



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 09.03.04 Программная инженерия

О Т Ч Е Т

по лабораторной работе № 3

Название: Работа с POST и GET запросами, шаблонизаторами и cookie

Дисциплина: Архитектура ЭВМ

Студент	<u>ИУ7-52Б</u>	<u></u>	<u>Брянская Е.В.</u>
	(Группа)	(Подпись, дата)	(И.О. Фамилия)

Преподаватель	<u></u>	<u>Попов А.Ю.</u>
	(Подпись, дата)	(И.О. Фамилия)

Москва, 2020

Цель: научиться работать с POST и GET запросами, корректно взаимодействовать с сервером, также изучить cookie. Выполнить соответствующие задания на основе изученного материала.

Часть 1

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" введенная 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 идет 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.

index.js

```
4  "use strict";
5
6  const express = require("express");
7  const fs = require("fs");
8  const file = "file.txt";
9
10 const app = express();
11 const port = 5000;
12 app.listen(port);
13 console.log(`Server on port ${port}`);
14
15 // отправка статических файлов
16 const way = __dirname + "/static";
17 app.use(express.static(way));
18
19 // body
20 function loadBody(request, callback) {
21   let body = [];
22   request.on('data', (chunk) => {
23     body.push(chunk);
24   }).on('end', () => {
25     body = Buffer.concat(body).toString();
26     callback(body);
27   });
28 }
```

```

31 function is_exist(surname, phone, email)
32 {
33     let data = [];
34
35     data.push(fs.readFileSync(file, "utf-8"));
36
37     if (data != '')
38     {
39         let temp = JSON.parse(data);
40
41         for (let i = 0; i < temp.length; i++)
42             if (temp[i].phone == phone || temp[i].email == email)
43                 return "ERROR: not unique";
44
45         temp.push({"Surname": surname, "phone": phone, "email": email});
46         data = temp;
47     }
48     else
49         data[0] = {"Surname": surname, "phone": phone, "email": email};
50
51     fs.writeFileSync(file, JSON.stringify(data))
52
53     return "Information was added";
54 }
55
56 // it is post
57 app.post("/save/info", function(request, response) {
58     loadBody(request, function(body) {
59         const obj = JSON.parse(body);
60         const surname = obj["surname"];
61         const phone = obj["phone"];
62         const email = obj["email"];
63         let msg = is_exist(surname, phone, email);
64         response.end(JSON.stringify({
65             result: msg
66         }));
67     });
68 });

```

code.js

```

3  window.onload = function() {
4      const f1 = document.getElementById("field-first");
5      const f2 = document.getElementById("field-second");
6      const f3 = document.getElementById("field-third");
7
8      const btn = document.getElementById("save-info-btn");
9
10     function ajaxPost(urlString, bodyString, callback) {
11         let r = new XMLHttpRequest();
12         r.open("POST", urlString, true);
13         r.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
14         r.send(bodyString);
15         r.onload = function() {
16             callback(r.response);
17         }
18     }
19
20     btn.onclick = function() {
21         const surname = f1.value;
22         const phone = f2.value;
23         const email = f3.value;
24
25         ajaxPost("/save/info", JSON.stringify([
26             surname, phone, email
27         ]), function(answerString) {
28             const answerObject = JSON.parse(answerString);
29             const result = answerObject.result;
30             alert(result);
31         });
32     };
33 };

