|  |  |
| --- | --- |
| 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 Куликов Дмитрий Алексеевич

*фамилия, имя, отчество*

Группа ИУ7-52Б

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Попов А.Ю.

*подпись, дата фамилия, и.о.*

*2020 г.*

**Цель работы:**

Приобретение и закрепление на практике следующих навыков:

1. Получение статических файлов

2. AJAX запросы GET

3. POST запросы в AJAX

4. Работа с шаблонизатором

5. Сессии 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.

В условие задачи не сказано, в каком формате хранятся данные в файле,

поэтому я храню их в следующем виде:



**Листинг:**

**index.js**

"use strict";

// импортируем библиотеку

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);

    });

}

function checkUniqe(obj, mail, phone) {

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

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

            return false;

        }

    }

    return true;

}

// 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"];

        let contentString = fs.readFileSync("./file.txt", "utf8");

        let answerString = "Не добавили человека";

        if (mail === '' || surname === '' || phone === '') {

            answerString = "Заполните поля";

        } else {

            let obj;

            if (contentString === '') {

                obj = [];

            } else {

                obj = JSON.parse(contentString);

            }

            if (checkUniqe(obj, mail, phone)) {

                obj.push({"mail": mail, "surname": surname, "phone":phone})

                contentString = JSON.stringify(obj);

                fs.writeFileSync("./file.txt", contentString);

                answerString = "Добавили человека"

            }

        }

        response.end(JSON.stringify({

            answer: answerString,

            curFile: contentString

        }));

    });

});

**static/page.html**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Моя страница</title>

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

</head>

<body>

    <p>Почта</p>

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

    <p>Фамилия</p>

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

    <p>Номер</p>

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

    <br>

    <br>

    <div id="btn-send" class="btn-class" onclick="makeAction()">Отправить</div>

    <br>

    <br>

    <h1 id="result-label"></h1>

    <script src="/code.js"></script>

</body>

</html>

**static/code.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() {

    // input fields

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

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

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

    // label

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

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

        mail, surname, phone

    }), function(answerString) {

        const answerObject = JSON.parse(answerString);

        const answer = answerObject.answer;

        alert(answer);

        const curFile = answerObject.curFile;

        label.innerHTML = `Текущий файл: ${curFile}`;

    });

}

**static/style.css**

body {

    padding: 30px;

    background: burlywood;

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

}

.btn-class {

    padding: 6px;

    background: blueviolet;

    color: white;

    cursor: pointer;

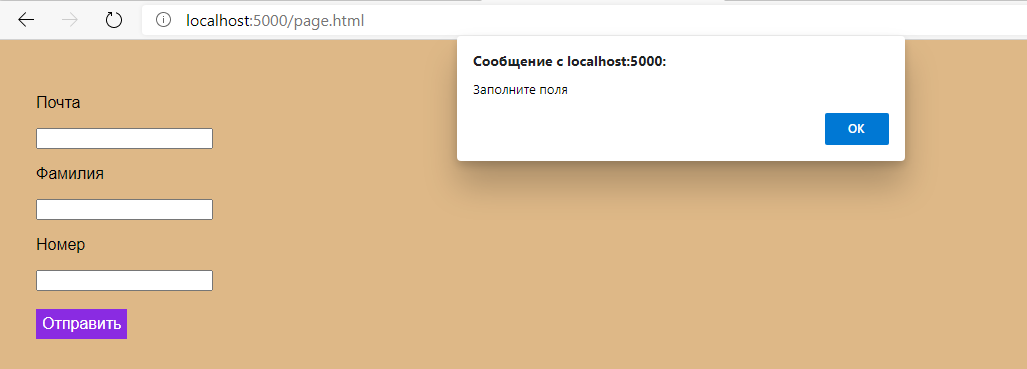
    display: inline-block;

