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

AJAX GET/POST запросы.

Студент Жигалкин Д.Р

Группа ИУ7-55Б

Оценка (баллы) _____

Москва.
2020 г.

Цель: Ознакомиться с получением статических файлов, ajax get/post запросами. Научиться работать с шаблонизатором и сессиями в NodeJS.

Task_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.

```
<!DOCTYPE html>
<html>
<head>
    <title>Регистрация</title>
    <link rel="stylesheet" href="/style.css">
    <meta charset="utf-8" />
</head>
<body>
    <h1>Введите данные</h1>
    <form name="registerForm">
        <label>email</label></br>
        <input type="text" name="userEmail" /></br></br>
        <label>subname</label></br>
        <input type="text" name="userSubname" /></br></br>
        <label>phone</label></br>
        <input type="text" name="userPhone" /></br></br>

        <button type="submit" id="submit">Отправить</button>
    </form>

    <script src="/code5.1.js"></script>
</body>
<html>
```

```
document.getElementById("submit").addEventListener("click", function (e) {
    e.preventDefault();

    // получаем данные формы
    let registerForm = document.forms["registerForm"];
```

```
    let userEmail = registerForm.elements["userEmail"].value;
    let userSubname = registerForm.elements["userSubname"].value;
    let userPhone = registerForm.elements["userPhone"].value;

    // сериализуем данные в json
    let user = JSON.stringify({email: userEmail, subname: userSubname, phone: use
rPhone});

    let request = new XMLHttpRequest();
    // посылаем запрос на адрес "/save/information/"
     request.open("POST", "/save/information", true);
     request.setRequestHeader("Content-Type", "application/json");
     request.send(user);

     request.addEventListener("load", function () {
        // получаем и парсим ответ сервера
         let receivedStatus = JSON.parse(request.response);
         alert(receivedStatus);   // смотрим ответ сервера
     });
 });
```

```
body {
    padding: 30px;
    background: burlywood;
    font-family: Geneva, Arial, Helvetica, sans-serif;
}
```

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

const app = express();
var port = 3000;
app.listen(port);
console.log(`Server on port ${port}`);

// создаем парсер для данных в формате json
const jsonParser = express.json();

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

    if (request.body.email == "" || request.body.subname == "" || request.body.ph
one == "")
    {
        response.json(JSON.stringify({result: "Don't save (bad input)"}));
    }
    else
```

```
    {
        checkRepetitions(request, response);
    }
});

function checkRepetitions(request, response)
{
    let fileContent = fs.readFileSync("file.txt", "utf8");
    if (fileContent.indexOf(request.body.email) == -
1 && fileContent.indexOf(request.body.phone) == -1)
    {
        fs.appendFileSync("file.txt", "\n" + JSON.stringify(request.body));
        response.json(JSON.stringify({result: "Save content ok"}));
    }
    else
    {
        response.json(JSON.stringify({result: "Don't save content (repetitions)"}
));
    }
}

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

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

    response.sendFile(__dirname + "/static/task5.1.html");
});
```

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.

```
<!DOCTYPE html>
<html>
<head>
    <title>Поиск информации</title>
    <link rel="stylesheet" href="/style.css">
    <meta charset="utf-8" />
</head>
<body>
    <h1>Введите данные</h1>
```

```
    <form name="checkForm">
        <label>email</label></br>
        <input type="text" name="userEmail" /></br></br>

        <button type="submit" id="submit">Отправить</button>
    </form>
    <script src="/code5.2.js"></script>

</body>
<html>
```

```
document.getElementById("submit").addEventListener("click", function (e) {
    e.preventDefault();

    // получаем данные формы
    let checkForm = document.forms["checkForm"];
    let userEmail = checkForm.elements["userEmail"].value;

    // сериализуем данные в json
    let user = JSON.stringify({email: userEmail});

    let request = new XMLHttpRequest();
    // посылаем запрос на адрес "/check/information/"
     request.open("GET", '/check/information?email=' + userEmail, true);
     request.setRequestHeader("Content-Type", "text/plain");
     request.send(user);

     request.addEventListener("load", function () {
        // получаем и парсим ответ сервера
         let receivedStatus = JSON.parse(request.response);
         alert(receivedStatus);   // смотрим ответ сервера
     });
 });
```

