



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 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника;

О Т Ч Е Т

по лабораторной работе № 3

Дисциплина: Архитектура ЭВМ

Студент

ИУ7-526

(Группа)

(Подпись, дата)

Кузин А.А.

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

Попов А.Ю.

(И.О. Фамилия)

Москва, 2020

Цель

Целью данной лабораторной работы является знакомство с POST и GET запросами в AJAX, а также изучение возможностей шаблонизаторов и cookie.

Часть 1

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.

Листинг программы:

```
"use strict"

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

const app = express();
const port = 5000;
app.listen(port);
console.log('Server on port 5000');

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

// Headers
```

```

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) => {
    body.push(chunk);
  }).on('end', () => {
    body = Buffer.concat(body).toString();
    callback(body);
  });
}

function checkUnique(post, phone){
  let unique = true;
  let lines = fs.readFileSync("part_1/records.txt", "utf-8");
  lines = lines.split("\n")
  for (let i = 0; i < lines.length; i++){
    const data = lines[i].split(" ");
    if (post === data[0] || phone === data[2]){
      unique = false;
      break;
    }
  }
  return unique;
}

app.post("/save/info", function (request, response) {
  loadBody(request, function (body) {
    const obj = JSON.parse(body);
    const post = obj["a"];
    const surname = obj["b"];
    const phone = obj["c"];

    if (checkUnique(post, phone)){
      const contentString = `${post} ${surname} ${phone}\n`;
      fs.appendFileSync("part_1/records.txt", contentString);
      response.end(JSON.stringify({
        result: "Save content ok"
      }));
    }
    else{
      response.end(JSON.stringify({
        result: "Content isn't unique"
      }));
    }
  })
}

```

```

    });
});
Код страницы:
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Добавление пользователя</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
    <h1>Добавление пользователя</h1>

    <p>Почта</p>
    <input id="post" type="text" spellcheck="false" autocomplete="off">

    <p>Фамилия</p>
    <input id="surname" type="text" spellcheck="false" autocomplete="off">

    <p>Номер телефона</p>
    <input id="phone" type="text" spellcheck="false" autocomplete="off">
    <br>
    <br>

    <button onclick="makeAction()" class="btn-class">Отправить</button>

    <script>
        "use strict";

        function ajaxPost(urlString, bodyString, callback) {
            let r = new XMLHttpRequest();
            r.open("POST", urlString, true);
            r.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
            r.send(bodyString);
            r.onload = function () {
                callback(r.response);
            }
        }

        function makeAction() {
            const a = document.getElementById("post").value
            const b = document.getElementById("surname").value
            const c = document.getElementById("phone").value

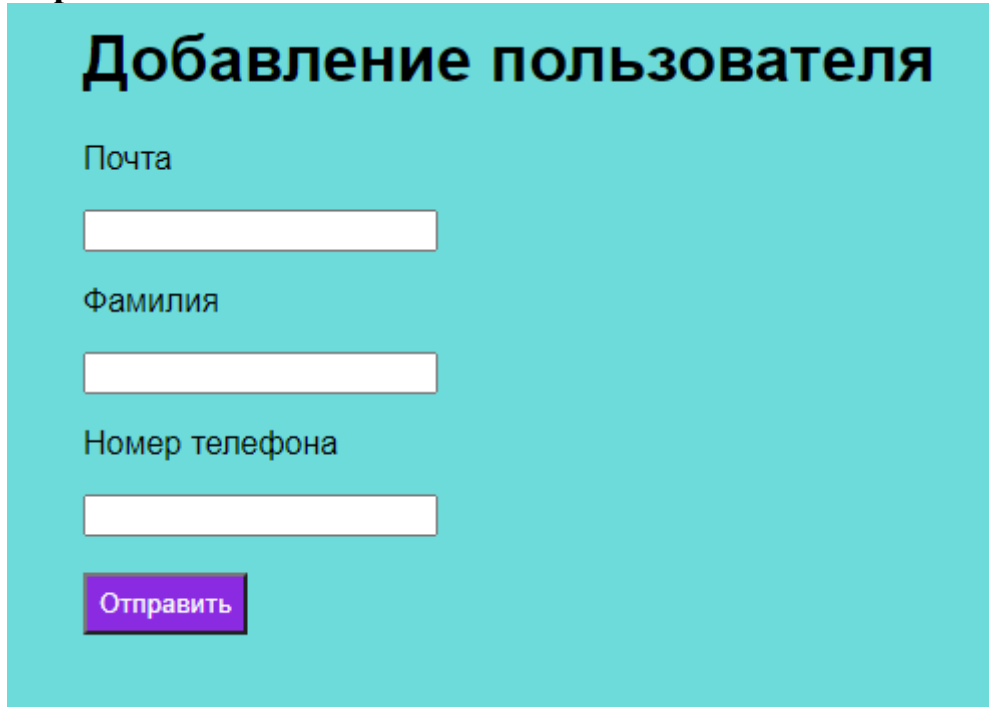
            ajaxPost("/save/info", JSON.stringify({
                a, b, c
            }), function (answerString) {
                const answerObject = JSON.parse(answerString);
                const result = answerObject.result;
                alert(result);
            });
        }
    </script>