}

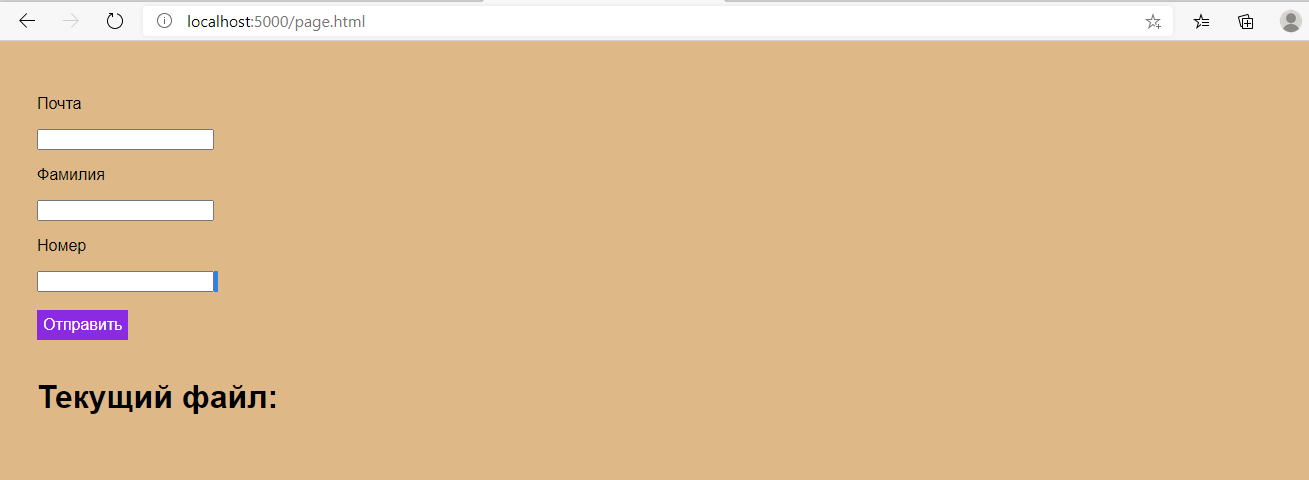
**Тестирование:**

Изначально файл, куда записываем данные - пуст.

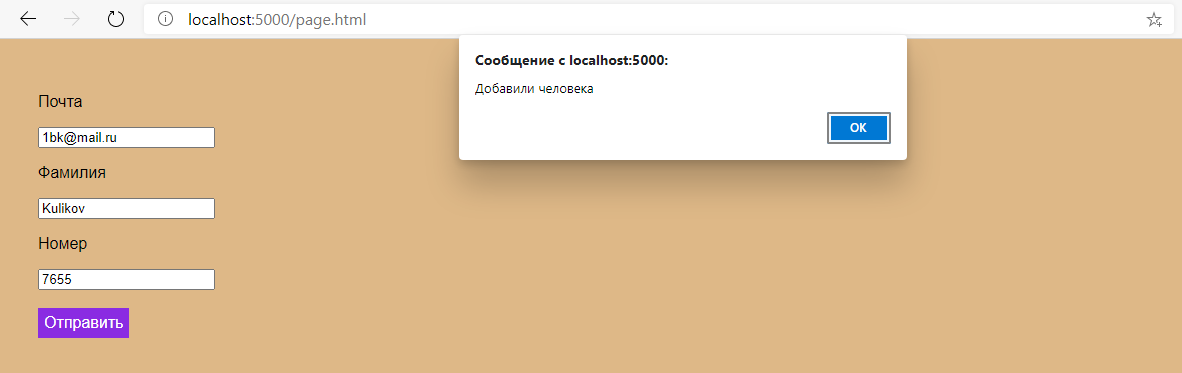
Отправим пустые поля



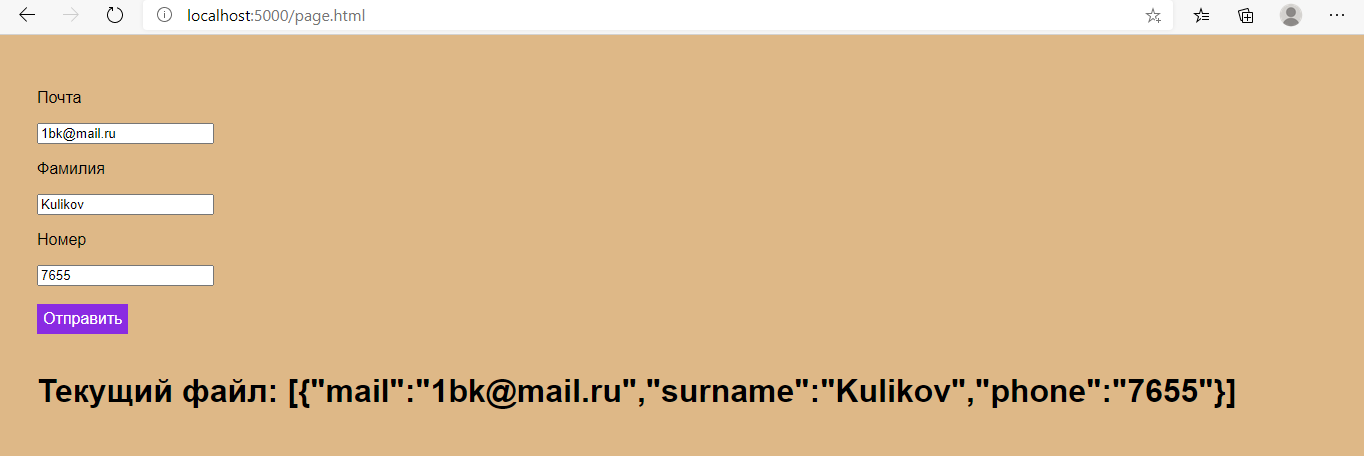
Файл как и прежде пуст.



