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 № 5-6

Тема: Знакомство с AJAX запросами, шаблонизаторами

Студент Унтилова А.О.

Группа ИУ7-56

Оценка (баллы) _____

Преподаватель Попов А.Ю.

Москва.
2020 г.

Цель: Ознакомиться с методом получения статических файлов, AJAX запросами, POST-запросами; научиться работать с шаблонизаторами, сессиями в NodeJS, изучить основы использования CSS.

Лабораторная работа №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.

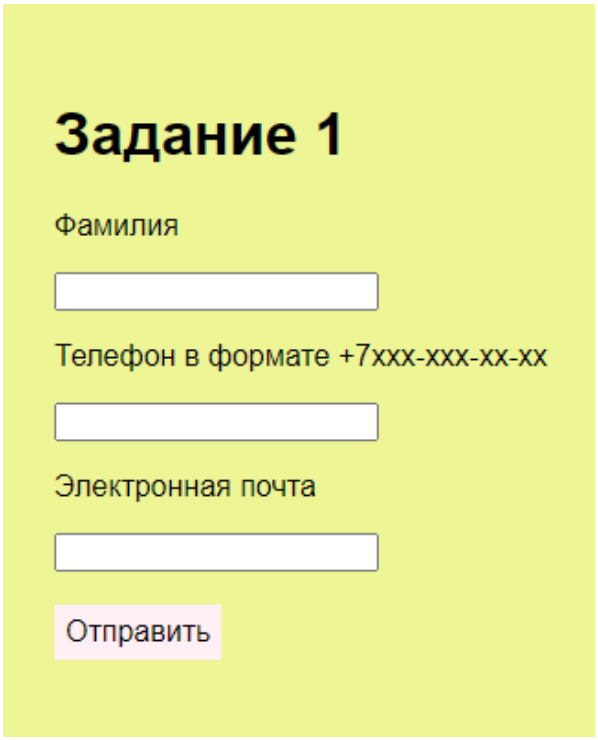
```
"use strict";
const filename = 'file.txt';

// импортируем библиотеку
const express = require("express");

// запускаем сервер
const app = express();
const port = 4000;
app.listen(port);
console.log(`Server on port ${port}`);

// отправка статических файлов
const way = __dirname + "/static_n";
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) => {  //chunk - либо Buffer
        body.push(chunk);
    }).on('end', () => {
        body = Buffer.concat(body).toString();
        console.log(body);
        callback(body);
    });
}

function check_file(file, string){
    let obj_str = JSON.parse(string);
    let arr = [];
    const fs = require("fs");
    let file_str = fs.readFileSync(filename, "utf8");
    if (file_str.length != 0)
    {
        arr = JSON.parse(file_str);
        for (let i = 0; i < arr.length; i++){
            if ((arr[i]["phone"] === obj_str["phone"]) || (arr[i]["email"] === obj_str["
email"]))
                return false;
        }
    }
    arr.push(obj_str);
    console.log(arr);
    fs.writeFileSync(filename, JSON.stringify(arr));
    return true;
}

// приём POST запроса
app.post("/save/info", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const surname = obj["surname"];
        const tel = obj["tel"];
        const email = obj["email"];
        const contentString = `{"surname" : "${surname}", "phone" : "${tel}", "email" :
"${email}"}`;
        let check = check_file(filename, contentString);
        let message = "Save not OK"
        if (check)
            message = "Save OK"
        response.end(JSON.stringify({  // отправка ответа
            result: message
        }));
    });
});
```

Page.html:

```
<body>
```

```
    <h1>Задание 1</h1>
 <p>Фамилия</p>
    <input id="field-surname" type="text" spellcheck="false" autocomplete="off">

    <p>Телефон в формате +7xxx-xxx-xx-
xx</p>

    <input id="field-tel" type="tel" pattern="+7[0-9]{3}-[0-9]{3}-[0-9]{2}-[0-
9]{2}" spellcheck="false" autocomplete="off"></p>

    <p>Электронная почта</p>
    <input id="field-email" type="email" spellcheck="false" autocomplete="off">

    <br>
    <br>

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

    <br>
    <br>

    <h1 id="result-label"></h1>

    <script src="/code.js"></script>
    <script src="bootstrap-formhelpers-phone.js"></script>
</body>
```



