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»

Отчет по лабораторной работе №3 по курсу «Архитектура ЭВМ»

Тема Работа с шаблонизаторами и сессиями

Студент Романов А.В.

Группа ИУ7-53Б

Преподаватели Попов А.Ю.

Москва — 2020 г.

Отчет по разделу #5

Цель работы

Ознакомиться с разработкой RESTful сервиса с использованием AJAX-запросов.

Задания 1 - 3

Условие

- 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.
- 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.
- 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.

Код программы

Язык: **Javascript**

src/app.js

```
"use strict";

const express = require("express");
const path = require("path");
const body_parser = require("body-parser");

const post = require("./controllers").post;
const router = require("./router");

const port = 5015;
const index_page = path.join(__dirname, "..", "public", "index.html")

const app = express();

app.use(body_parser.json());
app.use((req, _, next) => {
    console.log(req.url);

    if (req.method === "POST") {
        console.log(req.body);
    }
    else if (req.method === "GET") {
        console.log(req.query);
    }

    next();
})

app.use(express.static(path.join(__dirname, "..", "public")));

app.use(router);

app.use((req, res, next) => {
    if (req.method === "POST") {
        // обработка всех подряд POST-запросов
        res.send(`Address: ${req.url}, \nBody of POST: ${JSON.stringify(req.body)}`);
    } else {
        next();
    }
});

app.use((req, res) => {
    res.sendFile(index_page);
});

app.listen(port);

console.log(`Server running. Port: ${port}`);
```

src/controllers.js

```
"use strict";

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

const DATABASE = path.join(__dirname, "..", "data", "database.txt");
const SECOND = "index_second.html"
const FOLDER = "public"

function get_all_users() {
    return JSON.parse(fs.readFileSync(DATABASE));
}

function update_database(database, user) {
    database.push(user);
    fs.writeFileSync(DATABASE, JSON.stringify(database));
}

module.exports.add_user = (req, res) => {
    let user = req.body;
    let users = get_all_users();

    if (users.find(x => x.email === user.email || x.phone === user.phone)) {
        res.status(400).send({ result: "Пользователь уже существует в базе данных." });
        return;
    }

    update_database(users, user);
    res.status(200).send({ result: "Пользователь был добавлен" });
};

module.exports.find_user = (req, res) => {
    const email = req.query.email;

    if (!email) {
        res.status(400).send({ result: "FAIL", message: "Некорректный ввод." });
        return;
    }

    const user = get_all_users().find(x => x.email === email);

    if (user) {
        res.status(200).send({ result: "OK", message: "Пользователь найден!", user: user });
    } else {
        res.status(404).send({ result: "FAIL", message: "Такого пользователя нет в базе данных." });
    }
}

module.exports.second = (_, res) => {
    res.sendFile(path.join(__dirname, "..", FOLDER, SECOND));
}
```

src/router.js

```
"use strict";

const router = require("express").Router();
const controllers = require("./controllers");

router.get("/second", controllers.second);
router.post("/user", controllers.add_user);
router.get("/search", controllers.find_user)

module.exports = router;
```

public/index_01.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>NodeLab05</title>
        <link rel="stylesheet" href="style.css">
        <script defer src="scripts/app.js"></script>
    </head>
    <body>
        <div style="display: flex">
            <div>
                <label for="email"></label>
                <input placeholder="Введите email" id="email-input" type="email" name="email">
                <br>
                <label for="surname"></label>
                <input placeholder="Введите фамилию" id="surname-input" type="text" name="surname">
                <br>
                <label for="phone"></label>
                <input placeholder="Введите номер телефона" id="phone-input" type="text" name="phone">
                <br>
                <button>Добавить</button>
            </div>

            <div>
                <p id="result"></p>
            </div>

        </div>

        <a href="/second">Искать пользователя</a>

    </body>
</html>
```

public/index_02.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>NodeLab05</title>
        <link rel="stylesheet" href="style.css">
        <script defer src="scripts/app2.js"></script>
    </head>
    <body>
        <div style="display: flex">
            <div>
                <label for="email"></label>
                <input placeholder="Введите email" id="email-input" type="email" name="email">
                <br>
                <button>Искать</button>
            </div>

            <div>
                <p id="result"></p>
            </div>

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

public/style.css