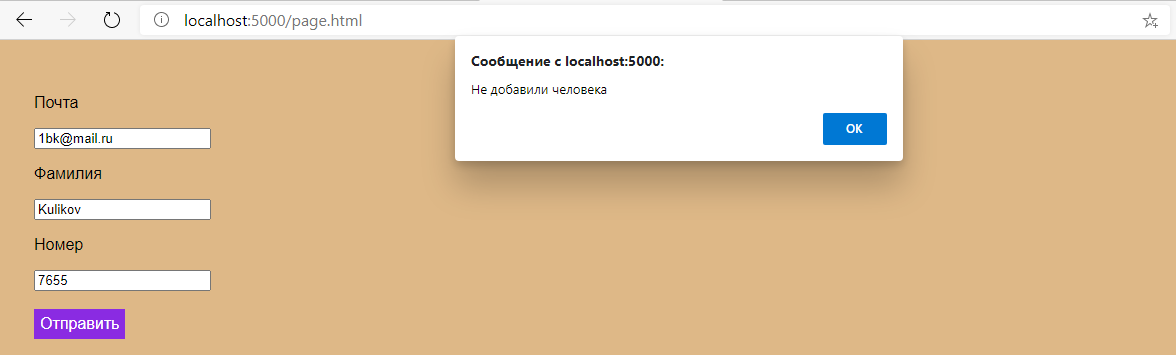
Добавим человека



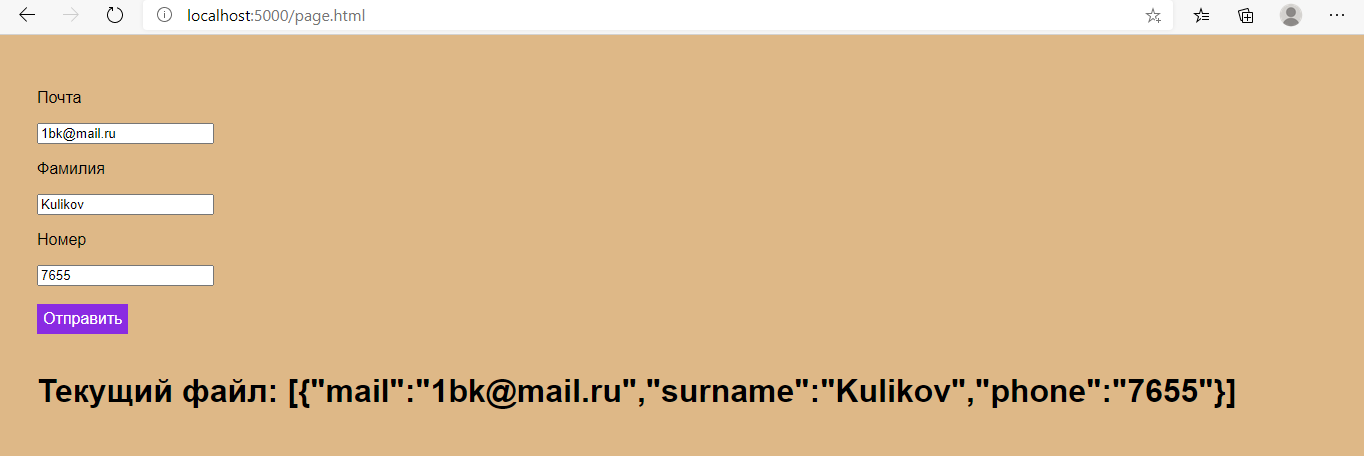
Содержимое файла изменилось.



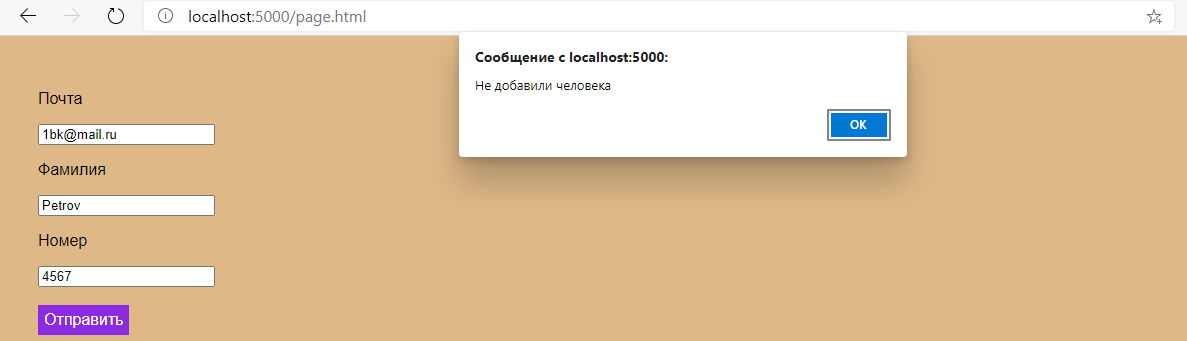
Добавим этого же человека еще раз.



