



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

Дисциплина: Архитектура ЭВМ

Студент

ИУ7-52Б

(Группа)

Сучков А.Д.

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Попов А.Ю.

(Подпись, дата)

(И.О. Фамилия)

Москва, 2020

Цель: изучить получение статических файлов, GET и POST запросы в AJAX, а также изучить шаблонизаторы и cookie.

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*" 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.

Листинг приведён для 1 и 2-го заданий одновременно.

Листинг класса сервера

```
"use strict";

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

const dataPath = "data/users.txt";

class Server {
  constructor(port = 5015) {
    this.port = port;
    this.app = express();
    this.staticPath = __dirname + "/static";
  }

  start() {
    try {
      this.app.listen(this.port);
      console.log("Server started on port " + this.port);
    }
    catch {
      console.log("Server startup error");
      throw new Error("Server start error");
    }

    this.app.use(this.headerFormation);
    this.app.use(express.static(this.staticPath));

    this.app.post("/addUser", this.addUser);
    this.app.get("/getUser", this.getUser);
  }

  headerFormation(request, response, next) {
    response.header("Cache-Control", "no-cache, no-store, must-revalidate");
    response.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
    response.header("Access-Control-Allow-Origin", "*");
    next();
  }

  addUser(request, response) {
    const email = request.query.email;
    const surname = request.query.surname;
    const phone = request.query.phone;

    const usersData = fs.readFileSync(dataPath, "utf8");
```

```

let usersStorage = usersData.length ? new Map(JSON.parse(usersData)) :
    new Map();

function findNumber(users, number) {
    for (let value of users.values()) {
        if (value.number == number) {
            return true;
        }
    }

    return false;
}

const emailExists = usersStorage.has(email);
const numberExists = !emailExists ? findNumber(usersStorage, phone) :
    true;

if (!numberExists) {
    usersStorage.set(email, {"surname" : surname, "phone" : phone});
    fs.writeFileSync(dataPath, JSON.stringify([...usersStorage]));
    response.end(JSON.stringify({added : true}));
}
else {
    response.end(JSON.stringify({added : false}));
}
}

getUser(request, response) {
    const email = request.query.email;

    const usersData = fs.readFileSync(dataPath, "utf8")
    let usersStorage = usersData.length ? new Map(JSON.parse(usersData)) :
        new Map();

    if (!usersStorage.has(email)) {
        response.end(JSON.stringify({found: false}));
    } else {
        const values = usersStorage.get(email);
        response.end(JSON.stringify({found: true, surname: values.surname,
phone: values.phone}));
    }
}

let server = new Server(5015);
server.start();

```

Листинг HTML страницы для добавления пользователей

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Добавление пользователя</title>
  <link rel="stylesheet" type="text/css" href="/style/addUser_style.css" />
</head>
<body>
  <p>Введите email</p>
  <input id="email" type="text" spellcheck="false" autocomplete="off">

  <p>Введите фамилию</p>
  <input id="surname" type="text" spellcheck="false" autocomplete="off">

  <p>Введите номер телефона</p>
  <input id="phone" type="text" spellcheck="false" autocomplete="off">

  <br>
  <br>

  <div id="addButton" class="btn-class">Добавить</div>

  <br>
  <br>

  <h3 id="status"></h3>
  <script src="/scripts/add_user.js"></script>
</body>
</html>
```

Листинг HTML страницы для поиска пользователей

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Найти пользователя</title>
  <link rel="stylesheet" type="text/css" href="/style/addUser_style.css" />
</head>
<body>
  <p>Введите email</p>
  <input id="email" type="text" spellcheck="false" autocomplete="off">

  <br>
  <br>
```

```
<div id="getButton" class="btn-class">Найти</div>

<br>
<br>

<h3 id="result"></h3>
<script src="/scripts/get_user.js"></script>
</body>
</html>
```

