



**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 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

О Т Ч Е Т

по лабораторной работе № 3

Дисциплина: Архитектура ЭВМ

Студент

ИУ7-52Б

(Группа)

(Подпись, дата)

В.А. Иванов

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

А.Ю. Попов

(И.О. Фамилия)

Москва, 2020

Цель работы

Целью лабораторной работы освоение методов работы с AJAX запросами, шаблонизаторами и cookie.

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.

Программная реализация

index.js (сервер)

```
"use strict";

const express = require("express");
const fs = require("fs");
const file_name = "users.txt"

const app = express();
const port = 5000;
app.listen(port);
console.log(`Server on port ${port}`);

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 = [];
  console.log(body);
  request.on('data', (chunk) => {
    body.push(chunk);
  }).on('end', () => {
    body = Buffer.concat(body).toString();
    callback(body);
  });
}

function get_f_arr(f_name)
{
  let f_arr;
  if (fs.existsSync(f_name)) {
```

```

        const f_str = fs.readFileSync(f_name, "utf-8");
        if (f_str === "")
            f_arr = [];
        else
            f_arr = JSON.parse(f_str);
    } else {
        f_arr = [];
    }
    return f_arr
}

app.post("/save/info", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        let f_arr = get_f_arr(file_name);

        let cont_flag = false;
        for (let i=0; i<f_arr.length && !cont_flag; i++) {
            if (f_arr[i].email == obj.email)
                cont_flag = true;
            else if (f_arr[i].phone == obj.phone)
                cont_flag = true;
        }

        let msg;
        if (cont_flag) {
            msg = "Entry didn't added";
        } else {
            msg = "Entry added";
            f_arr.push(obj);
            fs.writeFileSync(file_name, JSON.stringify(f_arr));
        }

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

```

req_code.js (клиент)

```

"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);
    }
}

window.onload = function() {
    // input fields
    const f_email = document.getElementById("field-email");
    const f_surname = document.getElementById("field-surname");
    const f_phone = document.getElementById("field-phone");

    // button
    const btn = document.getElementById("send-btn");

    // click event
    btn.onclick = function() {
        const msg = JSON.stringify({
            email: f_email.value,
            surname: f_surname.value,

```

```

        phone: f_phone.value
    });

    ajaxPost("/save/info", msg, function(answerString) {
        const answerObject = JSON.parse(answerString);
        const result = answerObject.result;
        alert(result);
    });
};
};
};

```

Тесты

Ввод:

Task 5

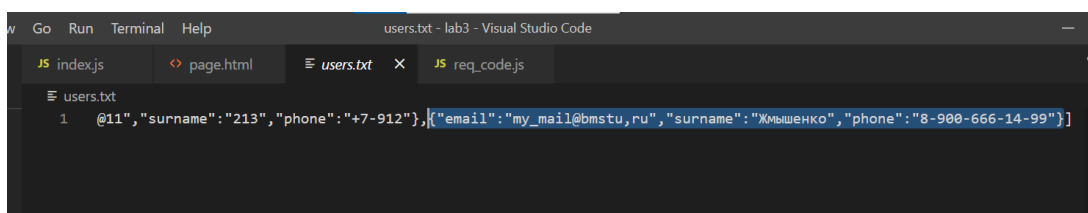
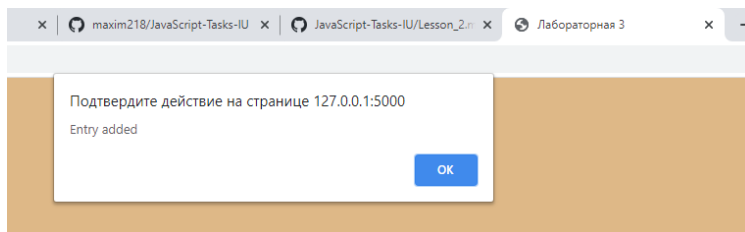
Почта

Фамилия

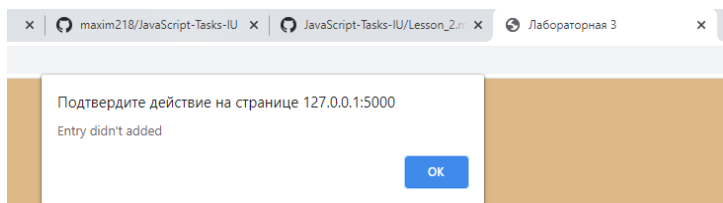
Номер телефона

Отправить

Вывод:



При повторном запросе:



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.