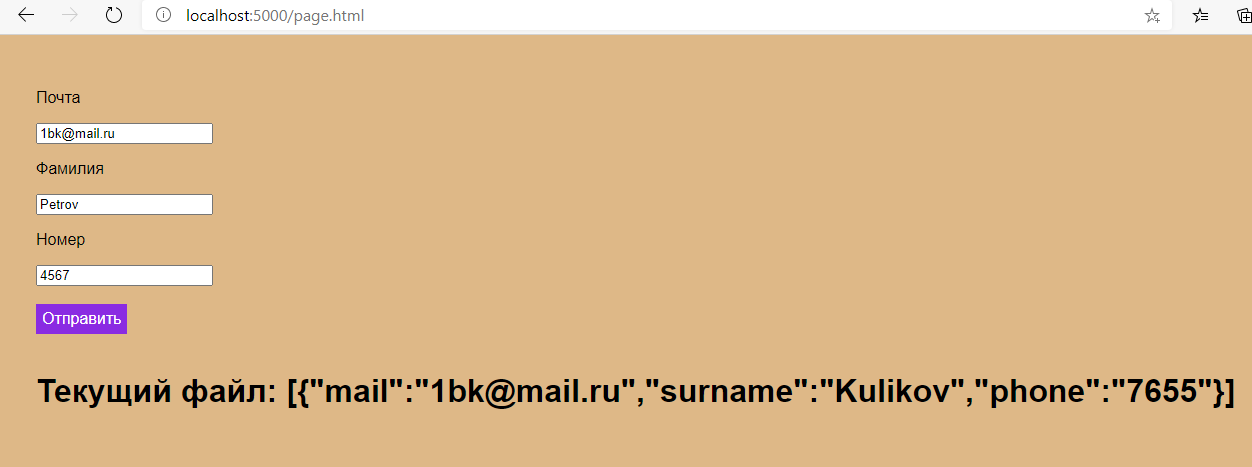
Содержимое файла не изменилось.



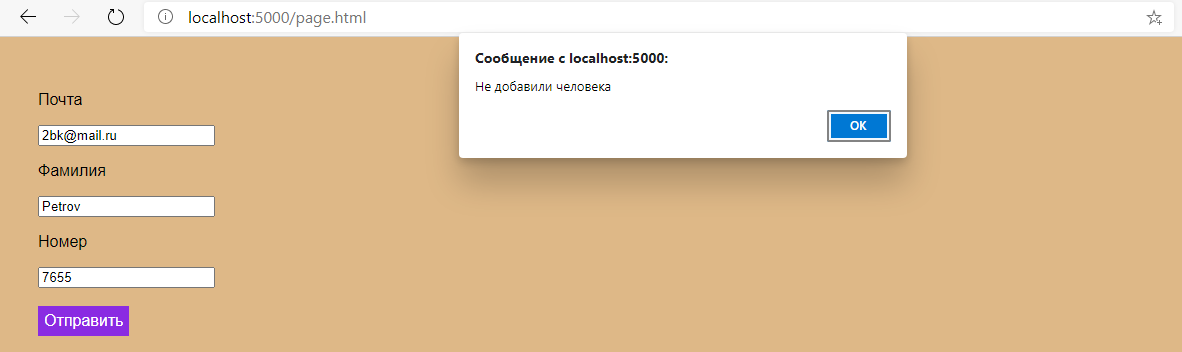
Добавим человека c той же почтой, но с другим именем и номером.



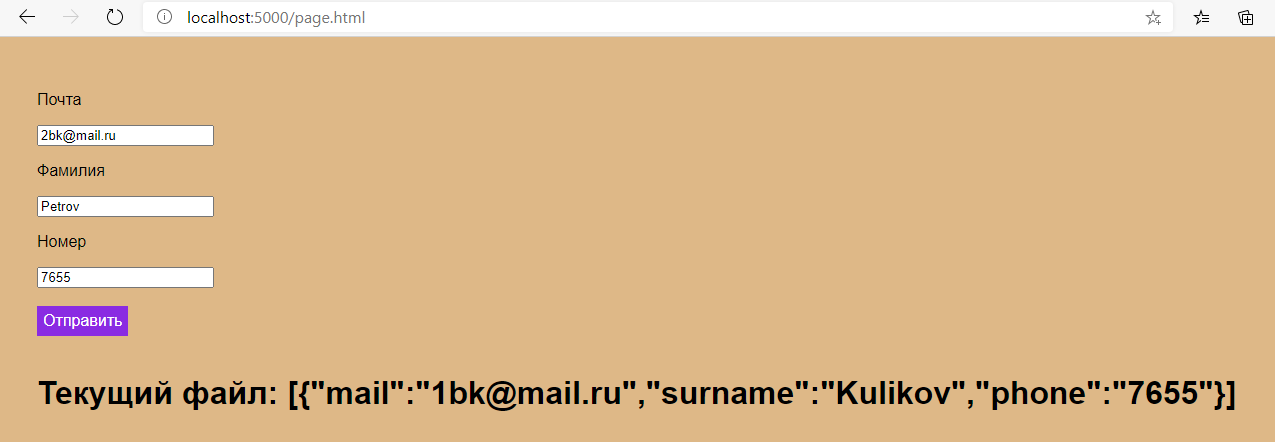
Содержимое файла не изменилось.