Скрипт для добавления пользователей

```
"use strict";

function checkValidEmail(email) {
    const regularEmail = /^[\\w]{1}[\\w-\\.]*@[\\w-]+\\. [a-z]{2,4}$/i
    return email.match(regularEmail);
}

function checkValidSurname(surname) {
    const regularSurname = /^[А-ЯЁ]{1}[а-яё]{1,23})$/
    return surname.match(regularSurname);
}

function checkValidPhone(phone) {
    const regularPhone = /^(8|\\+7)[\\- ])?(\\(\\?\\d{3}\\)?[\\- ])?[\\d\\- ]{7,10}$/
    return phone.match(regularPhone);
}

window.onload = function() {
    const emailField = document.getElementById("email");
    const surnameField = document.getElementById("surname");
    const phoneField = document.getElementById("phone");

    const addButton = document.getElementById("addButton");
    const statusLabel = document.getElementById("status");

    function ajaxGet(urlString, callback) {
        let request = new XMLHttpRequest();

        request.open("POST", urlString, true);
        request.setRequestHeader("Content-Type", "text/plain; charset=UTF-8");
        request.send(null);

        request.onload = function() {
            callback(request.response);
        }
    }
```

```

    }

    addButton.onclick = function() {
        const email = emailField.value;
        const surname = surnameField.value;
        const phone = phoneField.value;

        let statusString = "";
        let validInfo = true;

        if (!checkValidEmail(email)) {
            statusString += 'Wrong <font color="red">email</font><br>';
            validInfo = false;
        }

        if (!checkValidSurname(surname)) {
            statusString += 'Wrong <font color="green">surname</font><br>';
            validInfo = false;
        }

        if (!checkValidPhone(phone)) {
            statusString += 'Wrong <font color="blue">phone number</font><br>';
            validInfo = false;
        }

        if (!validInfo) {
            statusLabel.innerHTML = "Enter wrong data: <br>" + statusString;
        }
        else {
            const url = "/addUser?email=" + email + "&surname=" + surname + "&phone=" + phone;
            ajaxGet(url, function(stringAnswer) {
                const objectAnswer = JSON.parse(stringAnswer);
                const added = objectAnswer.added;

                statusLabel.innerHTML = added ? `User with mail <font color="red">${email}</font> and phone number <font color="blue">${phone}</font> has been successfully added` :
                    `User with mail <font color="red">${email}</font> and phone number <font color="blue">${phone}</font> already exists`;

            });
        }
    }
}

```

Скрипт для поиска пользователей

```
"use strict";

window.onload = function() {
    const emailField = document.getElementById("email");

    const getButton = document.getElementById("getButton");
    const resultLabel = document.getElementById("result");

    function ajaxGet(urlString, callback) {
        let request = new XMLHttpRequest();

        request.open("GET", urlString, true);
        request.setRequestHeader("Content-Type", "text/plain; charset=UTF-8");
        request.send(null);

        request.onload = function() {
            callback(request.response);
        }
    }

    getButton.onclick = function() {
        const email = emailField.value;
        const url = "/getUser?email=" + email;

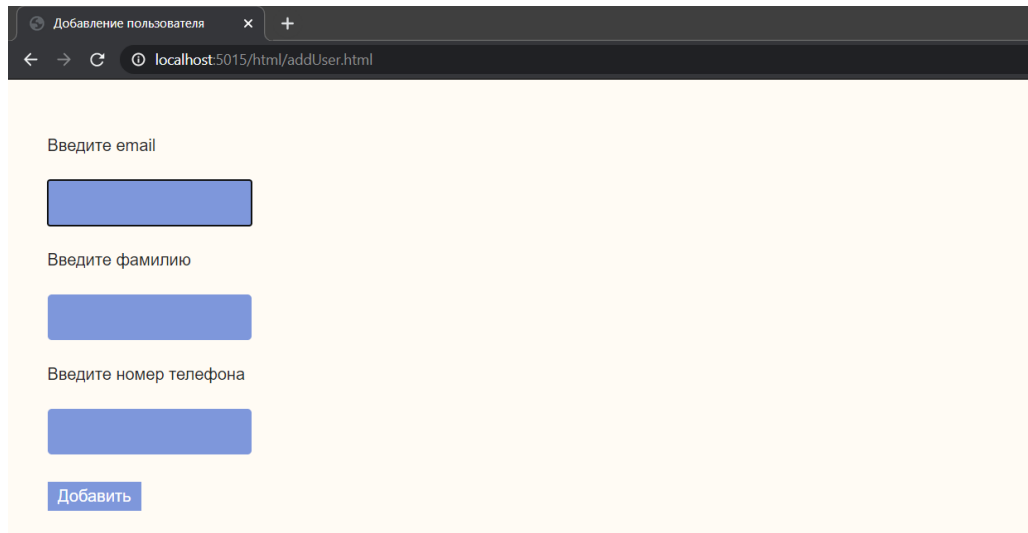
        ajaxGet(url, function(stringAnswer) {
            const objectAnswer = JSON.parse(stringAnswer);

            const found = objectAnswer.found;
            const surname = objectAnswer.surname;
            const phone = objectAnswer.phone;

            resultLabel.innerHTML = found ? `User with email <font color="red">
${email}</font> was found: <br>
                                     Surname - <font color="green">${su
rname}</font>;<br>
                                     Phone   - <font color="blue">${pho
ne}</font>;` :
            `User with email <font color="red">
${email}</font> not found`
        });
    }
}
```