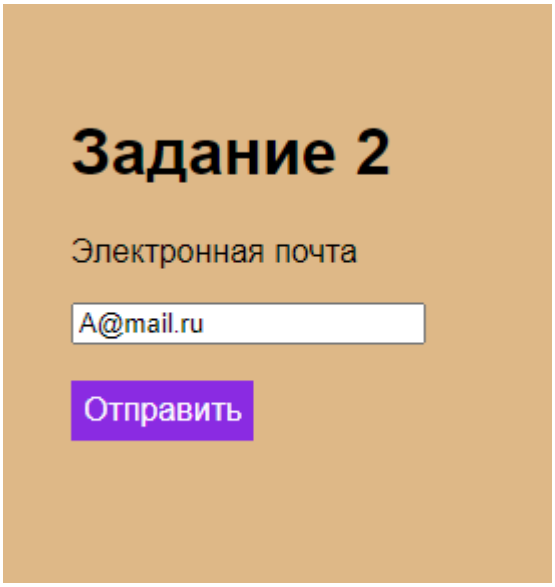
Задание 2

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.

```
"use strict";

// импортируем библиотеку
const express = require("express");
const filename = 'file.txt';
const fs = require("fs");

// запускаем сервер
const app = express();
const port = 4000;
app.listen(port);
console.log(`Server on port ${port}`);

// отправка статических файлов
const way = __dirname + "/static_get";
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", "*");    // Control -
    next();
});

function find_email(filename, email){
    let arr = [];
    const fs = require("fs");
    let file_str = fs.readFileSync(filename, "utf8");
    arr = JSON.parse(file_str);

    for (let i = 0; i < arr.length; i++){
        if ((arr[i]["email"] === email))
            return JSON.stringify(arr[i]);
    }
    return JSON.stringify("Пользователя с такой электронной почтой нет.");
}

app.get("/send", function(request, response) {
    const email = request.query.email;
    let js_string = find_email(filename, email);
    response.end(js_string);
});
```

Page2.html:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Задание 2</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
    <h1>Задание 2</h1>

    <p>Электронная почта</p>
    <input id="field-email" type="email" spellcheck="false" autocomplete="off">

    <br>
    <br>

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

    <br>
    <br>

    <h1 id="result-label"></h1>

    <script src="/code.js"></script>
</body>
</html>
```



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.

Для page2.html:

```
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;
}
```

Для page.html:

```
body {
    padding: 30px;
    background: rosybrown;
    font-family: Geneva, Arial, Helvetica, sans-serif;
}
.btn-class {
    padding: 6px;
    background: lavenderblush;
    color:black;
    cursor: pointer;
    display: inline-block;
}
```

Лабораторная работа №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

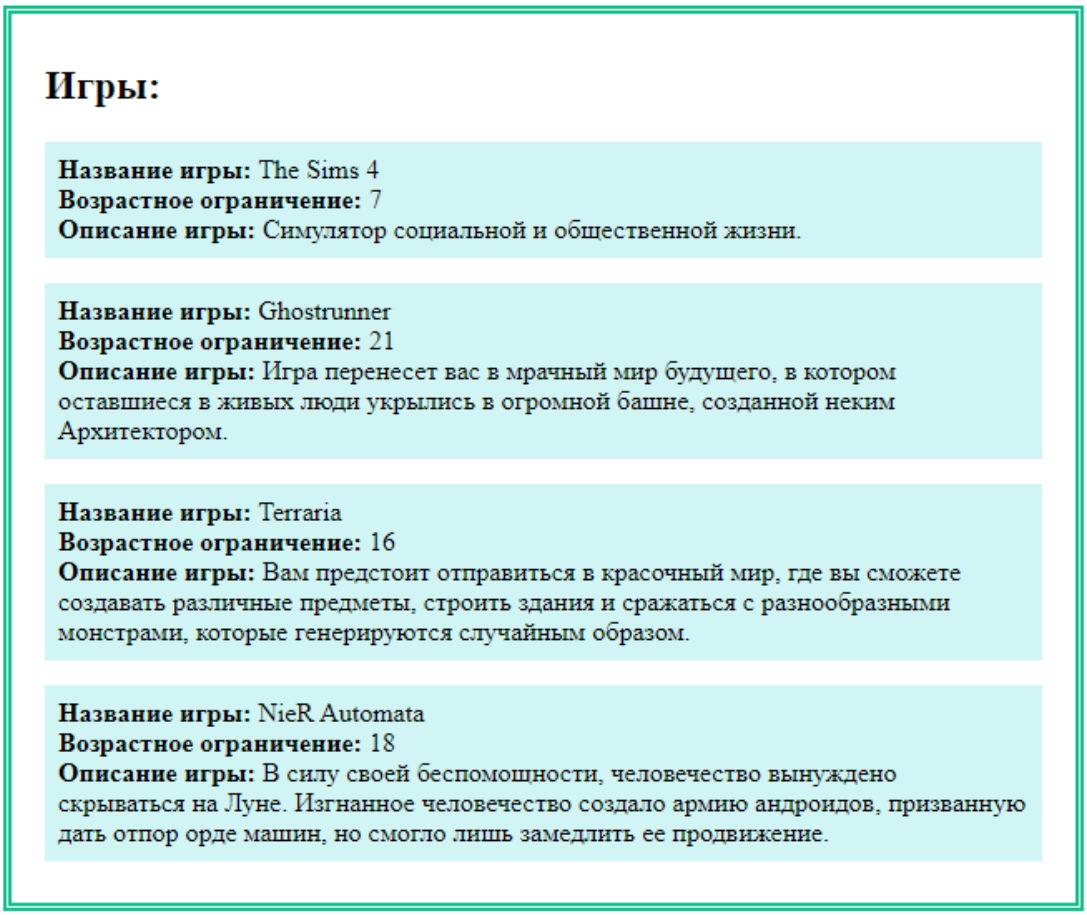
ограничения). Создать страницу с помощью шаблонизатора. В **url** передаётся параметр возраст (целое число). Необходимо отображать на этой странице только те игры, у которых возрастное ограничение меньше, чем переданное в **url** значение.