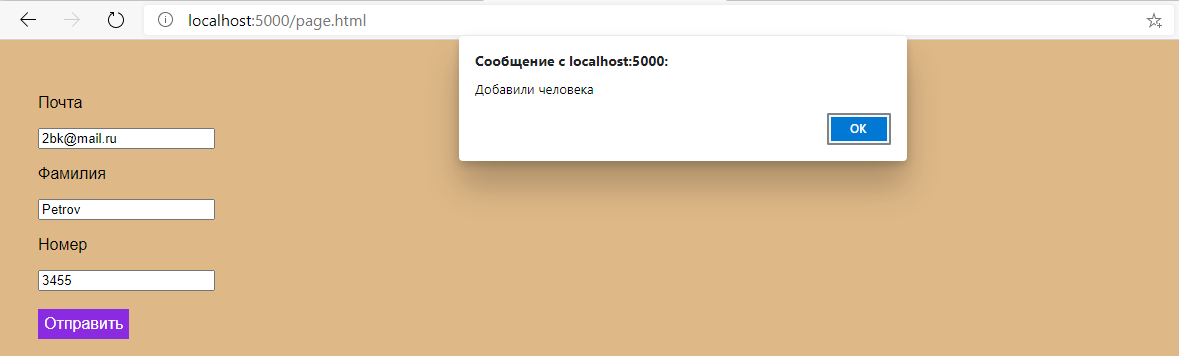
Добавим человека c тем же номером, но с другим именем и почтой.



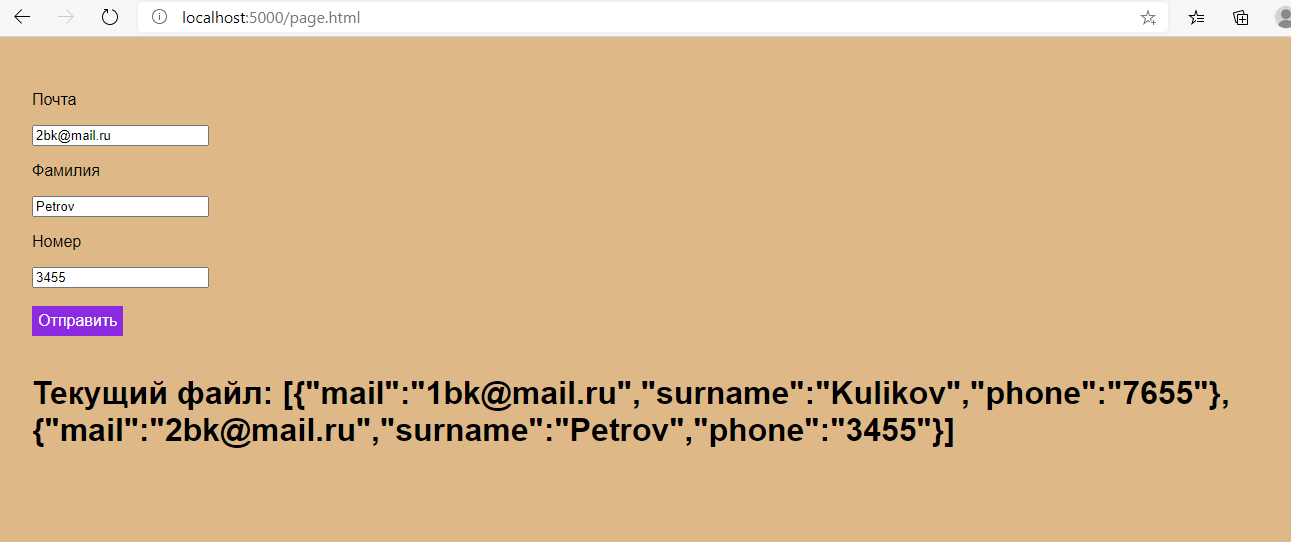
Содержимое файла не изменилось.



Добавим нового человека.



Содержимое файла изменилось.



# 2. 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.

**Листинг:**

В index.js добавим строчки кода

// выдать страницу

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

    response.sendFile(\_\_dirname + "/" + "page2.html");

});

function findObj(key, obj) {

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

        if (obj[i].mail === key) {

            return obj[i];

        }

    }

    return null;

}

// выдать запись

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

    const key = request.query.k;

    let value;

    let contentString = fs.readFileSync("./file.txt", "utf8");

    if (key === '') {

        value = 'Вы не заполняли поле почты'

    } else {

        if (contentString === '') {

            value = 'Нет людей в файле'

        } else {

            const obj = JSON.parse(contentString);

            let fobj = findObj(key, obj);

            if (fobj != null) {

                value = `mail: ${fobj.mail}, surname: ${fobj.surname}, phone: ${fobj.phone}`

            } else {

                value = "Нет такого человека"

            }

        }

    }

    response.end(JSON.stringify({

        v: value

    }));

});

static/page2.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Страница</title>

</head>

<body>

<p>Почта</p>

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

<br>

<br>

<button onclick="getRecord()">Отправить запрос</button>

<h1 id="result-label"></h1>

<script src="/code2.js"></script>

</body>

</html>

static/code2.js

"use strict";