```

```
    }  
</script>  
</body>  
</html>
```

Тесты

Страница:



Добавление пользователя

Почта

Фамилия

Номер телефона

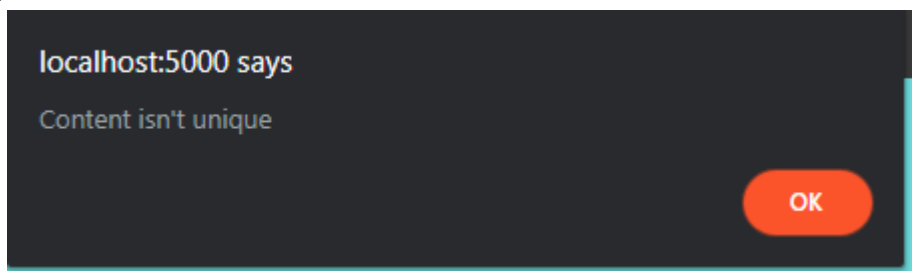
Отправить

Содержимое файла до операций:

1 1 1

2 2 2

Ввод: 2 4 3



Результат:

Ввод: 3 3 3



Результат:

Содержимое файла:

1 1 1

2 2 2

3 3 3

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.

Листинг программы:

К коду из предыдущего задания добавляем:

```
function findByPost(post) {
  let result = null;
  let lines = fs.readFileSync("records.txt", "utf-8");
  lines = lines.split("\n")
  for (let i = 0; i < lines.length; i++) {
    const data = lines[i].split(" ");
    if (post === data[0]) {
      result = lines[i];
      break;
    }
  }
  return result;
}
```

```
app.get("/find", function (request, response) {
  const post = request.query.a;

  const s = findByPost(post);

  if (s !== null)
    response.end(JSON.stringify({
      result: s
    }));
  else{
    const ans = "There is no such post";
    response.end(JSON.stringify({
      result: ans
    }));
  }
});
```

Код страницы:

```
<!DOCTYPE html>
<html>
```

```

<head>
  <meta charset="UTF-8">
  <title>Получение информации о пользователе</title>
  <link rel="stylesheet" href="/style_2.css">
</head>

<body>
  <h1>Получение информации о пользователе</h1>

  <p>Почта</p>
  <input id="post" type="text" spellcheck="false" autocomplete="off">
  <br>
  <br>

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

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

</html>

```

Код файла code.js:

```

"use strict";

window.onload = function () {
  const post = document.getElementById("post")

  const btn = document.getElementById("send-btn")

  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.onclick = function () {
    const a = post.value;
    const url = `/find?a=${a}`;
    ajaxGet(url, function (stringAnswer) {
      const objectAnswer = JSON.parse(stringAnswer);
      const result = objectAnswer.result;
      alert(result);
    });
  };
};

```

Тесты

Вид страницы:

Получение информации о пользователе

Почта

Отправить

Содержимое файла записей:

1 1 1

2 2 2

3 3 3

Входные данные: 1

localhost:5000 says

1 1 1

OK

Результат:

Входные данные: 4

localhost:5000 says

There is no such post

OK

Результат:

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 программы:

Файл style.css:

```
body {  
  padding: 30px;  
  background: rgb(110, 219, 219);  
  font-family: Geneva, Arial, Helvetica, sans-serif;  
}
```



```
.btn-class {  
  padding: 6px;  
  background: blueviolet;  
  color: white;  
  cursor: pointer;  
  display: inline-block;  
}
```

Файл style_2.css:

```
body {  
  padding: 30px;  
  background: rgb(168, 236, 208);  
  font-family: Geneva, Arial, Helvetica, sans-serif;  
}
```

```
.btn-class {  
  padding: 6px;  
  background: rgb(122, 75, 184);  
  color: white;  
  cursor: pointer;  
  display: inline-block;  
}
```

Часть 2

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.

Листинг программы

```
"use strict";  
  
const express = require("express");  
  
const app = express();
```

```

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

app.set("view engine", "hbs");

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 game1 = { name: "Doka", descr: "Описание", minAge: 12 };
const game2 = { name: "Cs:Go", descr: "Другое описание", minAge: 18 };
const game3 = { name: "Minecraft", descr: "Можно строить", minAge: 6 };
const games = [game1, game2, game3];

app.get("/page/games", function (request, response) {
  const age = request.query.age;
  let infoObject = {
    descriptionValue: "Список игр",
    gamesArray: []
  };

  for (let i = 0; i < games.length; i++)
    if (games[i].minAge < age)
      infoObject.gamesArray.push(games[i]);

  response.render("pageGames.hbs", infoObject);
});

```

Код файла шаблона:

```

<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Игры</title>
</head>

<body>

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

  {{#each gamesArray}}
  <div style="margin-bottom: 15px; padding: 8px;">
    Название: {{this.name}}
    <br>

```

```
        Описание: {{this.descr}}
        <br>
        Возрастное ограничение: {{this.minAge}}
    </div>
    {{/each}}

</body>

</html>
```

Тесты:

! localhost:5000/page/games?age=12

Список игр

Название: Minecraft
Описание: Можно строить
Возрастное ограничение: 6

! localhost:5000/page/games?age=24

Список игр

Название: Doka
Описание: Описание
Возрастное ограничение: 12

Название: Cs:Go
Описание: Другое описание
Возрастное ограничение: 18

Название: Minecraft
Описание: Можно строить
Возрастное ограничение: 6

! localhost:5000/page/games?age=5

Список игр

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 программы

```
"use strict";

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

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

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

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();
});

app.get("/api/save", function (request, response) {
  const login = request.query.login;
  const password = request.query.password;

  if (!login) return response.end("Login not set");
  if (!password) return response.end("Password not set");

  request.session.login = login;
  request.session.password = password;

  response.end("Set cookie ok");
});

class User{
  constructor(login, password, hobby, age){
    this.login = login;
```

```

        this.password = password;
        this.hobby = hobby;
        this.age = age;
    }
}

const users = [new User("qwer", "qwer", "Tennis", 16),
    new User("smt", "123", "Chess", 25),
    new User("Chad", "Cool", "Being cool", 20)]

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

    if (!request.session.login) return response.end("Not exists");
    if (!request.session.password) return response.end("Not exists");

    const login = request.session.login;
    const password = request.session.password;
    let hobby = "";
    let age = 0;

    let found = false;
    for (let i = 0; i < users.length; i++)
        if (login == users[i].login && password == users[i].password){
            found = true;
            hobby = users[i].hobby;
            age = users[i].age;
            break;
        }

    if (found){
        response.end(JSON.stringify({
            login,
            password,
            hobby,
            age
        }));
    }
    else{
        response.end("There is no such user");
    }
});

```

Тесты:

Устанавливаем cookie:

```
localhost:5000/api/save?login=Chad&password=Cool
```

Set cookie ok

Получаем информацию:

```
{"login":"Chad","password":"Cool","hobby":"Being cool","age":20}
```

Устанавливаем cookie:

```
localhost:5000/api/save?login=Virgin&password=Nerd
```

```
Set cookie ok
```

Получаем информацию:

```
There is no such user
```

Вывод

В результате выполнения данной лабораторной работы я получил навыки работы с форматом JSON, его обработкой, работы с файлами, а также навыки поднятия и использование серверов используя express.