|  |  |
| --- | --- |
| 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»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ДИСЦИПЛИНА «Архитектура ЭВМ»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 3**

|  |  |
| --- | --- |
| **Студент** \_\_Аникин И. А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Группа** \_\_ИУ7-51Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Преподаватель** \_\_Попов А. Ю.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |

Москва.

2020 г.

Оглавление

[Задание 5.1 3](#_Toc53869183)

[Задание 5.2 3](#_Toc53869184)

[Задание 5.3 3](#_Toc53869185)

[Задание 6.1 7](#_Toc53869186)

[Задание 6.2 9](#_Toc53869187)

Цель работы: научиться работать с cookie и ajax, а также научиться использовать шаблонизатор.

# Задание 5.1

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 5.2

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 5.3

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 формы add.html  
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Добавление человека в базу</title>

</head>

<body>

<h1>Добавление человека в базу</h1>

<div class="button7" onclick="makeAction()">Отправить запрос</div>

<link rel="stylesheet" type="text/css" href="addstyle.css">

<script>

"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 inputMessage = "Введите 3 строки через пробел";

const inputDefault = "email phone surname";

const input = prompt(inputMessage, inputDefault);

if(!input) return;

if(!input.trim()) return;

const arr = input.trim().split(" ");

const a = arr[0];

const b = arr[1];

const c = arr[2];

ajaxPost("/add", JSON.stringify({

a, b, c

}), function(answerString) {

const answerObject = JSON.parse(answerString);

const result = answerObject.result;

alert(result);

});

}

</script>

</body>

</html>

Код формы get.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Получение данных по email</title>

</head>

<body>

<h1>Получение данных по email</h1>

<form method="GET" action="/get">

<p>Введите email</p>

<input id="email" spellcheck="false" autocomplete="off">

<br>

<br>

<div class="knopka" id="get-user">Отправить запрос</div>

<link rel="stylesheet" type="text/css" href="getstyle.css">

</form>

<h1 id="result"></h1>

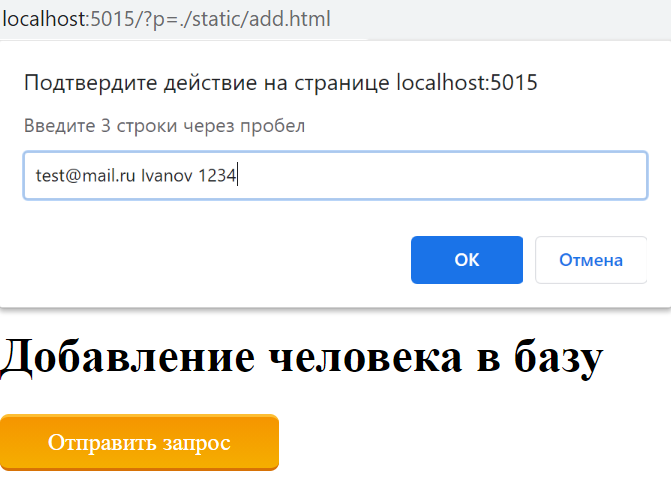
<script src="/code\_get.js"></script>

</body>

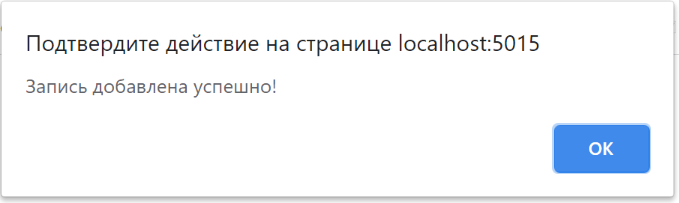
</html>

Примеры работы

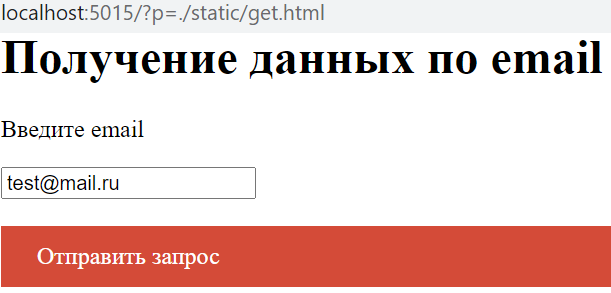
Проверим возможность добавления.