// 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);

    };

};

    // ввод ключа и отправка запроса на сервер

function getRecord() {

    // input fields

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

    const mail\_a = encodeURIComponent(mail);

    const url = `/record?k=${mail\_a}`;

    // label

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

    // отправка запроса на сервер

    ajaxGet(url, function(answerString) {

        const answerObject = JSON.parse(answerString);

        const result = answerObject.v;

        label.innerHTML = `Ответ: ${result}`;

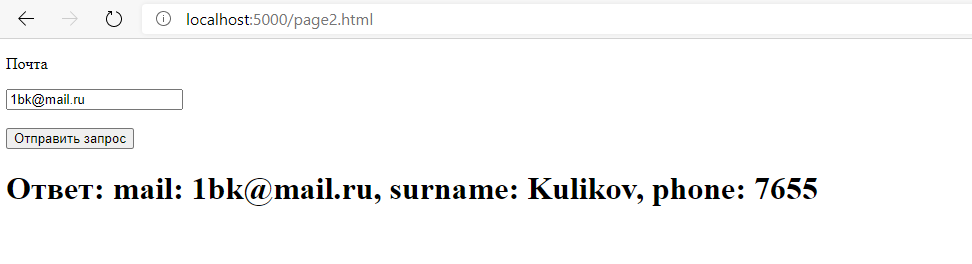
    });

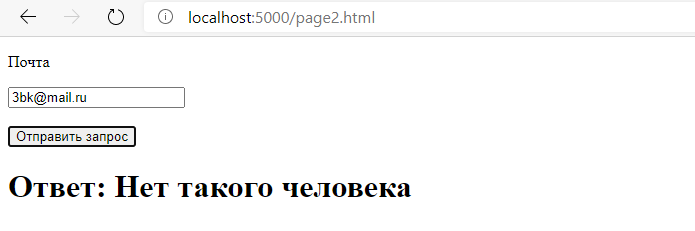
}

**Тестирование:**

В файл хранится следующая информация

****





# 3. 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:**

style1.css

body {

    padding: 30px;

    background: tomato;

}

p {

    font-family: Courier New;

    font-weight: bold;

}

h1 {

    font-family: Arial;

    font-weight: bold;

    color: white;

}

.btn-class {

    padding: 6px;

    background: orange;

    color: white;

    cursor: default;

    display: inline-block;

}

style2.css

body {

    margin-left: 50%;

    background: pink;

}

p {

    font-family: Cambria;

    font-weight: lighter;

    color: gray

}

h1 {

    font-family: Cochin;

    font-weight: bold;

    color:firebrick;

}

.btn-class {

    background: chartreuse;

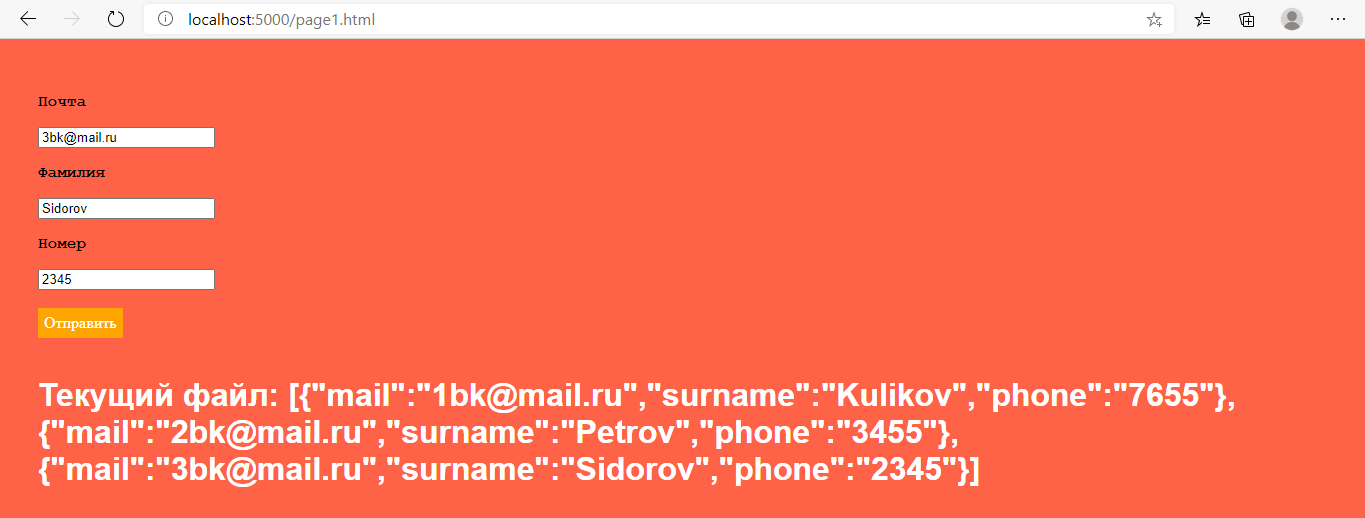
    color: blueviolet;

    cursor: pointer;

