



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»

Отчёт

по лабораторной работе № 3

Название: POST и GET запросы. Работа с шаблонизатором.

Дисциплина: Архитектура ЭВМ

Студент

ИУ7-55Б

(Группа)

Д.О. Склифасовский

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

А.Ю. Попов

(Подпись, дата)

(И.О. Фамилия)

Москва, 2020

Содержание

Введение	3
1 Задание 5	4
1.1 Условие	4
1.2 Решение	4
1.3 Тесты	7
2 Задание 6.1	10
2.1 Условие	10
2.2 Решение	10
2.3 Тесты	11
3 Задание 6.2	13
3.1 Условие	13
3.2 Решение	13
3.3 Тесты	15
Вывод	17

Введение

Цель работы: познакомиться с POST и GET запросами. Поработать с шаблонизаторами.

1 Задание 5

1.1 Условие

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.
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.
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.

1.2 Решение

Листинг 1 – Файл index.js

```

1  "use strict";
2
3  const express = require("express");
4  const fs = require("fs");
5
6  const app = express();
7  const port = 5000;
8  app.listen(port);
9  console.log('Server on port ${port}');
10
11  const way = __dirname + "/static";
12  app.use(express.static(way));
13
14  app.use(function(req, res, next) {
15      res.header("Cache-Control", "no-cache, no-store, must-revalidate");
16      res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
17          Content-Type, Accept");
18      res.header("Access-Control-Allow-Origin", "*");
19      next();
20  });
21
22  function loadBody(request, callback) {
23      let body = [];
24      request.on('data', (chunk) => {
25          body.push(chunk);
26      }).on('end', () => {
27          body = Buffer.concat(body).toString();
28          callback(body);
29      });
30  }
31
32  function checkUnique(a, b, c) {
33      let info = JSON.parse(fs.readFileSync("file.txt", "utf-8"));
34      let check = true;
35
36      for (let i = 0; i < info.length && check; i++) {
37          let obj = info[i];
38
39          if (obj["mail"] == a || obj["number"] == c) {

```

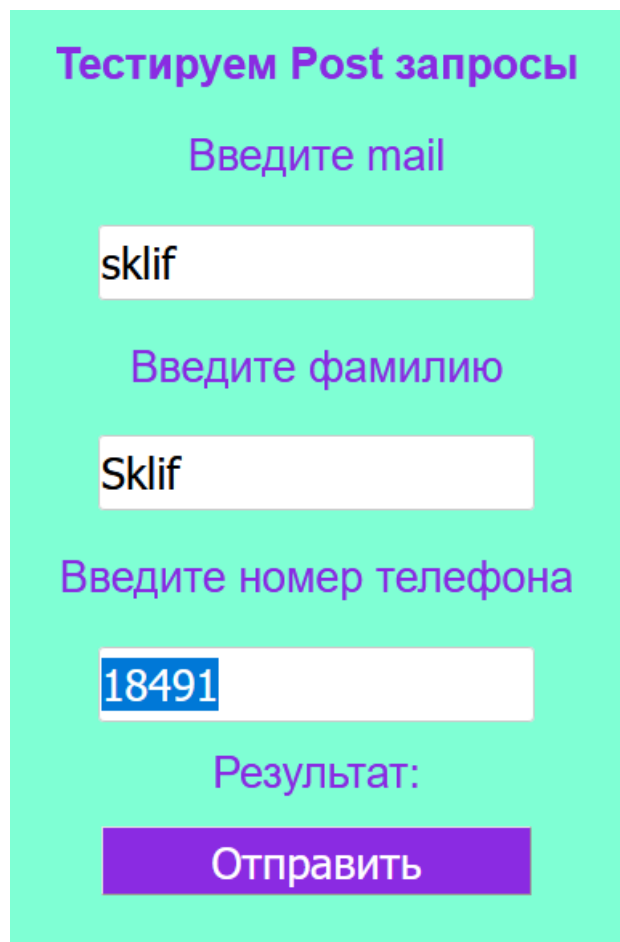
```

40     }
41 }
42     return check;
43 }
44
45 app.post("/save/info", function(request, response) {
46     loadBody(request, function(body) {
47         const obj = JSON.parse(body);
48         const mail = obj["mainInfo"];
49         const name = obj["nameInfo"];
50         const number = obj["numberInfo"];
51
52         let res = checkUnique(mail, name, number);
53
54         if (res) {
55             let info = JSON.parse(fs.readFileSync("file.txt", "utf-8"));
56             info.push({"mail" : mail, "name" : name, "number" : number});
57             const obj = JSON.stringify(info);
58             fs.writeFileSync("file.txt", obj);
59             response.end(JSON.stringify({
60                 result: "Save content ok"
61             }));
62         }
63         else {
64             response.end(JSON.stringify({
65                 result: "Not unique mail or number"
66             }));
67         }
68     });
69 });
70
71 function getInfo(mail) {
72     let info = JSON.parse(fs.readFileSync("file.txt", "utf-8"));
73     let result = null;
74     for (let i = 0; i < info.length; i++) {
75         let obj = info[i];
76         if (obj["mail"] == mail) {
77             result = obj;
78             break;
79         }

