МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Кафедра комп’ютерних наук та інформаційних систем

**Лабораторна робота № 4**

з курсу “Серверне програмування”

на тему:

“Створення серверів засобами Express.js ”

**Виконав:**

студент групи ПМ-41

Михайлюк І.І

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Перевірив:**

ст. викладач кафедри КНІС

к.т.н. Ізмайлов А. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка, підпис)

м. Івано-Франківськ

2024

Мета роботи: Навчитись застосовувати Express.js для створення серверів.

1.1 Теоретичні відомості

Node.js – це середовище виконання (runtime) для JavaScript на основі V8

JavaScript Engine (високопродуктивний JavaScript та WebAssembly рушій з

відкритим вихідним кодом від Google, написаний на С++). Використовує

асинхронну подійно-залежну (event-driven) неблокуючу модель вводу-виводу,

яка дає змогу запускати асинхронні JavaScript-застосунки на desktop-системах.

У загальному випадку, Node.js має два способи застосування:

– локальне середовище виконання для JavaScript-сценарії та застосунків;

– серверний функціонал для роботи бізнес-логіки веб-застосунку.

Для створення застосунків у Node.js можна використовувати майже всі

стандартні конструкції мови JavaScript.

Для роботи з файлами у Node.js призначений модуль fs.

Потік (Stream) представляє собою потік даних. Існують різні типи потоків,

серед яких можна виділити потоки для читання і потоки для запису. Канал Pipe

– це зв'язок між потоком для читання та потоком для запису, що дозволяє одразу

зчитувати дані з потоку для читання у потік для запису.

Фреймворк Express використовує модуль Node.js http, але при цьому надає

ряд готових абстракцій, які спрощують створення сервера та серверної логіки,

зокрема, обробка відправлених форм, робота з куками, CORS та інше.

1.2 Хід роботи

1.2.1 У створеному на GitHub репозиторії для курсу створити нову підпапку

для завдання лабораторної.

1.2.2 Написати сервер на Express.js, який запускається за адресою

http://localhost:3000 та забезпечує наступний функціонал:

– При зверненні клієнта за адресою «/» сервер повертає html-сторінку

із заголовком першого рівня, який містить текст «Сервер на

Express.js [ПІБ]», де [ПІБ] – прізвище, ім’я та по-батькові виконавця

лабораторної роботи;

– При зверненні клієнта за адресою «/about.html» сервер повертає

html-сторінку із заголовком першого рівня, який містить текст «Про

нас», та текстовий абзац деякої інформації (можливе використання

тексту-заповнювача Lorem ipsum). Файл сторінки повинен

знаходитись у деякій папці та подаватись як статичний файл;

– При зверненні клієнта за адресою «/getdata» сервер повертає json-

файл, який містить інформацію у наступному форматі:

{

date: дата і час (timestamp) надходження запиту на сервер за

часовим поясом UTC,

user: ім’я користувача, під обліковим записом якого

виконується сервер (модуль os)

}

– При зверненні клієнта за адресою «/myfile» сервер повертає txt-

файл file1.txt (із довільним вмістом, файл має бути створений

заздалегідь ще до запуску сервера), який знаходиться у папці data у

корені папки сервера. Файл має відобразитися у браузері;

– При зверненні клієнта за адресою «/mydownload» сервер повертає

txt-файл file2.txt (із довільним вмістом, файл має бути створений

заздалегідь ще до запуску сервера), який знаходиться у папці data у

корені папки сервера. Файл має завантажитись браузером на

пристрій користувача. Для цього можна, наприклад, повідомити

браузеру про необхідність завантаження наступним заголовком

відгуку:

{

'Content-Type': 'text/plain',

'Content-Disposition': 'attachment; filename="file2.txt"'

}

– При зверненні клієнта за адресою «/myarchive» сервер повертає

gzip-архів file1.txt.gz, у який заархівовано файл file1.txt (із

довільним вмістом, файл має бути створений заздалегідь ще до

запуску сервера), який знаходиться у папці data у корені папки

сервера. Архів має завантажитись браузером на пристрій

користувача. Для цього можна, наприклад, повідомити браузеру

про необхідність завантаження наступним заголовком відгуку:

{

'Content-Type': 'application/gzip',

'Content-Disposition': 'attachment; filename="file1.txt.gz"'

}

– При зверненні за будь-якою із ненаведених вище адрес сервер

повертає статусний код 404 та текст «There is no such resource».

1.2.3 Надати викладачеві посилання на репозиторій проекту (посилання

повинне бути у форматі: https://github.com/username/userrepo/subfolder).

const express = require("express");

const fs = require('fs');

const os = require('os');

const zlib = require('zlib');

const server = express();

server.get( ["/home",'/'],(request,response) =>{

    response.sendFile(\_\_dirname + "/home.html")

});

server.get('/about',(request,response) =>{

    response.sendFile(\_\_dirname + "/about.html")

});

server.get('/getdata',(request,response) =>{

    let now = Date.now();

    let timestamp = new Date(now).toISOString();

    let user = os.userInfo().username;

    let data = {

        date: timestamp,

        user: user

    };

    response.writeHead(200, { 'Content-Type': 'application/json' });

    response.end(JSON.stringify(data));

});

server.get('/myfile',(request,response) =>{

    // var fileData = "data error";

    fs.readFile('./myfile.txt',(error,fileData) =>{

        if(error){

            response.sendStatus(500);

        }

        fs.readFile(\_\_dirname + '/myfile.html', (error, data) => {

            if (error) {

                response.sendStatus(500);

            } else {

                const htmlText = data.toString().replace(/{fileData}/g, fileData);

                response.send(htmlText);

            }

        });

    });

});

server.get('/mydownload',(request,response) =>{

    fs.access('./myfile2.txt',(error)=>{

        if(error){

            response.sendStatus(500)

            console.log(500)

        }

        else{

              response.download('./myfile2.txt');

        }

    })

});

server.get('/myarchive',(request,response) =>{

    fs.access('./myfile.txt',(error)=>{

        if(error){

            response.sendStatus(500);

            console.log(0)

        }

        else{

            fs.readFile("./myfile.txt",(error,data)=>{

                if(error){

                    response.sendStatus(500);

                    console.log(1)

                }

                else{

                    zlib.gzip(data,(error,compressedData)=>{

                        if(error){

                            response.sendStatus(500);

                            console.log(2)

                        }

                        else{//тимчасовий файл для скачування стиснутих даних "compressedData"

                            const tempFilePath = \_\_dirname + "/tempFile.txt.gz";

                            fs.writeFile(tempFilePath,compressedData,(error)=>{

                                if(error){

                                    response.sendStatus(500);

                                    console.log(3)

                                    console.log(error)

                                }

                                else{

                                    response.download(tempFilePath,"myfile.txt.gz",(error)=>{

                                        if(error){

                                            response.sendStatus(500);

                                            console.log(4)

                                        };

                                        fs.unlink(tempFilePath, (error) => {

                                            if (error) {

                                                console.error('Failed to delete temporary file:', error);

                                            }

                                        });

                                    });

                                }

                            });

                        };

                    });

                }

            });

        }

    })

});

server.get("\*",(\_,response) =>{

    response.status(404).send("«There is no such resource».");

});

server.listen(3000);