



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»

ДИСЦИПЛИНА «Архитектура ЭВМ»

Лабораторная работа № 3

Тема Работа с AJAX запросами. Работа с шаблонизаторами. Сессии в NodeJS.

Студентка Воякин А. Я.

Группа ИУ7-54Б

Оценка (баллы) _____

Преподаватель Попов А. Ю.

Москва.
2020 г.

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.

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.

Три задания объединил в одно.

Листинг index.js:

```
"use strict";

// Импортируем необходимые библиотеки.
const express = require("express");
const fs = require("fs");

// Запускаем сервер.
const app = express();
const port = 5015;
app.listen(port);
console.log(`Server port: ${port}`);

// Отправка статических файлов.
const way = dirname + "/static";
app.use(express.static(way));

// 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();
});

// Получение информации о человеке по его почте.
app.get("/info", function(request, response) {
  const email = request.query.email;
  if (fs.existsSync("file.txt")) {
    const jsonStr = fs.readFileSync("file.txt", "utf8");
    const arr = JSON.parse(jsonStr);
    for (let i = 0; i < arr.length; i++) {
      if (arr[i].email === email) {
        response.end(JSON.stringify({result: JSON.stringify(arr[i])}));
        return;
      }
    }
    response.end(JSON.stringify({result: "EMAIL NOT FOUND"}));
  }
});

function loadBody(request, callback) {
  let body = [];
  request.on('data', (chunk) => {
    body.push(chunk);
  }).on('end', () => {
    body = Buffer.concat(body).toString();
    callback(body);
  });
}

app.post("/reg", function(request, response) {
  loadBody(request, function(body) {
    const obj = JSON.parse(body);
    const email = obj["email"];
    const phone = obj["phone"];
    let arr = [];
    if (fs.existsSync("file.txt")) {
      const jsonStr = fs.readFileSync("file.txt", "utf8");
      arr = JSON.parse(jsonStr);
      let unique = true;
      for (let i = 0; i < arr.length && unique; i++) {
        const element = arr[i];
        if (element.email === email || element.phone === phone) unique = false;
      }
      if (!unique) {
        response.end(JSON.stringify({
          result: "NOT SAVED"
        }));
        return;
      }
    }
    arr.push(obj);
    fs.writeFileSync("file.txt", JSON.stringify(arr));
    response.end(JSON.stringify({
      result: "SAVED"
    }));
  });
});
```

```

    ));
  });
});

```

Листинг reg_code.js:

```

"use strict";

window.onload = function() {
  const email_in = document.getElementById("email");
  const surname_in = document.getElementById("surname");
  const phone_in = document.getElementById("phone");

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

  const label = document.getElementById("result-label");

  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);
    }
  }

  btn.onclick = function() {
    const email = email_in.value;
    const surname = surname_in.value;
    const phone = phone_in.value;

    ajaxPost("/reg", JSON.stringify({
      email, surname, phone
    }), function(answerString) {
      const objectAnswer = JSON.parse(answerString);
      label.innerHTML = objectAnswer.result;
    });
  };

  email_in.onkeydown = email_in.onkeypress = email_in.onkeyup = function () {
    label.innerHTML = "";
  }

  phone_in.onkeydown = phone_in.onkeypress = phone_in.onkeyup = function () {
    label.innerHTML = "";
  }

  surname_in.onkeydown = surname_in.onkeypress = surname_in.onkeyup = function () {
    label.innerHTML = "";
  }
};

```

Листинг reg.html:

```

<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>REGISTRATION</title>
  <link rel="stylesheet" href="/reg_style.css">
</head>
<body>
<h1>USER REGISTRATION</h1>

<p>
  <span>EMAIL</span><br>
  <input id="email" type="text" spellcheck="false" autocomplete="off">
</p>

<p>
  <span>SURNAME</span><br>
  <input id="surname" type="text" spellcheck="false" autocomplete="off">
</p>

```

```

<p>
  <span>PHONE NUMBER</span><br>
  <input id="phone" type="text" spellcheck="false" autocomplete="off">
</p>

<br>

<div id="reg-send-btn" class="btn-class">SEND</div>

<br>
<br>

<h1 id="result-label"></h1>

<script src="/reg_code.js"></script>
</body>
</html>

```

Листинг reg_style.css:

```

body {
  padding: 40px;
  background-color: #000;
  color: #fff;
  text-align: center;
}

.btn-class {
  padding: 7px;
  background: blueviolet;
  color: white;
  cursor: pointer;
  display: inline-block;
}

