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"

Студент Наместник Анастасия
Группа ИУ7-53Б
Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ7
Тема Изучение запросов. Шаблонизатор. Cookie.

Студент: _____ Наместник А.А.
подпись, дата Фамилия, И.О.

Преподаватель: _____ Попов А. Ю.
подпись, дата Фамилия, И. О.

Москва — 2020 г.

TASK_1.

Цель работы:

— Создать сервер;
— Работа с POST запросами;
— Работа с GET запросами;
— Работа с CSS.

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.

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 1 — Код программы. TASK_1. Код программы

```
1
2     "use strict";
3     const fs = require("fs");
4     //Фреймворк Express сам использует модуль http, но вместе с тем предост
          авляет ряд готовых абстракций,
5     //которые упрощают создание сервера и серверной логики, в частности, об
          работка отправленных форм, работа с куками, CORS и т.д.
6
7     // подключение express
8     const express = require("express");
9
```

```
10      //В отличие от GET - запросов данные POST - запросов передаются не в ст
            роке запроса, а в его теле.
11      //Распространенным примеров подобных запросов является отправка данных
            формы на сервер.
12      //Для отправки POST - запросов предназначен метод post.
13      //Его объявление и использование в целом аналогично методу get.Он прини
            мает следующие параметры:
14      //url: обязательный параметр, содержащий адрес ресурса, к которому буде
            т обращаться запрос
15      //data: необязательный параметр, содержащий простой объект javascript и
            ли строку, которые будут отправлены на сервер вместе с запросом
16      //success(data, textStatus, jqXHR): необязательный параметр - функция о
            братного вызова, которая будет выполняться при успешном выполнении з
            апроса.Она может принимать три параметра: data - данные, полученные
            с сервера, textStatus - - статус запроса и jqXHR - специальный объек
            т jQuery, который представляет расширенный вариант объекта
            XMLHttpRequest.
17      //dataType: необязательный параметр, содержащий тип данных в виде строк
            и, например, "xml" или "json"
18      //На выходе метод post возвращает объект jqXHR.
19
20
21      function Main() {
22          // создаем объект приложения
23          const app = express();
24          //Настройка порта
25          const port = 5000;
26          app.listen(port);
27          console.log("My server on port " + port);
28
29          // отправка статических файлов
30          const way = __dirname + "/static";
31          app.use(express.static(way));
32
33          // заголовки в ответ клиенту
34          app.use(function (req, res, next) {
35              res.header("Cache-Control", "no-cache, no-store,
                    must-revalidate");
36              res.header("Access-Control-Allow-Headers", "Origin,
                    X-Requested-With, Content-Type, Accept");
37              res.header("Access-Control-Allow-Origin", "*");
38              next();
39          });
40
41          app.get("/find", function (request, response) {
42              const mail = request.query.mail;
43
```

```
44              //Открытие файла contacts.json
45              const file = fs.readFileSync("contacts.json", "utf-8");
46              const fileContent = JSON.parse(file);
47              let result = "Не найдено";
48
49              //Проверка на наличие
50              for (let i in fileContent) {
51                  if (mail == fileContent[i].mail) {
52                      result = fileContent[i];
53                      break;
54                  }
55              }
56
57              response.end(JSON.stringify({
58                  result: JSON.stringify(result)
59              }));
60          });
61
62          app.get("/get_info", (_request, response) => {
63              const fileContent = fs.readFileSync("static/" +
                    "get_info.html", "utf-8");
64              response.end(fileContent);
65          });
66
67
68          //В этом коде идёт описание функции загрузки тела POST запроса
69          function loadBody(request, callback) {
70              let body = [];
71              request.on('data', (chunk) => {
72                  body.push(chunk);
73              }).on('end', () => {
74                  body = Buffer.concat(body).toString();
75                  callback(body);
76              });
77          }
78
79          // it is post
80          app.post("/save/info", function (request, response) {
81              loadBody(request, function (body) {
82
83                  const obj = JSON.parse(body);
84                  const mail = obj["mail"];
85                  const lastname = obj["lastname"];
86                  const number = obj["number"];
87
88                  const file = fs.readFileSync("contacts.json", "utf-8");
89                  const fileContent = JSON.parse(file);
```

```
                let unique = true;
                let message = "";

                // Проверка на уникальность.
                for (let i in fileContent) {
                    if (mail == fileContent[i].mail) {
                        unique = false;
                        message = "Почта уже зарегистрирована"
                        break;
                    }
                    if (number == fileContent[i].number) {
                        unique = false;
                        message = "Номер уже зарегистрирован"
                        break;
                    }
                }

                if (unique) {
                    fileContent.push({ mail, lastname, number })
                    fs.writeFileSync("contacts.json",
                       JSON.stringify(fileContent, null, 4));
                    message = "Прльзователь добавлен"
                }

                // Ответ запроса.
                response.end(JSON.stringify({
                    result: message
                }));
            });
        });
    }


    Main();
```