Программная реализация

Дополнение index.js (сервер)

```
function search_email(email) {
    let obj = null;
    const f_arr = get_f_arr(file_name);
    for (let i=0; i<f_arr.length && !obj; i++) {
        if (f_arr[i].email == email)
            obj = f_arr[i];
    }
    return obj;
}

app.get("/search", function(request, response) {
    const email = request.query.email;
    const res_obj = search_email(email);
    let answer = {is_found : true, obj:res_obj};
    if (res_obj === null)
        answer.is_found = false;

    response.end(JSON.stringify(answer));
});
```

search.js (клиент)

```
"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);
    };
};

window.onload = function() {
    // input fields
    const f_email = document.getElementById("field-email");
    // button
    const btn = document.getElementById("send-btn");
    // Output fields
    const out_email = document.getElementById("result-email");
    const out_surname = document.getElementById("result-surname");
    const out_phone = document.getElementById("result-phone");

    function show_user(obj) {
        out_email.innerHTML = `Почта:      ${obj.email}`;
        out_surname.innerHTML = `Фамилия:   ${obj.surname}`;
        out_phone.innerHTML = `Телефон:   ${obj.phone}`;
    }
};
```

```

    }

    function show_void() {
        out_email.innerHTML = ``;
        out_surname.innerHTML = ``;
        out_phone.innerHTML = ``;
    }

    // click event
    btn.onclick = function() {
        const email = f_email.value;
        const url = `/search?email=${email}`;

        ajaxGet(url, function(stringAnswer) {
            const objectAnswer = JSON.parse(stringAnswer);
            if (objectAnswer.is_found)
                show_user(objectAnswer.obj);
            else {
                show_void();
                alert(`Пользователь с почтой ${email} не найден`);
            }
        });
    };
};

```

obj

Тесты

Поиск человека по почте

Введите почту:

Найти

Результат поиска:

Почта: @11
 Фамилия: 213
 Телефон: +7-912

← → ↻ 🏠 ⓘ 127.0.0.1:5000/search.html

Поиск человека по почте

Введите почту:

Найти

Подтвердите действие на странице 127.0.0.1:5000

Пользователь с почтой @12 не найден

ОК

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.

Программная реализация

search.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Лабораторная 3</title>
  <link rel="stylesheet" href="/search.css">
</head>
<body>
  <h2>Поиск человека по почте</h2>
  <p>Введите почту:</p>
  <input id="field-email" type="text" spellcheck="false" autocomplete="off">
  <br>
  <br>
  <div id="send-btn" class="btn-class">Найти</div>
  <br>
  <br>
  <h3>Результат поиска:</h3>
  <p id="result-email"></p>
  <p id="result-surname"></p>
  <p id="result-phone"></p>
  <script src="/search.js"></script>
</body>
</html>
```

search.css

```
body {
  margin-inline-end: auto;
  background: rgb(237, 238, 240);
  font-family: Geneva, Arial, Helvetica, sans-serif;
}

.btn-class {
  padding: 6px;
  background: rgb(74, 118, 168);
  color: white;
  cursor: help;
  display: inline-block;
}
```


page.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Лабораторная 3</title>
  <link rel="stylesheet" href="/page.css">
</head>
<body>

  <h1>Задание 5</h1>

  <p>Почта</p>
  <input id="field-email" type="text" spellcheck="false" autocomplete="off">
  <br>

  <p>Фамилия</p>
  <input id="field-surname" type="text" spellcheck="false" autocomplete="off">
  <br>

  <p>Номер телефона</p>
  <input id="field-phone" type="text" spellcheck="false" autocomplete="off">
  <br>

  <br>
  <div id="send-btn" class="btn-class">Отправить</div>

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

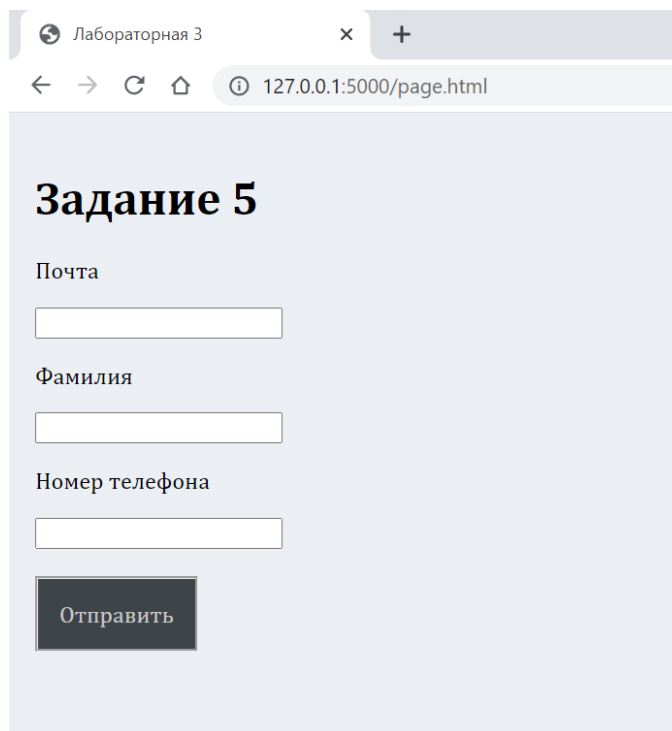
</body>
</html>
```