```

Листинг info_code.js:

```

"use strict";

window.onload = function() {
  const email_in = document.getElementById("email");

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

  const label = document.getElementById("result-label");

  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 email = email_in.value;
    const url = `/info?email=${email}`;
    ajaxGet(url, function(stringAnswer) {
      const obj = JSON.parse(stringAnswer);
      label.innerHTML = obj.result;
    });
  };

  email_in.onkeydown = email_in.onkeypress = email_in.onkeyup = function () {
    label.innerHTML = "";
  }
};

```

Листинг info.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>INFO</title>
  <link rel="stylesheet" href="/info_style.css">
</head>
<body>
<h1>USER INFO</h1>

<p>
  <label>
    <span>EMAIL</span><br>
    <input id="email" type="text" spellcheck="false" autocomplete="off">
  </label>
</p>

<br>

<div id="info-send-btn" class="btn-class">SEND</div>

<br><br>

<h2 id="result-label"></h2>

<script src="/info code.js"></script>
</body>
</html>
```

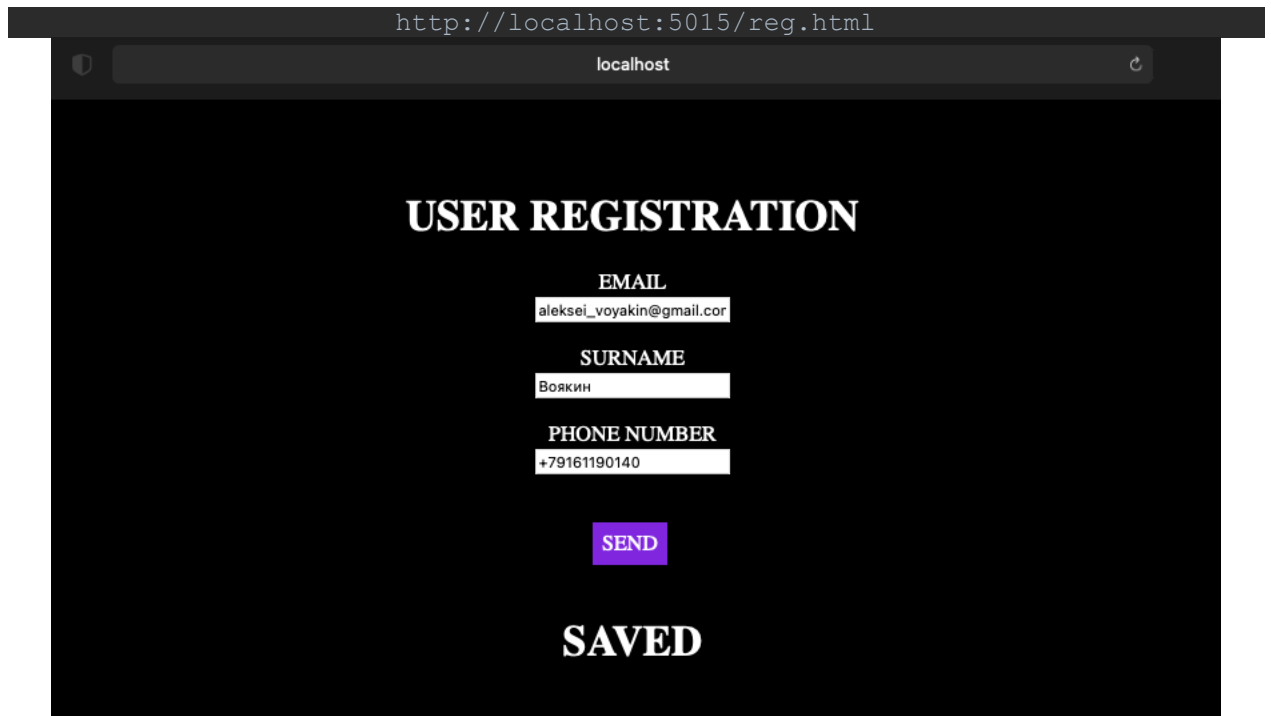
Листинг info_style.css:

```
body {
  padding: 40px;
  background-color: #000;
  color: #fff;
  text-align: center;
}