```

```
80     }
81     return result;
82 }
83
84 app.get("/getInfo", function(request, response) {
85     const mail = request.query.mail;
86     let res = getInfo(mail);
87     response.end(JSON.stringify({
88         result: res
89     }));
90 });
```

1.3 Тесты



Тестируем Post запросы

Введите mail

Введите фамилию

Введите номер телефона

Результат:

Рисунок 1 – Изначальная страница

Тестируем Post запросы

Введите mail

Введите фамилию

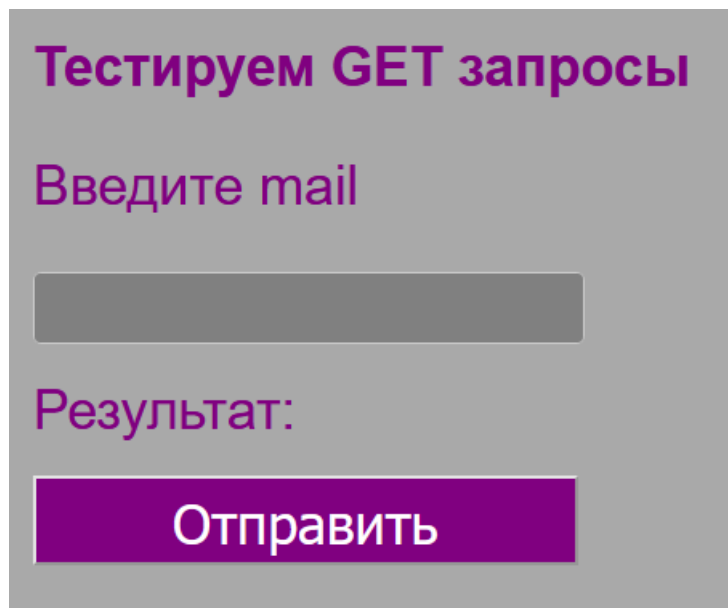
Введите номер телефона

Результат: Save content ok

Рисунок 2 – Результат записи

```
[{"mail":"alex@mail.ru","name":"Alexander","number":"8800553535"}, {"mail":"sklif","name":"SkLif","number":"18491"}]
```

