|  |  |
| --- | --- |
| **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 дисциплине «Архитектура ЭВМ»**

|  |  |
| --- | --- |
| **Студент: Садулаева Т. Р.**  **Группа:** ИУ7-54Б  **Преподаватели:** Попов А.Ю., Колотовкин М.И., Куклина Н.И.  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

### **Цель работы**

### Изучение библиотеки express и шаблонизатора hbs.

### **Подзадача 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.

**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.

**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.

**Реализация**

**index.js**

"use strict";

const express = require("express");

const fs = require("fs");

const app = express();

const port = 5000;

app.listen(port);

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

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();

});

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

const a = request.query.a;

const b = request.query.b;

const s = parseInt(a) + parseInt(b);

response.end(JSON.stringify({

result: s

}));

});

function getPersonByEmail(email, arr) {

let reqIndex = -1;

for (let i = 0; i < arr.length && reqIndex === -1; i++) {

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

reqIndex = i;

}

}

if (reqIndex === -1) {

return null

}

return arr[reqIndex];

}

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

const email = request.query.email;

let res = "";

const fileName = "task1.txt";

if (!fs.existsSync(fileName)) {

res = "There is no such email.";

}

else {

const contentString = fs.readFileSync(fileName, "utf-8");

const arr = JSON.parse(contentString);

const reqObj = getPersonByEmail(email, arr);

if (!reqObj) {

res = "There is no such email.";

}

else {

res = JSON.stringify(reqObj);

}

}

response.end(JSON.stringify({

result: res

}));

})

function loadBody(request, callback) {

let body = [];

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

body.push(chunk);

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

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

callback(body);

});

}

function isUniqueNumberAndEmail(obj, arr) {

let res = true;

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

if (obj["email"] === arr[i]["email"] ||

obj["phoneNumber"] === arr[i]["phoneNumber"]) {

res = false;

}

}

return res

}

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

loadBody(request, function(body) {

const fileName = "task1.txt";

if (!fs.existsSync(fileName)) {

fs.writeFileSync(fileName, "[]");

}

const obj = JSON.parse(body);

const prevContentString = fs.readFileSync(fileName, "utf-8");

const arr = JSON.parse(prevContentString);

let resultMessage = "Success";

if (isUniqueNumberAndEmail(obj, arr)) {

arr.push(obj);

const contentString = JSON.stringify(arr, null, 2);

fs.writeFileSync("task1.txt", contentString);

}

else {

resultMessage = "Failure: Not Unique Phone Number or Email";

}

response.end(JSON.stringify({

result: resultMessage

}));

});

});

**task1.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>task5\_1</title>

<link rel="stylesheet" href="/style1.css">

</head>

<body>

<h1>task5\_1</h1>

<p>Email</p>

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

<p>Surname</p>

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

<p>Phone Number</p>

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

<br>

<br>

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

<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 email = document.getElementById("field-email").value;

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

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

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

email, surname, phoneNumber

}), function(answerString) {

const answerObject = JSON.parse(answerString);

const result = answerObject.result;

alert(result);

});

}

</script>

</body>

</html>

**style1.css**

body {

padding: 30px;

background: lightgray;

font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif

}

.btn-class {

padding: 6px;

background: darkgreen;

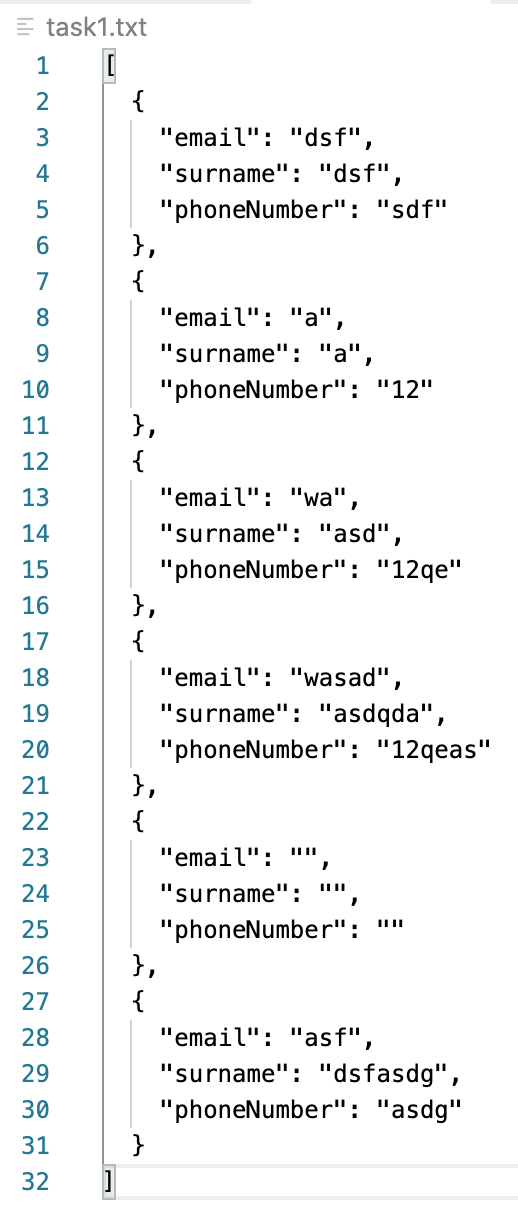
color: white;

cursor: pointer;

display: inline-block;

