|  |  |
| --- | --- |
|  | **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**  Чалый А. А.  **Группа** ИУ7 – 52 Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Попов А.Ю. |  |

Москва.

2020 г.

# Оглавление

[Оглавление 1](#_Toc12)

[Цель 3](#_Toc13)

[Task 5 4](#_Toc14)

[Задание 1 4](#_Toc15)

[Задание 2 4](#_Toc16)

[Задание 3 4](#_Toc17)

[Задание 4 4](#_Toc18)

[Task 6 5](#_Toc19)

[Задание 1 5](#_Toc20)

[Задание 2 5](#_Toc21)

[Вывод 6](#_Toc22)

# Цель

По окончании данной работы будут приобретены такие умения как:

1. получение статических файлов;
2. AJAX запросы POST и GET;
3. работа со спецсимволами;
4. работа с шаблонизатором;
5. работа с сессиями в NodeJS.

# Task 5

## Задание 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.

**"use strict"**;

*// импортируем необходимые библиотеки*

**const** express = require(**"express"**);

**const *fs*** = require(**"fs"**);

*// 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));

*// 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();

});

*// body*

**function** loadBody(request, callback) {

**let** body = [];

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

body.push(chunk);

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

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

callback(body);

});

}

**let** masobj = [];

*// it is post*

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

loadBody(request, **function**(body) {

**const** obj = ***JSON***.parse(body);

**const** mail = obj[**"mail"**];

**const** surname = obj[**"surname"**];

**const** phone = obj[**"phone"**];

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

**if** (masobj[i][**"mail"**] === mail || masobj[i][**"phone"**] === phone) {

response.end(***JSON***.stringify({

**result**: **"Save content ne ok"**,

**obj**: **null**

}));

**return**;

}

}

masobj.push(obj);

**const** contentString = **`Mail:** ${mail} **Surname:** ${surname} **Phone:** ${phone}**\n`**;

***fs***.*appendFileSync*(**"file.txt"**, contentString);

response.end(***JSON***.stringify({

**result**: **"Save content ok"**,

**obj**: contentString

}));

});

});

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

**const** mail = request.**query**.mail;

**const** person = **null**;

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

**if** (masobj[i][**"mail"**] === mail) {

**const** contentString = **`Mail:** ${masobj[i][**"mail"**]} **Surname:** ${masobj[i][**"surname"**]} **Phone:** ${masobj[i][**"phone"**]}**\n`**;

response.end(***JSON***.stringify({

**result**: contentString

}));

**return**;

}

}

response.end(***JSON***.stringify({

**result**: **null**

}));

});

Листинг 1. Файл task5\_1.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(**"field-first"**).**value**;

**const** surname = ***document***.getElementById(**"field-second"**).**value**;

**const** phone = ***document***.getElementById(**"field-third"**).**value**;

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

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

mail, surname, phone

}), **function**(answerString) {

**const** answerObject = ***JSON***.parse(answerString);

**const** result = answerObject.**result**;

**const** person = answerObject.**obj**;

**if** (person === **null**)

label.**innerHTML** = **`Такой пользователь уже существует`**;

**else**

label.**innerHTML** = **`Был добавлен пользователь:** ${person}**`**;

*alert*(result);

});

