|  |  |
| --- | --- |
| 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](#_Toc54790147)

[Task 3 4](#_Toc54790148)

[1. Задание 1 4](#_Toc54790149)

[2. Задание 2 5](#_Toc54790150)

[3. Задание 3 6](#_Toc54790151)

[4. Задание 4 8](#_Toc54790152)

[5. Задание 5 9](#_Toc54790153)

[6. Задание 6 10](#_Toc54790154)

[7. Задание 7 11](#_Toc54790155)

[Task 4 13](#_Toc54790156)

[1. Задание 1 13](#_Toc54790157)

[2. Задание 2 15](#_Toc54790158)

[3. Задание 3 16](#_Toc54790159)

[4. Задание 4 18](#_Toc54790160)

[Вывод: 21](#_Toc54790161)

# Цель работы:

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

# Task 5

# 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)

        })

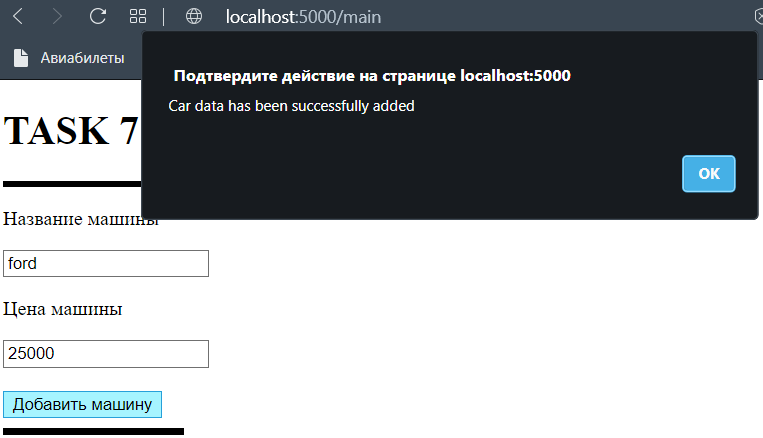
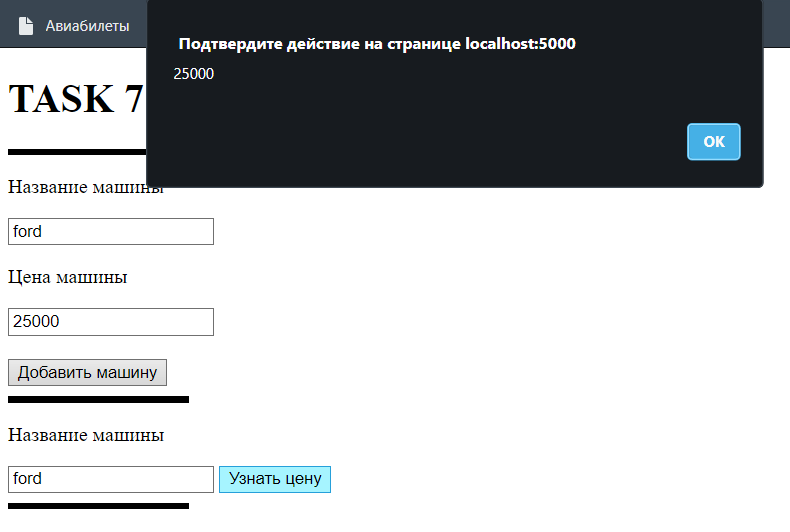
    }