}

**task1.txt**

****

**task2.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>task5\_2</title>

<link rel="stylesheet" href="/style2.css">

</head>

<body>

<h1>task5\_2</h1>

<p>Email</p>

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

<br>

<br>

<div id="get-btn" class="btn-class">Отправить</div>

<br>

<br>

<p1 id="result-label"></p1>

<script src="/code2.js"></script>

</body>

</html>

**code2.js**

"use strict";

window.onload = function() {

const emailField = document.getElementById("field-email");

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

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

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);

};

};

btn.onclick = function() {

const email = emailField.value;

const url = `/person?email=${email}`;

ajaxGet(url, function(stringAnswer) {

const objectAnswer = JSON.parse(stringAnswer);

const result = objectAnswer.result;

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

});

};

};

**style2.css**

body {

padding: 30px;

background: rgb(40, 40, 40);

color: lightgray;

font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif

}

.btn-class {

padding: 6px;

background: purple;

color: white;

cursor: pointer;

display: inline-block;

}

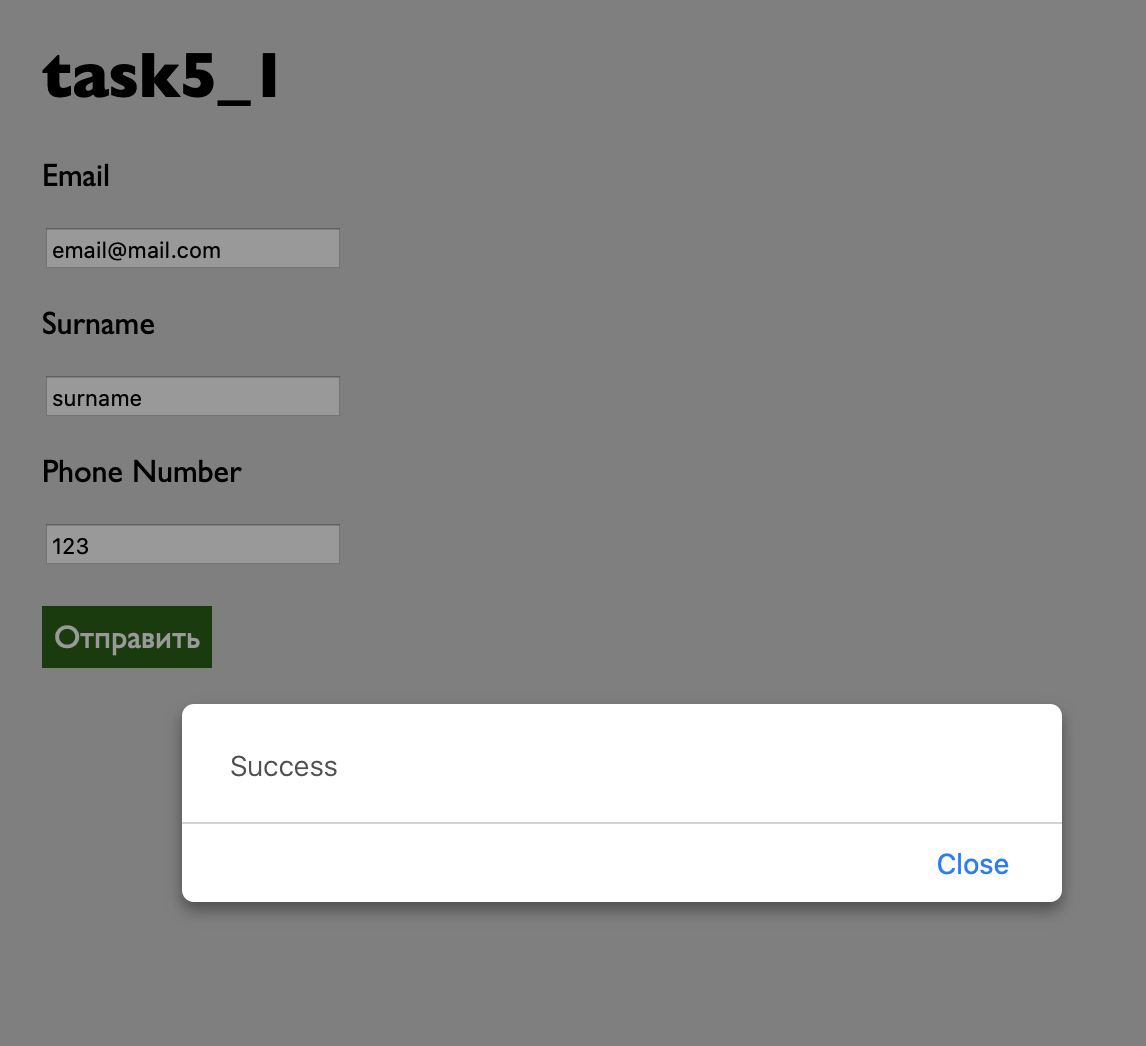
.input-class {

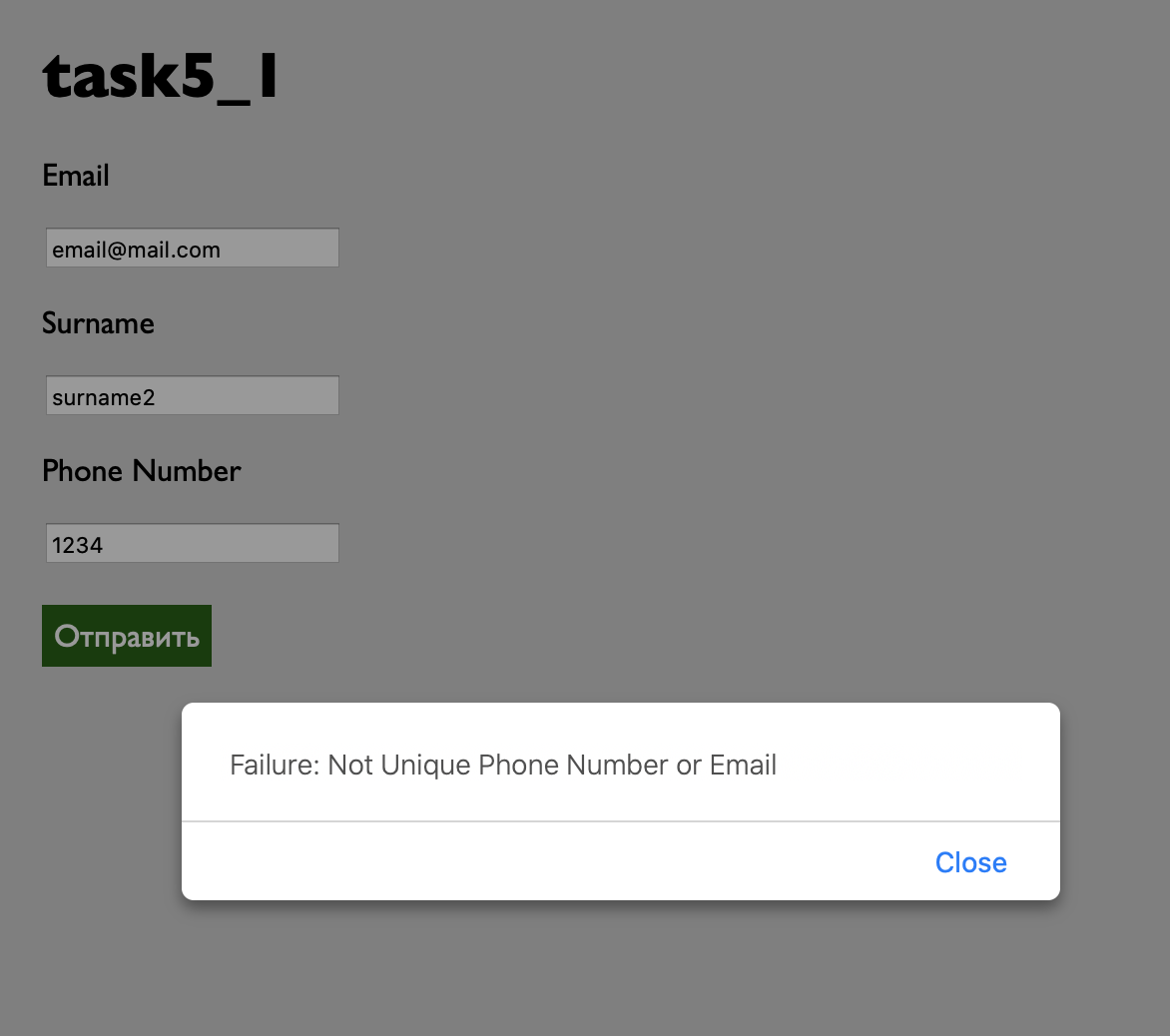
background-color: rgb(30, 30, 30); /\* Цвет фона \*/

