МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

по дисциплине «РПИ»

на тему: «Http/Https»

Выполнил: студент гр. ИП-32

Суховенко Э.С.

Принял: преподаватель

Процкая М.А.

Гомель 2022

**Цель работы:** освоить возможности NodeJs по созданию локальных серверов с выводом статической информации, ознакомиться с фреймворком Express.

**Задание**

**Задание 1** Создать сервер на NodeJs без использования фреймворка express. На базе сервера разработать статический web-сайт, состоящий из нескольких страниц. Тема выбирается студентом самостоятельно. К страницам сайта применить стилизацию. Предусмотреть переходы между станицами. Страницы храняться как статические файлы.

**Задание 2** Выполнить задание1 с использованием фреймворка Express. Дополнительно: все переходы логгировать в отдельный файл server.log.

**Задание 3** Используя фреймворк Express, разработать REST API. В качестве предметной области взять задание на курсовую работу. В качестве источника данных создать файл в формате JSON. Установить Postman и протестировать все разработанный REST API.

Предметная область: пиццерия.

**Код программы**

**//** WEB\_SITE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// out/express.txt

HOME --- Mon, 07 Mar 2022 17:28:07 GMT  
SECOND --- Mon, 07 Mar 2022 17:28:11 GMT  
HOME --- Mon, 07 Mar 2022 17:28:12 GMT  
FIRST --- Mon, 07 Mar 2022 17:28:13 GMT  
HOME --- Mon, 07 Mar 2022 17:28:13 GMT

// out/http.txt

HOME --- Mon, 07 Mar 2022 17:27:52 GMT  
FIRST --- Mon, 07 Mar 2022 17:27:54 GMT  
HOME --- Mon, 07 Mar 2022 17:27:54 GMT  
SECOND --- Mon, 07 Mar 2022 17:27:55 GMT  
HOME --- Mon, 07 Mar 2022 17:27:55 GMT  
SECOND --- Mon, 07 Mar 2022 17:27:55 GMT  
HOME --- Mon, 07 Mar 2022 17:27:56 GMT  
FIRST --- Mon, 07 Mar 2022 17:27:57 GMT  
HOME --- Mon, 07 Mar 2022 17:27:58 GMT

// public/pages/first/index.html

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Page 1</title>  
 <link rel="stylesheet" href="index.css">  
</head>  
<body>  
 <h1>First page!</h1>  
 <button id="alertButton">Click to show alert!</button>  
 <button id="switchButton">Click to switch page</button>  
 <button id="homeButton">HOME</button>  
 <script src="index.js"></script>  
</body>  
</html>

// public/pages/first/index.css

body {  
 font-size: 20pt;  
 font-weight: bold;  
 font-style: italic;  
}  
  
h1 {  
 color: brown;  
}

// public/pages/first/index.js

const alertButton = document.getElementById('alertButton')  
const switchButton = document.getElementById('switchButton')  
const homeButton = document.getElementById('homeButton')  
  
alertButton.addEventListener('click', () => {  
 *alert*('FIRST PAGE, ALERT!')  
})  
  
switchButton.addEventListener('click', () => {  
 history.pushState({}, null, '/second')  
 location.reload()  
 *// const response = await fetch('http://localhost:3000', {  
 // method: 'POST',  
 // body: JSON.stringify({  
 // message: 'Switch page',  
 // }),  
 // })  
 // const json = await response.text()  
 // const data = JSON.parse(json)  
 // console.log(data)*})  
  
homeButton.addEventListener('click', () => {  
 history.pushState({}, null, '/')  
 location.reload()  
})

// public/pages/second/index.html

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Page 2</title>  
 <link rel="stylesheet" href="index.css">  
</head>  
<body>  
 <h1>Second page!</h1>  
 <button id="alertButton">Click to show alert!</button>  
 <button id="switchButton">Click to switch page</button>  
 <button id="homeButton">HOME</button>  
 <script src="index.js"></script>  
</body>  
</html>

