

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 IІ курсу  групи КП-91  Ігор Цой Леонідович  (*прізвище, ім’я, по батькові*)  варіант №23 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

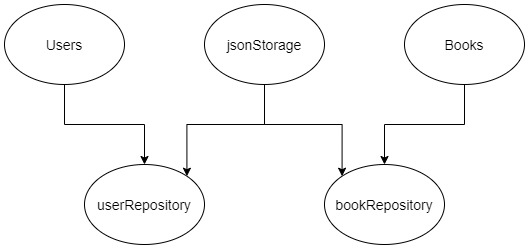
Познайомитись з мовою програмування JavaScript та середовищем Node.js.  
Навчитись створювати модулі, синхронно працювати із файловими потоками та обробляти JSON-текст.

**Постановка завдання**

Реалізувати на мові JavaScript (або TypeScript) консольну програму для керування двома типами сутностей із файлового сховища у форматі JSON: сутність "Користувач системи" та сутність за варіантом.

В якості сутностей за варіантом використати основну або додаткову сутність за темою вашої курсової роботи з "Основ програмування".

**Діаграма залежності модулів**



**JSON файли**

**Вміст після виконання команд**

|  |
| --- |
| **users** |
| {      "items": [          {              "id": 1,              "login": "admin",              "fullname": "Admin Admin",              "role": 1,              "registered\_at": "2020-09-15T18:29:06+00:00",              "ava\_url": null,              "is\_enabled": true          },          {              "id": 4,              "login": "ui1",              "fullname": "uu",              "role": 1,              "registered\_at": "2020-09-26",              "ava\_url": "url2",              "is\_enabled": true          },          {              "id": 5,              "login": "lul1",              "fullname": "lul lul",              "role": 0,              "registered\_at": "2020-09-27",              "ava\_url": "url22",              "is\_enabled": true          }      ]  } |

|  |
| --- |
| **users\_ids** |
| {      "next\_id": 6  } |

|  |
| --- |
| **books** |
| {      "items": [          {              "id": 2,              "title": "gf",              "original\_language": "re",              "publishment\_year": 12,              "pages": 34,              "added": "2020-09-23"          },          {              "id": 5,              "title": "bb",              "original\_language": "bob",              "publishment\_year": 1999,              "pages": 29,              "added": "2020-09-27"          }      ]  } |

|  |
| --- |
| **books\_ids** |
| {      "next\_id": 6  } |

**Тексти коду програм**

|  |
| --- |
| **users** |
| class User  {      constructor(id, login, fullname, role, registered\_at, ava\_url, is\_enabled)      {          this.id = id;          this.login = login;          this.fullname = fullname;          this.role = role;          this.registered\_at = registered\_at;          this.ava\_url = ava\_url;          this.is\_enabled = is\_enabled;      }   };     module.exports = User; |

|  |
| --- |
| **books** |
| class Book  {      constructor (id, title, original\_language, publishment\_year, pages, added)      {          this.id = id;          this.title = title;          this.original\_language = original\_language;          this.publishment\_year = publishment\_year;          this.pages = pages;          this.added = added;      }  };  module.exports = Book; |

|  |
| --- |
| **jsonStorage** |
| const fs = require('fs');  function jsonParser(json\_text)  {      try      {          let obj = JSON.parse(json\_text);          return obj;      }      catch (err)      {          throw(err);      }  }  function jsonStringify(object)  {      let json\_text = JSON.stringify(object, null, 4);      return json\_text;  }  function readFile(file\_path)  {      const buffer = fs.readFileSync(file\_path);      const text = buffer.toString();      return text;  }  function writeFile(file\_path, json\_text)  {      fs.writeFileSync(file\_path, json\_text);  }  class JsonStorage  {      constructor(file\_path) {          this.\_\_file\_path = file\_path;      }      get nextId()      {          try          {              const obj = jsonParser(readFile(this.\_\_file\_path+'\_ids.json'));              return obj['next\_id'];          }          catch (err)          {              throw(err);          }      }      incrementNextId()      {          let obj = jsonParser(readFile(this.\_\_file\_path+'\_ids.json'));          obj['next\_id']++;          writeFile(this.\_\_file\_path+'\_ids.json', jsonStringify(obj))      }      readItems()      {          try          {              const obj = jsonParser(readFile(this.\_\_file\_path+'.json'));              return obj;          }          catch (err)          {              throw(err);          }      }      writeItems(items)      {          writeFile(this.\_\_file\_path+'.json', jsonStringify(items));      }  };  module.exports = JsonStorage; |