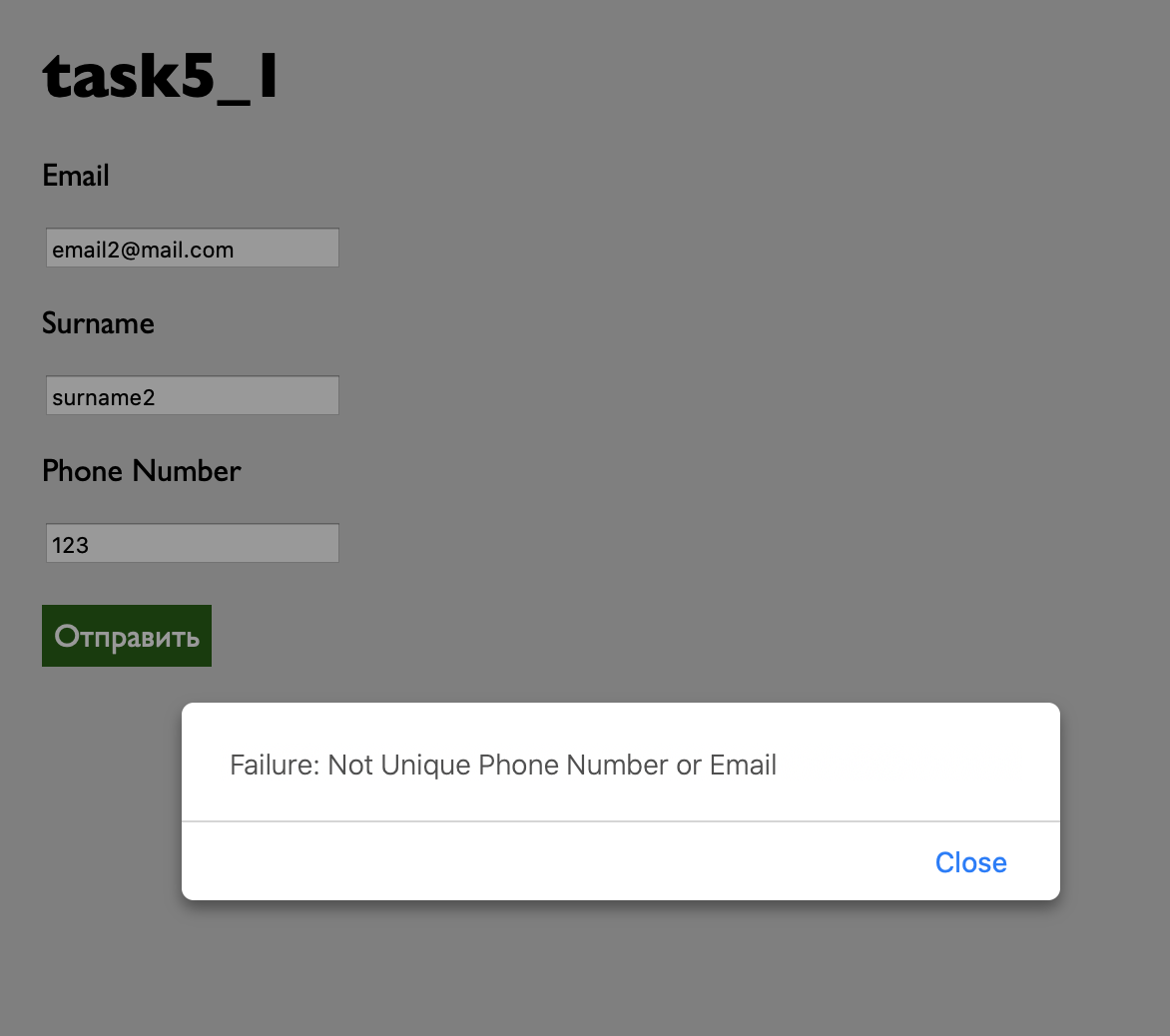
color: lightgray; /\* Цвет текста \*/

}

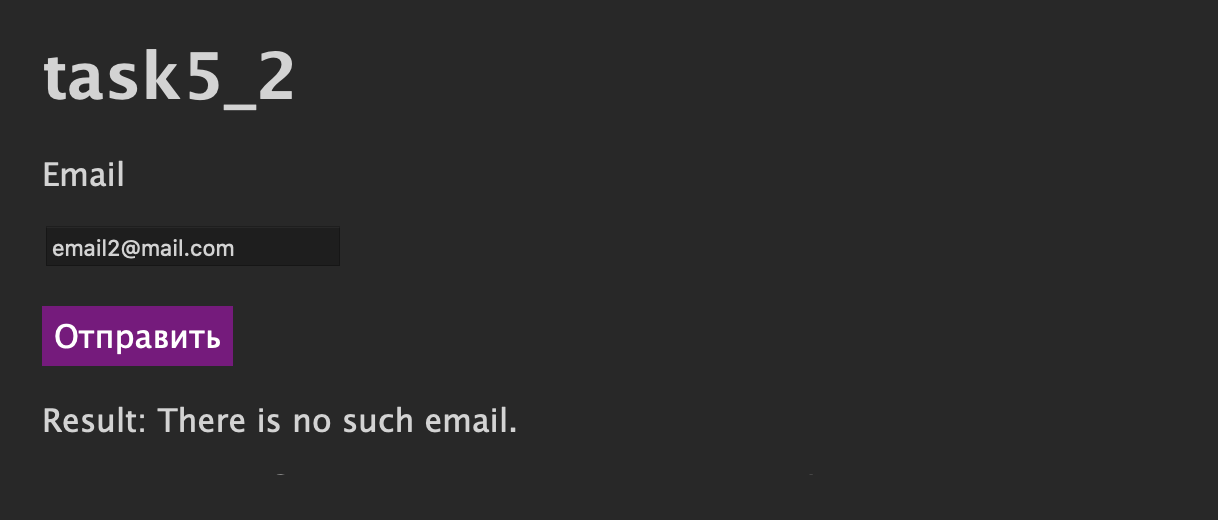
**Проверка работоспособности**

****









**Подзадача 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.

**Реализация**

**index.js**

"use strict";

const express = require("express");

const app = express();

const port = 5015;

app.listen(port);

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

app.set("view engine", "hbs");

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();