```
body {
    padding: 30px;
    background: burlywood;
    font-family: Geneva, Arial, Helvetica, sans-serif;
}
```

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

const app = express();
var port = 3000;
```

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

// создаем парсер для данных в формате json
const jsonParser = express.json();

app.get("/check/information", jsonParser, function (request, response) {

    if (request.body.email == "")
    {
        response.json(JSON.stringify({result: "Don't check (bad input)"}));
    }
    else
    {
        checkInformation(request, response);
    }
});

function checkInformation(request, response)
{
    let fileContent = fs.readFileSync("file.txt", "utf8");
    let indexStartcontent = fileContent.indexOf('"email":' + '"' + request.query.
email + '"');

    if (indexStartcontent != -1)
    {
        let indexEndcontent = fileContent.indexOf("\n", indexStartcontent);
        response.json(JSON.stringify(fileContent.slice(indexStartcontent -
 1, indexEndcontent)));
    }
    else
    {
        response.json(JSON.stringify({result: "Information not found"}));
    }
}

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

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

    response.sendFile(__dirname + "/static/task5.2.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.

Оформление с помощью css сделано выше

Task_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.

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

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

{{#each gamesArray}}

    <div style="background: yellow; margin-bottom: 15px; padding: 8px;">
        Название: {{this.name}}
        <br>
        Описание: {{this.description}}
        <br>
        Возрастное ограничение: {{this.age}}
    </div>

{{/each}}

</body>
</html>
```

```
"use strict";

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

// запускаем сервер
const app = express();
```

```
const port = 5000;
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();
});

const infoObject = {
    descriptionValue: "Games list (taking into account age)",
    gamesArray: [
        {name: "GTA", description: "cool", age: 15},
        {name: "kenshi", description: "very good strategy", age: 6},
        {name: "happy farm", description: "so-so", age: 1},
        {name: "TLoUS 2", description: "cool", age: 20}
    ]
};

// выдача страницы с массивом учеников
app.get("/games", function(request, response) {
    let currentAge = parseInt(request.query.age);
    let resultObject = { descriptionValue: infoObject.descriptionValue, gamesArra
y: [] };
    resultObject.gamesArray = infoObject.gamesArray.filter(item => item.age < cur
rentAge);

    response.render("pageGames.hbs", resultObject);
});
```

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";

// импортируем библиотеки
const express = require("express");
```

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

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

// работа с сессией
app.use(cookieSession({
    name: 'session',
    keys: ['hhh', 'qqq', 'vvv'],
    maxAge: 24 * 60 * 60 * 1000 * 365
}));

const users = [
    {login: "mqa", password: "AJHjghbj", hobby: "a1", age: 5},
    {login: "qqqq", password: "Znjb5", hobby: "a2", age: 16},
    {login: "w", password: "123456", hobby: "a3", age: 15},
    {login: "v", password: "zxcvb", hobby: "a4", age: 50},
    {login: "s", password: "asdfgh", hobby: "a5", age: 19},
    {login: "gg", password: "q12w3e", hobby: "a6", age: 24},
];

// сохранить cookie
app.get("/auth/user", 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");

    const user = users.find(s => s.login === login && s.password === password);

    if (!user)
    {
        response.statusCode = 400;
        return response.end("inccorect login or password");
    }
    else
    {
        // выставляем cookie
        request.session.login = login;
        request.session.password = password;
        return response.end("ok auth");
    }
```

```
});

app.get("/user", function(request, response) {
    if (!request.session.login || !request.session.password)
    {
        response.statusCode = 401;
        return response.end("not auth");
    }

    // отправляем ответ с содержимым cookie
    const login = request.session.login;
    const password = request.session.password;

    const user = users.find(s => s.login === login && s.password == password);
    if (!user)
    {
        response.statusCode = 401;
        return response.end("not auth");
    }

    return response.end(JSON.stringify(user));
});

// удалить все cookie
app.get("/delete", function(request, response) {
    request.session = null;
    response.end("Delete cookie ok");
});
```

Вывод: Я ознакомился с получением статических файлов, ajax get/post запросами. Научился работать с шаблонизатором и сессиями в NodeJS.