}

Листинг 2. Файл code5 \_1.js

<!DOCTYPE **html**>

<**html**>

<**head**>

<**meta charset="UTF-8"**>

<**title**>Task5\_1</**title**>

</**head**>

<**body**>

<**link rel="stylesheet" type="text/css" href="task5\_1.css"**>

<**h1**>Заполните поля</**h1**>

<**br**>

<**p**>Почта</**p**>

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

<**br**>

<**p**>Фамилия</**p**>

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

<**br**>

<**p**>Телефон</**p**>

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

<**br**>

<**button onclick="**makeAction()**"**>Отправить запрос</**button**>

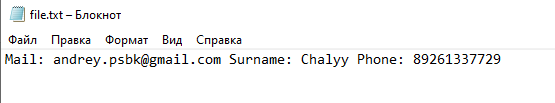
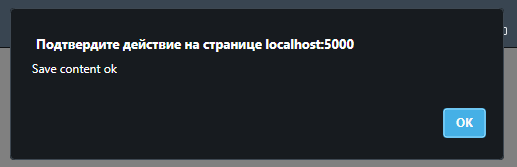
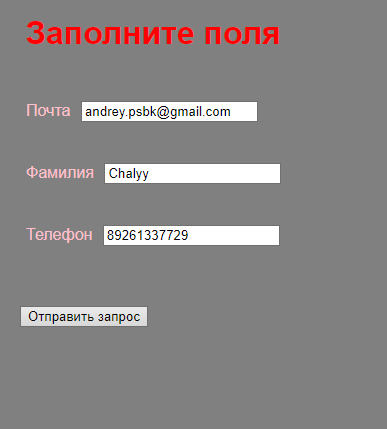
<**h1 id="result-label"**></**h1**>

<**script src="code5\_1.js"**></**script**>

</**body**>

</**html**>

Листинг 3. Файл task5 \_1.html



## Задание 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.

**"use strict"**;

**function** *makeAction*(){

*// input fields*

**const** mail = ***document***.getElementById(**"field-first"**).**value**;

**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**);

};

};

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

ajaxGet(url, **function**(stringAnswer) {

**const** objectAnswer = ***JSON***.parse(stringAnswer);

**const** result = objectAnswer.**result**;

**if** (result === **null**)

label.**innerHTML** = **`Пользователя с указанной почтой не существует`**;

**else**

label.**innerHTML** = **`Пользователь с данной почтой:** ${result}**`**;

});

};

Листинг 4. Файл code5 \_2.js

<!DOCTYPE **html**>

<**html**>

<**head**>

<**meta charset="UTF-8"**>

<**title**>Task5\_2</**title**>

</**head**>

<**body**>

<**link rel="stylesheet" type="text/css" href="task5\_2.css"**>

<**h1**>Найти пользователя по почте</**h1**>

<**br**>

<**p**>Почта</**p**>

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

<**br**>

<**button onclick="**makeAction()**"**>Отправить запрос</**button**>

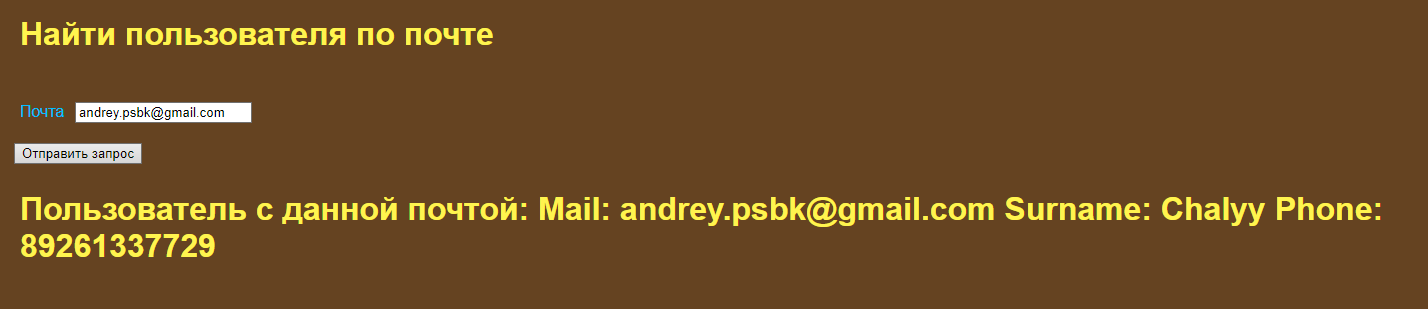
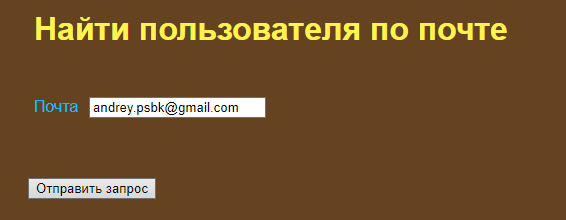
<**h1 id="result-label"**></**h1**>