Листинг 2 — Код программы. TASK_1. Дополнительный скрипт

```
    "use strict";

// onload - функция, которая вызывается когда собрался HTML.
// window - это глобальной объект.
window.onload = function () {
    // Получаем (ссылку) на поля.
    const field_find_mail = document.getElementById("field-get-info");

    // Получаем кнопку, при нажатии на которую должна выдаваться информация
        .
```

```
    const btn_get_info = document.getElementById("get-info-btn");

    // ajax get
    function ajaxGet(urlString, callback) {
        let r = new XMLHttpRequest();
        r.open("GET", urlString, true);
        r.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");
        r.send(null);
        r.onload = function () {
            callback(r.response);
        };
    };

    btn_get_info.onclick = function () {
        const find_mail = field_find_mail.value;

        const url = `/find?mail=${find_mail}`;

        ajaxGet(url, function (stringAnswer) {
            const objectAnswer = JSON.parse(stringAnswer);
            const result = objectAnswer.result;
            alert(result);
        });
    }
};
```

Листинг 3 — Код программы. TASK_1. Дополнительный скрипт

```
    "use strict";

    window.onload = function () {
        // input fields
        const f1 = document.getElementById("field-first");
        const f2 = document.getElementById("field-second");
        const f3 = document.getElementById("field-third");

        // button
        const btn = document.getElementById("add-contact-btn");

        // label (результат добавления (добавлено/не добавлено))
        const label = document.getElementById("result-label");

        // ajax post
        function ajaxPost(urlString, body, callback) {
            let r = new XMLHttpRequest();
            //Инициализация соединения
            r.open("POST", urlString, true);
```

```
20              r.setRequestHeader("Content-Type",
                    "application/json;charset=UTF-8");
21              r.send(body);
22              r.onload = function () {
23                  callback(r.response);
24              };
25          };
26
27          // click event
28          btn.onclick = function () {
29              const mail = f1.value;
30              const lastname = f2.value;
31              const number = f3.value;
32
33              //Создание POST запроса
34
35              ajaxPost("/save/info", JSON.stringify({
36                  mail, lastname, number
37              }), function (stringAnswer) {
38                  const objectAnswer = JSON.parse(stringAnswer);
39                  const result = objectAnswer.result;
40                  label.innerHTML = result;
41              });
42          };
43      };
```

Вывод:

— Был создан сервер;
— Была реализована работа с POST запросами;
— Была реализована работа с GET запросами;
— Была реализована работа с CSS.

Пример работы:

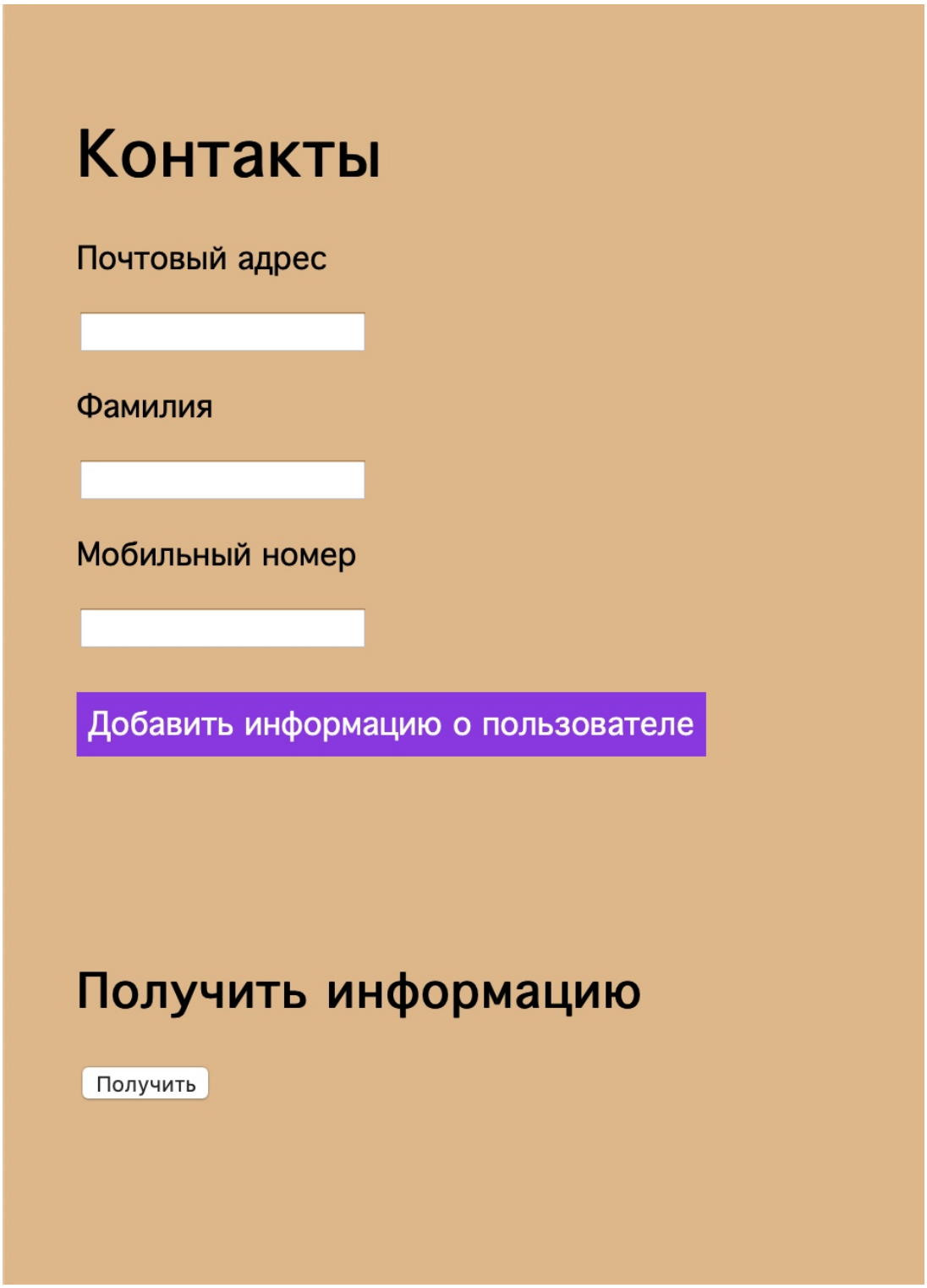

Рисунок 0.1 — Пример работы программы

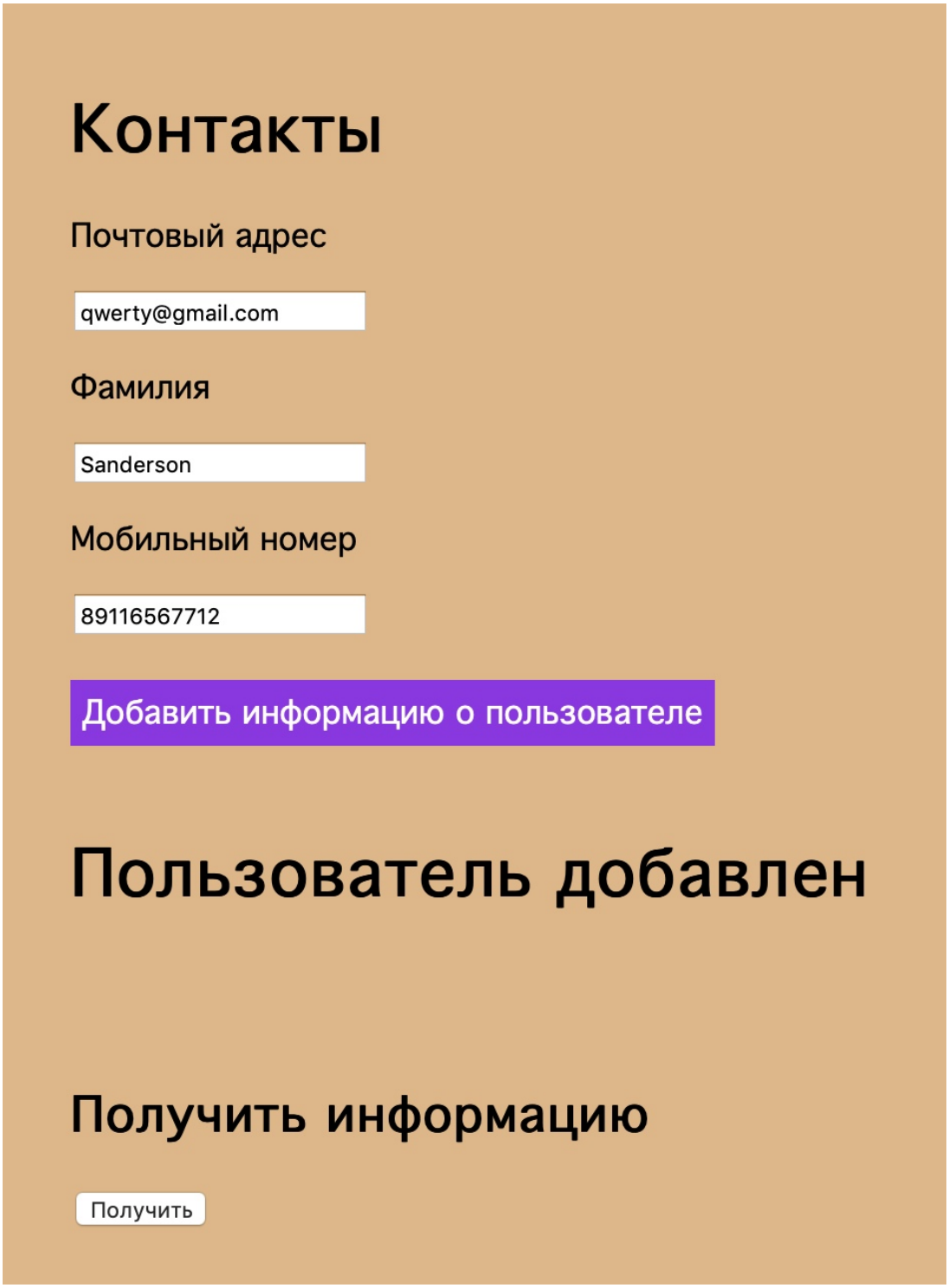

Рисунок 0.2 — Пример работы программы

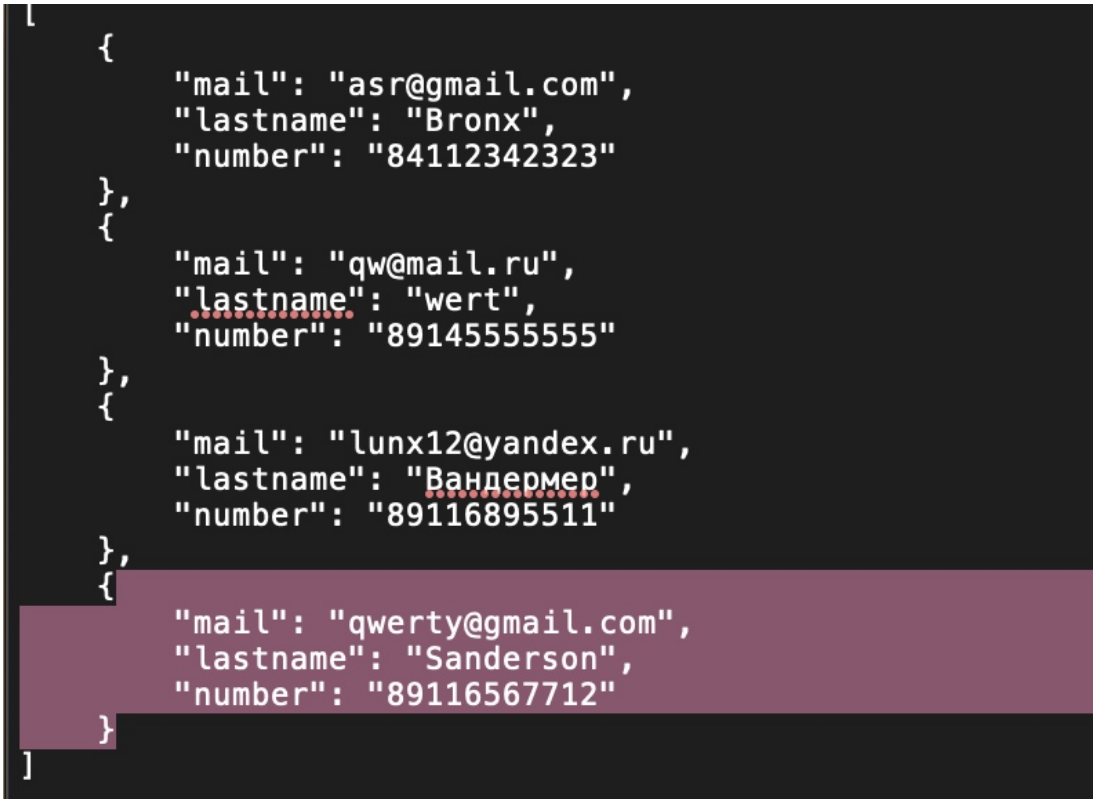

Рисунок 0.3 — Пример работы программы

Рисунок 0.4 — Пример работы программы

TASK_2.

Цель работы:

— Создать сервер.
— Реализовать страницу с использованием шаблонизатора.
— Изучить и реализовать работу с cookie.

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.

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 4 — Код программы. TASK_2. Реализация задания 1

```
"use strict";

// импорт библиотек
const express = require("express");
const fs = require("fs");

// 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();
});
```

```
23
24 // выдача страницы с информацией о кафедре
