МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра стратегічного управління

ЗВІТ

з лабораторної роботи №4

з дисципліни «Програмування веб-застосунків»

Перевірила

Виконав:

Студент групи КН-27

Харків – 2019

**Тема: Реалізувати REST API у Node.js.**

**Варіант:** №2 Сутності: Книги по програмуванню, Мова програмування

**Завдання:**

1. Реалізувати REST API бекенду вбудованими засобами Node.js.
2. Реалізувати REST API бекенду за допомогою Express.
3. Вимоги до завдання 2 та 3:

* описати обрану предметну область двома сутностями, зв’язаними між собою відношенням один до багатьох, в форматі \*.json;
* кожна сутність повинна мати від 3 до 5 полів (можна і більше);
* реалізувати базові функції CRUD (create, read, update, delete)
* управління даними для кожної сутності: створення, зчитування, зміна і видалення;
* привести декілька запросів, які показують взаємозв’язок між двома сутностями;
* продемонструвати роботу REST API бекенду за допомогою вільно-розповсюдженого програмного забезпечення Postman або Fiddler.

**Хід роботи**

1. Для здійснення умов лабораторної роботи було створено дві сутності (у форматі json) для опису Книг рограммування та Мов програмування. Вміст цих файлів наведено нижче:

|  |
| --- |
| books.json |
| [{"name":"C# для чайников","price":20,"language":"C#"},{"name":"Иванов Л. Курс лекций","price":20,"language":"Java"},{"name":"Node JS Basics","price":40,"language":"js"}] |
| languages.json |
| [{"name":"C#"},{"name":"Java"},{"name":"Python"}] |

1. Реалізацію базових функцій CRUD у Node.js за допомогою фреймворку express наведено нижче:

|  |
| --- |
| Файл db.js |
| const fs = require('fs')  const Db = function(){  this.books = undefined  this.languages = undefined  }  Db.prototype = {  booksTable : './database/books.json',  languagesTable : './database/languages.json',  loadDatabase : function(){  this.books = JSON.parse(fs.readFileSync(this.booksTable, 'utf-8'))  this.languages = JSON.parse(fs.readFileSync(this.languagesTable, 'utf-8'))  },  updateBookByID : function(book\_id, book){  this.books[book\_id] = book  fs.writeFileSync(this.booksTable, JSON.stringify(this.books), {encoding : 'utf-8'})  return this.books  },  updateLanguageByID : function(language\_id, language){  this.languages[language\_id] = language  fs.writeFileSync(this.languagesTable, JSON.stringify(this.languages), {encoding : 'utf-8'})  return this.languages  },  deleteBookByID : function(book\_id){  let deleted\_book = this.books[book\_id]  this.books.splice(book\_id,1)  return { deleted\_book : deleted\_book, books : this.books}  },  deleteLanguageByID : function(language\_id){  let deleted\_language = this.books[language\_id]  this.languages.splice(language\_id,1)  return { deleted\_language : deleted\_language, languages : this.languages}  },  getBooks : function(){  return this.books  },  getLanguages : function(){  return this.languages  },  getBookById : function(book\_id){  const result = this.books[book\_id]  return result == undefined ? 'Not Found' : result  },  getLanguageById : function(language\_id){  const result = this.languages[language\_id]  return result == undefined ? 'Not Found' : result  },  addBook : function(book){  this.books.push(book)  fs.writeFileSync(this.booksTable, JSON.stringify(this.books), {encoding : 'utf-8'})  },  addLanguage : function(language){  this.languages.push(language)  fs.writeFileSync(this.languagesTable, JSON.stringify(this.languages), {encoding : 'utf-8'})  }  }  module.exports = Db |
| books.js |
| const express = require('express')  const router = express.Router()  const server = require('../../server')  router.get('/', (req, res, next) => {  result = server.db.getBooks()  res.status(200).json(result)  })  router.post('/', (req, res, next) => {  const book = {  name : req.body.name,  price : req.body.price,  language : req.body.language  }  server.db.addBook(book)  res.status(200).json({  book\_added : book  })  })  router.get('/:bookName', (req, res, next) => {  const id = req.params.bookName  res.status(200).json(server.db.getBookById(id))  })  router.delete('/:book\_id', (req, res, next) => {  const book\_id = req.params.book\_id  res.status(200).json(server.db.deleteBookByID(book\_id))  })  router.patch('/:book\_id', (req, res, next) => {  const book\_id = req.params.book\_id    const book = {  name : req.body.name,  price : req.body.price,  language : req.body.language  }  res.status(200).json(server.db.updateBookByID(book\_id, book))  })  module.exports = router |
| languages.js |
| const express = require('express')  const router = express.Router()  const server = require('../../server')  router.get('/', (req, res, next) => {  result = server.db.getLanguages()  res.status(200).json(result)  })  router.post('/', (req, res, next) => {  const language = {  name : req.body.name,  }  server.db.addLanguage(language)  res.status(200).json({  language\_added : language  })  })  router.get('/:languageId', (req, res, next) => {  const languageId = req.params.languageId  res.status(200).json(server.db.getLanguageById(languageId))  })  router.delete('/:language\_id', (req, res, next) => {  const language\_id = req.params.language\_id  res.status(200).json(server.db.deleteLanguageByID(language\_id))  })  router.patch('/:language\_id', (req, res, next) => {  const language\_id = req.params.language\_id  const language = {  name : req.body.name,  }  res.status(200).json(server.db.updateLanguageByID(language\_id, language))  })  module.exports = router |

1. Приклад відправки запитів за допомою Postman наведено у рисунках 1-3

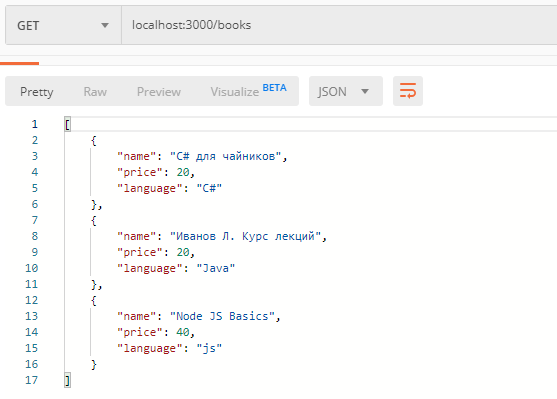


Рисунок 1 – GET запит за адресою /books

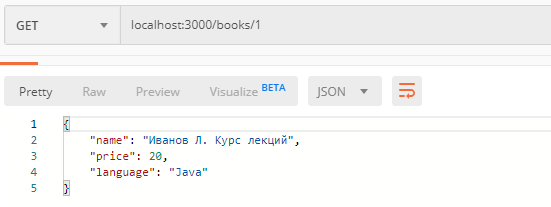


Рисунок 1 – GET запит за адресою /books/:id

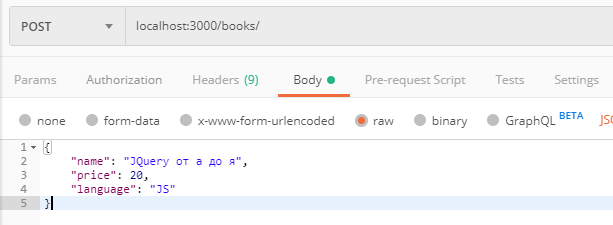




Рисунок 3 – POST запит за адресою /books

Висновки

При виконанні цієї лабораторної було створено сервер та API для доступу до даних що знаходятся у теках серверу. Були реалізовані дві сутності, а також додані даних до них за допомогою запитів.