|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки 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 \_\_\_\_\_\_

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

Дисциплина Архитектура ЭВМ

Студент Мишина Елена Викторовна

*фамилия, имя, отчество*

Группа ИУ7-54Б

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Попов А.Ю.

*подпись, дата фамилия, и.о.*

*2020 г.*

Оглавление

[Цель работы: 3](#_Toc54994414)

[Task 7 4](#_Toc54994415)

[1. Задание 1 4](#_Toc54994416)

[2. Задание 2 10](#_Toc54994417)

[Task 8 12](#_Toc54994418)

[1. Задание 1 12](#_Toc54994419)

[Вывод: 12](#_Toc54994420)

# Цель работы:

Написание программ, демонстрирующих работу взаимодействие между серверами, передачу параметров скрипту, дочерние процессы.

# Task 7

# Задание 1

**Условие задачи:**

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 C. 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).

**Листинг кода:**

**Сервер А:**

"use strict";

const fs = require("fs")

const express = require("express");

const app = express();

const port = 5001;

app.listen(port);

console.log("Server on port " + port);

app.get("/insert/record", function(request, response) {

    const name = request.query.n

    if (!name) return response.end(JSON.stringify({result: "Error name"}))

    const price = request.query.p

    if (!price) return response.end(JSON.stringify({result: "Error price"}))

    console.log(price)

    const priceInt = parseInt(price)

    const JSONstr = fs.readFileSync("data.txt", "utf8");

    let data = JSON.parse(JSONstr)

    data.push({"name": name, "price": priceInt})

    console.log(priceInt)

    data = JSON.stringify(data)

    fs.writeFileSync("data.txt", data);

    response.end(JSON.stringify({result: "Car data has been successfully added"}))

})

app.get("/select/record", function(request, response) {

    const JSONstr = fs.readFileSync("data.txt", "utf8");

    response.end(JSONstr)

})

**Сервер B**

"use strict";

const fs = require("fs")

const express = require("express");

const app = express();

const port = 5002;

app.listen(port);

console.log("Server on port " + port);

app.get("/insert/record", function(request, response) {

    const StoreHouseName = request.query.s

    if (!StoreHouseName) return response.end(JSON.stringify({result: "Error StoreHouseName"}))

    const cars = request.query.c

    if (!cars) return response.end(JSON.stringify({result: "Error cars"}))

    const JSONstr = fs.readFileSync("data.txt", "utf8");

    let data = JSON.parse(JSONstr)

    let arr\_cars = cars.split(",")

    data.push({"StoreHouseName": StoreHouseName, "cars": arr\_cars})

    data = JSON.stringify(data)

    fs.writeFileSync("data.txt", data);

    response.end(JSON.stringify({result: "Storehouse data has been successfully added"}))

})

app.get("/select/record", function(request, response) {

    const JSONstr = fs.readFileSync("data.txt", "utf8");

    response.end(JSONstr)

})

**Сервер С:**

"use strict";

const express = require("express");

const fs = require("fs")

const request = require("request");

const app = express();

const port = 5000;

app.listen(port);

console.log(`Server on port ${port}`);

function sendPost(url, callback) {

    request({method: 'GET', url: url},

        function (error, response, body) {

            if(error) {

                callback(null);

            }

            else {

                callback(body);

            }

        }

    )

}

app.get("/main", function(request, response) {

    const nameString = request.query.page;

    if (fs.existsSync(nameString)) {

        const contentString = fs.readFileSync(nameString, "utf8");

        response.end(contentString);

    } else {

        const contentString = fs.readFileSync("bad.html", "utf8");

        response.end(contentString);

    }

})

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

    const n = request.query.n

    const p = request.query.p

    console.log(p)

    sendPost("http://localhost:5001/insert/record?n=" + n + "&p=" + p, function(answerString) {

        response.end(answerString)

    })

})

app.get("/find/cost", function(request, response) {

    const n = request.query.n

    sendPost("http://localhost:5001/select/record", function(answerString) {

        const arr = JSON.parse(answerString)

        for (let i = 0; i < arr.length; i++) {

            if (arr[i].name === n) {

                response.end(JSON.stringify({result: arr[i].price}))

                return

            }

        }

        response.end(JSON.stringify({result: "Not found"}))

    })

})

app.get("/new/housestore", function(request, response) {

    const s = request.query.s

    const c = request.query.c

    sendPost("http://localhost:5002/insert/record?s=" + s + "&c=" + c, function(answerString) {

        response.end(answerString)