25 app.get("/page/department", function (request, response) {
26     const infoObject = {
27         facultyValue: "Информатика и системы управления",
28         departmentValue: "Компьютерные системы и сети",
29         indexValue: 6
30     };
31     response.render("pageDepartment.hbs", infoObject);
32 });
33
34
35
36 // выдача страницы с массивом игр
37 app.get("/page/games", function (request, response) {
38     //__dirname - возвращает путь к каталогу текущего исполняемого файла
39     //(__dirname использует локацию выполняемого скрипта(файла))
40     let age = request.query.age;
41     console.log(age);
42     age = parseInt(age);
43
44     //Проверка на корректность параметра URL-адреса
45     if (!age) {
46         response.end("Input Error!");
47         return
48     }
49
50     const file = "games.json";
51     const path = __dirname + "/" + file;
52     if (!fs.existsSync(path)) {
53         console.log("\nThe file" + file + "does not exist!")
54     };
55     const fileContent = fs.readFileSync(path, "utf-8");
56     const GamesArray = JSON.parse(fileContent);
57
58     //Массив игр, возрастное ограничение которых будет меньше переданного в
           URL-адресе
59     const ResultArray = []
60
61     for (let i = 0; i < GamesArray.length; i++)
62         if (GamesArray[i].restriction < age)
63             ResultArray.push(GamesArray[i]);
64
65     //Формирование объекта для передачи в шаблон
66     const infoObject = {
67         descriptionValue: "Список компьютерных игр",
68         gamesArray: ResultArray
```

```
69      };
70      response.render("pageGames.hbs", infoObject);
71  });
```