<**script src="code5\_2.js"**></**script**>

</**body**>

</**html**>

Листинг 5. Файл task5 \_2.html



## Задание 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.

**body** {

**padding**: 30**px**;

**background**: **#808080**;

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

}

**h1** {

**padding**: 6**px**;

**color**: **#ff0000**;

**cursor**: **pointer**;

**display**: **inline-block**;

}

**p** {

**padding**: 6**px**;

**color**: **#ffc0cb**;

**cursor**: **pointer**;

**display**: **inline-block**;

}

Листинг 4. Файл task5\_1.css

**body** {

**padding**: 30**px**;

**background**: **#654321**;

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

}

**h1** {

**padding**: 6**px**;

**color**: **#fff44f**;

**cursor**: **pointer**;

**display**: **inline-block**;

}

**p** {

**padding**: 6**px**;

**color**: **rgb**(0, 191, 255);

**cursor**: **pointer**;

**display**: **inline-block**;

}

Листинг 5. Файл task5\_2.css

# Task 6

## Задание 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.

**"use strict"**;

*// 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 массивом игр*

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

**let** age = request.query.age;

**const** infoObject = {

descriptionValue: **"Список игр"**,

gamesArray: [

{name: **"Dota2"**, desc: **"MMO draka"**, age: 18},

{name: **"CSGO"**, desc: **"Strelyat draka"**, age: 12},

{name: **"AOW"**, desc: **"Strategia draka"**, age: 9},

{name: **"WOW"**, desc: **"Draka draka"**, age: 0}

]

};

**const** resObj = {

descriptionValue: **"Список игр"**,

gamesArray: []

};

**for** (**let** i = 0; i < infoObject.gamesArray.length; i++){

**let** curgame = infoObject.gamesArray[i];

**if** (curgame.age < age)

resObj.gamesArray.push(curgame);

}

response.render(**"pageGames.hbs"**, resObj);

});

Листинг 6. Файл task6\_1.js

<!DOCTYPE **html**>

<**html**>

<**head**>

<**meta charset="UTF-8"**>

<**title**>Ученики</**title**>

</**head**>

<**body**>

<**h2**>

{{**descriptionValue**}}

</**h2**>

{{#**each gamesArray**}}

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

Название: {{**this**.**name**}}

<**br**>

Описание: {{**this**.**desc**}}

<**br**>

Возрастное ограничение: {{**this**.**age**}}

</**div**>

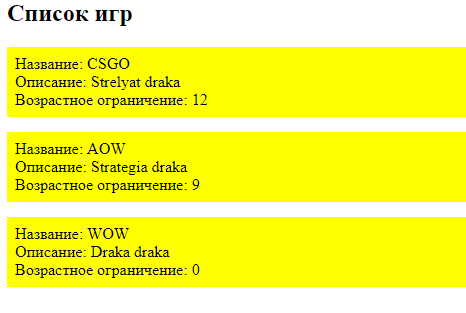
{{/**each**}}

</**body**>

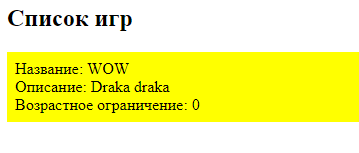
</**html**>

Листинг 7. Файл pageGames.hbs

http://localhost:5000/page/games?age=14



localhost:5000/page/games?age=8



## Задание 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 8. Файл task6\_2.js

**"use strict"**;

*// 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}**`**);

**const** way = \_\_dirname + **"/static"**;

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

*// 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();

});

**let** users = [

{**'login'**: **'sanya'**,

**'password'**: **'123'**,

**'hobby'**: **'eda'**,

**'age'**: 19},

{**'login'**: **'leha'**,

**'password'**: **'321'**,

**'hobby'**: **'chel'**,

**'age'**: 2},

{**'login'**: **'andrey'**,

**'password'**: **'555'**,

**'hobby'**: **'games'**,

**'age'**: 20},

];

*// сохранить cookie*

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

*// получаем параметры запроса*

**const** login = request.query.**login**;

**const** password = request.query.**password**;

**let** flagAuth = **false**;

**let** answerString = **"false"**;

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

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

flagAuth = **true**;