    })

})

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

    const s = request.query.s

    sendPost("http://localhost:5002/select/record", function(answerString) {

        const arr = JSON.parse(answerString)

        for (let i = 0; i < arr.length; i++) {

            if (arr[i].StoreHouseName === s) {

                response.end(JSON.stringify({result: arr[i].cars}))

                return

            }

        }

        response.end(JSON.stringify({result: "Not found"}))

    })

})

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>TASK 7</title>

</head>

<body>

    <h1>TASK 7</h1>

    <hr width="20%" color="black" align="left" size="5">

    <p>Название машины</p>

    <input id="field-first" type="text" spellcheck="false" autocomplete="off">

    <p>Цена машины</p>

    <input id="field-second" type="text" spellcheck="false" autocomplete="off">

    <br><br>

    <button onclick="NewCar()">Добавить машину</button>

    <hr width="20%" color="black" align="left" size="5">

    <p>Название машины</p>

    <input id="field-third" type="text" spellcheck="false" autocomplete="off">

    <button onclick="FindCost()">Узнать цену</button>

    <hr width="20%" color="black" align="left" size="5">

    <p>Название склада</p>

    <input id="field-fourth" type="text" spellcheck="false" autocomplete="off">

    <p>Содержимое склада</p>

    <input id="field-fifth" type="text" spellcheck="false" autocomplete="off">

    <br><br>

    <button onclick="NewHouseStore()">Создать склад</button>

    <hr width="20%" color="black" align="left" size="5">

    <p>Содержимое склада</p>

    <input id="field-sixth" type="text" spellcheck="false" autocomplete="off">

    <button onclick="DataHouseStore()">Показать содержимое</button>

    <hr width="20%" color="black" align="left" size="5">

    <script>

    "use strict";

    //obj.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");

    function ajaxGet(urlString, callback) {

        let obj = new XMLHttpRequest()

        obj.open("GET", urlString, true)

        obj.send(null)

        obj.onload = function() {

            callback(obj.response);

        }

    }

    function NewCar() {

        const a = document.getElementById("field-first");

        const strA = a.value

        const b = document.getElementById("field-second");

        const strB = b.value

        const url = "/new/car?n=" + strA + "&p=" + strB

        ajaxGet(url, function(stringAnswer) {

            const JSONresult = stringAnswer;

            const OBJECTresult = JSON.parse(JSONresult);

            const result = OBJECTresult.result

            alert(result)

        })

    }

    function FindCost() {

        const a = document.getElementById("field-third");

        const strA = a.value

        const url = "/find/cost?n=" + strA

        ajaxGet(url, function(stringAnswer) {

            const JSONresult = stringAnswer;

            const OBJECTresult = JSON.parse(JSONresult);

            const result = OBJECTresult.result

            alert(result)

        })

    }

    function NewHouseStore() {

        const a = document.getElementById("field-fourth");

        const strA = a.value

        const b = document.getElementById("field-fifth");

        const strB = b.value

        const url = "/new/housestore?s=" + strA + "&c=" + strB

        ajaxGet(url, function(stringAnswer) {

            const JSONresult = stringAnswer;

            const OBJECTresult = JSON.parse(JSONresult);

            const result = OBJECTresult.result

            alert(result)

        })

    }

    function DataHouseStore() {

        const a = document.getElementById("field-sixth");

        const strA = a.value

        const url = "/data/housestore?s=" + strA

        ajaxGet(url, function(stringAnswer) {

            const JSONresult = stringAnswer;

            const OBJECTresult = JSON.parse(JSONresult);

            const result = OBJECTresult.result

            alert(result)

        })