page.css

```
body {
  padding: 15px;
  background: rgb(235, 239, 243);
  font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
  float: left;
  margin: 3px;
}
.btn-class {
  padding: 15px;
  background: rgb(63, 68, 72);
  color: rgb(200, 200, 204);
  cursor: pointer;
  display: inline-table;
  border-style: groove;
}
```

Тесты

page.html



Лабораторная 3

127.0.0.1:5000/page.html

Задание 5

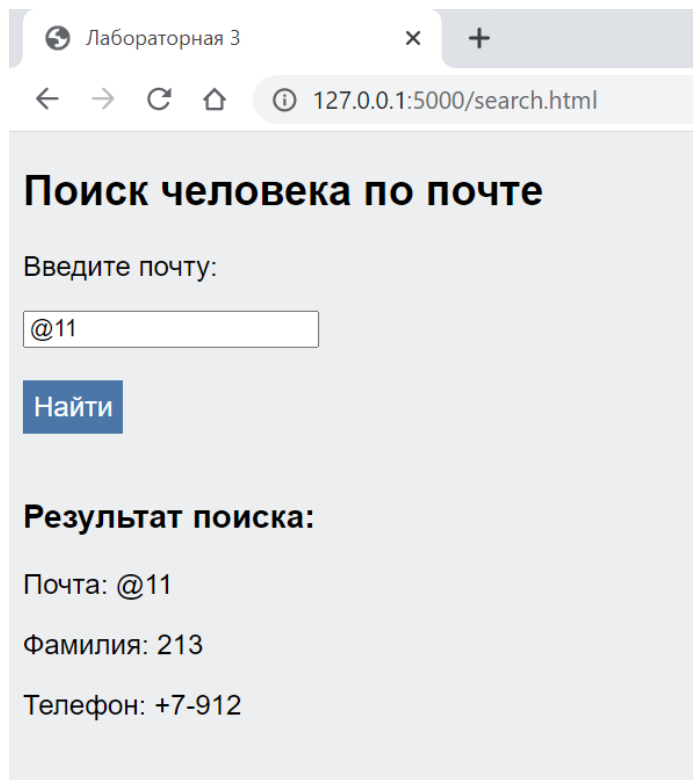
Почта

Фамилия

Номер телефона

Отправить

search.html



Лабораторная 3

127.0.0.1:5000/search.html

Поиск человека по почте

Введите почту:

Найти

Результат поиска:

Почта: @11

Фамилия: 213

Телефон: +7-912

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.

Программная реализация

index.js (сервер)

```
"use strict";

let gameArr = [];
function addGame(name_, descr_, limit_) {
  if (limit_ < 0)
    return;

  let game = {
    name: name_,
    description: descr_,
    age_limit: limit_
  };
  gameArr.push(game);
}
function fillGameArr(){
  addGame("LocoRoco", "Платформер, аркада и головоломка с оригинальными графическими решениями", 0);
  addGame("Crash Bandicoot", "Woah!", 6);
  addGame("Baldur's Gate 3", "Заставляют убивать людей, полуросликов и танцевать нагишом, кошмар", 17);
  addGame("Gulman 5", "Gulman - герой, Superman - нет", 16);
  addGame("Pretty Neko", "Оно того не стоит, не лезь", 21);
  addGame("Pretty Angel", "см. описание Pretty Neko", 21);
  addGame("Dota 2", "2 sides, 3 lines, 100 hours", 16);
}

function getGames(age) {
  let ansArr = [];
  for (let i=0; i < gameArr.length; i++)
    if (gameArr[i].age_limit <= age)
      ansArr.push(gameArr[i]);
  return ansArr;
}

const { request } = require("express");
const express = require("express");
const app = express();
```

```

const port = 5000;
let server = app.listen(port);
console.log(`Server on port ${port}`);

app.set("view engine", "hbs");
const way = __dirname + "/static";
app.use(express.static(way));

fillGameArr();

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("/show_games", function(request, response) {
  const age = request.query.age;
  let info = {
    ageValue: age,
    gameArr: getGames(age)
  }
  response.render("pageGames.hbs", info);
});