Листинг 5 — Код программы. TASK_2. Реализация задания 2

```
1   "use strict";
2
3   // импортируем библиотеки
4   const express = require("express");
5   const cookieSession = require("cookie-session");
6   //npm install fs --save
7   const fs = require("fs");
8
9   // запускаем сервер
10  const app = express();
11  const port = 5000;
12  app.listen(port);
13  console.log(`Server on port ${port}`);
14
15  // работа с сессией
16  app.use(cookieSession({
17      name: 'session',
18      keys: ['hhh', 'qqq', 'vvv'],
19      maxAge: 24 * 60 * 60 * 1000 * 365
20  }));
21
22  // // заголовки в ответ клиенту
23  // app.use(function (req, res, next) {
24  //  res.header("Cache-Control", "no-cache, no-store, must-revalidate");
25  //  res.header("Access-Control-Allow-Headers", "Origin,
        X-Requested-With, Content-Type, Accept");
26  //  next();
27  // });
28
29  // сохранить cookie
30  //Пример: http://localhost:5000/api/save?login=KiraSan&password=qwerty
31  app.get("/api/save", function (request, response) {
32      // Если пользователь уже авторизован,
33      //выдаем сообщение об этом
34      console.log(request.session);
35      if (request.session.login) {
36          response.end("You are signed in.");
37      }
38
39      //console.log(request.session);
40      // получаем параметры запроса
```

```
41          const login = request.query.login;
42          console.log(login)
43          const password = request.query.password;
44
45          // контролируем существование параметров
46          if (!login) return response.end("Login not set");
47          if (!password) return response.end("Password not set");
48
49          const file = "DB_users.json";
50          const path = __dirname + "/" + file;
51          const fileContent = fs.readFileSync(path, "utf-8");
52          const UsersArray = JSON.parse(fileContent);
53
54          //Если пользователь зарегистрирован (информация о нем находится в ф
                айле),
55          //при авторизации выставляем Cookie на стороне сервера
56          for (let i = 0; i < UsersArray.length; i++) {
57              if (UsersArray[i].login === login && UsersArray[i].password
                    === password) {
58                  // выставляем cookie
59                  request.session.login = login;
60                  request.session.password = password;
61                  // отправляем ответ об успехе операции
62                  response.end("Set cookie ok");
63              }
64          }
65          response.end("Authorization fail: invalid login or password");
66
67      });
68
69      //Проверка на то, авторизован ли пользователь (проверка на существовани
            е Cookie),
70      //и выдача информации о нем (если да) или сообщения о том, что его
            Cookie не существуют
71      // http://localhost:5000/api/get
72
73      app.get("/api/get", function (request, response) {
74          // контролируем существование cookie
75          if (!request.session.login) return response.end("Not exists");
76          if (!request.session.password) return response.end("Not exists");
77
78          //Если Cookie существуют (пользователь авторизован),
79          //получаем информацию о нем из файла с зарегистрированными пользова
                телями
80          const file = "DB_users.json";
81          const path = __dirname + "/" + file;
82          const fileContent = fs.readFileSync(path, "utf-8");
```

```
83          const UsersArray = JSON.parse(fileContent);
84
85          const login = request.session.login;
86          const password = request.session.password;
87          for (let i = 0; i < UsersArray.length; i++) {
88              if (UsersArray[i].login === login && UsersArray[i].password
                    === password) {
89
90                  return response.end("Info:\nLogin: " + UsersArray[i].login
                        +
91                      "\nAge : " + UsersArray[i].age + "\nHobby: " +
                            UsersArray[i].hobby);
92              }
93          }
94      });
95
96      // удалить все cookie
97      // http://localhost:5000/api/delete
98      app.get("/api/delete", function (request, response) {
99          request.session = null;
100         console.log(request.session);
101         response.end("Delete cookie ok");
102     });
103
104     //Зайдем не под своим паролем:
105     // http://localhost:5000/api/save?login=KiraSan&password=123
106     //Получим сообщение об этом: Authorization fail: invalid login or
            password
107     //Пробуем зайти с неверным логином
108     //http://localhost:5000/api/save?login=123Wol&password=1234567
109     //Получим сообщение об этом: Authorization fail: invalid login or
            password
110
111     //Зайдем под своим паролем:
112     // http://localhost:5000/api/save?login=KiraSan&password=qwerty
113     //Успех: сообщение "You are signed in"
114     //Получим информацию о пользователе:
115     // http://localhost:5000/api/get
116     //Успех
117     //Пробуем авторизоваться еще раз
118     // http://localhost:5000/api/save?login=123Wolf&password=1234567
119     //Неуспех (так как пользователь уже авторизован)
120     //Удалим Cookies и зайдем снова
121     //Успех
122 +т
```