```
"use strict";

const games_arr = [{'name' : 'The Sims 4', 'age' : 7, 'description' : 'Симулятор социаль
ной и общественной жизни.'},
{'name' : 'Ghostrunner', 'age' : 21, 'description' : 'Игра перенесет вас в мрачный мир б
удущего, в котором оставшиеся в живых люди укрылись в огромной башне, созданной неким Ар
хитектором.' },
{'name' : 'Terraria', 'age' : 16, 'description' : 'Вам предстоит отправиться в красочный
 мир, где вы сможете создавать различные предметы, строить здания и сражаться с разнообр
азными монстрами, которые генерируются случайным образом.'},
{'name' : 'NieR Automata', 'age' : 18, 'description' : 'В силу своей беспомощности, чело
вечество вынуждено скрываться на Луне. Изгнанное человечество создало армию андроидов, п
ризванную дать отпор орде машин, но смогло лишь замедлить ее продвижение.'}]

// импорт библиотеки
const express = require("express");

// запускаем сервер
const app = express();
const port = 4000;
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();
});

// выдача страницы с массивом игр
app.get("/page/games", function(request, response) {
    const max_age = request.query.age;
    let new_arr = [];
    for (let i = 0; i < games_arr.length; i++){
        if (games_arr[i]['age'] < max_age){
            new_arr.push(games_arr[i]);
        }
    }
    const infoObject = {
        descriptionValue: "Игры: ",
        array:new_arr
    };
```

```
// Генерация и отдача представления осуществляется с помощью метода render(), который пр
инимает два параметра:
// шаблон;данные для шаблона в виде объекта (если необходимо).
    response.render("pageGames.hbs", infoObject);
});
```

PageGames.hbs:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Игры</title>
    <style>
    body{
        {{!-- text-indent: 30px; --}}   // делает отступ у абзаца
        border: 5px double #0cc08a;
        padding: 10px 20px 10px 20px;
        margin: 20px auto;
        width: 600px;
    }
  </style>
</head>
<body>

<h2>
    {{descriptionValue}}
</h2>

{{#each array}}
    {{!-- margin-
bottom устанавливает величину внешнего отступа от нижнего края элемента --}}
    {{!-- padding - Устанавливает значение полей вокруг содержимого элемента --}}
    <div style="background: rgb(209, 245, 245); margin-bottom: 15px; padding: 8px;">
        <b>Название игры:</b> {{this.name}}
        <br>
        <b>Возрастное ограничение:</b> {{this.age}}
        <br>
        <b>Описание игры:</b> {{this.description}}
    </div>
{{/each}}

</body>
```



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.