```

page.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Моя страница</title>
6      <!--<link rel="stylesheet" href="/style_task_3.css">-->
7      <link rel="stylesheet" href="/style.css">
8  </head>
9  <body>
10     <h1>Информация</h1>
11
12     <p>Фамилия</p>
13     <input id="field-first" type="text" spellcheck="false" autocomplete="off">
14
15     <p>Номер телефона</p>
16     <input id="field-second" type="text" spellcheck="false" autocomplete="off">
17
18     <p>Почта</p>
19     <input id="field-third" type="text" spellcheck="false" autocomplete="off">
20
21     <br>
22     <br>
23
24     <div id="save-info-btn" class="btn-class">Отправить</div>
25
26     <br>
27     <br>
28
29     <h1 id="result-label"></h1>
30
31     <script src="/code.js"></script>
32 </body>
33 </html>

```

Тесты:

Тест	Результат
<p>Файл пустой</p> <div><h3>Информация</h3><p>Фамилия</p><input type="text" value="Иванов"/><p>Номер телефона</p><input type="text" value="123456789"/><p>Почта</p><input type="text" value="qwe"/><p>Отправить</p></div>	<p>Подтвердите действие на странице localhost:5000</p> <p>Information was added</p> <p>OK</p>
<p>В файле содержится информация с прошлого запроса</p> <div><h3>Информация</h3><p>Фамилия</p><input type="text" value="Иванов"/><p>Номер телефона</p><input type="text" value="123456789"/><p>Почта</p><input type="text" value="qwe"/><p>Отправить</p></div>	<p>Подтвердите действие на странице localhost:5000</p> <p>ERROR: not unique</p> <p>OK</p>
<div><h3>Информация</h3><p>Фамилия</p><input type="text" value="Лежнев"/><p>Номер телефона</p><input type="text" value="1000000"/><p>Почта</p><input type="text" value="qwe@123"/><p>Отправить</p></div>	<p>Подтвердите действие на странице localhost:5000</p> <p>Information was added</p> <p>OK</p>

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.

index.js

```
5  const file = "file.txt";
6
7
8  function get_data(filename)
9  {
10     let data = [];
11
12     data.push(fs.readFileSync(filename, "utf-8"));
13
14     if (data != '')
15         data = JSON.parse(data);
16
17     return data;
18 }
19
20 function find_person(email)
21 {
22     let res = null;
23     let data = get_data(file);
24
25     for (let i = 0; i < data.length; i++)
26         if (data[i].email == email)
27         {
28             res = data[i];
29             break;
30         }
31
32     return res;
33 }
```

```

35 // импортируем библиотеку
36 const express = require("express");
37
38 // запускаем сервер
39 const app = express();
40 const fs = require("fs");
41 const port = 5000;
42 app.listen(port);
43 console.log(`Server on port ${port}`);
44
45 // отправка статических файлов
46 const way = __dirname + "/static";
47 app.use(express.static(way));
48
49 // заголовки в ответ клиенту
50 app.use(function(req, res, next) {
51     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
52     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
53     res.header("Access-Control-Allow-Origin", "*");
54     next();
55 });
56
57 app.get("/find", function(request, response) {
58     const email = request.query.email;
59     let res = find_person(email);
60
61     response.end(JSON.stringify(res));
62 });

```

code.js

```

3 window.onload = function() {
4     // input fields
5     const f1 = document.getElementById("field-email");
6
7     // button
8     const btn = document.getElementById("find-btn");
9
10    // labels
11    const label_result = document.getElementById("result");
12    const label_surname = document.getElementById("result-surname");
13    const label_phone = document.getElementById("result-phone");
14    const label_email = document.getElementById("result-email");
15
16    // ajax get
17    function ajaxGet(urlString, callback) {
18        let r = new XMLHttpRequest();
19        r.open("GET", urlString, true);
20        r.setRequestHeader("Content-Type", "text/plain; charset=UTF-8");
21        r.send(null);
22        r.onload = function() {
23            callback(r.response);
24        };
25    };

```

