Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 7

Тема: Изучение взаимодействия между серверами, Знакомство с дочерними процессами

Студент Унтилова А.О.

Группа ИУ7-56

Оценка (баллы) _____

Преподаватель Попов А.Ю.

Москва.
2020 г.

Цель: Ознакомиться с взаимодействием между серверами, передачей параметров по скрипту, научиться работать с дочерними процессами.

Лабораторная работа №7

Задание 1

Создать сервер А. На стороне сервера хранится файл с содержимым в формате JSON. При получении запроса на /insert/record идёт добавление записи в файл. При получении запроса на /select/record идёт получение записи из файла. Каждая запись хранит информацию о машине (*название* и *стоимость*).

Создать сервер Б. На стороне сервера хранится файл с содержимым в формате JSON. Каждая запись в файле хранит информацию о складе и массиве машин, находящихся на данном складе. То есть каждая запись хранит в себе название склада (*строку*) и массив названий машин (*массив строк*). При получении запроса на /insert/record идёт добавление записи в файл. При получении запроса на /select/record идёт получение записи из файла.

Создать сервер С. Сервер выдаёт пользователю страницы с формами для ввода информации. При этом сервер взаимодействует с серверами А и Б. Реализовать для пользователя функции:

- создание нового типа машины
- получение информации о стоимости машины по её типу
- создание нового склада с находящимися в нём машинами
- получение информации о машинах на складе по названию склада

Реализовать удобный для пользователя интерфейс взаимодействия с системой (использовать поля ввода и кнопки).

Для работы с сервером С:

```
"use strict";

// импорт библиотек
const express = require("express");
const request = require("request");

// запускаем сервер
const app = express();
```

```
const port = 4000;
app.listen(port);
console.log(`Server on port ${port}`);

const way = __dirname + "/static";
app.use(express.static(way));

// заголовки в ответ клиенту
app.use(function(req, res, next) {
    res.header("Cache-Control", "no-cache, no-store, must-revalidate");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-
Type, Accept");
    res.header("Access-Control-Allow-Origin", "*");
    next();
});

// функция для отправки POST запроса на другой сервер
function sendPost(url, body, callback) {
    // задаём заголовки
    const headers = {};
    headers["Cache-Control"] = "no-cache, no-store, must-revalidate";
    headers["Connection"] = "close";
    // отправляем запрос
    request.post({
        url: url,
        body: body,
        headers: headers,
    }, function (error, response, body) {
        if(error) {
            callback(null);
        } else {
            callback(body);
        }
    });
}

// принимаем GET запрос и отправляем POST запрос на другой сервер
app.get("/create/new/car", function(request, response) {
    const car = request.query.car;
    const cost = request.query.cost;
    console.log(car + ' ' + cost);
    sendPost("http://localhost:4002/insert/record", JSON.stringify({
        car: car,
        cost: cost
    }), function(answerString) {
        //const answerObject = JSON.parse(answerString);
        //const answer = answerObject.answer;
        //response.end("Answer: " + answer);
        response.end(answerString);
    });
});

app.get("/get/data/car", function(request, response) {
```

```
    const car = request.query.car;
    sendPost("http://localhost:4002/select/record", JSON.stringify({
        car: car
    }), function(answerString) {
        const answerObject = JSON.parse(answerString);
        //const answer = answerObject.answer;
        response.end(answerString);
    });
});

app.get("/create/new/warehouse", function(request, response) {
    const warehouse = request.query.warehouse;
    const cars_arr = request.query.cars_arr;
    sendPost("http://localhost:4001/insert/record", JSON.stringify({
        warehouse: warehouse,
        cars_arr: cars_arr
    }), function(answerString) {
        // const answerObject = JSON.parse(answerString);
        // const answer = answerObject.answer;
        response.end(answerString);
    });
});

app.get("/get/data/warehouse", function(request, response) {
    const warehouse = request.query.warehouse;
    sendPost("http://localhost:4001/select/record", JSON.stringify({
        warehouse: warehouse
    }), function(answerString) {
        // const answerObject = JSON.parse(answerString);
        // const answer = answerObject.answer;
        response.end(answerString);
    });
});
```