```
html {
    margin: 0;
    padding: 0;
}

body {
    padding-top: 2em !important;
    margin: 0;
    padding: 0;
    background-color: #808080;
    color: #808080 !important;
}

label {
    font-size: 1em;
    font-weight: bold;
}

p {
    color: white;
    font-size: 1.5em;
    text-align: center;
    white-space: pre;
}

input {
    justify-self: left;
    width: 80%;
    background-color: #000000;
    color: #F0FFF0;
    padding: 1em;
    font-size: 1em;
}

div, input, button {
    display: block;
    margin: 0 auto;
}

div {
    border-radius: 2em;
    color: #F0FFF0;
    background-color: #000000;
    width: 60%;
    padding: 2em;
    border: 1px solid #F0FFF0;
}

p {
    color: #808080;
}

button {
    font-size: 1em;
    cursor: pointer;
    margin-top: 1em;
    width: 60%;
    color: #F0FFF0;
    background: #000000;
    border-radius: 2em;
}

h1 {
    margin-top: 2em;
```

```
    margin-top: 2em;
    margin-bottom: 1em;
    width: 70%;
    margin: 0 auto;
    text-align: center;
}

form {
    text-align: left;
    margin-top: 1em;
}

a {
    display: block;
    width: 60%;
    text-align: center;
    text-decoration: none;
    color: #F0FFF0;
    font-size: 1.2em;
    border-radius: 1em;
    margin: 0 auto;
    margin-top: 2em;
    padding: 0.4em;
    background-color: #000000;
    border: 1px solid #F0FFF0;
}
```

scripts/app_01.js

```
"use strict";

const post = async (url, body) => {
    return fetch(url, {
        method: "POST",
        headers: {
            "Content-Type": "application/json;charset=utf-8"
        },
        body
    })
        .then(response => response.json())
}

const get_content = (res) => {
    return JSON.stringify(res.result);
}

const send = () => {
    let surname = document.querySelector("#surname-input");
    let email = document.querySelector("#email-input");
    let phone = document.querySelector("#phone-input");
    let res = document.querySelector("#result");

    if (email && surname && phone && result) {
        email = email.value;
        surname = surname.value;
        phone = phone.value;

        const str_body = JSON.stringify({ email, surname, phone });
        post("/user", str_body)
            .then(result => {
                res.textContent = get_content(result);
        });
    }
}

const btn = document.querySelector("button");
btn.addEventListener("click", send);
```

scripts/app_02.js

```
"use strict";

const get = async (url, body) => {
    url = Object.keys(body).reduce((acc, p) => acc + `${p}=${encodeURIComponent(body[p])}&`, url + `?`);

    return fetch(url, {
        method: "GET",
        headers: {
            "Content-Type": "application/json;charset=utf-8"

        },
    })
        .then(response => response.json())
}

const get_content = (res) => {
    if (res.result === "FAIL") {
        return res.message;
    }

    return res.message + Object.keys(res.user).reduce((acc, field) => acc + `${field}: ${res.user[field]}\n`, "\n");
}

const get_info = () => {
    let email = document.querySelector("#email-input");
    const label = document.querySelector("#result");

    if (email && label) {
        email = email.value;

        get("/search", { email })
            .then(result => {
                label.textContent = get_content(result);
        });
    }
}

const btn = document.querySelector("button");
btn.addEventListener("click", get_info);
```

Результаты тестирования

Все тесты были пройдены успешно.

Вывод

В результате работы был разработан **RESTful** сервис с применением **AJAX**-запросов, выяснены плюсы и минусы использования асинхронных запросов.

Отчет по разделу #6

Цель работы

Ознакомиться с работой шаблонизаторов и управлением сессиями в **Node.js**.

Задания 1-2

- 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.
- 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.

Код программы

Язык: **Javascript**

src/app.js