|  |
| --- |
| **userRepository** |
| const User = require('../models/user');  const JsonStorage = require('../jsonStorage');    class UserRepository  {      constructor(file\_path)      {          this.storage = new JsonStorage(file\_path);      }        getUsers()      {          try {              const items = this.storage.readItems();              let users = [];              for (const item of items['items']) {                  users.push(new User(                      item['id'],                      item['login'],                      item['fullname'],                      item['role'],                      item['registered\_at'],                      item['ava\_url'],                      item['is\_enabled']                  ));              }              return users;          } catch (err) {              throw err;          }      }        getUserById(user\_id)      {          try {              const items = this.storage.readItems();              for (const item of items['items']) {                  if (item['id'] === user\_id) {                      return new User(                          user\_id,                          item['login'],                          item['fullname'],                          item['role'],                          item['registered\_at'],                          item['ava\_url'],                          item['is\_enabled']                      );                  }              }  return null;          } catch (err) {              throw err;          }      }        addUser(user\_model)      {          try {              let items = this.storage.readItems();              user\_model.id = this.storage.nextId;              this.storage.incrementNextId();              items['items'].push({                  id: user\_model.id,                  login: user\_model.login,                  fullname: user\_model.fullname,                  role: user\_model.role,                  registered\_at: user\_model.registered\_at,                  ava\_url: user\_model.ava\_url,                  is\_enabled: user\_model.is\_enabled              });              this.storage.writeItems(items);          } catch (err) {              throw err;          }      }        updateUser(user\_model)      {          try {              let items = this.storage.readItems();              for (const item of items['items']) {                  if (item['id'] === user\_model.id) {                      item['login'] = user\_model.login;                      item['fullname'] = user\_model.fullname;                      item['role'] = user\_model.role;                      item['registered\_at'] = user\_model.registered\_at;                      item['ava\_url'] = user\_model.ava\_url;                      item['is\_enabled'] = user\_model.is\_enabled;                      this.storage.writeItems(items);                      return true;                  }              }              return false;          } catch (err) {              throw err;          }      }        deleteUser(user\_id)      {          try {              let items = this.storage.readItems();              for (const pair of items['items'].entries()) {                  if (pair[1]['id'] === user\_id) {                      items['items'].splice(pair[0], 1);                      this.storage.writeItems(items);                      return true;                  }              }              return false;          } catch (err) {              throw err;          }      }  };    module.exports = UserRepository; |

|  |
| --- |
| **bookRepository** |
| const Book = require('../models/book');  const JsonStorage = require('../jsonStorage');    class BookRepository  {      constructor(file\_path)      {          this.storage = new JsonStorage(file\_path);      }        getBooks()      {          try {              const items = this.storage.readItems();              let books = [];              for (const item of items['items']) {                  books.push(new Book(                      item['id'],                      item['title'],                      item['original\_language'],                      item['publishment\_year'],                      item['pages'],                      item['added']                  ));              }              return books;          } catch (err) {              throw err;          }      }        getBookById(book\_id)      {          try {              const items = this.storage.readItems();              for (const item of items['items']) {                  if (item['id'] === book\_id) {                      return new Book(                          item['id'],                          item['title'],                          item['original\_language'],                          item['publishment\_year'],                          item['pages'],                          item['added']                      );                  }              }  return null;          } catch (err) {              throw err;          }      }        addBook(book\_model)      {          try {              let items = this.storage.readItems();              book\_model.id = this.storage.nextId;              this.storage.incrementNextId();              items['items'].push({                  id: book\_model.id,                  title: book\_model.title,                  original\_language: book\_model.original\_language,                  publishment\_year: book\_model.publishment\_year,                  pages: book\_model.pages,                  added: book\_model.added              });              this.storage.writeItems(items);          } catch (err) {              throw err;          }      }        updateBook(book\_model)      {          try {              let items = this.storage.readItems();              for (const item of items['items']) {                  if (item['id'] === book\_model.id) {                      item['title'] = book\_model.title;                      item['original\_language'] = book\_model.original\_language;                      item['publishment\_year'] = book\_model.publishment\_year;                      item['pages'] = book\_model.pages;                      item['added'] = book\_model.added;                      this.storage.writeItems(items);                      return true;                  }              }              return false;          } catch (err) {              throw err;          }      }        deleteBook(book\_id)      {          try {              let items = this.storage.readItems();              for (const pair of items['items'].entries()) {                  if (pair[1]['id'] === book\_id) {                      items['items'].splice(pair[0], 1);                      this.storage.writeItems(items);                      return true;                  }              }              return false;          } catch (err) {              throw err;          }      }  };    module.exports = BookRepository; |