Тесты

```
PS C:\Repositories\bmstu_archEvm\lab_03\task_11> npm start  
  
> task_01@1.0.0 start C:\Repositories\bmstu_archEvm\lab_03\task_11  
> node index.js  
  
Server started on port 5015
```



Добавление пользователя

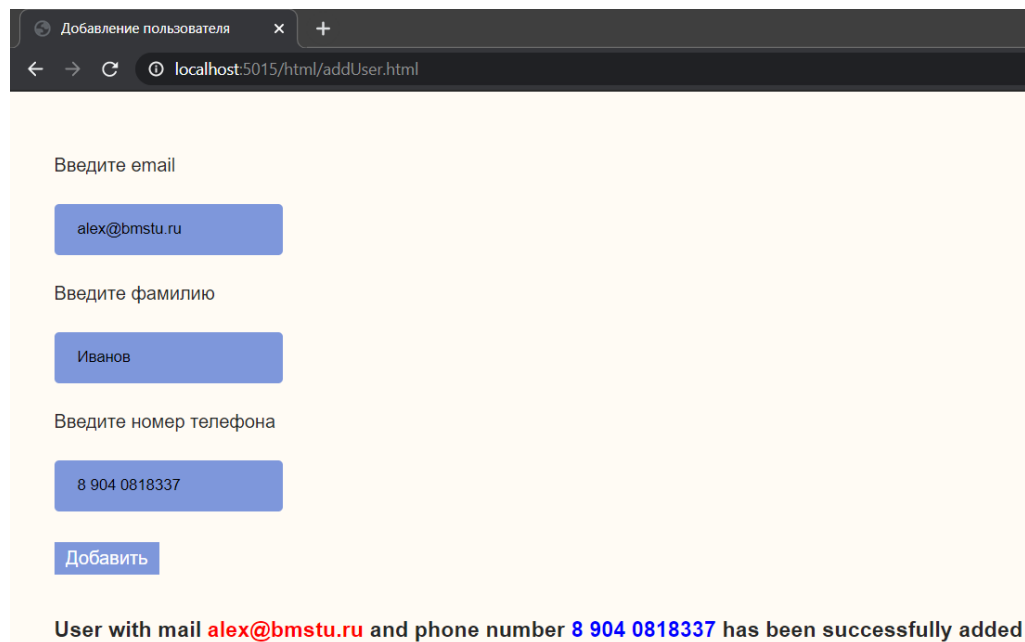
localhost:5015/html/addUser.html

Введите email

Введите фамилию

Введите номер телефона

Добавить



Добавление пользователя

localhost:5015/html/addUser.html

Введите email

alex@bmstu.ru

Введите фамилию

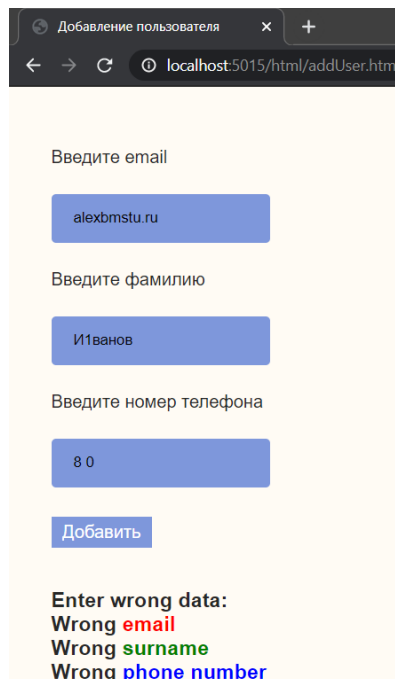
Иванов

Введите номер телефона

8 904 0818337

Добавить

User with mail alex@bmstu.ru and phone number 8 904 0818337 has been successfully added



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.