```

27 // click event
28 btn.onclick = function() {
29     const email = f1.value;
30
31     const url = `/find?email=${email}`;
32     ajaxGet(url, function(stringAnswer) {
33         const objectAnswer = JSON.parse(stringAnswer);
34
35         if (objectAnswer != null)
36         {
37             label_result.innerHTML = `Ответ:`;
38             label_surname.innerHTML = `Фамилия: ${objectAnswer.Surname}`;
39             label_phone.innerHTML = `Телефон: ${objectAnswer.phone}`;
40             label_email.innerHTML = `Почта: ${objectAnswer.email}`;
41         }
42         else
43         {
44             label_result.innerHTML = `Ответ:`;
45             label_surname.innerHTML = `Фамилия: -`;
46             label_phone.innerHTML = `Телефон: -`;
47             label_email.innerHTML = `Почта: -`;
48             alert(`Человек с данной почтой ${email} НЕ найден`);
49         }
50     });
51 };
52
53

```

page.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Моя страница</title>
6
7      <link rel="stylesheet" href="/style.css">
8      <!--<link rel="stylesheet" href="/style_task_3.css"-->
9  </head>
10 <body>
11     <h1>Найти человека</h1>
12
13     <p>Введите его почту:</p>
14     <input id="field-email" type="text" spellcheck="false" autocomplete="off">
15
16     <br>
17     <br>
18
19     <div id="find-btn" class="btn-class">Отправить</div>
20
21     <br>
22     <br>
23
24     <h1 id="result"></h1>
25     <h2 id="result-surname"></h2>
26     <h2 id="result-phone"></h2>
27     <h2 id="result-email"></h2>
28
29     <script src="/code.js"></script>
30 </body>
31 </html>

```

Тесты:

Тест	Результат
<p>Пользователь существует</p> <div> <h3>Найти человека</h3> <p>Введите его почту:</p> <input type="text" value="qaz@"/> <p>Отправить</p> </div>	<div> <h3>Найти человека</h3> <p>Введите его почту:</p> <input type="text" value="qaz@"/> <p>Отправить</p> </div> <div> <p>Ответ:</p> <p>Фамилия: Букина</p> <p>Телефон: 987</p> <p>Почта: qaz@</p> </div>
<p>Пользователя нет</p> <div> <h3>Найти человека</h3> <p>Введите его почту:</p> <input type="text" value="123456789"/> <p>Отправить</p> </div>	<p>Подтвердите действие на странице localhost:5000</p> <p>Человек с данной почтой 123456789 НЕ найден</p> <p>OK</p>

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.

style.css (для первого задания)

```

1  body {
2      padding: 30px;
3      background: lightskyblue;
4      font-family: Geneva, Arial, Helvetica, sans-serif;
5      font-style: italic;
6  }
7
8  .btn-class {
9      padding: 6px;
10     background: blue;
11     color: white;
12     cursor: pointer;
13     display: inline-block;
14 }
15
16 .btn-class:active{
17     background: Navy;
18     color: white;
19 }
20
21 .btn-class:hover {
22     background: red;
23     color: white;
24 }

```

Результат:

(при наведении на кнопку происходит смена цвета)

Информация	Информация
Фамилия	Фамилия
<input type="text"/>	<input type="text"/>
Номер телефона	Номер телефона
<input type="text"/>	<input type="text"/>
Почта	Почта
<input type="text"/>	<input type="text"/>
<input type="button" value="Отправить"/>	<input type="button" value="Отправить"/>

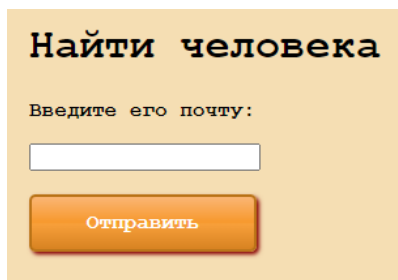
style.css (для второго задания)

```

1  body {
2      padding: 30px;
3      background: #f5f5dc;
4      font-family: 'Courier New', Courier, monospace;
5      font-weight: bold;
6  }
7
8  .btn-class {
9      display: inline-block;
10     width: 170px;
11     line-height: 40px;
12     text-align: center;
13     color: #fff;
14     border: 2px solid #b67014ee;
15     border-radius: 5px;
16     box-shadow: 0 0 0 60px rgba(0,0,0,0) inset, .1em .1em .2em #800;
17     background: linear-gradient(to top right, #fbb675 48%, #f8a03b 52%, #d88322);
18 }