});

const gamesArray = [

{

name: "Blacksad: Under the Skin",

description: "Интерактивный квест по мотивам популярного комикса BlackSad, где главные герои — антропоморфные (похожие на людей) животные. История разворачивается в 50-х годах ХХ века и рассказывает о детективе-коте Джоне Блэксэде. Он вместе с дочерью расследует групповое убийство. Жертвы — владелец боксерского клуба и его помощник.",

requiredAge: 18

},

{

name: "Mechwarrior 5: Mercenaries",

description: "Шутер от первого лица, где геймер может почувствовать себя боевым роботом! События происходят в 3015 году — во времена третьей галактической Войны за наследие. Игроки будут выполнять роль оператора специальных роботов. Но это не бездушная «стрелялка»! По ходу сюжета придется выбирать контракты, зарабатывать деньги, чтобы открывать новые детали, да еще и успевать продвигаться по карьерной лестнице.",

requiredAge: 16

},

{

name: "Red Dead Redemption 2",

description: "Эта игра успела менее чем за год перейти в статус «культовой», потому что она смешала в себе культуру дикого запада и дерзость Rockstar.\nКак и в GTA V, здесь есть как одиночный сюжетный, так и многопользовательский онлайн-режим, что позволяет «выходить на охоту» за плохишами, убегать от федералов с головорезами и грабить богачей целыми бандами. Также авторы игры не забыли про графику: увидев степень проработки каждой мелочи, геймерам стало понятно, почему разработка затянулась на целых 5 лет.",

requiredAge: 18

},

{

name: "Need For Speed Heat",

description: "Продолжение гоночной саги: в этот раз — в жарком Майами. Геймеров ждут сумасшедшие гонки на дорогах, побеги от полиции и невероятные вариации для кастомизации авто. Полицейские стали злее — от них некуда спрятаться. Они будут следовать за игроком по пятам. Не поможет даже крушение рекламного щита и заносы на резких поворотах.",

requiredAge: 12

},

{

name: "Call Of Duty: Modern Warfare",

description: "Служба зовет! В 16-ой игре военной франшизы Call of Duty игроки смогут перенестись в Сирию, Ирак и даже на Донбасс. Чтобы избежать моральных проблем, разработчики абстрагировались от реальности, частично изменив события внутри игрового мира.",

requiredAge: 18

},

{

name: "Trine 4: The Nightmare Prince",

description: "Продолжение одного из самых популярных 2.5D-платформеров. Перед игроками окажется знакомая троица: рыцарь Понтий, воровка Зоя и волшебник Амадей. Они будут бороться против монстров, которых случайно выпустил на волю принц Селиус. Теперь троице нужно успеть победить врагов до того времени, как кошмары захватят Вселенную.",

requiredAge: 12

},

{

name: "Fifa 20",

description: "Fifa 20 — продолжение главной футбольной игры с обновленной системой искусственного интеллекта. Матч похож на реальный. В игрушке улучшили работу с мячом и без него. Продумали тщательную и реалистичную физику самого мяча. Также были улучшены высокие пасы, дриблинг, попытки отдать, перехватить мяч и многое другое. Бежать по полю стало так, что аж дух захватывает.",

requiredAge: 0

},

{

name: "Borderlands 3",

description: "Триквел самого безумного шутера последнего десятилетия рассказывает о четверке Искателей Убежища, противостоящих злодеям-близнецам. ",

requiredAge: 18

},

{

name: "The Blackout Club",

description: "Кооперативный хоррор про тайны странного городка, расположенного в секретной военной зоне. И компанию подростков, которая решила стоять на страже малой Родины. По сюжету, геймерам придется объединиться в команду до 4 человек, чтобы защитить город от странного коллективного лунатизма. ",

requiredAge: 16

}

]

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

let age = request.query.age;

if (!age) {

age = "0";

}

const ageInt = parseInt(age);

const resGamesArray = [];

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

if (ageInt >= gamesArray[i]["requiredAge"]) {

resGamesArray.push(gamesArray[i]);

}

}

const infoObject = {

title: "Список игр",

gamesArray: resGamesArray

};

response.render("task1.hbs", infoObject);

});

**task1.hbs**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Компьютерные игры</title>

</head>

<body style="font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif; background: rgb(40, 40, 40); color: darkgray">

<h2>

{{title}}

</h2>

{{#each gamesArray}}

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

Название: <div style="font-weight:bold"> {{this.name}} </div>

<br>

Описание: <br>{{this.description}}

<br>

<br>

Возрастное ограничение: <div style="font-weight:bold">{{this.requiredAge}}+ </div>