Листинг оформления для страницы добавления пользователей

```
body {  
    padding: 30px;  
    background: rgb(255, 251, 244);  
    font-family: Geneva, Arial, Helvetica, sans-serif;  
}  
  
.btn-class {  
    padding-left: 10px;  
    padding-right: 10px;  
    padding-top: 5px;  
    padding-bottom: 5px;  
    background: rgb(126, 151, 219);  
    color: white;  
    cursor: pointer;  
    display: inline-block;  
}  
  
input {  
    background-color: rgb(126, 151, 219);  
    color: black;  
    padding: 14px 20px;  
    margin: 8px 0;  
    border: none;  
    border-radius: 4px;  
}
```

```
}

p {
  color: rgb(39, 39, 39);
}

h3 {
  color: rgb(39, 39, 39);
}
```

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.

Листинг класса сервера

```
"use strict";

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

const dataPath = "data/games.txt";

class Server {
  constructor(port = 5015) {
    this.port = port;
    this.app = express();
    this.staticPath = __dirname + "/static";
  }

  start() {
    try {
      this.app.listen(this.port);
      console.log("Server started on port " + this.port);
    }
    catch {
      console.log("Server startup error");
      throw new Error("Server start error");
    }
  }
}
```

```

    this.app.use(this.headerFormation);
    this.app.use(express.static(this.staticPath));
    this.app.set("view engine", "hbs");

    this.app.get("/getGames", this.getGames);
  }

  headerFormation(request, response, next) {
    response.header("Cache-Control", "no-cache, no-store, must-revalidate");
    response.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
    response.header("Access-Control-Allow-Origin", "*");
    next();
  }

  getGames(request, response) {
    const age = request.query.age;

    const gamesData = fs.readFileSync(dataPath, "utf8");
    const gamesStorage = JSON.parse(gamesData);

    const infoObject = {
      description: "Games for " + age,
      games: gamesStorage.filter((game) => {return game.age <= age})
    };

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

let server = new Server(5015);
server.start();

```

Листинг шаблона для вывода списка игр

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Игры</title>
  <link rel="stylesheet" type="text/css" href="/style/style.css" />
</head>
<body>

<h2>
  {{description}}

```

```

</h2>

{{#each games}}
  <div class = "row">
    <ul class = "hr">
      <li>Название: <font color ="green">{{this.name}}</font></li>
      <li>Описание: <font color ="blue">{{this.description}}</font></li>
      <li>Возрастное ограничение: <font color ="red">{{this.age}}+</font></li>
    </ul>
  </div>
</each>

</body>
</html>

```

Тесты

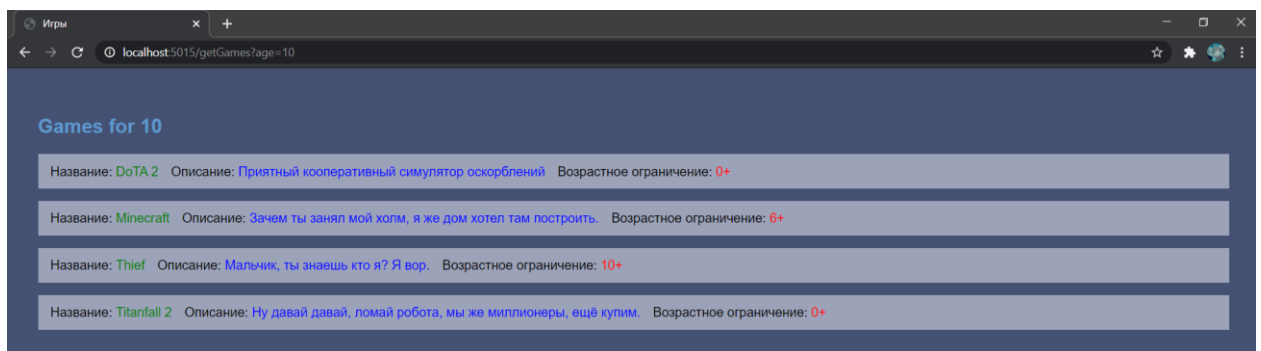
```

PS C:\Repositories\bmstu_archEvm\lab_03\task_21> npm start

> task_02@1.0.0 start C:\Repositories\bmstu_archEvm\lab_03\task_21
> node index.js