Рисунок 3 – Новое содержимое файла на сервере



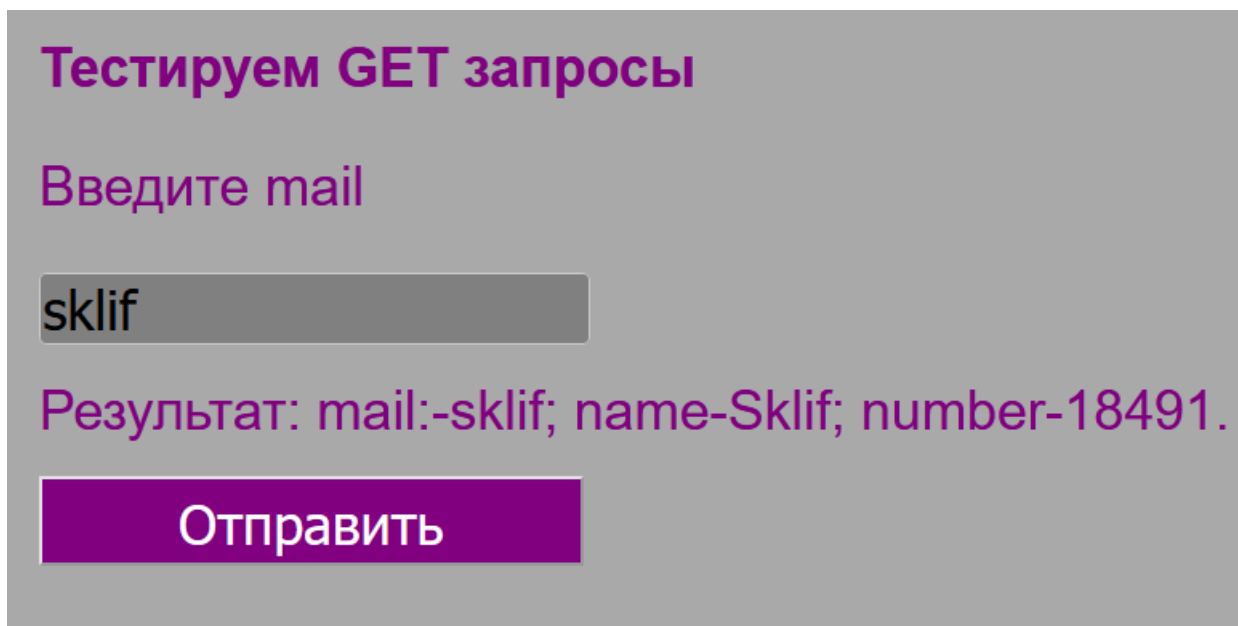
Тестируем GET запросы

Введите mail

Результат:

Отправить

Рисунок 4 – Вторая страница получения информации



Тестируем GET запросы

Введите mail

Результат: mail:-sklif; name-Sklif; number-18491.

Отправить

Рисунок 5 – Результат получения информации

2 Задание 6.1

2.1 Условие

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.

2.2 Решение

Листинг 2 – Файл index.js

```
1  "use strict";
2
3  let games = [
4    {name : "SKYRIM", description : "some info", restrictions : 16},
5    {name : "Grand Theft Auto 5", description : "some info", restrictions :
      18},
6    {name : "MineCraft", description : "some info", restrictions : 0},
7    {name : "Assassin's Creed", description : "some info", restrictions :
      18},
8    {name : "Mass Effect", description : "some info", restrictions : 16},
9    {name : "Mafia", description : "some info", restrictions : 18},
10   {name : "Far Cry", description : "some info", restrictions : 18},
11   ];
12
13   const express = require("express");
14
15   const app = express();
16   const port = 5000;
17   app.listen(port);
18   console.log('Server on port ${port}');
19
20   app.set("view engine", "hbs");
21
22   app.use(function(req, res, next) {
```

```

23     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
24     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
        Content-Type, Accept");
25     res.header("Access-Control-Allow-Origin", "*");
26     next();
27 });
28
29 function findNeedGames(age) {
30     let arr = [];
31     for (let i = 0; i < games.length; i++) {
32         let obj = games[i];
33         if (Number(age) > obj.restrictions) {
34             arr.push(obj);
35         }
36     }
37     return arr;
38 }
39
40 app.get("/page/getGames", function(request, response) {
41     const age = request.query.age;
42     let res = findNeedGames(age);
43     console.log(res);
44     const infoObject = {
45         games : res
46     };
47     response.render("pageGetGames.hbs", infoObject);
48 });

```

2.3 Тесты

localhost:5000/page/getGames?age=21

Рисунок 6 – Передаем возраст равный 21

Результат:

Компьютерные игры

Игра: SKYRIM

Описание: приключенческая ролевая игра

Возрастные ограничения: 16

Игра: Grand Theft Auto 5

Описание: Игроку предоставлена полная свобода действий - он может путешествовать по городу и его окрестностям, грабить прохожих, воровать машины - или же жить простой жизнью обычного человека.

Возрастные ограничения: 18

Игра: Minecraft

Описание: инди-игра в жанре песочницы с элементами выживания и открытым миром.

Возрастные ограничения: 0

Игра: Assassin's Creed

Описание: приключенческий экшен от третьего лица.