Для работы с сервером B:

```
"use strict";

// импорт библиотеки
const express = require("express");
const filename = "warehouse.txt"

// запускаем сервер
const app = express();
const port = 4001;
app.listen(port);
console.log("Server on port " + port);
```

```
// заголовки для ответа
app.use(function(req, res, next) {
    res.header("Cache-Control", "no-cache, no-store, must-revalidate");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-
Type, Accept");
    res.header("Access-Control-Allow-Origin", "*");
    next();
});

function add(filename, obj_str){
    let arr = [];
    const fs = require("fs");
    let file_str = fs.readFileSync(filename, "utf8");
    arr = JSON.parse(file_str);
    arr.push(obj_str);
    fs.writeFileSync(filename, JSON.stringify(arr));
}

function get(filename, key){
    let arr = [];
    const fs = require("fs");
    let file_str = fs.readFileSync(filename, "utf8");
    arr = JSON.parse(file_str);

    for (let i = 0; i < arr.length; i++){
        if (arr[i]['warehouse'] === key)
            return JSON.stringify(arr[i]);
    }
    return null;
}

// загрузка тела
function loadBody(request, callback) {
    let body = [];
    request.on('data', (chunk) => {
        body.push(chunk);
    }).on('end', () => {
        body = Buffer.concat(body).toString();
        callback(body);
    });
}

// приём запроса
app.post("/insert/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const warehouse = obj.warehouse;
        const cars_arr = obj.cars_arr;
        add(filename, {'warehouse' : warehouse, 'cars_arr' : cars_arr});
        let s = "Warehouse added.";
        response.end(JSON.stringify({
            answer: s
        }));
```

```
    });
});

app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const warehouse = obj.warehouse;
        const s = get(filename, warehouse);
        response.end(JSON.stringify({
            answer: s
        }));
    });
});
```

Для работы с сервером А:

```
"use strict";

// импорт библиотеки
const express = require("express");
const filename = "cars.txt"

// запускаем сервер
const app = express();
const port = 4002;
app.listen(port);
console.log("Server on port " + port);

// заголовки для ответа
app.use(function(req, res, next) {
    res.header("Cache-Control", "no-cache, no-store, must-revalidate");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-
Type, Accept");
    res.header("Access-Control-Allow-Origin", "*");
    next();
});

function add(filename, obj_str){
    let arr = [];
    const fs = require("fs");
    let file_str = fs.readFileSync(filename, "utf8");
    arr = JSON.parse(file_str);
    arr.push(obj_str);
    fs.writeFileSync(filename, JSON.stringify(arr));
}

function get(filename, key){
    let arr = [];
    const fs = require("fs");
    let file_str = fs.readFileSync(filename, "utf8");
    arr = JSON.parse(file_str);
```

```
    for (let i = 0; i < arr.length; i++){
        if (arr[i]['car'] == key){
            return JSON.stringify(arr[i]);
        }
    }
    return null;
}

// загрузка тела
function loadBody(request, callback) {
    let body = [];
    request.on('data', (chunk) => {
        body.push(chunk);
    }).on('end', () => {
        body = Buffer.concat(body).toString();
        callback(body);
    });
}

// приём запроса
app.post("/insert/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const car = obj.car;
        const cost = obj.cost;
        add(filename, {'car' : car, 'cost' : parseInt(cost)});
        let s = "Car added.";
        response.end(JSON.stringify({
            answer: s
        }));
    });
});

app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const car = obj.car;
        const s = get(filename, car);
        response.end(JSON.stringify({
            answer: s
        }));
    });
});
```

Задание 2

Написать скрипт, который принимает на вход число и считает его факториал. Скрипт должен получать параметр через process.argv.

Написать скрипт, который принимает на вход массив чисел и выводит на экран факториал каждого числа из массива. Скрипт принимает параметры через process.argv.

При решении задачи вызывать скрипт вычисления факториала через execSync.

```
const execSync = require('child_process').execSync;

// функция для вызова программы и получения результата её работы
function useCmd(s) {
    const options = {encoding: 'utf8'};
    const cmd = s.toString();
    //метод не вернется, пока дочерний процесс не будет полностью закрыт.
    const answer = execSync(cmd, options);
    return answer.toString();
}

// получаем факториал (убираем первыеы 2 параметра из переданных)
let arr = process.argv.slice(2);
// console.log(arr);

for (let i = 0; i < arr.length; i++){
    let factorialCommand = `node factorial.js ${arr[i]}`;
    console.log(factorialCommand);
    let factorial = useCmd(factorialCommand);
    console.log(factorial);
}

"use strict";

function factorial(x){
    let result = 1;
    for (let i = 2; i <= parseInt(x); i++){
        result *= i;
    }
    console.log(result);
}

factorial(process.argv[2]);
```

```
PS D:\перенос  25.08.20\labs\5_sem_EVM\task7\t2> node index.js '1' '5' '10'
node factorial.js 1
1

node factorial.js 5
120

node factorial.js 10
3628800
```

Вывод: Во время выполнения лабораторной работы я ознакомилась с взаимодействием между серверами, передачей параметров по скрипту, научитлась работать с дочерними процессами.