Server started on port 5015

```



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 класса сервера

```
"use strict";

const fs = require("fs");

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

const dataPath = "data/users.txt";

class Server {
  constructor(port = 5015) {
    this.port = port;
    this.app = express();
    this.staticPath = __dirname + "/static";
  }

  start() {
    try {
      this.app.listen(this.port);
      console.log("Server started on port " + this.port);
    }
    catch {
      console.log("Server startup error");
      throw new Error("Server start error");
    }

    this.app.use(this.headerFormation);
    this.app.use(express.static(this.staticPath));

    this.app.use(cookieSession({
      name: "session",
      keys: ['hhh', 'qqq', 'vvv'],
      maxAge: 25 * 60 * 60 * 1000
    }));

    this.app.set("view engine", "hbs");
    this.app.set("views", "static/views");

    this.app.get("/api/save", this.saveCookie);
    this.app.get("/api/get", this.getCookie);
    this.app.get("/api/delete", this.deleteCookie);

    this.app.get("/getUser", this.getUser);
    this.app.get("/account", this.getAccount);
    this.app.get("/logIn", this.getAuth);
  }
}
```

```

    headerFormation(request, response, next) {
        response.header("Cache-Control", "no-cache, no-store, must-
revalidate");
        response.header("Access-Control-Allow-Headers", "Origin, X-Requested-
With, Content-Type, Accept");
        next();
    }

    // Cookies
    saveCookie(request, response) {
        const login = request.query.login;
        const password = request.query.password;

        if (!login) {
            return response.end("Login is not set!");
        }

        if (!password) {
            return response.end("Password is not set!");
        }

        request.session.login = login;
        request.session.password = password;

        response.end("Cookie set succesfully!");
    }

    getCookie(request, response) {
        if (!request.session.login || !request.session.password) {
            return response.end(JSON.stringify({"exists" : false}));
        }

        const login = request.session.login;
        const password = request.session.password;

        response.end(JSON.stringify({
            "exists": true,
            "login": login,
            "password": password
        })));
    }

    deleteCookie(request, response) {
        request.session = null;
        response.end(" Cookie deleted succesfully!");
    }

    // Other methods
    getUser(request, response) {

```

```

    const login = request.query.login;
    const password = request.query.password;

    const usersData = fs.readFileSync(dataPath, "utf8");
    const usersStorage = new Map(JSON.parse(usersData));

    if (!usersStorage.has(login) || (usersStorage.has(login) && usersStorage.get(login).password !== password)) {
        response.end(JSON.stringify({found: false}));
    }
    else {
        const value = usersStorage.get(login);
        response.end(JSON.stringify({found: true, hobbie: value.hobbie, age: value.age}));
    }
}

getAccount(request, response) {
    const login = request.query.login;
    const hobbie = request.query.hobbie;
    const age = request.query.age;

    const infoObject = {
        login: login,
        hobbie: hobbie,
        age: age
    };

    response.render("account.hbs", infoObject);
}

getAuth(request, response) {
    response.end(fs.readFileSync("static/html/log_in.html", "utf8"));
}
}

let server = new Server(5015);
server.start();

```


Листинг HTML страницы для

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Authorization</title>
  <link rel="stylesheet" type="text/css" href="/style/style.css" />
</head>
<body>
  <h1>Authorization</h1>

  <p>Login</p>
  <input id="login" type="text" spellcheck="false" autocomplete="off">

  <br>
  <br>

  <p>Password</p>
  <input id="password" type="text" spellcheck="false" autocomplete="off">

  <br>
  <br>

  <div id="get_btn" class="btn-class">Log in</div>

  <br>
  <br>

  <h5 id="status"></h5>
  <script src="/scripts/script.js"></script>
</body>
</html>
```

