|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 № 3**

**По курсу «Архитектура ЭВМ»**

|  |  |
| --- | --- |
| **Тема**  AJAX запросы и работа с шаблонизатором  **Студент** Якуба Д. В.  **Группа** ИУ7-53Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Попов А. Ю. |  |

Москва

2020 г.

# Цели работы

* Освоение работы с AJAX запросами в ЯП JavaScript.
* Освоение работы с шаблонизатором.
* Изучение Cookie.

# Отчёт по разделу №5

## Задания раздела

### Условия

* 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.
* 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.
* 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.

### Код программы

Язык: JavaScript

**index.js**

"use strict";

const express = require("express");

const fs = require("fs");

let name = "data.txt";

const app = express();

const port = 5000;

app.listen(port);

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

app.use(express.static(\_\_dirname));

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 getInfo(mail)

{

    if (!fs.existsSync(name))

        return "Have no any data.";

    let readContents = fs.readFileSync(name, "utf-8");

    readContents = readContents.split(/\n| /);

    let index = readContents.indexOf(mail);

    let out = "";

    if (index >= 0)

        out += readContents[index] + ' ' + readContents[index + 1] + ' ' + readContents[index + 2];

    else

        out = "No user found";

    return out;

}

// http://localhost:5000/getUser.html

app.get("/getInfo", function(request, response)

{

    const mail = request.query.mail;

    let out = getInfo(mail);

    response.end(JSON.stringify({

        result: out

    }));

});

function loadBody(request, callback)

{

    let body = [];

    request.on('data', (chunk) =>

    {

        body.push(chunk);

    }).on('end', () => {

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

        callback(body);

    });

}

function loadInfoToFile(mail, lastName, telNum)

{

    if (!fs.existsSync(name))

        fs.writeFileSync(name, "");

    let readContents = fs.readFileSync(name, "utf-8");

    if (readContents.indexOf(mail) >= 0)

        return "Fail: mail already exists";

    if (readContents.indexOf(telNum) >= 0)

        return "Fail: telephone number already exists";

    readContents += '\n';

    readContents += mail + ' ';

    readContents += lastName + ' ';

    readContents += telNum;

    fs.writeFileSync(name, readContents);

    return "Successfully added";

}

// http://localhost:5000/registration.html

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

{

    loadBody(request, function(body)

    {

        const obj = JSON.parse(body);

        const mail = obj["mail"];

        const name = obj["name"];

        const telephone = obj["telephone"];

        let result = loadInfoToFile(mail, name, telephone);

        response.end(JSON.stringify(

        {

            result: result

        }));

    });

});

**Файл getUser.html**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Получение информации о пользователе</title>

    <link rel="stylesheet" href="/getStyle.css">

</head>

<body>

    <h1>GET запрос. Готов отдать всё, что угодно...</h1>

    <p>Почта человека, который Вам нужен:</p>

    <input class="inp" id="mail" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <button class="button" onclick="getInfo()">Получить данные!</button>

    <br>

    <br>

    <h1 id="result-label"></h1>

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

</body>

</html>

**Файл getStyle.css**

body {

    padding: 60px;

    background: rgb(0, 255, 234);

    font-family: sans-serif;

}

.button {

    padding: 10px;

    background: rgb(0, 255, 136);

    color: rgb(0, 0, 0);

    cursor: pointer;

    display: inline-block;

}

.inp {

    padding: 13px;

    background: rgb(0, 0, 0);

    color: rgb(255, 255, 255);

}

**Файл getInfo.js**

"use strict";

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

function getInfo() {

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

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

    ajaxGet(url, function(stringAnswer) {

        const objectAnswer = JSON.parse(stringAnswer);

        const result = objectAnswer.result;

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

    });

};

**Файл registration.html**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Отправка данных о пользователе на сервер</title>

    <link rel="stylesheet" href="/regStyle.css">

</head>

<body>

    <h1>POST запрос. Дайте нам свои данные!</h1>

    <p>Почта</p>

    <input class="inp" id="mail" type="text" spellcheck="false" autocomplete="off">

    <p>Фамилия</p>

    <input class="inp" id="name" type="text" spellcheck="false" autocomplete="off">

    <p>Номер телефона</p>

    <input class="inp" id="telephone" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <button class="button" onclick="registrate()">Отправить данные, кому попало!</button>

    <br>

    <br>

    <h1 id="result-label"></h1>

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

</body>

</html>

**Файл regStyle.css**

body {

    padding: 30px;

    background: rgb(21, 255, 0);

    font-family: fantasy;

}

.button {

    padding: 6px;

    background: rgb(247, 252, 0);

    color: rgb(0, 0, 0);

    cursor: pointer;

    display: inline-block;

}

.inp {

    padding: 6px;

    background: rgb(0, 0, 0);

    color: rgb(255, 255, 255);