// public/pages/second/index.css

body {  
 font-size: 20pt;  
 font-weight: bold;  
 font-style: italic;  
}  
  
h1 {  
 color: orange;  
}

// public/pages/second/index.js

const alertButton = document.getElementById('alertButton')  
const switchButton = document.getElementById('switchButton')  
const homeButton = document.getElementById('homeButton')  
  
alertButton.addEventListener('click', () => {  
 *alert*('SECOND PAGE, ALERT!')  
})  
  
switchButton.addEventListener('click', () => {  
 history.pushState({}, null, '/first')  
 location.reload()  
})  
  
homeButton.addEventListener('click', () => {  
 history.pushState({}, null, '/')  
 location.reload()  
})

// public/pages/home/index.html

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Home page</title>  
 <link rel="stylesheet" href="index.css">  
</head>  
<body>  
 <h1>Home page!</h1>  
 <button id="firstPage">Go to first page</button>  
 <button id="secondPage">Go to second page</button>  
 <script src="index.js"></script>  
</body>  
</html>

// public/pages/home/index.css

body {  
 font-size: 20pt;  
 font-weight: bold;  
 font-style: italic;  
}  
  
h1 {  
 color: red;  
}

// public/pages/home/index.js

const firstPage = document.getElementById('firstPage')  
const secondPage = document.getElementById('secondPage')  
  
firstPage.addEventListener('click', () => {  
 history.pushState({}, null, '/first')  
 location.reload()  
})  
  
secondPage.addEventListener('click', () => {  
 history.pushState({}, null, '/second')  
 location.reload()  
})

// src/index.ts

import { httpServer } from './servers/http/http'  
import { expressServer } from './servers/express/express'  
import { LogPath, Port } from './constants/constants'  
import \* as fs from 'fs'  
  
fs.*truncateSync*(LogPath.*http*)  
fs.*truncateSync*(LogPath.*express*)  
  
httpServer.listen(Port.*http*, () => {  
 console.log(`Http server listening on port ${Port.*http*}!`)  
})  
  
expressServer.listen(Port.*express*, () => {  
 console.log(`Express server listening on port ${Port.*express*}!`)  
})

// src/constants/constants.ts

export const PAGES\_PATH = './public/pages'  
  
export enum Port {  
 *http* = 3000,  
 *express* = 4000,  
}  
  
export enum LogPath {  
 *http* = './out/http.txt',  
 *express* = './out/express.txt',  
}  
  
export const FILE\_PATHS = {  
 '/': `${PAGES\_PATH}/home`,  
 '/first': `${PAGES\_PATH}/first`,  
 '/second': `${PAGES\_PATH}/second`,  
}  
  
export enum FileExtension {  
 *html* = 'html',  
 *css* = 'css',  
 *js* = 'js',  
}  
  
export enum Route {  
 *home* = '/',  
 *first* = '/first',  
 *second* = '/second',  
}  
  
export enum FileRoute {  
 *indexCss* = '/index.css',  
 *indexJs* = '/index.js',  
}

// src/helpers/helpers.ts

import { FileExtension } from '../constants/constants'  
  
export const *getFilePathByExtension* = (pagesPath: string, extension: FileExtension) => `${pagesPath}/index.${extension}`

// servers/express.ts

import express from 'express'  
import { FILE\_PATHS, FileExtension, FileRoute, LogPath, Route } from '../../constants/constants'  
import { *getFilePathByExtension* } from '../../helpers/helpers'  
import path from 'path'  
import fs from 'fs'  
  
let currentRoute = Route.*home*export const expressServer = *express*()  
  
expressServer.get(FileRoute.*indexCss*, (req: *express*.Request, res: *express*.Response) => {  
 res.sendFile(path.resolve(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*css*)))  
})  
  
expressServer.get(FileRoute.*indexJs*, (req: *express*.Request, res: *express*.Response) => {  
 res.sendFile(path.resolve(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*js*)))  
})  
  
expressServer.get(Route.*home*, (req: *express*.Request, res: *express*.Response) => {  
 currentRoute = Route.*home* fs.*appendFileSync*(LogPath.*express*, `HOME --- ${new Date().toUTCString()}\n`)  
 res.sendFile(path.resolve(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*html*)))  
})  
  