Листинг подключаемого скрипта

```
"use strict";

window.onload = function() {
  const loginField = document.getElementById("login");
  const passwordField = document.getElementById("password");

  const button = document.getElementById("get_btn");
  const status = document.getElementById("status");

  function ajaxGet(urlString, callback) {
    let request = new XMLHttpRequest();
```

```

request.open("GET", urlString, true);
request.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");
request.send(null);

request.onload = function() {
    callback(request.response);
};

function getUserFromData(login, password) {
    const url = `/getUser?login=${login}&password=${password}`;
    let found;

    ajaxGet(url, function(stringAnswer) {
        const objectAnswer = JSON.parse(stringAnswer);
        found = objectAnswer.found;

        if (!found) {
            status.innerHTML = "Login or password incorrect, please try again";
        }
        else {
            const hobby = objectAnswer.hobby;
            const age = objectAnswer.age;

            const accountUrl = `/account?login=${login}&hobby=${hobby}&age=${age}`;
            window.open(accountUrl);
        }
    });

    return found;
}

function authByCookies(stringAnswer) {
    console.log(stringAnswer);
    const objectAnswer = JSON.parse(stringAnswer);

    if (objectAnswer.exists) {
        const login = objectAnswer.login;
        const password = objectAnswer.password;

        getUserFromData(login, password);
    }
}

let url = "/api/delete";
ajaxGet(url, function(stringAnswer) {});

```

```

url = "/api/get";
ajaxGet(url, authByCookies);

button.onclick = function() {
    const login = loginField.value;
    const password = passwordField.value;

    const found = getUserFromData(login, password);

    if (!found) {
        url = `/api/save?login=${login}&password=${password}`;
        ajaxGet(url, function(stringAnswer){});
    }
}
}

```

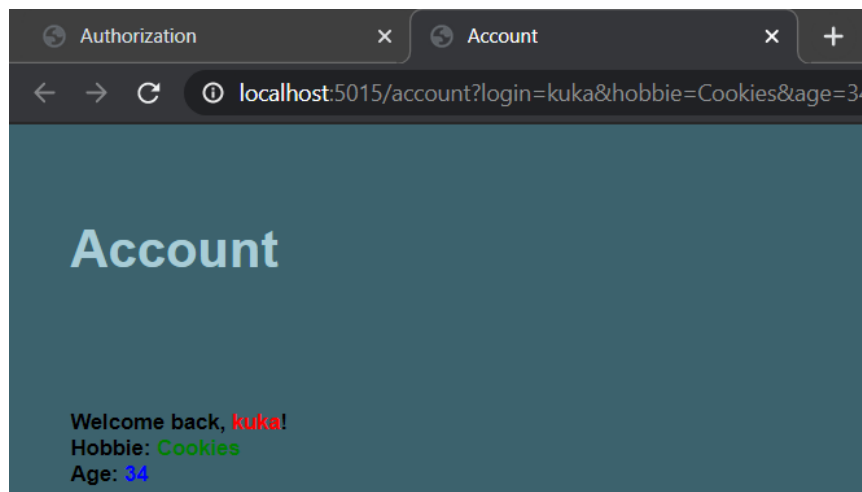
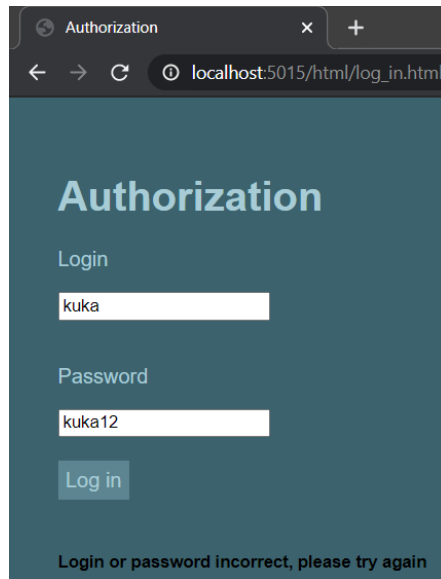
Листинг шаблона для вывода аккаунта

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Account</title>
    <link rel="stylesheet" type="text/css" href="/style/style.css" />
</head>
<body>
    <h1>Account</h1>
    <br>
    <br>
    <h5 id="result_label">Welcome back, <font color="red">{{login}}</font>!<br>
        Hobbie: <font color="green">{{hobbie}}</font><br>
        Age: <font color="blue">{{age}}</font><br></h5>
</body>
</html>

```

Тесты



Вывод: в ходе выполнения лабораторной работы были изучены и реализованы получение статических файлов, GET и POST запросы в AJAX, шаблонизаторы и cookie.