```
"use strict";

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

const body_parser = require("body-parser");

const controllers = require("./controllers");
const router = require("./router");

const PAGES_FOLDER = path.join(__dirname, "..", "public");
const VIEWS_FOLDER = path.join(__dirname, "..", "views");
const MAX_AGE = 60 * 1000;

const port = process.env.PORT | 3000;

const app = express();

app.set("view engine", "ejs");
app.set("views", VIEWS_FOLDER)

app.use(cookieSession({
    name: 'session',
    keys: ['a', 'b', 'c'],
    maxAge: MAX_AGE
}));

app.use(controllers.set_headers);

app.use(controllers.logger);
app.use(express.static(PAGES_FOLDER));
app.use(body_parser.urlencoded({ extended: true }));

app.use(router);
app.listen(port);

console.log(`Сервер запущен. Порт ${port}`);
```

src/router.js

```
"use strict";

const express = require("express");
const controllers = require("./controllers");

const router = express.Router();

router.get("/games", controllers.render_games)
router.get("/personal", controllers.render_personal);
router.post("/login", controllers.login);
router.get("/exit", controllers.exit);

router.get("/", controllers.default_controller);

module.exports = router;
```

src/controllers.js

```
"use strict";
```

```
const PATH_TO_MAIN_PAGE = "index.ejs";
const GAMES_FILE = "games.ejs";
const PERSONAL_FILE = "personal.ejs";

const path = require("path");
const db_obj = require("./db_obj");

module.exports.default_controller = (_, res) => res.render(PATH_TO_MAIN_PAGE);

module.exports.set_headers = (_, 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();
}

module.exports.render_games = (req, res) => {
    let age, games = [], error = null;

    try {
        age = parseInt(req.query.age);
        games = db_obj.games.filter(game => game.age <= age);
    } catch(_) {
        console.log(err);
    }

    if (games.length) {
        error = "По вашему запросу нет игр";
    }

    res.render(GAMES_FILE, { games, error });
}

module.exports.logger = (req, _, next) => {
    if (req.method === "POST")
        console.log(`body:\n${JSON.stringify(req.body, null, 4)}`);
    else if (req.method === "GET")
        console.log(`query params:\n${JSON.stringify(req.query, null, 4)}`);

    next();
}

const render_personal = (req, res) => {
    let metainfo = { isAuth: false };

    const login = req.session.login;
    const password = req.session.password;

    if (login && password) {
        metainfo.isAuth = true;
        metainfo.user = db_obj.users.find(elem => elem.login === login && elem.password === password);
    }

    if (req.error)
        metainfo.error = req.error;

    if (!metainfo.error)
        metainfo.error = false;

    res.render(PERSONAL_FILE, metainfo);
}

module.exports.login = (req, res) => {
    const login = req.body.login;
    const password = req.body.password;
```

```
    const password = req.body.password;

    if (!login && !password)
        req.error = "Не введен логин или пароль";
    else {
        if (db_obj.users.find(user => user.login === login && user.password === password)) {
            req.session.login = req.body.login;
            req.session.password = req.body.password;
        } else {
            req.error = "Такого пользователя нет в базе данных";
        }
    }

    render_personal(req, res);
}

module.exports.exit = (req, res) => {
    req.session = null;
    res.redirect("/personal");
}

module.exports.render_personal = render_personal;
```

src/db_obj.js

```
"use strict";

module.exports.games = [
    {
        name: "Dota 2",
        description: "Defence of the Ancient 2",
        age: 0
    },
    {
        name: "CS 1.6",
        description: "Counter-Strike 1.6",
        age: 6
    },
    {
        name: "Курс NodeJS на архитектуре ЭВМ",
        description: "Спасибо, Попов.",
        age: 10
    },
    {
        name: "CS Source",
        description: "Counter Strike Source",
        age: 12
    },
    {
        name: "CS GO",
        description: "Counter Strike Global Offensive",
        age: 14
    },
    {
        name: "FIFA 2020",
        description: "FIFA",
        age: 18
    },
    {
        name: "The Witcher 3",
        description: "The Witcher 3",
        age: 21
    },
    {
        name: "DOTA",
        description: "Defence of the Ancient",
```

