|  |  |
| --- | --- |
| 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 №3**

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 с AJAX запросами GET, POST запросами в AJAX, а также работу с шаблонизатором и c сессиями в NodeJS.

# Task 5

# Задание 1

**Условие задачи:**

Создать сервер. Сервер должен выдавать страницу с тремя текстовыми полями и кнопкой. В поля ввода вбивается информация о почте, фамилии и номере телефона человека. При нажатии на кнопку "Отправить" введённая информация должна отправляться с помощью POST запроса на сервер и добавляться к концу файла (в файле накапливается информация). При этом на стороне сервера должна происходить проверка: являются ли почта и телефон уникальными. Если они уникальны, то идёт добавление информации в файл. В противном случае добавление не происходит. При отправке ответа с сервера клиенту должно приходить сообщение с информацией о результате добавления (добавилось или не добавилось). Результат операции должен отображаться на странице.

# Задание 2

**Условие задачи:**

Добавить серверу возможность отправлять клиенту ещё одну страницу. На данной странице должно быть поле ввода и кнопка. В поле ввода вводится почта человека. При нажатии на кнопку "Отправить" на сервер отправляется GET запрос. Сервер в ответ на GET запрос должен отправить информацию о человеке с данной почтой в формате JSON или сообщение об отсутствии человека с данной почтой.

# Задание 3

**Условие задачи:**

Оформить внешний вид созданных страниц с помощью CSS. Информация со стилями CSS для каждой страницы должна храниться в отдельном файле. Стили CSS должны быть подключены к страницам.

**Листинг кода index.js :**

"use strict";

const fs = require("fs");

// импортируем библиотеку

const express = require("express");

// запускаем сервер

const app = express();

const port = 5000;

app.listen(port);

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

// отправка статических файлов

const way = \_\_dirname + "/static";

app.use(express.static(way));

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 loadBody(request, callback) {

    let body = []

    request.on('data', (chunk) => {

        body.push(chunk);

    }).on('end', () => {

        body = Buffer.concat(body).toString();

        callback(body);

    });

}

app.post("/save", function(request, response) {

    loadBody(request, function(body) {

        const obj = JSON.parse(body);

        const stringJson = fs.readFileSync("data.json", "utf-8");

        let arr = JSON.parse(stringJson);

        let flag = false;

        for (const elem of arr) {

            if (elem.mail === obj["mail"] || elem.phone === obj["phone"]) {

                flag = true;

                response.end(JSON.stringify({

                    result: "Error. Already exists."

                }));

            }

        }

        if (!flag) {

            arr.push(obj);

            fs.writeFileSync("data.json", JSON.stringify(arr));

            response.end(JSON.stringify({

                result: "Ok. Save new user."

            }));

        }

    });

});

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

    const mail = request.query.mail;

    const stringJson = fs.readFileSync("data.json", "utf-8");

    let arr = JSON.parse(stringJson);

    let flag = false;

    for (const elem of arr) {

        if (elem.mail === mail) {

            flag = true;

            let ans = "<br><br>Mail: " + mail + "<br>";

            ans += "Phone number: " + elem.phone + "<br>";

            ans += "Surname: " + elem.surname;

            response.end(JSON.stringify({

                result: "Success! Found user: " + ans

            }));

        }

    }

    if (!flag) {

        response.end(JSON.stringify({

            result: "Error. User not found."

        }));

    }

});

**Листинг кода find.js :**

"use strict";

window.onload = function() {

    // input fields

    const f1 = document.getElementById("in1");

    // button

    const btn = document.getElementById("btn");

    // label

    const label = document.getElementById("result-label");

    // ajax get

    function ajaxGet(urlString, callback) {

        let r = new XMLHttpRequest();

        r.open("GET", urlString, true);

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

        r.send(null);

        r.onload = function() {

            callback(r.response);

        };

    };

    // click event

    btn.onclick = function() {

        const mail = f1.value;

        const url = `/find?mail=${mail}`;

        ajaxGet(url, function(stringAnswer) {

            const objectAnswer = JSON.parse(stringAnswer);

            const result = objectAnswer.result;

            if (result == "Error. User not found.") {

                label.style.color = 'red';

            }

            else {

                label.style.color = 'green';

            }

            label.innerHTML = `<br>Результат: ${result}`;

        });

    };

};

**Листинг кода reg.js :**

"use strict";

function ajaxPost(urlString, bodyString, callback) {

    let r = new XMLHttpRequest();

    r.open("POST", urlString, true);

    r.setRequestHeader("Content-Type", "application/json;charset=UTF-8");

    r.send(bodyString);

    r.onload = function () {

        callback(r.response);

    };

}

function makeAction() {

    const mail = document.getElementById("in1").value;

    const surname = document.getElementById("in2").value;

    const phone = document.getElementById("in3").value;

    if (!mail || !surname || !phone) {

        return

    }

    const new\_person = {"mail": mail, "phone": phone, "surname": surname};

    const url = `/save?mail=${mail}&phone=${phone}&surname=${surname}`;

    ajaxPost(

        url,

        JSON.stringify(new\_person),

        function (answerString) {

            const answerObject = JSON.parse(answerString);

            const result = answerObject.result;

            const elem = document.getElementById("result-label");

            if (result === "Ok. Save new user.") {

                elem.style.color = 'green';

            }

            else {

                elem.style.color = 'red';

            }

            elem.innerHTML = `${result}`;

        }

    );

}

**Листинг кода reg\_page.css и reg\_page.html :**

html, body {

    height: 100%;

    width: 100%;

    background: khaki;

    font-family: Geneva, Arial, Helvetica, sans-serif;

}

body {

    position: absolute;

    top: 30%;

    left: 22%;

    height: 300px;

    width: 600px;

    margin-top: -250px;

    margin-left: -300px;