}

**Тестирование:**

page1.html



page2.html



# 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:**

index.js

"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();

});

let gameArray = [

    {name: "DOTA2", description: "multiplayer online battle arena", age: 6},

    {name: "GTA5", description: "action-adventure", age: 18},

    {name: "CS-GO", description: "first-person shooter", age: 16}

];

// выдача страницы c играми

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

    const key = request.query.k;

    console.log(key);

    const infoObject = {

        descriptionValue: `Список компьютерных игр, с возрастным ограничением ${key}`,

        games : gameArray.filter(obj => obj.age <= key)

    };

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

});

pageGames.hbs

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Компьютерные игры</title>

</head>

<body>

<h2>

    {{descriptionValue}}

</h2>

{{#each games}}

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

        Название: {{this.name}}

        <br>

        Описание: {{this.description}}

        <br>

        Возраст: {{this.age}}

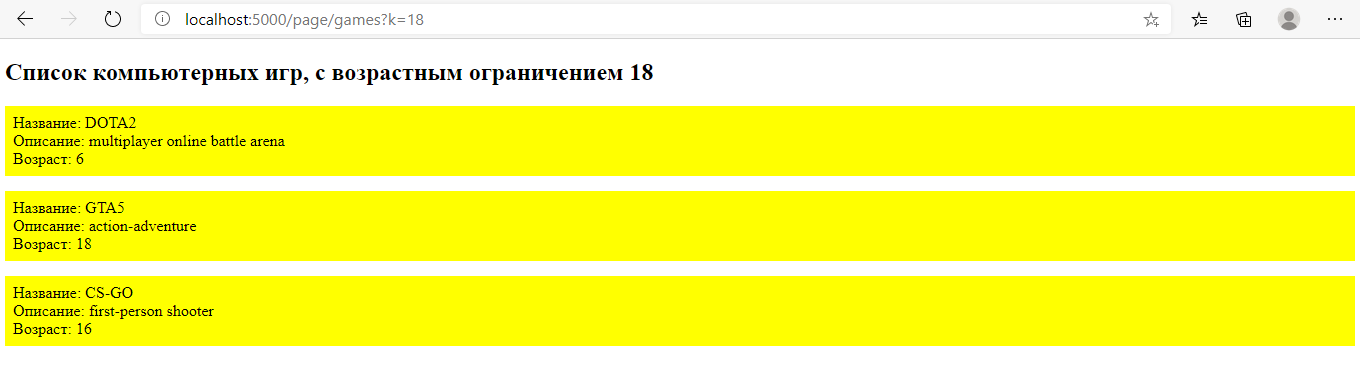
    </div>

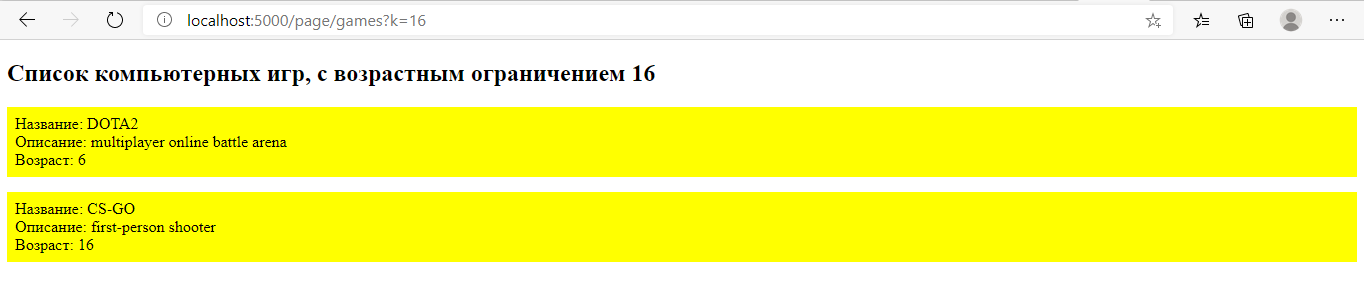
{{/each}}

</body>

</html>

**Тестирование:**





# 2. 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:**

index.js

"use strict";

// импортируем библиотеки

const express = require("express");

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

const fs = require("fs");

// 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));

const users = [

    {"login" : "Dimakrut", "password" : "2280", "age" : 9, "hobby" : "tennis"},

    {"login" : "Vamar", "password" : "1488", "age" : 12, "hobby" : "football"},

    {"login" : "Zeus", "password" : "3220", "age" : 18, "hobby" : "basketball"},

    {"login" : "Morg", "password" : "1234", "age" : 26, "hobby" : "hockey"}

]

// 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();

});

// 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");

});

auth.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Авторизация</title>

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

</head>

<body>

    <p>Логин</p>

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

    <p>Пароль</p>

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

    <br>

    <br>

    <div id="btn-send" class="btn-class" onclick="makeAction()">Войти</div>

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

</body>

</html>

codeAuth.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 login = document.getElementById("field-login");

    const password = document.getElementById("field-password");

    const l = login.value;

    const p = password.value;