}

**Файл registration.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 registrate()

    {

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

        const f2 = document.getElementById("name");

        const f3 = document.getElementById("telephone");

        let mail = f1.value;

        let name = f2.value;

        let telephone = f3.value;

        ajaxPost("/save/info", JSON.stringify(

            {

                mail, name, telephone

            }),

            function(answerString)

            {

                const answerObject = JSON.parse(answerString);

                const result = answerObject.result;

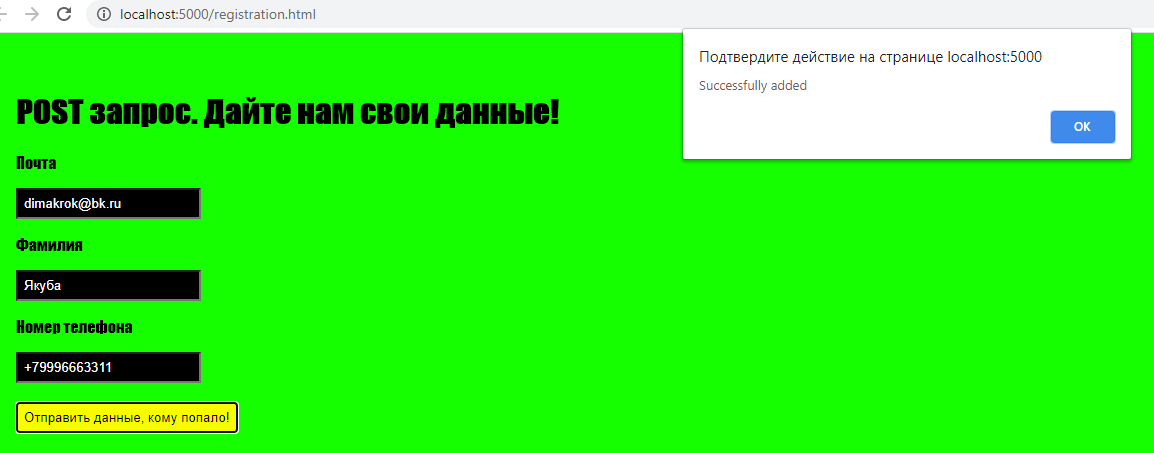
                alert(result);

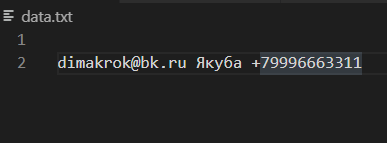
                document.getElementById("result-label").innerHTML = result;

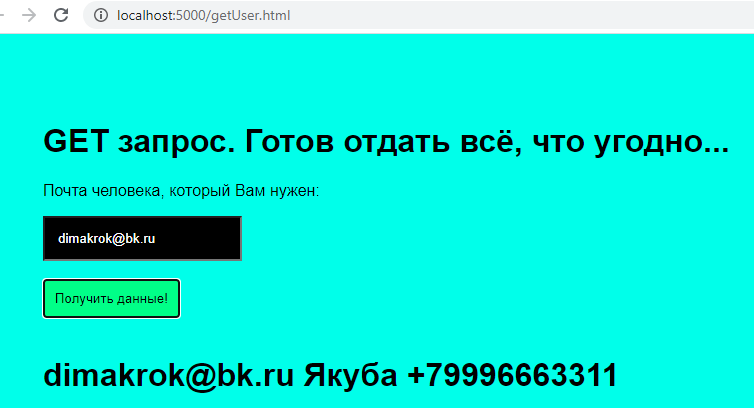
            });

    }

### Результаты тестирования







# Отчёт по разделу №6

## Задание 1

### Условие

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.

### Код программы

Язык: JavaScript

**index.js**

"use strict";

let gamesArr = [{name: "Dota", descr: "Gladly IceFrog", age: 16},

                {name: "Dota2", descr: "Place", age: 16},

                {name: "Furry Tales", descr: "UWU", age: 8},

                {name: "Zabiv", descr: "Hmm, smth wrong", age: 21},

                {name: "SuperSsunek", descr: "Sonic goes brrrrrrr", age: 0}

]

const express = require("express");

const app = express();

const port = 5000;

app.listen(port);

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

app.set("view engine", "hbs");

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();

});

// http://localhost:5000/page/games?age=16

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

    let out = [];

    for (let game of gamesArr)

        if (game.age <= request.query.age)

            out.push(game);

    const infoObject = {

        games: out

    };

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

});

**Файл getUser.html**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Получение информации о пользователе</title>

    <link rel="stylesheet" href="/getStyle.css">

</head>

<body>

    <h1>GET запрос. Готов отдать всё, что угодно...</h1>

    <p>Почта человека, который Вам нужен:</p>

    <input class="inp" id="mail" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <button class="button" onclick="getInfo()">Получить данные!</button>