</div>

{{/each}}

</body>

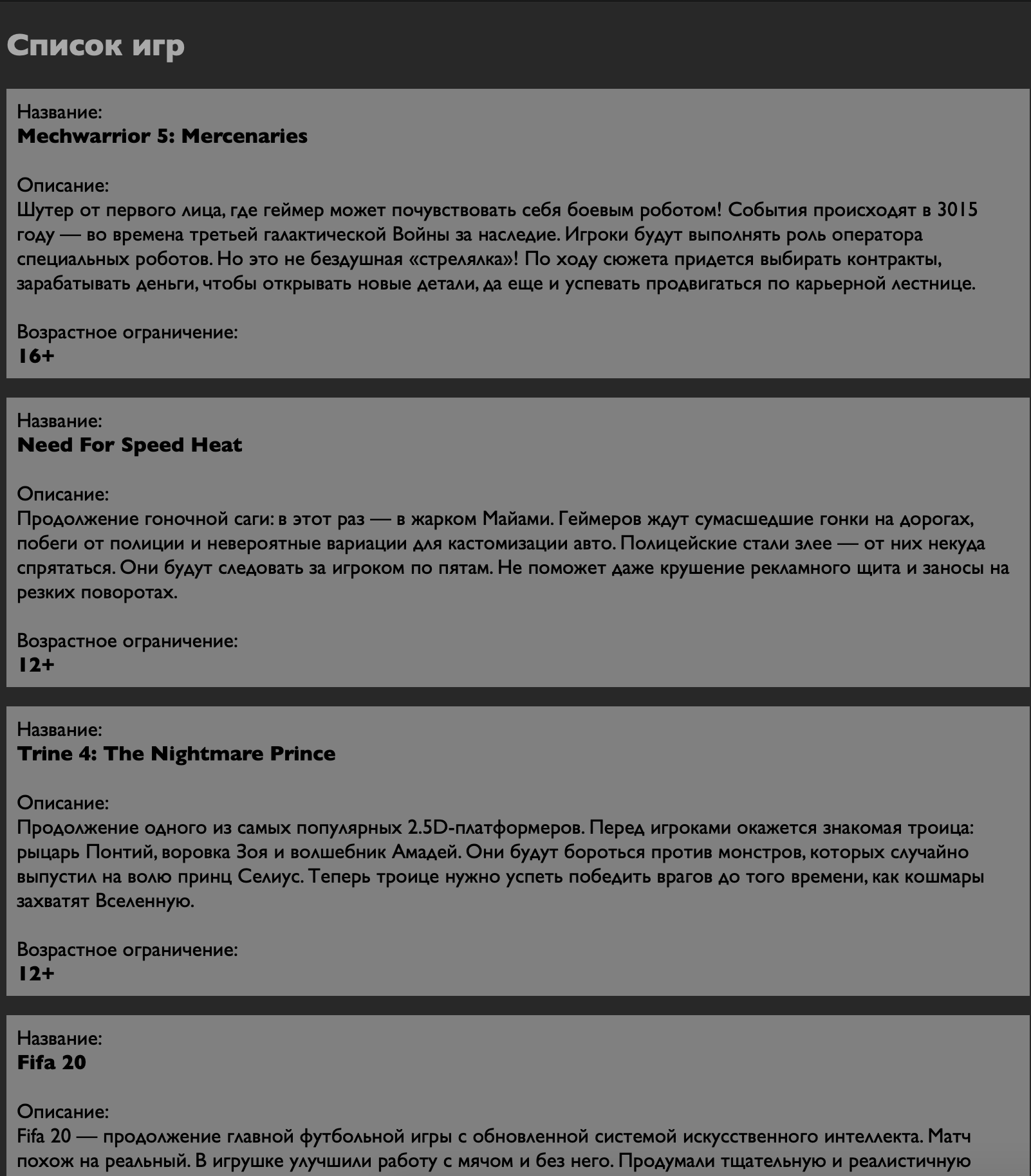
</html>

**Проверка работоспособности**

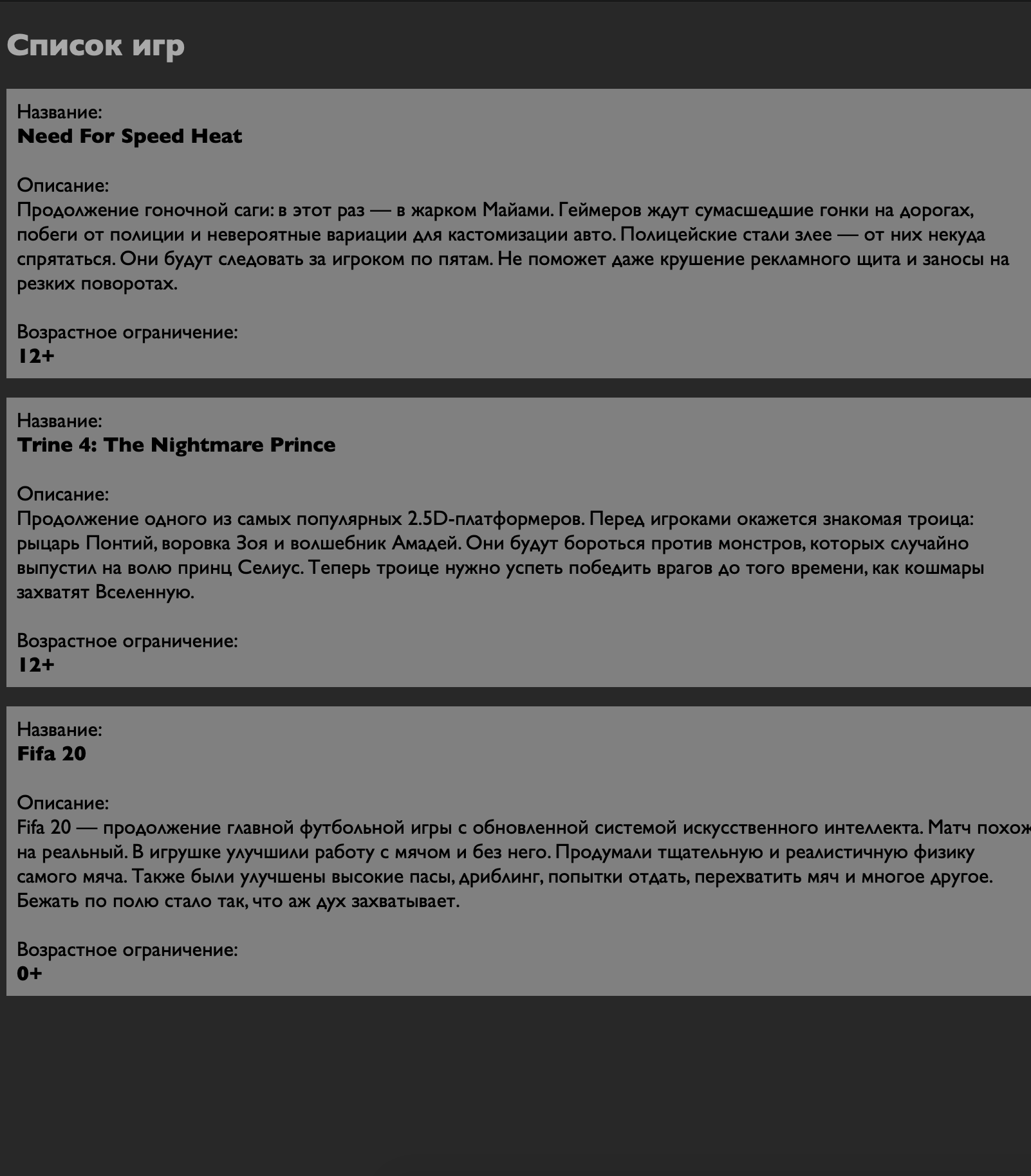
<http://localhost:5015/page/games?age=18>