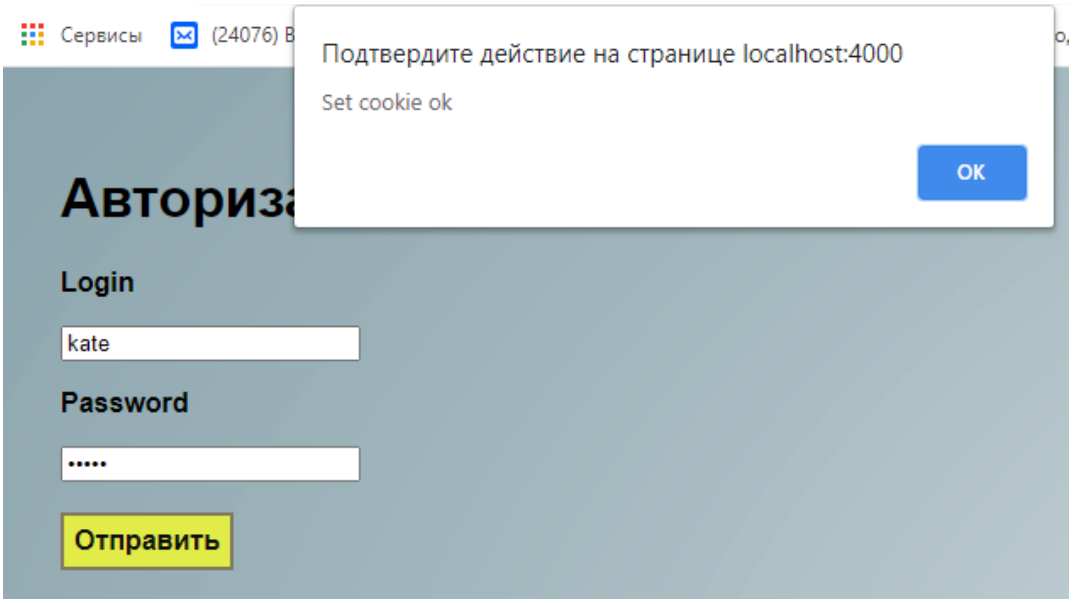
```
"use strict";

const data = [{'login' : 'Kate', 'password' : 'hello', 'hobby' : 'cooking', 'age' : 25},
{'login' : 'Tim', 'password' : 'qwerty', 'hobby' : 'football', 'age' : 10},
{'login' : 'Sam', 'password' : '1234', 'hobby' : 'reading', 'age' : 16}]

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

// запускаем сервер
const app = express();
const port = 4000;
app.listen(port);
console.log(`Server on port ${port}`);
```

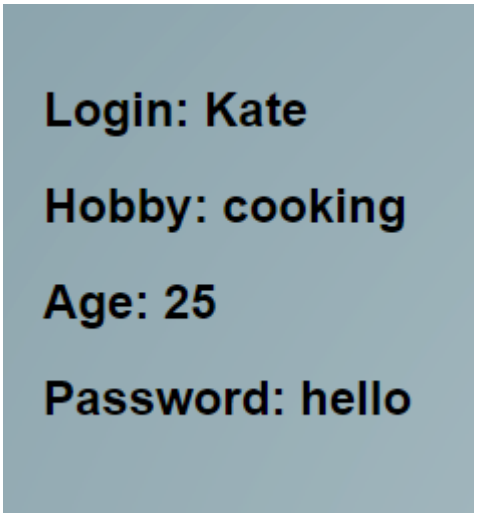
```
// работа с сессией
app.use(cookieSession({
    // Имя устанавливаемого файла cookie, по умолчанию session.

    name: 'session',
    keys: ['hhh', 'qqq', 'vvv'],
    // определяет время жизни файла в секундах;
    maxAge: 24 * 60 * 60 * 1000 * 365
}));

const way = __dirname + "/static";
app.use(express.static(way));

function get_user(array, login, password){
    for (let i = 0; i < array.length; i++){
        if (array[i]['login'] === login && array[i]['password'] === password)
        {
            let user = {};
            // копирует из исходных объектов в целевой объект только перечисляемые и соб
ственные свойства
            Object.assign(user, array[i]);
            user['status'] = true;
            return user;
        }
    }
    return {'status' : false};
}

// 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("/api/save", function(request, response) {
    // получаем параметры запроса
    const login = request.query.login;
    const password = request.query.password;
    // контролируем существование параметров
    if(!login) return response.end("Login not set");
    if(!password) return response.end("Password not set");
    // выставляем cookie
    request.session.login = login;
    request.session.password = password;
    // отправляем ответ об успехе операции
    response.end("Set cookie ok");
});

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

```
    // контролируем существование cookie
    if(!request.session.login) return response.end("Not exists");
    if(!request.session.password) return response.end("Not exists");
    let user = get_user(data, request.session.login, request.session.password);
    response.end(JSON.stringify(user));
});
```

Login.html:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Авторизация</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
    <h1>Авторизация</h1>
    <p><b>Login</b></p>
    <input id="field-login" type="text" spellcheck="false" autocomplete="off">
    <p><b>Password</b></p>
    <input id="field-password" type="password" spellcheck="false" autocomplete="off">
    <br>
    <br>
    <div id="send-btn" class="btn-class"><b>Отправить</b></div>
    <br>
    <br>
    <h1 id="result-label"></h1>
    <script src="/code.js"></script>
</body>
</html>
```





Вывод: Во время выполнения данной лабораторной работы я ознакомилась с методом получения статических файлов, AJAX запросами, POST-запросами; научилась работать с шаблонизаторами, сессиями в NodeJS, изучила основы использования CSS.