**break**;

}

}

**if** (flagAuth) {

request.**session**.**login** = login;

request.**session**.**password** = password;

answerString = **"true"**;

}

response.**end**(***JSON***.stringify({**"answer"** : answerString}));

});

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

*// получаем параметры запроса*

**const** login = request.**session**.**login**

**const** password = request.**session**.**password**

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

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

response.**end**(***JSON***.stringify(users[i]))

}

});

*// получить cookie*

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

*// контролируем существование cookie*

**if**(!request.**session**.**login**) **return** response.**end**(**"Not exists"**);

**if**(!request.**session**.**age**) **return** response.**end**(**"Not exists"**);

*// отправляем ответ с содержимым cookie*

**const** login = request.**session**.**login**;

**const** age = request.**session**.**age**;

response.**end**(***JSON***.stringify({

login,

age

}));

});

*// удалить все cookie*

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

request.**session** = **null**;

response.**end**(**"Delete cookie ok"**);

});

Листинг 9. Файл login.js

**"use strict"**;

**function** *ajaxGet*(urlString, callback) {

**let** r = **new *XMLHttpRequest***();

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

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

r.send(**null**);

r.**onload** = **function**() {

callback(r.**response**);

}

}

**function** *makeAction*() {

**const** l = ***document***.getElementById( **"login"** ).**value**;

**const** p = ***document***.getElementById( **"pass"** ).**value**;

***console***.log(l, p);

**const** url = **`request?login=**${l}**&password=**${p}**`**

*ajaxGet*(url, **function**(answerString) {

**const** answerObject = ***JSON***.parse(answerString);

**if** (answerObject.answer === **"true"**)

***window***.**location**.replace(**"http://localhost:5000/getInfo.html"**)

**else**

*alert*(**"Incorrect input"**)

});

}

Листинг 10. Файл getinfo.js

**"use strict"**;

**const *f*** = ***document***.getElementById(**"data"**)

**function** *ajaxGet*(urlString, callback) {

**let** r = **new *XMLHttpRequest***();

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

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

r.send(**null**);

r.**onload** = **function**() {

callback(r.**response**);

}

}

**function** *makeAction*() {

**const** label = ***document***.getElementById(**"data"**);

**const** url = **`/info`**

*ajaxGet*(url, **function**(answerString) {

**const** obj = ***JSON***.parse(answerString);

label.**innerHTML** = **`Логин:** ${obj.**login**} **Возраст:** ${obj.**age**} **Хобби:** ${obj.**hobby**}**`**

});

}

Листинг 11. Файл login.html

<!DOCTYPE **html**>

<**html**>

<**head**>

<**meta charset="UTF-8"**>

<**title**>Task6\_2</**title**>

</**head**>

<**body**>

<**p**>Логин</**p**>

<**input id="login" type="text" spellcheck="false" autocomplete="off"**>

<**p**>Пароль</**p**>

<**input id="pass" type="text" spellcheck="false" autocomplete="off"**>

<**br**>

<**br**>

<**button onclick="***makeAction*()**"**>Авторизоваться</**button**>

<**script src="login.js"**></**script**>

</**body**>

</**html**>

Листинг 12. Файл getinfo.html

<!DOCTYPE **html**>

<**html**>

<**head**>

<**meta charset="UTF-8"**>

<**title**>Task6\_2</**title**>

</**head**>

<**body**>

<**br**>

<**br**>

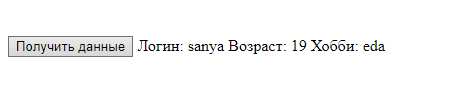
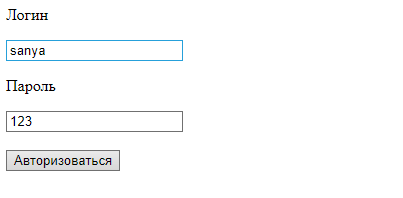
<**button onclick="***makeAction*()**"**>Получить данные</**button**>

<**label id="data"**></**label**>

<**script src="getinfo.js"**></**script**>

</**body**>

</**html**>



# Вывод

Все поставленные задачи были выполнены. Цель лабораторной работы достигнута.