expressServer.get(Route.*first*, (req: *express*.Request, res: *express*.Response) => {  
 currentRoute = Route.*first* fs.*appendFileSync*(LogPath.*express*, `FIRST --- ${new Date().toUTCString()}\n`)  
 res.sendFile(path.resolve(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*html*)))  
})  
  
expressServer.get(Route.*second*, (req: *express*.Request, res: *express*.Response) => {  
 currentRoute = Route.*second* fs.*appendFileSync*(LogPath.*express*, `SECOND --- ${new Date().toUTCString()}\n`)  
 res.sendFile(path.resolve(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*html*)))  
})  
  
expressServer.get('\*', (req: *express*.Request, res: *express*.Response) => {  
 res.send('Wrong url, try "/"!')  
})

// servers/http.ts

import \* as fs from 'fs'  
import \* as http from 'http'  
import { IncomingMessage } from 'http'  
import { *getFilePathByExtension* } from '../../helpers/helpers'  
import { FILE\_PATHS, FileExtension, FileRoute, LogPath, Route } from '../../constants/constants'  
  
let currentRoute = Route.*home*const getResolvers: { [key: string]: () => string } = {  
 [FileRoute.*indexCss*]: () => fs.*readFileSync*(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*css*)).toString(),  
 [FileRoute.*indexJs*]: () => fs.*readFileSync*(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*js*)).toString(),  
 [Route.*first*]: () => {  
 currentRoute = Route.*first* fs.*appendFileSync*(LogPath.*http*, `FIRST --- ${new Date().toUTCString()}\n`)  
 return fs.*readFileSync*(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*html*)).toString()  
 },  
 [Route.*second*]: () => {  
 currentRoute = Route.*second* fs.*appendFileSync*(LogPath.*http*, `SECOND --- ${new Date().toUTCString()}\n`)  
 return fs.*readFileSync*(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*html*)).toString()  
 },  
 [Route.*home*]: () => {  
 currentRoute = Route.*home* fs.*appendFileSync*(LogPath.*http*, `HOME --- ${new Date().toUTCString()}\n`)  
 return fs.*readFileSync*(*getFilePathByExtension*(FILE\_PATHS[currentRoute], FileExtension.*html*)).toString()  
 },  
}  
  
const process = (req: IncomingMessage, res: http.ServerResponse, jsonString: string) => {  
 const parsedData = JSON.parse(jsonString)  
 console.log(parsedData)  
 res.end(JSON.stringify({  
 receivedData: parsedData,  
 message: 'success',  
 }))  
}  
  
const handlePOSTRequest = (req: IncomingMessage, res: http.ServerResponse) => {  
 let jsonString = ''  
  
 req.on('data', (chunk: string) => jsonString += chunk)  
 req.on('end', () => process(req, res, jsonString))  
}  
  
const requestListener = (req: IncomingMessage, res: http.ServerResponse) => {  
 if (req.method === 'POST') {  
 handlePOSTRequest(req, res)  
 } else if (req.method === 'GET' && getResolvers[req.url]) {  
 res.end(getResolvers[req.url]())  
 } else {  
 res.end('Wrong url, try "/"!')  
 }  
}  
  
export const httpServer = http.*createServer*(requestListener)

**//** API \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// res/pizza.json

[  
 {  
 "name": "Hut",  
 "size": "small",  
 "hasCheese": true,  
 "hasMeat": true  
 },  
 {  
 "name": "Margherita",  
 "size": "middle",  
 "hasCheese": true,  
 "hasMeat": false  
 },  
 {  
 "name": "Sicilian",  
 "size": "large",  
 "hasCheese": false,  
 "hasMeat": true  
 },  
 {  
 "name": "Neapolitan",  
 "size": "small",  
 "hasCheese": false,  
 "hasMeat": false  
 },  
 {  
 "name": "Calzone",  
 "size": "large",  
 "hasCheese": false,  
 "hasMeat": true  
 },  
 {  
 "name": "Greek",  
 "size": "middle",  
 "hasCheese": true,  
 "hasMeat": true  
 }  
]