```
        age: 35
    }
];

module.exports.users = [
    {
        login: "GULZA",
        password: "d1ma5",
        hobby: "Kotlin",
        age: 7
    },
    {
        login: "mrrvz",
        password: "admin",
        hobby: "Haskell",
        age: 19
    },
    {
        login: "Justarone",
        password: "pashok",
        hobby: "Rust",
        age: 20
    },
    {
        login: "CATFELLA",
        password: "misha",
        hobby: "B3JIOM",
        age: 20
    },
    {
        login: "hackfeed",
        password: "serega",
        hobby: "nodejs",
        age: 20
    },
]
```

views/index.ejs

```
<%- include("include/header.ejs") %>
    <link rel="stylesheet" href="/style.css">
    <title>Lab 06</title>
</head>
<body>
    <%- include("include/navbar.ejs", { page: "Home" }) %>
    <div class="simple-card">
        <h1>Игры</h1>
        <form method="GET" action="/games">
            <label for="age"></label>
            <input type="text" name="age" placeholder="Введите возраст">
            <button type="submit">Проверить</button>
        </form>
    </div>
    <div class="simple-card">
        <h1>Персональная страница</h1>
        <a class="ba" href="/personal">Ссылка на персональную страницу</a>
    </div>

<%- include("include/footer.ejs") %>
```

views/games_page.ejs

```
<%- include("include/header.ejs") %>
    <link rel="stylesheet" href="/style.css">
    <title>Игры</title>
</head>
<body>
    <%- include("include/bar.ejs", { page: "games" }) %>

    <% if (error) {%>
        <div class="err-card">
            <h1><%=error%></h1>
        </div>
    <% } %>

    <% for (const game of games) { %>
        <div class="simple-card">
            <h1 style="text-align: center"><%= game.name %></h1>
            <p class="bp">Описание: <%= game.description %></p>
            <p class="bp">Возраст: <%= game.age %></p>
        </div>
    <% } %>

<%- include("include/footer.ejs") %>
```

views/personal_page.ejs

```
<%- include("include/header.ejs") %>
    <link rel="stylesheet" href="/style.css">
    <title>Персональная страница</title>
</head>
<body>
    <%- include("includes/navbar.ejs", { page: "personal" }) %>
    <% if (error) {%>
        <div class="err-card">
            <h1><%=error%></h1>
        </div>
    <%}%>

    <% if (isAuth) {%>
        <div class="simple-card">
            <h1>Пользователь</h1>
            <p class="bp">Логин: <%= user.login %></p>
            <p class="bp">Пароль: <%= user.password %></p>
            <p class="bp">Хобби: <%= user.hobby %></p>
            <p class="bp">Возраст: <%= user.age %></p>
        </div>
        <a class="ba exit-a" href="/exit">Выход</a>
    <%} else {%>
        <div class="simple-card">
            <form METHOD="POST" action="/login">
                <label for="login"></label>
                <input type="text" name="login" spellcheck="false" autocomplete="off">
                <br>
                <label for="password"></label>
                <input type="password" name="password" spellcheck="false" autocomplete="off">
                <br>
                <button type="submit">Логин</button>
            </form>
        </div>
    <%}%>

<%- include("include/footer.ejs") %>
```

views/include/header.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

views/include/bar.ejs

```
<header>
    <nav>
        <a class=<%= page === "Home" ? "active" : "passive"%> href="/">Домашнаяя страница</a>
        <a class=<%= page === "games" ? "active" : "passive"%> href="/games?age=0">Игры</a>
        <a class=<%= page === "personal" ? "active" : "passive"%> href="/personal">Персональная страница</a>
    </nav>
</header>
```

Результаты тестирования

Все тесты были пройдены успешно.

Вывод

В результате работы был освоен шаблонизатор **EJS** и работа с сессиями при помощи **express-session**.