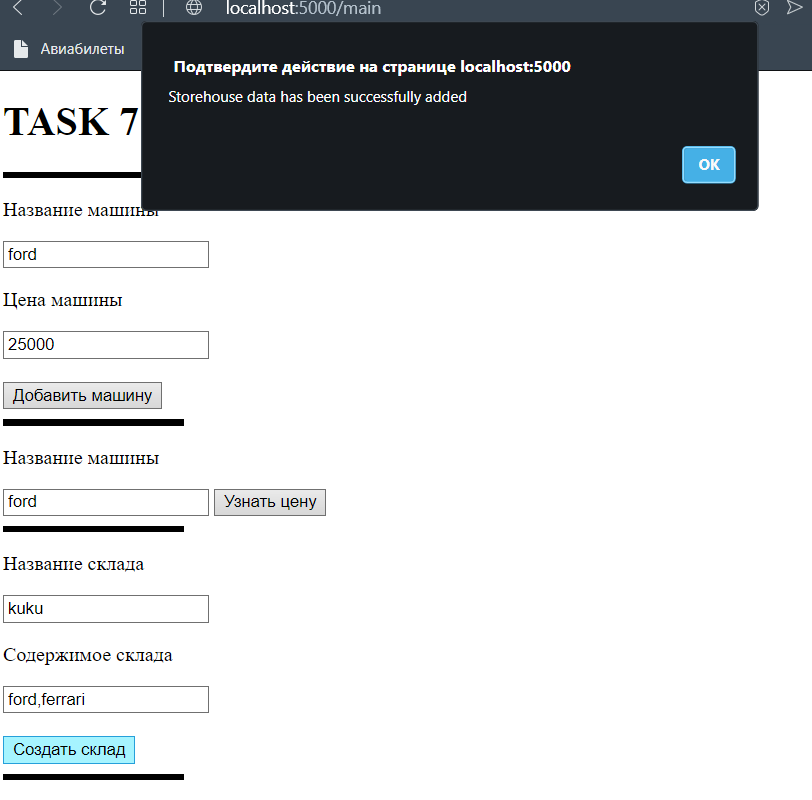
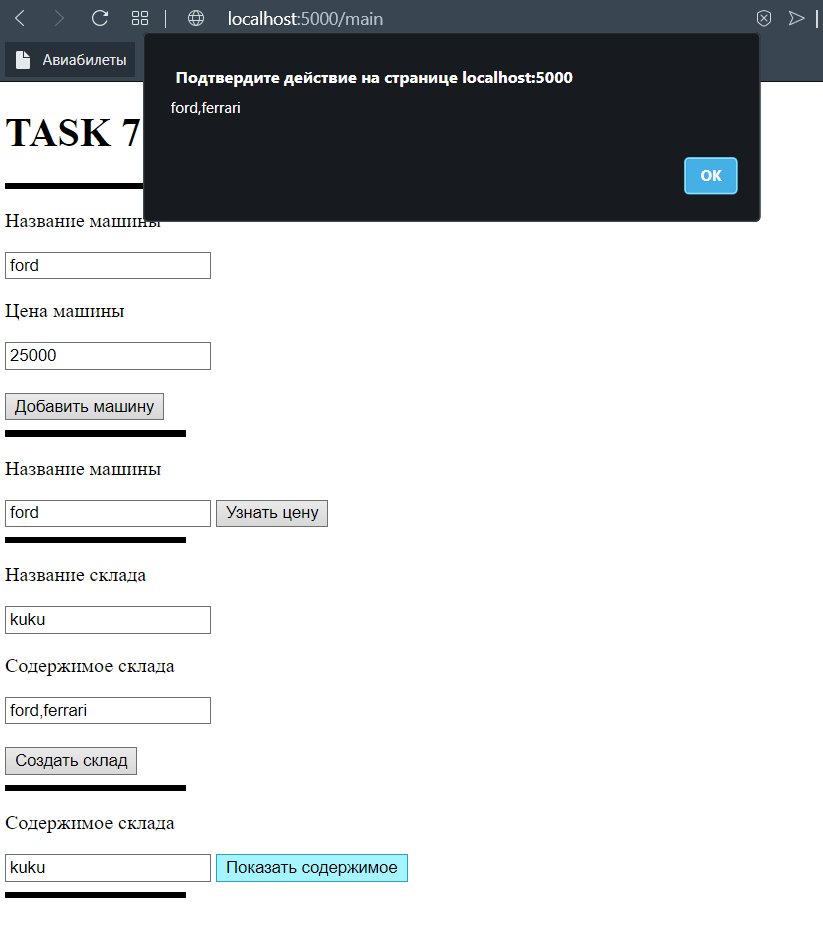
    </script>

</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.

