesn't exist`);          }      } else if (command === 'post') {          if (entity === 'university') {              const items = universityRepository.getUniversities();              const item = {};              console.log(`Fill in the fields for the new university`);              item.uuid = uuidv4();              do {                  item.name = await ask(`name: `);              } while (!item.name);              do {                  item.country = await ask(`country: `);              } while (!item.country);              do {                  item.numOfStudents = +(await ask(`num of students: `));              } while (!item.numOfStudents);              do {                  item.campus = +(await ask(`campus: `));              } while (!item.campus);              do {                  item.foundationDate = await ask(`foundation date: `);              } while (!isIsoDate(item.foundationDate));              items.push(item);              universityRepository.addUniversity(items);          } else console.log(`You haven't selected an index`);      } else if (command === 'exit') {          process.exit(0);      } else {          console.log(`Not supported command: '${command}'`);      }      while (true) await onInput();  }  function isIsoDate(str) {      if (!/\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}.\d{3}Z/.test(str)) return false;      const d = new Date(str);      return d.toISOString() === str;  } |

**Вміст .json файлів:**

|  |
| --- |
| **users** |
| [      {          "uuid": "9d5ad13a-de1a-4f21-90fa-0a180301041b",          "login": "admin",          "fullname": "Admin Admin",          "role": 1,          "registeredAt": "2020-06-10T14:12:48.622Z",          "avaUrl": "https://i.ibb.co/1bXgp60/photo-2015-09-24-22-09-17.jpg",          "isEnabled": true      },      {          "uuid": "6e8d22c3-8d93-415e-9f10-eca4d1879f6e",          "login": "testUser",          "fullname": "Test User",          "role": 0,          "registeredAt": "2020-09-10T14:18:48.622Z",          "avaUrl": "#",          "isEnabled": true      }  ] |

|  |
| --- |
| **universities** |
| [      {          "uuid": "9d5ad13a-de1a-4f21-90fa-0a123301041b",          "name": "KPI",          "country": "Ukraine",          "numOfStudents": 36532,          "campus": 1.6,          "foundationDate": "1898-08-31T00:00:00.000Z"      },      {          "uuid": "63f3523b-354c-4035-a340-28aad441df30",          "name": "Unik",          "country": "Zambalia",          "numOfStudents": 123,          "campus": 234,          "foundationDate": "2010-01-26T00:00:00.000Z"      }  ] |

**Приклади роботи програми:**

|  |
| --- |
| F:\Projects\kpi\webprogbase\labs\lab1>node app  Enter your command: get/user  --------------------  UUID: 9d5ad13a-de1a-4f21-90fa-0a180301041b  name: admin  --------------------  UUID: 6e8d22c3-8d93-415e-9f10-eca4d1879f6e  name: testUser  --------------------  Enter your command: get/user/324324  This index is missing  Enter your command: get/user/6e8d22c3-8d93-415e-9f10-eca4d1879f6e  User {  uuid: '6e8d22c3-8d93-415e-9f10-eca4d1879f6e',  login: 'testUser',  fullname: 'Test User',  role: 0,  registeredAt: '2020-09-10T14:18:48.622Z',  avaUrl: '#',  isEnabled: true  }  Enter your command: get/university  --------------------  UUID: 9d5ad13a-de1a-4f21-90fa-0a123301041b  name: KPI  country: Ukraine  --------------------  UUID: 63f3523b-354c-4035-a340-28aad441df30  name: Unik  country: Zambalia  --------------------  Enter your command: get/university/9d5ad13a-de1a-4f21-90fa-0a123301041b  University {  uuid: '9d5ad13a-de1a-4f21-90fa-0a123301041b',  name: 'KPI',  country: 'Ukraine',  numOfStudents: 36532,  campus: 1.6,  foundationDate: '1898-08-31T00:00:00.000Z'  }  Enter your command: update/university  You haven't selected an index  Enter your command: update/university/9d5ad13a-de1a-4f21-90fa-0a123301041b  Input new value(or press 'ENTER' to keep the original)  name: CoolUniversity  country:  num of students: 4444  campus: 1.55  foundation date:  Enter your command: get/university/9d5ad13a-de1a-4f21-90fa-0a123301041b  University {  uuid: '9d5ad13a-de1a-4f21-90fa-0a123301041b',  name: 'CoolUniversity',  country: 'Ukraine',  numOfStudents: 4444,  campus: 1.55,  foundationDate: '1898-08-31T00:00:00.000Z'  }  Enter your command: delete/university/9d5ad13a-de1a-4f21-90fa-0a123301041b  Enter your command: get/university  --------------------  UUID: 63f3523b-354c-4035-a340-28aad441df30  name: Unik  country: Zambalia  --------------------  Enter your command: post/university  Fill in the fields for the new university  name:  name: regree  country: Ukraine  num of students: 32432  campus: 3.33  foundation date: 2442grgerg  foundation date: 1999-03-14T00:00:00.000Z  Enter your command: get/university  --------------------  UUID: 63f3523b-354c-4035-a340-28aad441df30  name: Unik  country: Zambalia  --------------------  UUID: 9158ac39-d8ad-4d5f-860c-0a7784c32379  name: regree  country: Ukraine  --------------------  Enter your command: get/university/9158ac39-d8ad-4d5f-860c-0a7784c32379  University {  uuid: '9158ac39-d8ad-4d5f-860c-0a7784c32379',  name: 'regree',  country: 'Ukraine',  numOfStudents: 32432,  campus: 3.33,  foundationDate: '1999-03-14T00:00:00.000Z'  }  Enter your command: |

**Висновок**

Виконавши цю лабораторну роботу ми познайомились з мовою програмування JavaScript та середовищем Node.js.

А також, реалізувавши консольну програму для керування різних типів даних, навчились створювати модулі, синхронно працювати із файловими потоками та обробляти JSON-текст.