  }

.field {

    clear:both;

    text-align:right;

    padding: 10px;

}

label {

    float:left;

    font-size: 24px;

    padding: 20px;

}

.main {

    float:left;

}

.btn-class {

    border-radius: 10px;

    padding: 20px 190px;

    text-align: center;

    vertical-align: middle;

    font-size: 40px;

    background: orange;

    color: white;

    cursor: pointer;

    display: inline-block;

}

.in-class {

    font-size:24px;

    height: 26px;

    padding: 20px;

    border-radius: 10px;

    cursor: pointer;

    display: inline-block;

}

.res-lab {

    text-align: center;

}

.main-lab {

    text-align: center;

    font-size: 50px;

}

<!DOCTYPE html>

<html>

    <head>

        <meta charset="UTF-8">

        <title>Моя страница</title>

        <link rel="stylesheet" href="/reg\_page.css">

    </head>

    <body>

        <h1 class="main-lab"> Регистрация</h1>

        <div class="main">

            <div class="field">

                <label for="n">Email: </label>

                <input type="text" id="in1", class="in-class", maxlength="25" size="20" value="">

            </div>

            <div class="field">

                <label for="ln">Surname: </label>

                <input type="text" id="in2", class="in-class", maxlength="25" size="20" value="">

            </div>

            <div class="field">

                <label for="a">Phone number: </label>

                <input type="text" id="in3", class="in-class", maxlength="25" size="20" value="">

            </div>

            <p><div id="btn" class="btn-class" onclick="makeAction()"> Добавить </div></p>

            <h2 id="result-label", class="res-lab"></h2>

        </div>

        <script src="./reg.js"></script>

    </body>

</html>

**Листинг кода find\_page.css и find\_page.html :**

html, body {

    height: 100%;

    width: 100%;

    background: khaki;

    font-family: Geneva, Arial, Helvetica, sans-serif;

}

body {

    position: absolute;

    top: 30%;

    left: 22%;

    height: 300px;

    width: 600px;

    margin-top: -250px;

    margin-left: -300px;

  }

.field {

    clear:both;

    text-align:right;

    padding: 10px;

}

label {

    float:left;

    font-size: 24px;

    padding: 20px;

}

.main {

    float:left;

}

.btn-class {

    border-radius: 10px;

    padding: 20px 190px;

    text-align: center;

    vertical-align: middle;

    font-size: 40px;

    background: orange;

    color: white;

    cursor: pointer;

    display: inline-block;

}

.in-class {

    font-size:24px;

    height: 26px;

    padding: 20px;

    border-radius: 10px;

    cursor: pointer;

    display: inline-block;

}

.res-lab {

    text-align: left;

}

.main-lab {

    text-align: left;

    font-size: 50px;

}

<!DOCTYPE html>

<html>

    <head>

        <meta charset="UTF-8">

        <title>Моя страница</title>

        <link rel="stylesheet" href="/find\_page.css">

    </head>

    <body>

        <h1 class="main-lab"> Поиск</h1>

        <div class="main">

            <div class="field">

                <label for="n">Email: </label>

                <input type="text" id="in1", class="in-class", maxlength="25" size="20" value="">

            </div>

            <p><div id="btn" class="btn-class"> Найти </div></p>

            <h2 id="result-label", class="res-lab"></h2>

        </div>

        <script src="./find.js"></script>

    </body>

</html>

**Тестирование:**

# 

# Task 6

# Задание 1

**Условие задачи:**

Создать сервер. В оперативной памяти на стороне сервера создать массив, в котором хранится информация о компьютерных играх (название игры, описание игры, возрастные ограничения). Создать страницу с помощью шаблонизатора. В url передаётся параметр возраст (целое число). Необходимо отображать на этой странице только те игры, у которых возрастное ограничение меньше, чем переданное в url значение.

**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>

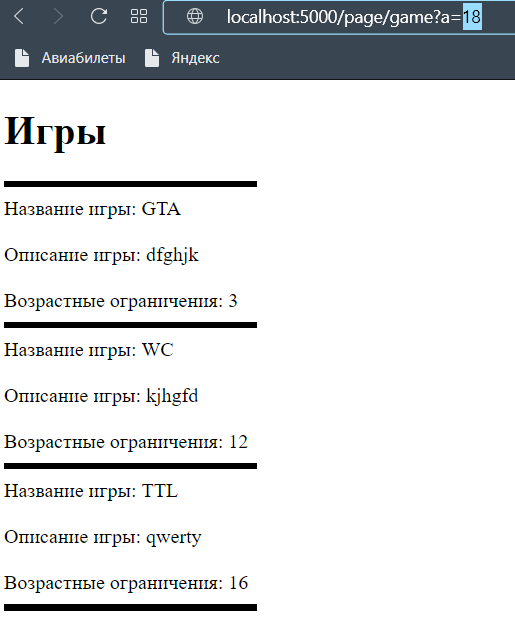
    <hr align="left" width="200" size="5" color="black" />

{{/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}}

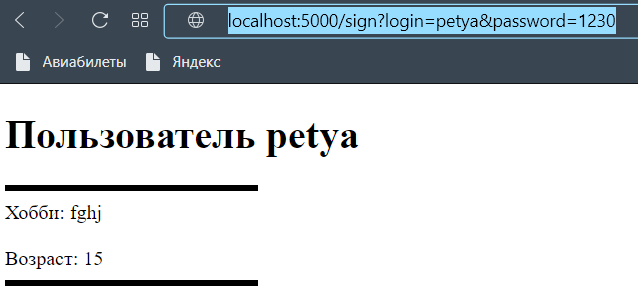
    <br>

    <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.