Вывод:

— Был создан сервер.

— Была реализована страница с использованием шаблонизатора.

— Была изучена и реализована работа с cookie.

Пример работы:

Рисунок 0.1 — Пример работы программы

Список компьютерных игр

Название: Minecraft
Описание: Компьютерная инди-игра в жанре песочницы, разработанная шведским программистом Маркусом Перссоном и выпущенная его компанией Mojang AB
Возрастное ограничение: 6

Название: The Sims 4
Описание: Однопользовательская компьютерная игра в жанре симулятора жизни, четвёртая по счёту из серии игр The Sims, разработанная компанией Maxis и издаваемая Electronic Arts для Windows и macOS
Возрастное ограничение: 16

Название: Need for Speed: Most Wanted
Описание: Компьютерная игра серии Need for Speed в жанре аркадной автогонки, разработанная студией EA Canada и изданная компанией Electronic Arts для консолей, персональных компьютеров и мобильных телефонов в 2005 году
Возрастное ограничение: 12

Рисунок 0.2 — Пример работы программы

Рисунок 0.3 — Пример работы программы

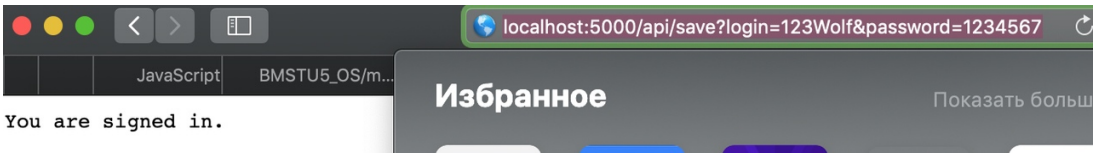

Рисунок 0.4 — Пример работы программы

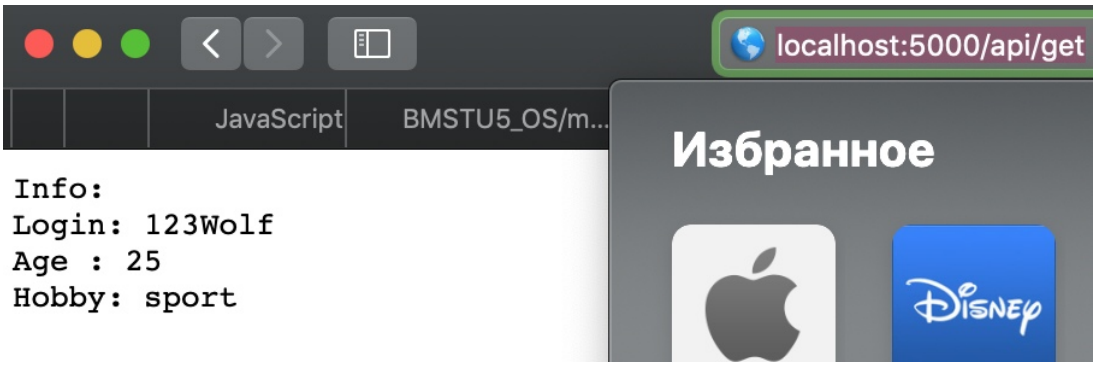

Рисунок 0.5 — Пример работы программы