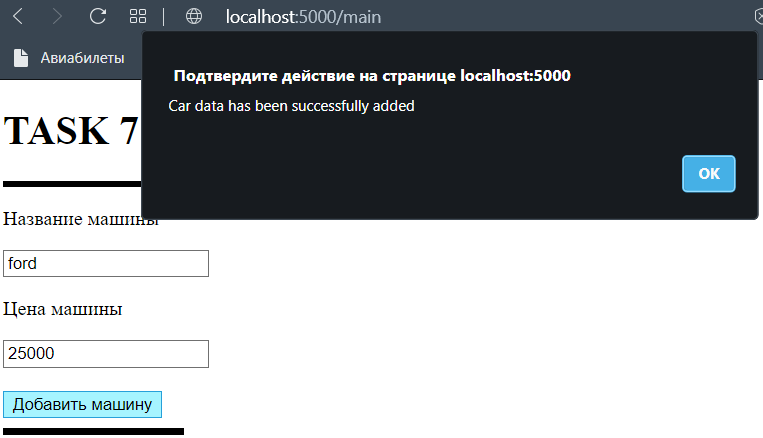
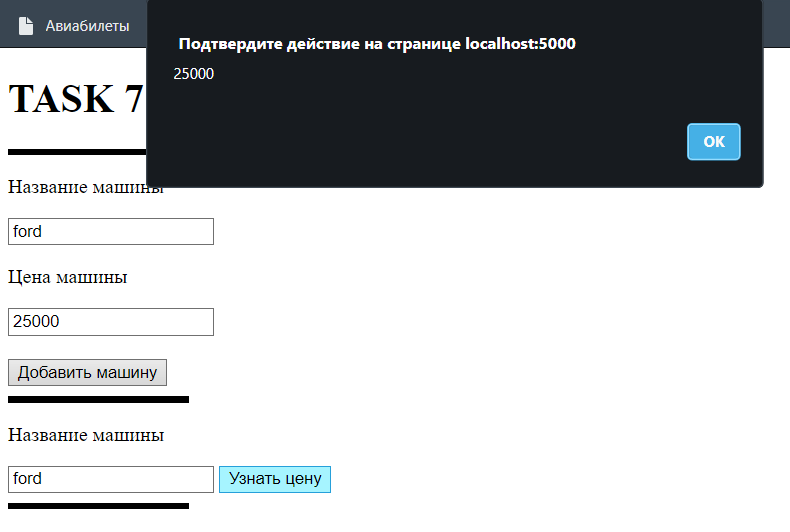
    }

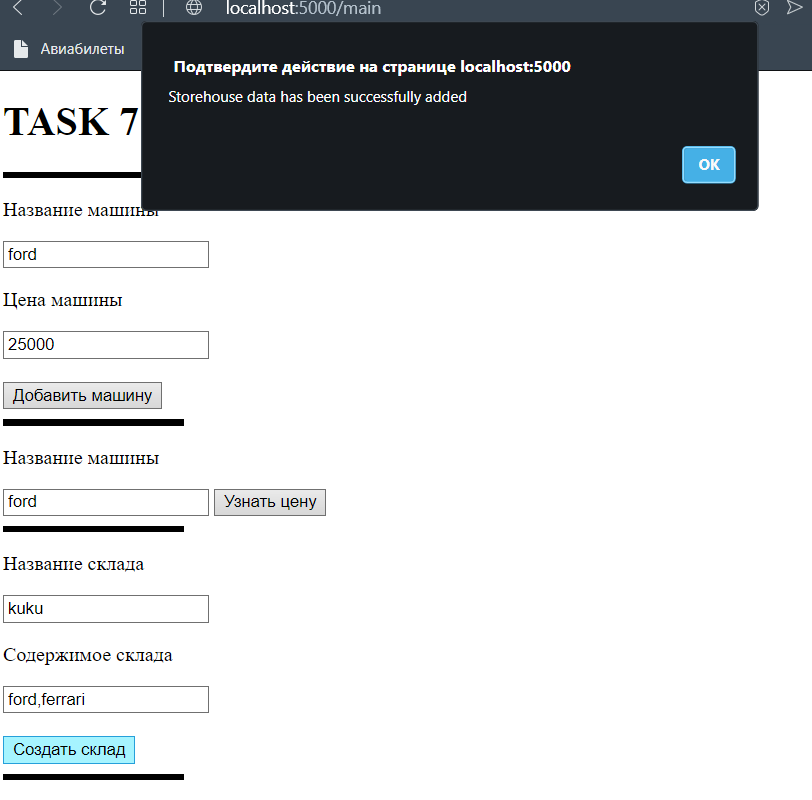
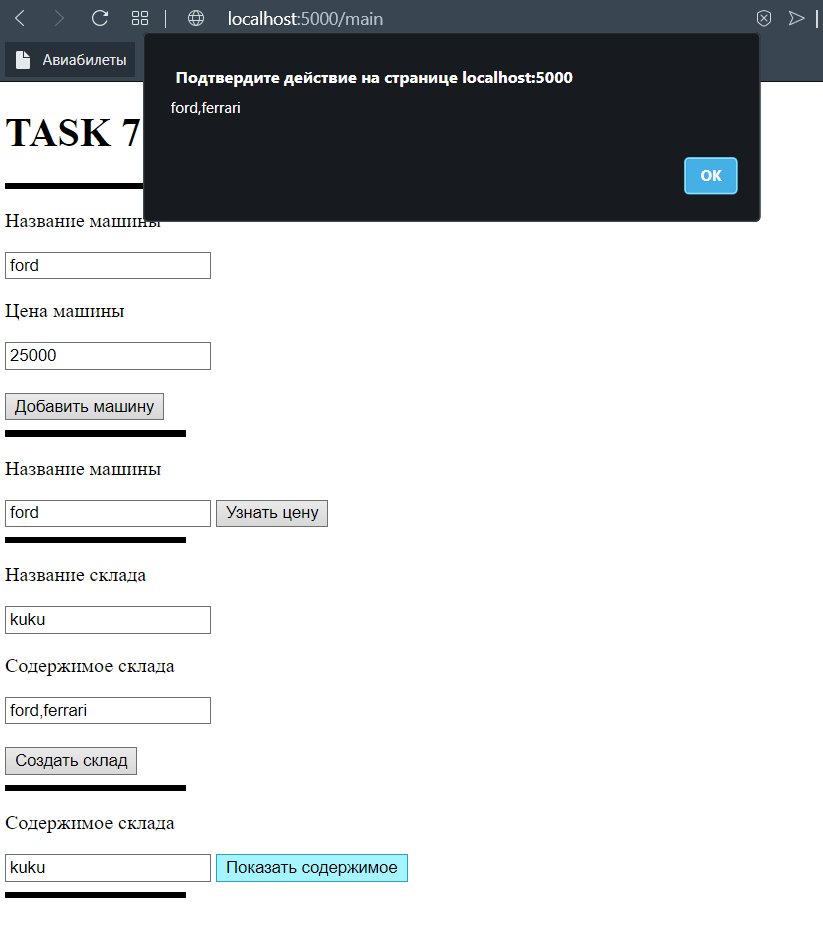
    </script>