# 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:**

"use strict";

let mas = [

    { NameGameValue: 'GTA', DescGameValue: 'dfghjk', AgeValue: 3 },

    { NameGameValue: 'WC', DescGameValue: 'kjhgfd', AgeValue: 12 },

    { NameGameValue: 'WWQ', DescGameValue: 'fghjkl', AgeValue: 18 },

    { NameGameValue: 'TTL', DescGameValue: 'qwerty', AgeValue: 16 }

  ]

  // импорт библиотеки

  const express = require("express");

  // запускаем сервер

  const app = express();

  const port = 5000;

  app.listen(port);

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

  // активируем шаблонизатор

  app.set("view engine", "hbs");

  // 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();

  });

  // выдача страницы с информацией о кафедре

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

    const a = request.query.a;

    let count = 0

    const infoObject = {

        game\_mas: []

    }

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

        if (mas[i].AgeValue < a)

        {

            infoObject.game\_mas[count] = mas[i]

            count++

        }

    }

    response.render("pageGame.hbs", infoObject);

  });

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Игры</title>

</head>

<body>

<h1>Игры</h1>

<hr align="left" width="200" size="5" color="black" />

{{#each game\_mas}}

    Название игры: {{NameGameValue}}

    <br>

    <br>

    Описание игры: {{DescGameValue}}

    <br>

    <br>

    Возрастные ограничения: {{AgeValue}}

    <br>

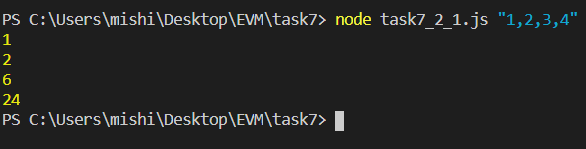
    <hr align="left" width="200" size="5" color="black" />

{{/each}}

</body>

</html>

**Тестирование:**



# Задание 2

**Условие задачи:**

Создать сервер. В оперативной памяти на стороне сервера создать массив, в котором хранится информация о пользователях (логин, пароль, хобби, возраст). На основе cookie реализовать авторизацию пользователей. Реализовать возможность для авторизованного пользователя просматривать информацию о себе.

**Листинг кода:**

"use strict";

let mas\_user = [

    { login: 'petya', password: '1230', hobbi: 'fghj', age: 15 },

    { login: 'vasya', password: 'qweert', hobbi: 'dfghjk', age: 22 },

    { login: 'grisha', password: 'sdfgh', hobbi: 'dfghjk', age: 33 },

    { login: 'olya', password: '1111', hobbi: 'dertyu', age: 45 },

    { login: 'katya', password: '3456', hobbi: 'qwerrty', age: 17 }

  ]

// импортируем библиотеки

const express = require("express");

const cookieSession = require("cookie-session");

// запускаем сервер

const app = express();

const port = 5000;

app.listen(port);

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

// работа с сессией

app.use(cookieSession({

    name: 'session',

    keys: ['hhh', 'qqq', 'vvv'],

    maxAge: 24 \* 60 \* 60 \* 1000 \* 365

}));

// 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");

    next();

});

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

    // получаем параметры запроса

    const login = request.query.login;

    const password = request.query.password;

    let flag = 0

    for (let i = 0; i < mas\_user.length; i++){

        if (mas\_user[i].login === login && mas\_user[i].password === password) {

            // выставляем cookie

            request.session.login = login;

            request.session.password = password;

            let infoObject = {

                login: mas\_user[i].login,

                HobbiValue: mas\_user[i].hobbi,

                AgeValue: mas\_user[i].age

            }

            flag = 1

            response.render("pageUser.hbs", infoObject);

        }

    }

    if (flag == 0){

        response.end("User doesn't exist")

    }

}

)

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Пользователь</title>

</head>

<body>

<h1>Пользователь {{login}}</h1>

<hr align="left" width="200" size="5" color="black" />

    Хобби: {{HobbiValue}}

    <br>

    <br>

    Возраст: {{AgeValue}}

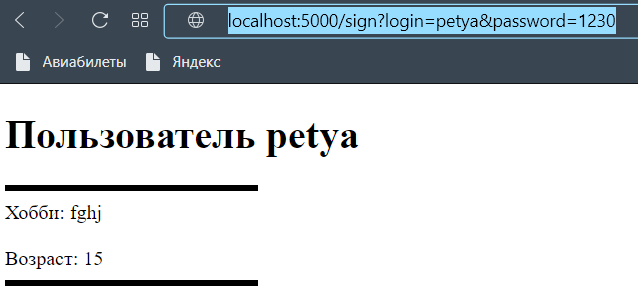
    <br>

    <hr align="left" width="200" size="5" color="black" />

</body>

</html>

**Тестирование:**



# Вывод:

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