```

pageGames.hbs

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Подборка игр</title>
</head>
<body>

<h2>
  Список игр для возраста <i>{{ageValue}}</i>:
</h2>

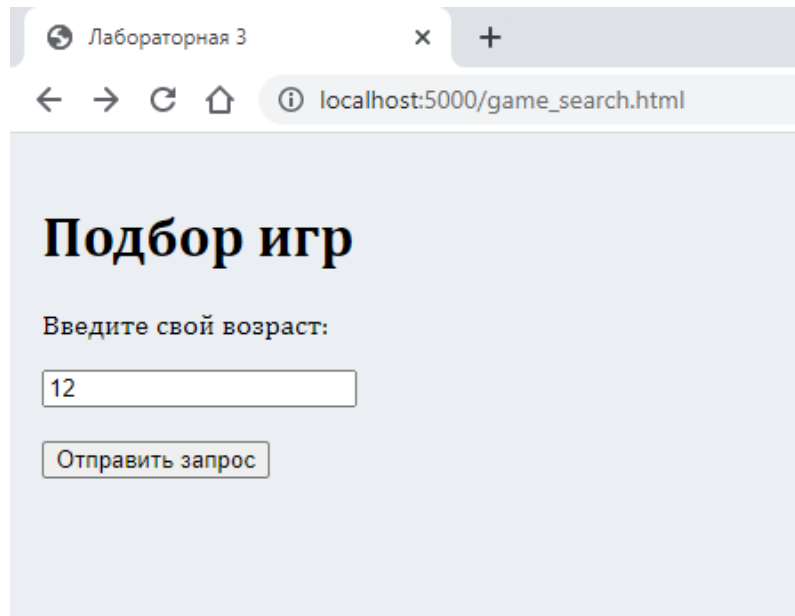
{{#each gameArr}}
  <div style="background: rgb(232, 240, 254); margin-
bottom: 10px; padding: 5px; font-
family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;">
    <h3>{{this.name}} (<i>{{this.age_limit}}</i>)</h3>
    <div style="background: rgb(134, 188, 238); margin-
bottom: 3px; padding: 7px; font-family: Verdana, Geneva, Tahoma, sans-serif;">
      Описание: {{this.description}}
    </div>
  </div>
</div>
{{/each}}

```

```
</body>  
</html>
```

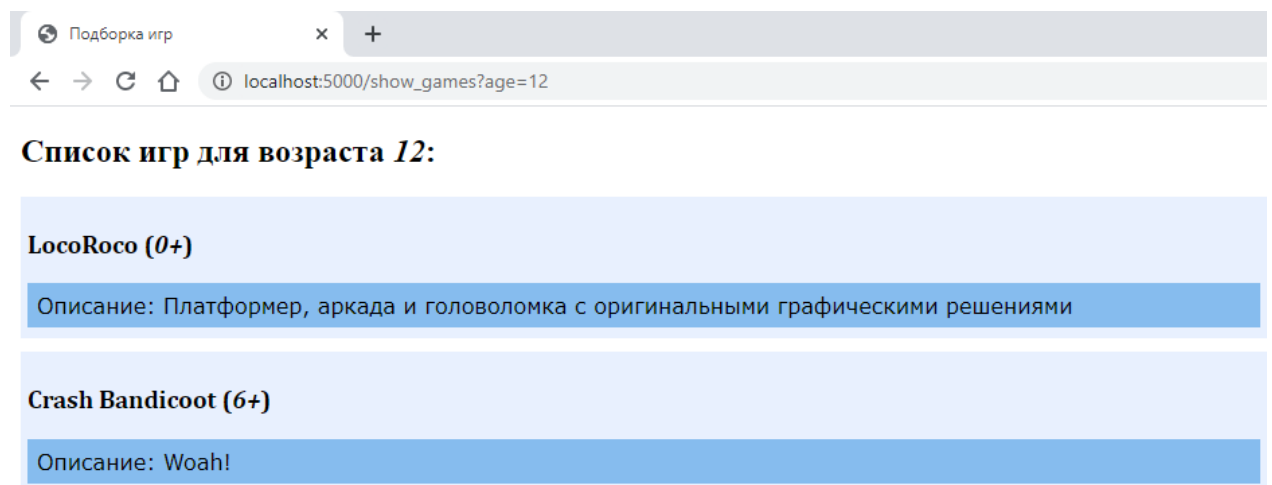
Тесты

Ввод:



The screenshot shows a web browser window with the title "Лабораторная 3". The address bar displays "localhost:5000/game_search.html". The page content includes a heading "Подбор игр" (Game Selection), a label "Введите свой возраст:" (Enter your age:), a text input field containing the number "12", and a button labeled "Отправить запрос" (Send request).

Вывод:



The screenshot shows a web browser window with the title "Подборка игр" (Game Selection). The address bar displays "localhost:5000/show_games?age=12". The page content includes a heading "Список игр для возраста 12:" (List of games for age 12:). Below this, there are two game entries, each with a title and a description:

LocoRoco (0+)	Описание: Платформер, аркада и головоломка с оригинальными графическими решениями
Crash Bandicoot (6+)	Описание: Woah!

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.

Программная реализация

index.js (сервер)