```

Результат:



Часть 2

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.

index.js

```

5   let data_games = [];
6
7   function fill_data()
8   {
9       data_games.push({ "name": "Portal 2", "decsr": "Вторая часть
10      data_games.push({ "name": "Minecraft", "decsr": "Исследуйте
11      data_games.push({ "name": "Gris", "decsr": "Грис – наивная д
12      data_games.push({ "name": "Deponia", "decsr": "Главгерой по
13      data_games.push({ "name": "National Geographic Challenge!",
14  }
15
16  function get_games(age)
17  {
18      let res = [];
19
20      for (let i = 0; i < data_games.length; i++)
21      {
22          if (data_games[i].age_limit <= age)
23              res.push(data_games[i]);
24      }
25      return res;
26  }
27
28  // импорт библиотеки
29  const express = require("express");
30
31  // запускаем сервер
32  const app = express();
33  const port = 5000;
34  app.listen(port);
35  console.log(`Server on port ${port}`);
36
37  // отправка статических файлов
38  const way = __dirname + "/static";
39  app.use(express.static(way));
40
41  // активируем шаблонизатор
42  app.set("view engine", "hbs");
43
44  fill_data();

```

```

45  // заголовки в ответ клиенту
46  app.use(function(req, res, next) {
47      res.header("Cache-Control", "no-cache, no-store, must-revalidate");
48      res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
49      res.header("Access-Control-Allow-Origin", "*");
50      next();
51  });
52
53  app.get("/find_games", function(request, response) {
54      const age = request.query.age;
55
56      const infoObject = {
57          age : age,
58          games : get_games(age)
59      }
60
61      response.render("pageGames.hbs", infoObject);
62  });

```

Page.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Моя страница</title>
6
7      <link rel="stylesheet" href="/style.css">
8  </head>
9  <body>
10     <h1>Подобрать игры по возрасту</h1>
11     <form method="GET" action="/find_games">
12
13         <p>Введите возраст:</p>
14         <input name = "age" type="number" spellcheck="false" autocomplete="off">
15
16         <br>
17         <br>
18
19         <input type="submit" value="Найти">
20     </form>
21 </body>
22 </html>

```

pageGames.hbs

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Подобрать игры по возрасту</title>
6  </head>
7  <body>
8
9      <h2>
10         Результат
11     </h2>
12
13     {{#each games}}
14         <div style="background: ■rgb(201, 248, 244); margin-bottom: 15px; padding: 4px;">
15             <h3>Название: {{this.name}}</h3>
16             <br>
17             Описание: {{this.decsr}}
18             <br>
19             Возрастное ограничение: <i>{{this.age_limit}}</i>
20         </div>
21     </div>
22     {{/each}}
23
24 </body>
25 </html>

```

Тесты:

Тест	Результат
------	-----------

<h2>Подобрать игры по возрасту</h2> <p>Введите возраст:</p> <input type="text" value="7"/> <p>Найти</p>	<h2>Результат</h2> <div> <p>Название: Gris</p> <p>Описание: Грис — наивная девочка, запершаяся в собственном мире из-за боли, воплощение в ее платье, которое дает разные способности, позволяющие лучше ориентироваться в мире. Эмоционально крепнет и начинает видеть свой мир немного иначе, открывая все новые и новые тайны.</p> <p>Возрастное ограничение: 0+</p> </div> <div> <p>Название: National Geographic Challenge!</p> <p>Описание: Пазл от National Geographic - разгадывайте (можно вместе с друзьями!)</p> <p>Возрастное ограничение: 0+</p> </div>
<h2>Подобрать игры по возрасту</h2> <p>Введите возраст:</p> <input type="text" value="55"/> <p>Найти</p>	<h2>Результат</h2> <div> <p>Название: Portal 2</p> <p>Описание: Вторая часть Portal - идеальная игра для любителей ностальгии. Та же самая загадка будут более сложными и хитроумными, а часть сюжета пройдет за пределами н</p> <p>Возрастное ограничение: 10+</p> </div> <div> <p>Название: Minecraft</p> <p>Описание: Исследуйте случайным образом генерируемые миры и стройте самые разные</p> <p>Возрастное ограничение: 10+</p> </div> <div> <p>Название: Gris</p> <p>Описание: Грис — наивная девочка, запершаяся в собственном мире из-за боли, окутывающ воплощение в ее платье, которое дает разные способности, позволяющие лучше ориентироваться в мире. Эмоционально крепнет и начинает видеть свой мир немного иначе, открывая все новые и новые тайны.</p> <p>Возрастное ограничение: 0+</p> </div> <div> <p>Название: Deponia</p> <p>Описание: Главгерой по имени Руфус живет на планете, превращенной в гигантскую мусор и мечтает о лучшей жизни. И однажды судьба дает ему шанс - надо лишь вернуть на соседнюю кучу мусора...</p> <p>Возрастное ограничение: 13+</p> </div> <div> <p>Название: National Geographic Challenge!</p> <p>Описание: Пазл от National Geographic - разгадывайте (можно вместе с друзьями!) лег</p> <p>Возрастное ограничение: 0+</p> </div>

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.

index.js

```
5 let data_users = [];
6
7 function fill_data()
8 {
9     data_users.push({"login": "_qwe_", "password": "1234", "hobby": "Собираю марки", "age": 15});
10    data_users.push({"login": "elena75", "password": "123qwe", "hobby": "Книги", "age": 40});
11    data_users.push({"login": "oleg0205", "password": "password", "hobby": "Дайвинг", "age": 37});
12    data_users.push({"login": "gorik", "password": "5", "hobby": "Охота и рыбалка", "age": 48});
13    data_users.push({"login": "my_login_2", "password": "1a2b3c", "hobby": "Оперное пение", "age": 10});
14 }
15
16 function is_exist(login)
17 {
18     for (let i = 0; i < data_users.length; i++)
19         if (data_users[i].login === login)
20             return data_users[i];
21
22     return null;
23 }
24
25 // импортируем библиотеки
26 const express = require("express");
27 const cookieSession = require("cookie-session");
28
29 // запускаем сервер
30 const app = express();
31 const port = 5000;
32 app.listen(port);
33 console.log(`Server on port ${port}`);
34
35 // активируем шаблонизатор
36 app.set("view engine", "hbs");
37
38 app.use(cookieSession({
39     login: '',
40     password: '',
41     keys: ['hhh', 'qqq', 'vvv']
42 }));
43
44 fill_data();
```

```

46 // отправка статических файлов
47 const way = __dirname + "/static";
48 app.use(express.static(way));
49
50 // заголовки в ответ клиенту
51 app.use(function(req, res, next) {
52     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
53     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
54     next();
55 });
56
57 app.get("/api/save", function(request, response) {
58     const login = request.query.login;
59     const password = request.query.password;
60
61     if(!login) return response.end("Login not set");
62     if(!password) return response.end("Password not set");
63
64     let res = is_exist(login);
65
66     if (res === null)
67         return response.end("Such login NOT FOUND");
68
69     if (res.password !== password)
70         return response.end("Wrong password");
71
72     // выставляем cookie
73     request.session.login = login;
74     request.session.password = password;
75
76     response.end();
77 });

```

```

79 app.get("/show_profile", function(request, response) {
80     if(!request.session.login) return response.end(" Login is not exists");
81     if(!request.session.password) return response.end("Password is not exists");
82
83     const login = request.session.login;
84     const password = request.session.password;
85
86     let res = is_exist(login);
87
88     if (res === null)
89         return response.end("Such login NOT FOUND");
90
91     if (res.password !== password)
92         return response.end("Wrong password");
93
94     const infoObject = {
95         login : res.login,
96         age : res.age,
97         hobby : res.hobby
98     }
99
100     response.render("User.hbs", infoObject);
101 });