    <br>

    <br>

    <h1 id="result-label"></h1>

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

</body>

</html>

**Файл gamesPage.hbs**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Игры всякие</title>

</head>

<body>

<h1>Игры по Вашему запросу:</h1>

{{#each games}}

    <div style="background: yellow; margin-bottom: 15px; padding: 8px;">

        Название игры: {{this.name}}

        <br>

        Описание: {{this.descr}}

        <br>

        Возраст: {{this.age}}

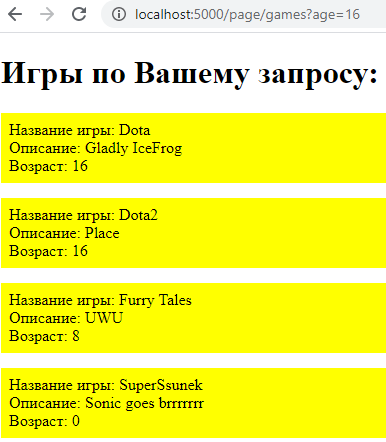
    </div>

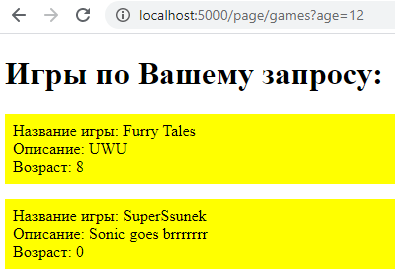
{{/each}}

</body>

</html>

### Результаты тестирования





## Задание 2

### Условие

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.

### Код программы

Язык: JavaScript

**index.js**

 "use strict";

 let users = [

     {login: "Hitler", password: "1945", hobby: "Suicide", age: 56},

     {login: "Stalin", password: "1941", hobby: "Repression", age: 74},

     {login: "Mussolini", password: "MamaMia,Pepperoni!", hobby: "Pizza", age: 61}

 ];

const express = require("express");

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

const app = express();

const port = 5000;

app.listen(port);

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

app.use(cookieSession({

    name: 'session',

    keys: ['hhh', 'qqq', 'vvv'],

    maxAge: 300 \* 1000

}));

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();

});

// http://localhost:5000/login?login=Stalin&&password=1941

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

    const login = request.query.login;

    const password = request.query.password;

    if(!login) return response.end("Login not set");

    if(!password) return response.end("Password not set");

    let found = false;

    let index = 0;

    for (let i = 0; i < users.length && !found; i++)

        if (users[i].login === login && users[i].password === password)

            found = true;

    if (!found)

    {

        response.end("Login failed!");

        return;

    }

    request.session.login = login;

    request.session.password = password;

    response.end("Authorized!");

});

// http://localhost:5000/auth

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

    console.log(request.session.hobby, request.session.age);

    if(!request.session.login || !request.session.password) return response.end("No cookies of auth!");

    const login = request.session.login;

    const password = request.session.password;

    let hobby = "";

    let age = 0;

    let found = false;

    for (let i = 0; i < users.length && !found; i++)

        if (users[i].login === login && users[i].password === password)

        {

            hobby = users[i].hobby;

            age = users[i].age;

        }

    const infoObject = {

        login: login,

        password: password,

        hobby: hobby,

        age: age

    };

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

});

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

    request.session = null;

    response.end("Cookies are deleted");

});

**Файл getUser.html**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Получение информации о пользователе</title>

    <link rel="stylesheet" href="/getStyle.css">

</head>

<body>

    <h1>GET запрос. Готов отдать всё, что угодно...</h1>

    <p>Почта человека, который Вам нужен:</p>

    <input class="inp" id="mail" type="text" spellcheck="false" autocomplete="off">

    <br>

    <br>

    <button class="button" onclick="getInfo()">Получить данные!</button>

    <br>

    <br>

    <h1 id="result-label"></h1>

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

</body>

</html>

**Файл showUserInfo.hbs**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Добро пожаловать на выставку!</title>

</head>

<body>

<h1>Информация о Вас:</h1>

<div style="background: rgb(0, 255, 76); margin-bottom: 15px; padding: 8px;">

    Имя пользователя: {{login}}

    <br>

    Пароль: {{password}}

    <br>

    Хобби: {{hobby}}

    <br>

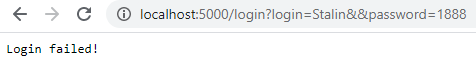
    Возраст: {{age}}

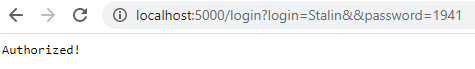
</div>

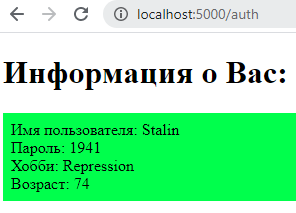
</body>

</html>

### Результаты тестирования







# Вывод

В результате выполнения работы:

* Была освоена работа с AJAX запросами в ЯП JavaScript.
* Была освоена работа с шаблонизатором.
* Были изучены Cookie.