```
"use strict";

let userArr = [];
function userPos(login_) {
  for (let i=0; i<userArr.length; i++)
    if (userArr[i].login == login_)
      return i;
  return -1;
}
function addUser(login_, password_, hobby_, age_) {
  if (age_ <= 0 || password_.length == 0 || !login_)
    return;
  if (userPos(login_) != -1)
    return;

  let user = {
    login: login_,
    password: password_,
    hobby: hobby_,
    age: age_
  };
  userArr.push(user);
}
function fillUserArr(){
  addUser("$Kekotic_2000$", "dumb_password", "Манная каша", 20);
  addUser("Martin", "1221", "Радужные покатушки", 13);
  addUser("Void", "0000", "Спать по 12 часов", 21);
}
fillUserArr();
console.log(userArr);

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

const app = express();
const port = 5000;
let server = app.listen(port);
```

```

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

app.use(cookieSession({
  login: '',
  password: '',
  keys: ['hhh', 'qqq', 'vvv']
}));

app.set("view engine", "hbs");
const way = __dirname + "/static";
app.use(express.static(way));

app.get("/api/save", function(request, response) {
  const login = request.query.login;
  const password = request.query.password;

  if(!login) return response.end("Input login");
  if(!password) return response.end("Input password");

  let i = userPos(login);
  if (i !== -1) {
    if (userArr[i].password !== password)
      return response.end(`Wrong password`);
  } else
    return response.end(`No user "${login}" in users list`);

  request.session.login = login;
  request.session.password = password;
  response.end();
});

app.get("/api/get", function(request, response) {
  response.end(JSON.stringify(request.session));
})

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("/show_me", function(request, response) {
  const login = request.session.login;
  const password = request.session.password;

  let i = userPos(login);
  if (i !== -1) {

```

```

        if (userArr[i].password != password)
            return response.end(`Wrong password`);
    } else
        return response.end(`No user "${login}" in users list`);

    let info = {
        login: userArr[i].login,
        hobby: userArr[i].hobby,
        age: userArr[i].age
    }
    response.render("pageUser.hbs", info);
});

```

page.js(клиент)

```

"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);
    };
}

function shut_down() {
    ajaxGet(`/shut_down`, function(stringAnswer) {
        alert(stringAnswer);
    });
}

function updateCookies(upd_func) {
    ajaxGet(`/api/get`, function(stringAnswer) {
        document.cookie = stringAnswer;
        upd_func();
    });
}

window.onload = function() {
    const login_in = document.getElementById("field-login");
    const password_in = document.getElementById("field-password");

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

    function profile_block() {
        profile_btn.style.color = "rgb(0, 0, 0)"
        profile_btn.style.background = "rgb(206, 63, 63)";
    }
}

```



```

}
function profile_open() {
    profile_btn.style.color = "rgb(0, 0, 0)"
    profile_btn.style.background = "rgb(61, 187, 78)";
}
function profile_update() {
    if (!document.cookie) {
        profile_block();
        return;
    }

    const cookies = JSON.parse(document.cookie);
    if (cookies.login)
        profile_open();
    else
        profile_block();
}
updateCookies(profile_update);

auth_btn.onclick = function() {
    const login = login_in.value;
    const password = password_in.value;
    const url = `/api/save?login=${login}&password=${password}`;

    ajaxGet(url, function(strAns) {
        if (strAns)
            alert(strAns);
        updateCookies(profile_update);
    });
};
}

```

Тесты

← → ↻ 🏠 ⓘ localhost:5000/auth.html

Авторизация:

Логин

Пароль

Войти

Профиль

← → ↻ 🏠 ⓘ localhost:5000/auth.html

Авторизация:

Логин

Пароль

Войти

Профиль

Данные пользователя *Void*:

Возраст: 21

Хобби: Спать по 12 часов

Вывод

В рамках лабораторной работы было выполнено ознакомление и практическое закрепление основ работы с AJAX запросами GET и POST, шаблонизаторами и cookies в языке JS.