```

code.js


```

3 window.onload = function() {
4     // input fields
5     const f1 = document.getElementById("login");
6     const f2 = document.getElementById("password");
7
8     // button
9     const btn_in = document.getElementById("btn_in");
10    const btn_prof = document.getElementById("btn_prof");
11
12    // ajax get
13    function ajaxGet(urlString, callback) {
14        let r = new XMLHttpRequest();
15        r.open("GET", urlString, true);
16        r.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");
17        r.send(null);
18        r.onload = function() {
19            callback(r.response);
20        };
21    };
22
23    function change_btn()
24    {
25        document.getElementById("btn_in").hidden = true;
26        document.getElementById("btn_prof").hidden = false;
27        document.getElementById("login").setAttribute('disabled', 'disabled');
28        document.getElementById("password").setAttribute('disabled', 'disabled');
29    }
30
31    // click event
32    btn_in.onclick = function() {
33        const login = f1.value;
34        const password = f2.value;
35
36        const url = `/api/save?login=${login}&password=${password}`;
37        ajaxGet(url, function(stringAnswer) {
38            if (stringAnswer)
39                alert(stringAnswer);
40            else
41                change_btn();
42        });
43    };
44 };

```

page.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Авторизация</title>
6
7      <h2>Авторизация</h2>
8
9      <link rel="stylesheet" href="/style.css">
10 </head>
11 <body>
12     <p>Введите логин:</p>
13     <input id = "login" type="text" spellcheck="false" autocomplete="off">
14
15     <br>
16     <br>
17
18     <p>Введите пароль:</p>
19     <input id = "password" type="password" spellcheck="false" autocomplete="off">
20
21     <br>
22     <br>
23
24     <div id = "btn_in" class="btn-class1">Войти</div>
25 <form method="GET" action="/show_profile">
26     <br>
27     <input id = "btn_prof" class="btn-class2" hidden=true type="submit" value="Просмотреть профиль">
28 </form>
29 <script src="/code.js"></script>
30
31 </body>
32 </html>

```

User.hbs

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Личная информация</title>
6  </head>
7  <body>
8
9      <h2>
10     Личная информация
11 </h2>
12
13     <div style="background: ■ rgba(185, 232, 238, 0.829); margin-bottom: 15px; padding: 4px;">
14         <h3>Логин: {{login}}</h3>
15         <br>
16         Возраст: {{age}}
17         <br>
18         Хобби: {{hobby}}
19     </div>
20 </div>
21
22 </body>
23 </html>

```

Тесты:

Тест	Результат
------	-----------

<p>Пользователь существует</p> <div data-bbox="225 241 531 620"> <h3>Авторизация</h3> <p>Введите логин:</p> <input type="text" value="_qwe_"/> <p>Введите пароль:</p> <input type="password" value="...."/> <p>Войти</p> </div>	<div data-bbox="770 152 1061 560"> <h3>Авторизация</h3> <p>Введите логин:</p> <input type="text" value="_qwe_"/> <p>Введите пароль:</p> <input type="password" value="...."/> <p>Просмотреть профиль</p> </div> <div data-bbox="770 616 1106 853"> <h3>Личная информация</h3> <p>Логин: _qwe_</p> <p>Возраст: 15 Хобби: Собираю марки</p> </div>
<p>Пользователя нет</p> <div data-bbox="225 1008 541 1415"> <h3>Авторизация</h3> <p>Введите логин:</p> <input type="text" value="1111111"/> <p>Введите пароль:</p> <input type="password" value="."/> <p>Войти</p> </div>	<p>Подтвердите действие на странице localhost:5000</p> <p>Such login NOT FOUND</p> <p>OK</p>

Вывод

В ходе лабораторной работы были изучены POST и GET запросы, шаблонизаторы, методы работы с cookie. На основе полученных знаний были выполнены задания лабораторной работы, результаты представлены в виде отчёта.