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

    ajaxGet(url, function(answerString) {

        const answerObject = JSON.parse(answerString);

        alert(answerObject.answer);

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

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

        else

            alert("Not auth user");

    });

}

profile.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Профиль</title>

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

</head>

<body>

    <div id="btn-send" class="btn-class" onclick="makeAction()">Получить данные</div>

    <br>

    <br>

    <h1 id="result-label"></h1>

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

</body>

</html>

codeProfile.js

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("result-label");

    const url = `/info`

    ajaxGet(url, function(answerString) {

        const obj = JSON.parse(answerString);

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

    });

}

style.css

body {

    padding: 30px;

    background: burlywood;

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

}

.btn-class {

    padding: 6px;

    background: blueviolet;

    color: white;

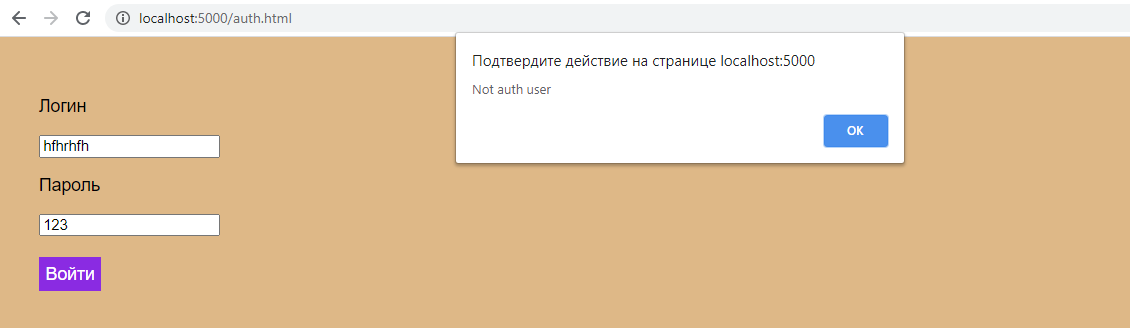
    cursor: pointer;

    display: inline-block;

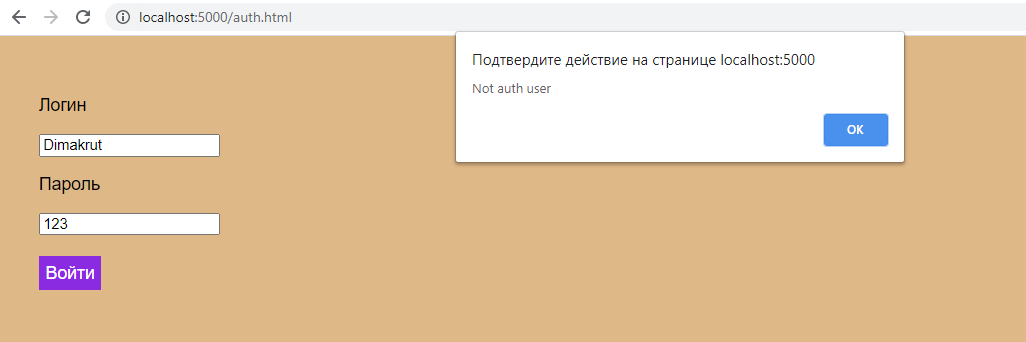
}

**Тестирование:**

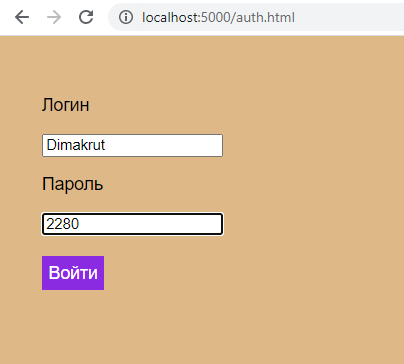
Нет такого пользователя

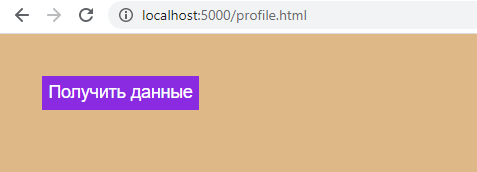


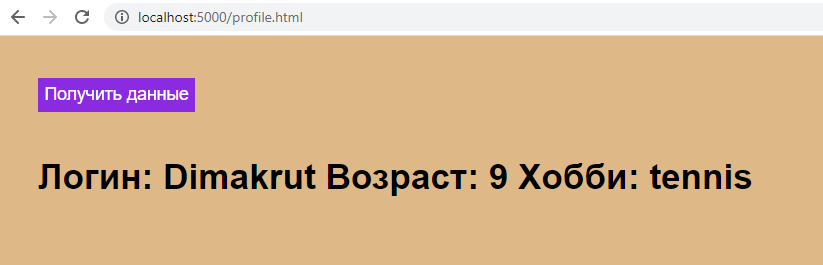
Неверный пароль



Авторизованный пользователь







**Вывод:**

Все поставленные задачи были выполнены. Цель лабораторной работы достигнута.