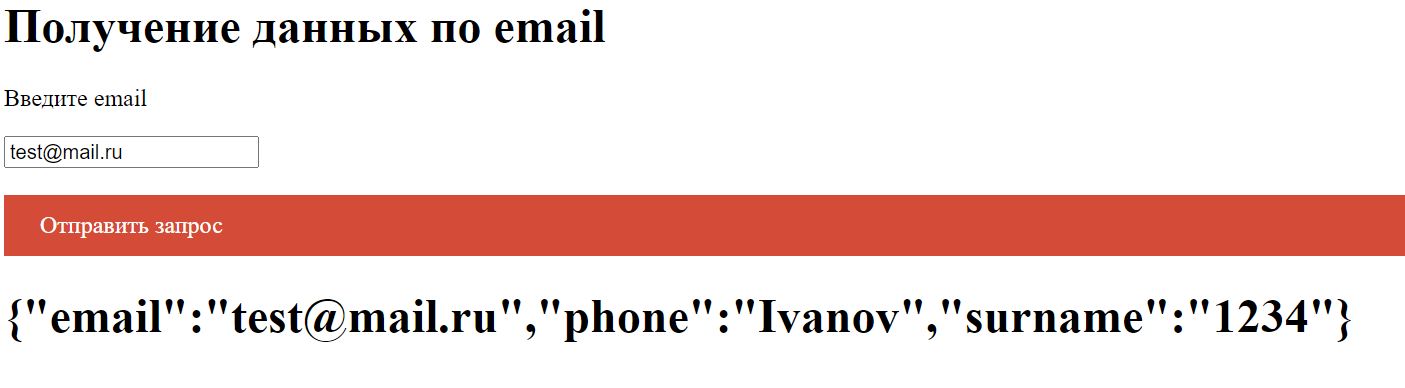
При нажатии на ОК появится следующее окно



Проверим возможность получения



При нажатии на красную кнопку получим следующий ответ без перезагрузки страницы



Листинг программы

"use strict"

const express = require("express")

const fs = require("fs")

const app = express();

const port = 5015;

app.listen(port);

const statpath = \_\_dirname + "/static";

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

console.log("server running");

let arr = JSON.parse(fs.readFileSync("./static/data.txt"));

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();

});

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

const nameString = request.query.p;

if (fs.existsSync(nameString)) {

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

response.end(contentString);

} else {

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

response.end(contentString);

}

});

function loadBody(request, callback) {

let body = [];

request.on('data', (chunk) => {

body.push(chunk);

}).on('end', () => {

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

callback(body);

});

}

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

loadBody(request, function(body) {

const obj = JSON.parse(body);

const email = obj["a"];

const phone = obj["b"];

const surname = obj["c"];

let f = true;

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

if (email === arr[i]["email"] || phone === arr[i]["phone"])

f = false;

if (f) {

arr.push({email: email, phone: phone, surname: surname});

console.log(arr);

fs.writeFileSync("./static/data.txt", JSON.stringify(arr));

response.end(JSON.stringify({ result: "Запись добавлена успешно!" }));

}

else

response.end(JSON.stringify({ result: "Повторная запись!" }));

});

});

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

let email = request.query.email;

let obj = null;

for (let i = 0; i < arr.length && !obj; i++)

if (arr[i]["email"] === email)

obj = arr[i];

response.end(JSON.stringify({result: JSON.stringify(obj)}));

});

# Задание 6.1

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 программы

"use strict"

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();

});

let games = [

{ name: "S.T.A.L.K.E.R.: Shadow of Chernobyl",

description: "BEST FPS shooter in the world",

age: 18 },

{ name: "Half-life 2",

description: "Quest in alternative reality",

age: 16 },

{ name: "Dota 2",

description: "Multiplayer Online Battle Arena",

age: 14 },

{ name: "Counter-Strike: Global Offensive",

description: "Cybersport team shooter",

age: 14 },

{ name: "Sims 4",

description: "Cool life simulator",

age: 12 },

{ name: "102 dalmatians",

description: "Interesting cartoon game",

age: 0 }

]

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

let lim = parseInt(request.query.lim);

let res = []

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

if (games[i]["age"] >= lim)

res.push(games[i]);