Возрастные ограничения: 18

Игра: Mass Effect

Описание: мультиплатформенный сиквел, вторая часть космической ролевой оперы.

Возрастные ограничения: 16

Игра: Мафия

Описание: приключенческий экшен, который переносит игроков в преступный мир Америки 40-50-х годов прошлого века.

Возрастные ограничения: 18

Игра: Far Cry

Описание: приключенческий шутер от первого лица с открытым миром и элементами RPG

Возрастные ограничения: 18

Рисунок 7 – Результат

3 Задание 6.2

3.1 Условие

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.

3.2 Решение

Листинг 3 – Файл index.js

```
1  "use strict";
2
3  let users = [
4    {login : "nagibator228", password : "228", hobby : "some info", age :
      16},
5    {login : "PBOTA_EHOTA", password : "1234", hobby : "some info", age :
      13},
6    {login : "PAK-OMAP", password : "1337", hobby : "some info", age : 18},
7    {login : "Zalypal", password : "1010", hobby : "some info", age : 20},
8    {login : "ideotko", password : "777", hobby : "some info", age : 11},
9    {login : "Alina-popa", password : "root", hobby : "some info", age :
      19},
10   ]
11
12   const express = require("express");
13   const cookieSession = require("cookie-session");
14
15   const app = express();
16   const port = 5000;
17   app.listen(port);
18   console.log('Server on port ${port}');
19
20   app.use(cookieSession({
21     name: 'session',
22     keys: ['hhh', 'qqq', 'vvv'],
```

```

23     maxAge: 24 * 60 * 60 * 1000 * 365
24   }));
25
26   app.set("view engine", "hbs");
27
28   app.use(function(req, res, next) {
29     res.header("Cache-Control", "no-cache, no-store, must-revalidate");
30     res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
      Content-Type, Accept");
31     next();
32   });
33
34   function checkSignIn(login, password) {
35     for (let i = 0; i < users.length; i++) {
36       let obj = users[i];
37       if (obj.login == login && obj.password == password) {
38         return true;
39       }
40     }
41     return false;
42   }
43
44   // http://localhost:5000/api/signIn?login=nagibator228&password=228
45   app.get("/api/signIn", function(request, response) {
46     const login = request.query.login;
47     const password = request.query.password;
48     if(!login) return response.end("Login not set");
49     if(!password) return response.end("Password not set");
50     if(!checkSignIn(login, password)) return response.end("Login or
      password was wrong.");
51     request.session.login = login;
52     request.session.password = password;
53     response.end("You signed in!");
54   });
55
56   function getInfo(login, password) {
57     let res = null;
58     for (let i = 0; i < users.length; i++) {
59       let obj = users[i];
60       if (obj.login == login && obj.password == password) {

```

```

61     res = obj;
62     break;
63 }
64 }
65 return res;
66 }
67
68 // http://localhost:5000/api/getInfo
69 app.get("/api/getInfo", function(request, response) {
70     if(!request.session.login) return response.end("Sign in first");
71     if(!request.session.password) return response.end("Sign in first");
72     const login = request.session.login;
73     const password = request.session.password;
74     let info = getInfo(login, password);
75     console.log(info);
76     response.render("pageInfo.hbs", info);
77 });
78
79 // http://localhost:5000/api/logOff
80 app.get("/api/logOff", function(request, response) {
81     request.session = null;
82     response.end("You logged off");
83 });

```

3.3 Тесты

Входим в аккаунт

`localhost:5000/api/signIn?login=nagibator228&password=228`

Рисунок 8 – Адрес с логином и паролем

Результат:

`You signed in!`

Рисунок 9 – Результат

Получаем информацию:



```
localhost:5000/api/getInfo
```

Рисунок 10 – Получение информации аккаунта

Информация:

Информация о Пользователе

Логин: nagibator228


Пароль: 228

Хобби: Копатели онлайн

Возраст: 16

Рисунок 11 – Информация

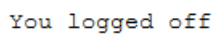
Выходим из аккаунта



```
localhost:5000/api/logOff
```

Рисунок 12 – Выход из аккаунта

Результат:



```
You logged off
```

Рисунок 13 – Результат

Вывод

В ходе выполнения лабораторной работы я научился работать с POST и GET запросами, также поработал с шаблонизаторами.