****

<http://localhost:5015/page/games?age=18>

****

http://localhost:5015/page/games?age=13

****

**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.

**Реализация**

**index.js**

"use strict";

const express = require("express");

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

const app = express();

const port = 5015;

app.listen(port);

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

app.set("view engine", "hbs");

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();

});

app.use(cookieSession({

name: 'session',

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

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

}));

function findUserIndex(login) {

let reqIndex = -1;

for (let i = 0; i < usersArray.length && reqIndex === -1; i++) {

if (usersArray[i]["login"] === login) {

reqIndex = i;

}

}

return reqIndex;

}

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

const login = request.query.login;

const password = request.query.password;

let res = ""

if(!login) return response.end(JSON.stringify({result: "Login not set"}));

if(!password) return response.end(JSON.stringify({result: "Password not set"}));

let index = findUserIndex(login)

if (index === -1) return response.end(JSON.stringify({result: "User not found"}));

if (usersArray[index]["password"] !== password)

return response.end(JSON.stringify({result: "Incorrect password"}));

request.session.login = login;

request.session.password = password;

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

});

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

if(!request.session.login) return response.end(JSON.stringify({result: "You are not authorised"}));

if(!request.session.password) return response.end(JSON.stringify({result: "You are not authorised"}));

const login = request.session.login;

const password = request.session.password;

let index = findUserIndex(login)

const hobby = usersArray[index]["hobby"];

const age = usersArray[index]["age"];

response.end(null);

});

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

request.session = null;

response.end("OK");

});

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

const login = request.session.login;

const password = request.session.password;

let index = findUserIndex(login)

response.render("task2.hbs", usersArray[index]);

});

const usersArray = [

{

login: "user1",

password: "passWord1",

hobby: "games",

age: 10

},

{

login: "user2",

password: "password2",

hobby: "sport",

age: 20

},

{

login: "user3",

password: "passWord3",

hobby: "chess",

age: 15

},

{

login: "admin",

password: "AdMiN",

hobby: "administrating",

age: 19

},

{

login: "user4",

password: "password4",

hobby: "web-surfing",

age: 26

},

{

login: "user5",

password: "passWord5",

hobby: "sitting home",

age: 11

}

]

**loginPage.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Authorization</title>

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

</head>

<body>

<h1>Authorization</h1>

<p>Login</p>

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

<p>Password</p>

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

<br>

<br>

<div id="auth-btn" class="btn-class">submit</div>

<br>

<br>

<p class="err-class" id="error-label"></p>

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

</body>

</html>

**login.js**

"use strict";

window.onload = function() {

const loginField = document.getElementById("field-login");

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

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

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

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);

};

};

btn.onclick = function() {

const login = loginField.value;

const password = passwordFiled.value;

let url = `/api/save?login=${login}&password=${password}`;

ajaxGet(url, function(stringAnswer) {

let objectAnswer = JSON.parse(stringAnswer);

let result = objectAnswer.result;

label.innerHTML = `${result}`;

if (result === "Success") {

url = `/api/get`;

ajaxGet(url, function(stringAnswer) {

label.innerHTML = stringAnswer

if (stringAnswer) {

objectAnswer = JSON.parse(stringAnswer);

result = objectAnswer.result;

label.innerHTML = `${result}`;

}

else {

location.href = `/profile`

}

});

}

});

};

};