// src/constants/constants.ts

export const PIZZA\_PATH = './res/pizza.json'  
  
export const SERVER\_PORT = 5000

// src/constants/types.ts

export enum Size {  
 *small* = 'small',  
 *middle* = 'middle',  
 *large* = 'large',  
}  
  
export type Pizza = {  
 name: string  
 size: Size  
 hasCheese: boolean  
 hasMeat: boolean  
}

// src/index.ts

import \* as fs from 'fs'  
import express from 'express'  
import { Pizza } from './constants/types'  
import { PIZZA\_PATH, SERVER\_PORT } from './constants/constants'  
  
const app = *express*()  
const pizzas: Pizza[] = JSON.parse(fs.*readFileSync*(PIZZA\_PATH).toString())  
  
app.get('/', (req: *express*.Request, res: *express*.Response) => {  
 const maxCount = +req.query?.maxCount ?? pizzas.length  
 res.json(pizzas.slice(0, maxCount))  
})  
  
app.post('/', (req: *express*.Request, res: *express*.Response) => {  
 let jsonString = ''  
 req.on('data', (chunk => jsonString += chunk))  
 req.on('end', () => {  
 try {  
 const pizzaQuery: Pizza = JSON.parse(jsonString)  
 const filteredPizzas: Pizza[] = pizzas  
 .filter(pizza => pizzaQuery.size === undefined || pizza.size === pizzaQuery.size)  
 .filter(pizza => pizzaQuery.hasMeat === undefined || pizza.hasMeat === pizzaQuery.hasMeat)  
 .filter(pizza => pizzaQuery.hasCheese === undefined || pizza.hasCheese === pizzaQuery.hasCheese)  
 res.json(filteredPizzas)  
 } catch (error) {  
 res.json({ message: 'Error', details: error.toString() })  
 }  
 })  
})  
  
app.listen(SERVER\_PORT, () => {  
 console.log(`Server is listening on port ${SERVER\_PORT}`)  
})

**Результат**



Рисунок 1

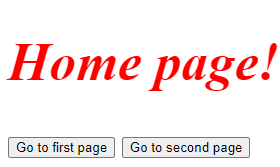


Рисунок 2

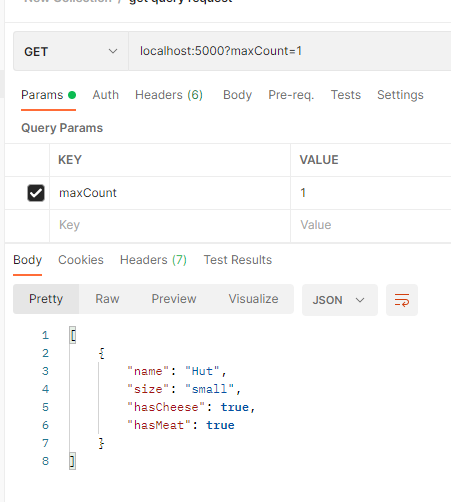


Рисунок 3

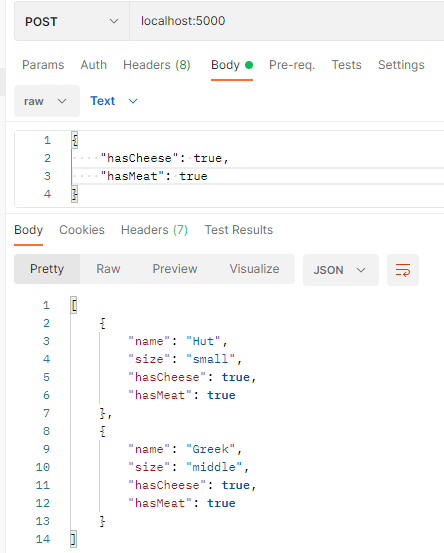


Рисунок 4

**Вывод**: освоил возможности NodeJs по созданию локальных серверов с выводом статической информации, ознакомился с фреймворком Express.