</body>

</html>

**Тестирование:**

# Задание 2

**Условие задачи:**

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.

**Листинг кода:**

"use strict";

// получаем параметры скрипта

const number = parseInt(process.argv[2])

function factorial(number){

    let fac = 1

    for (let i = 2; i <= number; i++){

        fac \*= i

    }

    return fac

}

// вывод сообщения на экран

console.log(factorial(number));

"use strict";

// получаем параметры скрипта

const arr = (process.argv[2]).split(",")

// импортируем библиотеку

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();

    const answer = execSync(cmd, options);

    return answer.toString();

}

function mas\_factorial(arr){

    for (let i = 0; i < arr.length; i++){

        const facCommand = `node task7\_2.js ${parseInt(arr[i])}`;

        let fac = useCmd(facCommand);

        fac = parseInt(fac);

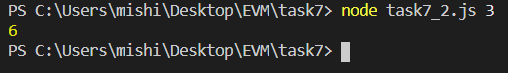
        console.log(fac);

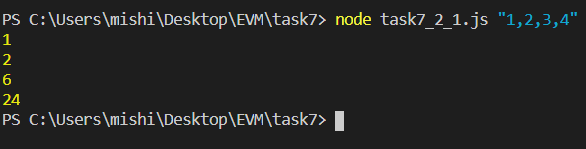
    }

}

mas\_factorial(arr)

**Тестирование:**





# Task 8

# Задание 1

**Условие задачи:**

С клавиатуры считываются числа A и B. Необходимо вывести на экран все числа Фибоначчи, которые принадлежат отрезку от A до B.

**Листинг кода:**

ok.

input(A) :- read(A); ok.

fib(N, Res) :-

N is 0, Res is 0, !;

N is 1, Res is 1, !;

N1 is N-1, N2 is N-2,

fib(N1, Buf1), fib(N2, Buf2),

Res is Buf1 + Buf2.

writeGreater(A, X) :- A =< X, write(X), nl; ok.

fibNumbers(N, A, B) :-

fib(N, Result), Result =< B,

writeGreater(A, Result), Y is (N+1), fibNumbers(Y, A, B);

ok.

f :- input(A), input(B), fibNumbers(0, A, B).

**Тестирование:**



# Вывод:

Были приобретены новые знания по JavaScript, а также по Prolog. Все задачи, представленные в данной лабораторной работе выполнены успешно.