**style.css**

body {

padding: 30px;

background: rgb(40, 40, 40);

color: lightgray;

font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif

}

.btn-class {

padding: 6px;

background: purple;

color: white;

cursor: pointer;

display: inline-block;

}

.input-class {

background-color: rgb(30, 30, 30); /\* Цвет фона \*/

color: lightgray; /\* Цвет текста \*/

}

.err-class {

color: rgb(200, 0, 0);

}

**task2.hbs**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Profile</title>

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

</head>

<bodystyle="font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif; background: rgb(40, 40, 40); color: darkgray">

<h1>{{login}}</h1>

Hobby: {{hobby}}

<br>

<br>

Age: {{age}}

<br>

<br>

<div id="quit-btn" class="btn-class">quit</div>

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

</body>

</html>

**quit.js**

"use strict";

window.onload = function() {

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

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);

};

};

btn.onclick = function() {

let url = `/api/delete`;

ajaxGet(url, function(stringAnswer) {

if (stringAnswer === "OK") {

location.href = `/loginPage.html`

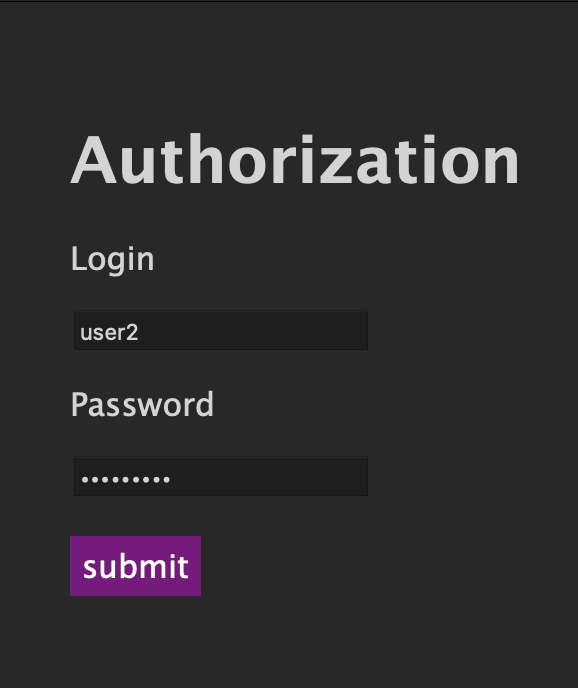
}

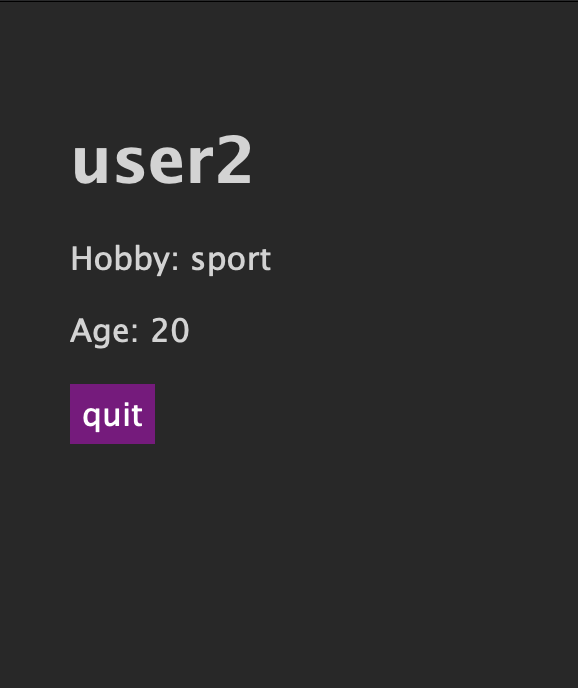
});

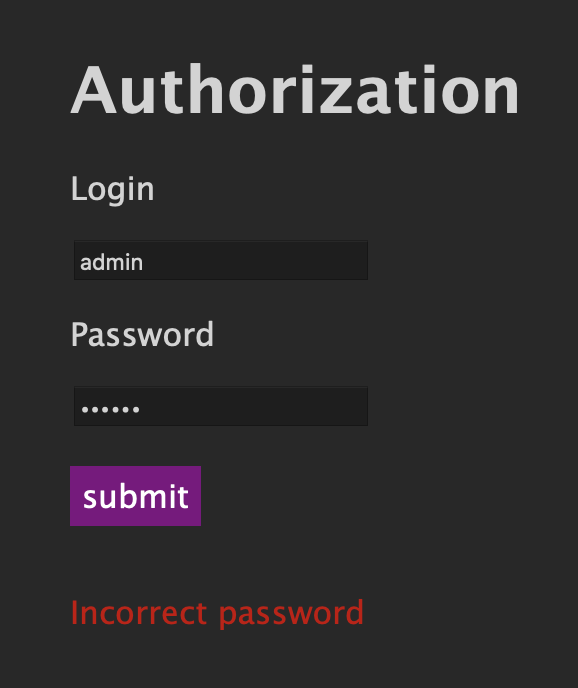
};

};

**Проверка работоспособности**

****

****

****

**Вывод**

Язык программирования JavaScript позволяет создавать веб-сервера при помощи библиотеки request. Сохранение информации на стороне клиента возможно при помощи Cookie.