**Приклад роботи програми**

|  |
| --- |
| **Enter command: get/users**  **[**  **User {**  **id: 1,**  **login: 'admin',**  **fullname: 'Admin Admin',**  **role: 1,**  **registered\_at: '2020-09-15T18:29:06+00:00',**  **ava\_url: null,**  **is\_enabled: true**  **},**  **User {**  **id: 2,**  **login: 'user 1',**  **fullname: 'User User',**  **role: 0,**  **registered\_at: '2020-09-16T20:07:54+00:00',**  **ava\_url: null,**  **is\_enabled: true**  **},**  **User {**  **id: 4,**  **login: 'u1',**  **fullname: 'uu',**  **role: 0,**  **registered\_at: '2020-09-27',**  **ava\_url: 'url1',**  **is\_enabled: true**  **}**  **]** |
| **Enter command: get/books**  **[**  **Book {**  **id: 1,**  **title: 'book 1',**  **original\_language: 'lul',**  **publishment\_year: 1998,**  **pages: 220,**  **added: '2020-09-18T19:36:55+00:00'**  **},**  **Book {**  **id: 2,**  **title: 'f',**  **original\_language: 'r',**  **publishment\_year: 1,**  **pages: 1,**  **added: '2020-09-24'**  **}**  **]** |
| **Enter command: get/users/2**  **User {**  **id: 2,**  **login: 'user 1',**  **fullname: 'User User',**  **role: 0,**  **registered\_at: '2020-09-16T20:07:54+00:00',**  **ava\_url: null,**  **is\_enabled: true**  **}** |
| **Enter command: get/books/1**  **Book {**  **id: 1,**  **title: 'book 1',**  **original\_language: 'lul',**  **publishment\_year: 1998,**  **pages: 220,**  **added: '2020-09-18T19:36:55+00:00'**  **}** |
| **Enter command: delete/users/2**  **User deleted** |
| **Enter command: delete/books/1**  **Book deleted** |
| **Enter command: update/users/4**  **User {**  **id: 4,**  **login: 'u1',**  **fullname: 'uu',**  **role: 0,**  **registered\_at: '2020-09-27',**  **ava\_url: 'url1',**  **is\_enabled: true**  **}**  **Enter login: ui1**  **Enter fullname: uu**  **Enter role: 1**  **Enter registration date (ISO 8601): 2020-09-26**  **Enter ava url: url2**  **Is enabled? true**  **Updating succesful** |
| **Enter command: update/books/2**  **Book {**  **id: 2,**  **title: 'f',**  **original\_language: 'r',**  **publishment\_year: 1,**  **pages: 1,**  **added: '2020-09-24'**  **}**  **Enter title: gf**  **Enter original language: re**  **Enter publishment year: 12**  **Enter number of pages (GOST R 7.0.3-2006): 34**  **Enter date added (ISO 8601): 2020-09-23**  **Updating succesful** |
| **Enter command: post/users**  **Enter login: lul1**  **Enter fullname: lul lul**  **Enter role: 0**  **Enter registration date (ISO 8601): 2020-09-27**  **Enter ava url: url22**  **Is enabled? true** |
| **Enter command: post/books**  **Enter title: bb**  **Enter original language: bob**  **Enter publishment year: 1999**  **Enter number of pages (GOST R 7.0.3-2006): 29**  **Enter date added (ISO 8601): 2020-09-27** |
| **Enter command: get/books/7**  **Book does not exist** |

**Висновки**

У цiй лабораторнiй роботi було ознайомлено з мовою програмування JavaScript, а також середовищем Node.js.

Було здобуто навичок створення модулей у цій мові програмування, синхронного працювання із файловими потоками та оброблення JSON тексту.

Було реалізовано консольну програму для керування двома типами сутностей із файлового сховища у форматі JSON: сутність «користувач системи» та сутність «книга», взяту із теми курсової роботи з «Основ програмування».