const infoObject = {

title: "Список игр с ограничением "+ lim + "+",

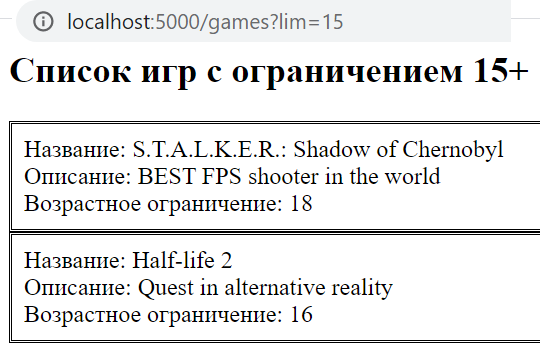
res: res

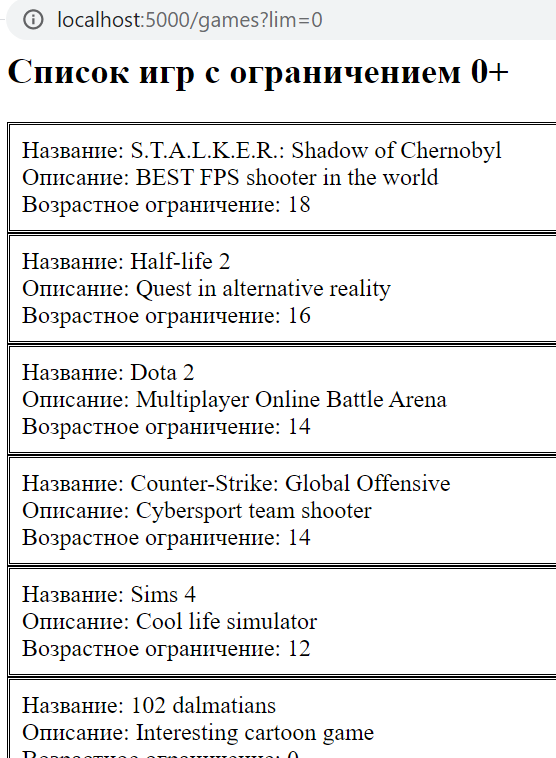
};

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

});

Примеры работы





# Задание 6.2

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 программы

"use strict"

const express = require("express");

const fs = require("fs");

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

const app = express();

const port = 5001;

app.listen(port);

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

app.use(cookieSession({

name: 'session',

keys: ['hhh', 'qqq', 'vvv'],

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

}));

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();

});

let users = [

{login: 'biba',

password: 'boba',

hobby: 'cybersport',

age:18},

{login: 'anikin',

password: '132',

hobby: 'prog',

age:20},

{login: 'putin',

password: 'vv',

hobby: 'president',

age:68},

{login: 'potter',

password: 'avada',

hobby: 'wizarding',

age:38}]

function sendGet(url, body, callback) {

const headers = {};

headers["Cache-Control"] = "no-cache, no-store, must-revalidate";

headers["Connection"] = "close";

request.get({

url: url,

body: body,

headers: headers,

}, function (error, response, body) {

if(error) {

callback(null);

} else {

callback(body);

}

});

}

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

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

response.end(contentString);

});

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

if (!request.session.login || !request.session.password)

if (!request.query.login || !request.query.password)

return response.end("At least one of fields is empty");

const login = request.query.login;

const password = request.query.password;

const c\_login = request.session.login;

const c\_password = request.session.password;

let f = false;

let ind = 0;

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

if (users[i]['login'] == login && users[i]['password'] == password ||

users[i]['login'] == c\_login && users[i]['password'] == c\_password) {

f = true;

ind = i;

}

if (f) {

if (request.query.login && request.query.password) {

request.session.login = login;

request.session.password = password;

}

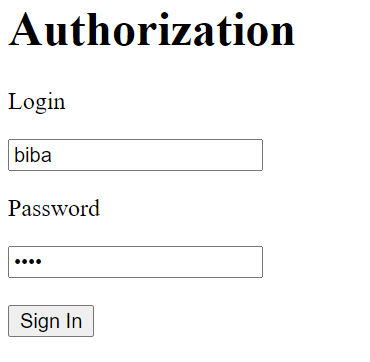
response.end("Authorization success\n" + JSON.stringify(users[ind]));

} else

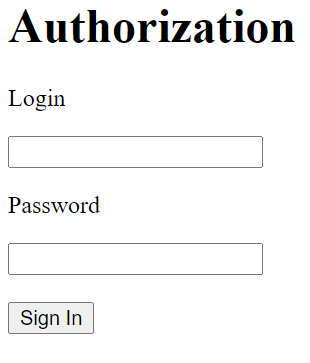
response.end("Authorization failed!")

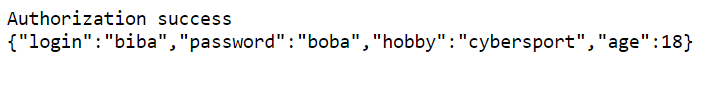
});

Примеры работы

При авторизации получаем результат .

Т. к. cookie сохранены можно повторно авторизоваться без ввода логина и пароля.





Вывод: по итогам лабораторной работы я научится работать с cookie и ajax, а также научился использовать шаблонизатор.