.btn-class {
  padding: 7px;
  background: blueviolet;
  color: white;
  cursor: pointer;
  display: inline-block;
}
```

Демонстрация работы программы.

Регистрация пользователя, которого нет в базе:



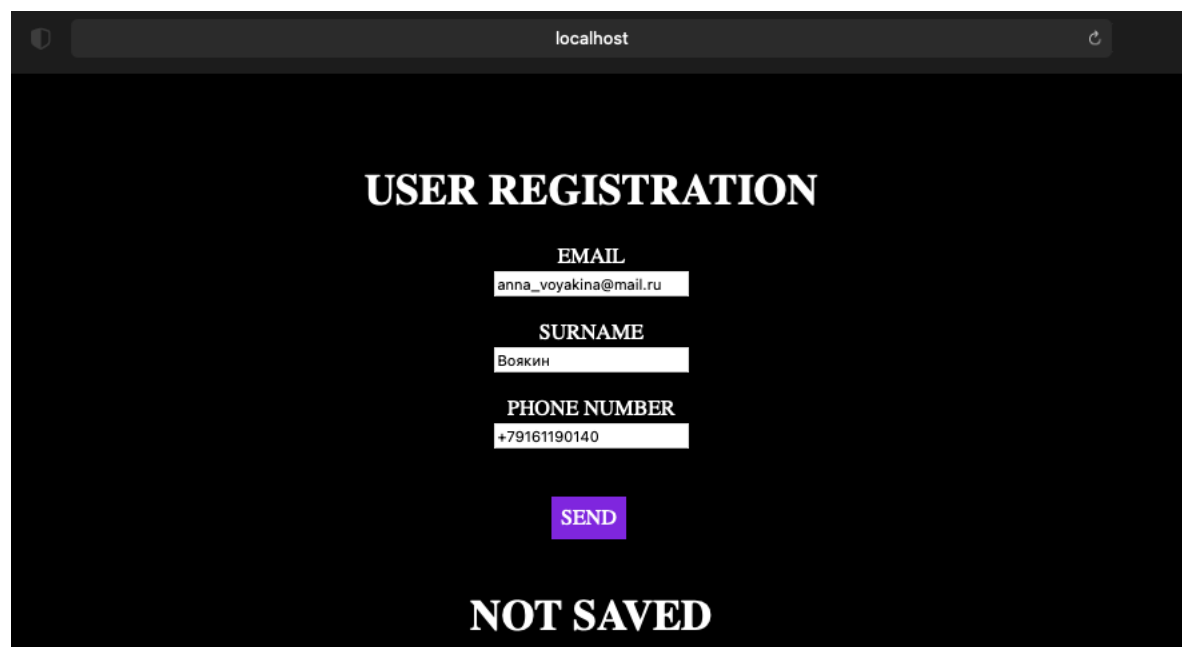
The screenshot shows a web browser window with the address bar displaying `http://localhost:5015/reg.html`. The page title is "localhost". The main content area has a black background with white text. At the top, it says "USER REGISTRATION". Below this, there are three input fields: "EMAIL" with the value "aleksei_voyakin@gmail.cor", "SURNAME" with the value "Воякин", and "PHONE NUMBER" with the value "+79161190140". Below these fields is a blue "SEND" button. At the bottom, it says "SAVED".

Пользователя с таким email и номером телефона ещё не было в базе данных, поэтому запись произошла и пользователю вывелось сообщение “SAVED”

В файл file.txt добавилась запись с данными пользователя:

```
{"email": "aleksei_voyakin@gmail.com", "surname": "Воякин", "phone": "+79161190140"}
```

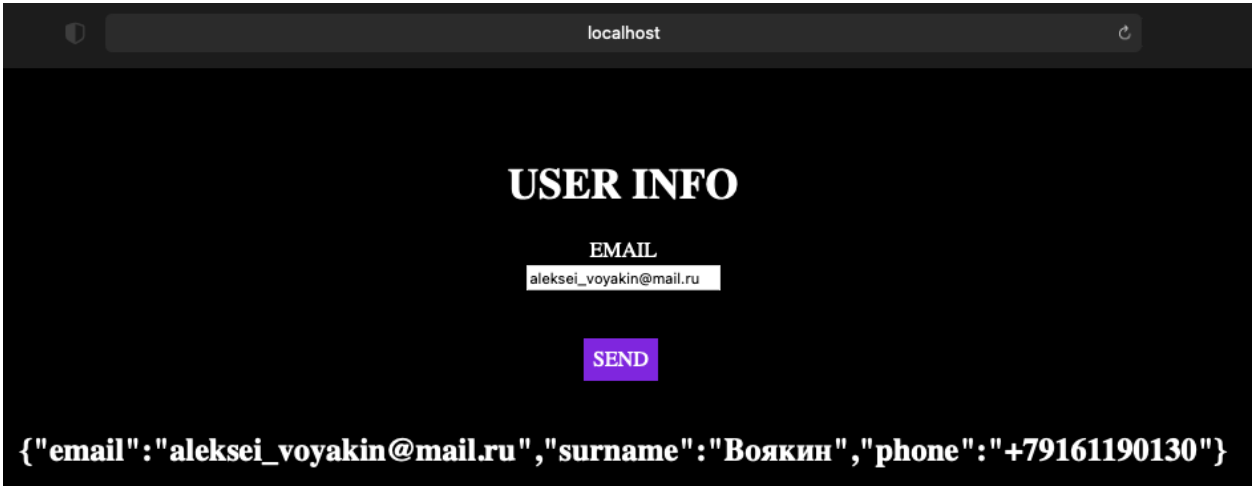
Регистрация пользователя, который есть в базе:



The screenshot shows a web browser window with the address bar displaying "localhost". The page title is "localhost". The main content area has a black background with white text. At the top, it says "USER REGISTRATION". Below this, there are three input fields: "EMAIL" with the value "anna_voyakina@mail.ru", "SURNAME" with the value "Воякин", and "PHONE NUMBER" with the value "+79161190140". Below these fields is a blue "SEND" button. At the bottom, it says "NOT SAVED".

Пользователь с таким номером телефона уже присутствует в базе, поэтому добавление не произошло и пользователю вывелось сообщение “NOT SAVED”

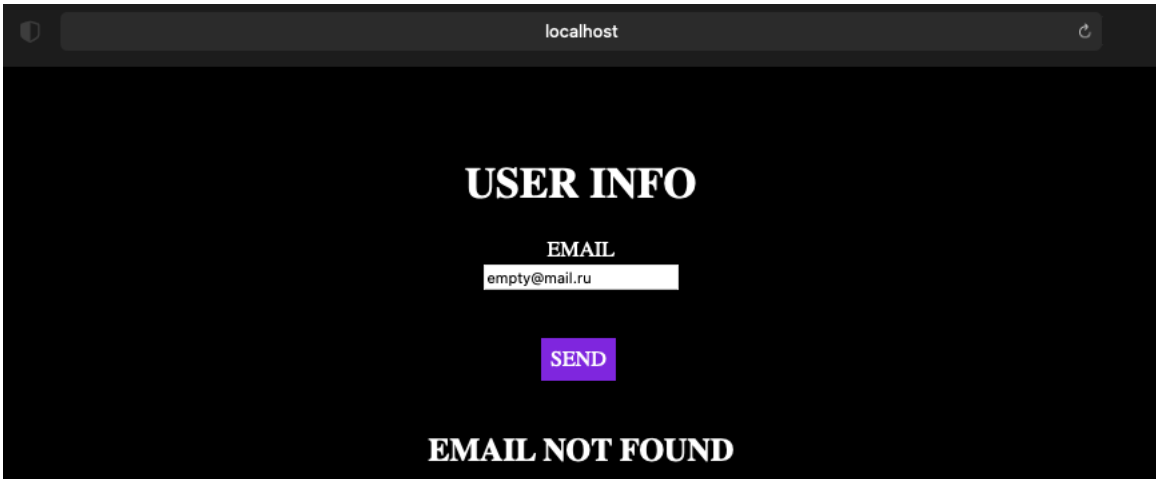
Получение информации о пользователе, который есть в базе:



The screenshot shows a web browser window with the address bar set to 'localhost'. The page has a black background with white text. At the top, it says 'USER INFO'. Below that is a label 'EMAIL' and a text input field containing 'aleksei_voyakin@mail.ru'. Under the input field is a blue button with the word 'SEND' in white. At the bottom of the page, a JSON object is displayed: {"email": "aleksei_voyakin@mail.ru", "surname": "Воякин", "phone": "+79161190130"}.

Пользователь с введённым адресом электронной почты есть в базе данных, поэтому пользователю вывелось на экран с информацией.

Получение информации о пользователе, которого нет базе:



The screenshot shows a web browser window with the address bar set to 'localhost'. The page has a black background with white text. At the top, it says 'USER INFO'. Below that is a label 'EMAIL' and a text input field containing 'empty@mail.ru'. Under the input field is a blue button with the word 'SEND' in white. At the bottom of the page, the text 'EMAIL NOT FOUND' is displayed.

Пользователя с таким email нет, вывелось соответствующее сообщение.

Задание 2.1

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 index.js:

```
"use strict";

// Импорт библиотеки.
const express = require("express");
const fs = require("fs");

// Запускаем сервер.
const app = express();
const port = 5015;
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();
});

// Выдача страницы с массивом игр, подходящих под возраст.
app.get("/page/games", function(request, response) {
  const age = parseInt(request.query.age);
  const games = JSON.parse(fs.readFileSync("games.json", "utf8"));
  const filteredGames = games.filter(item => item.age < age);
  const infoObject = {
    descriptionValue: `GAMES LIST WITH AGE UNDER ${age}`,
    gamesArray: filteredGames
  };
  response.render("pageGames.hbs", infoObject);
});
```

Листинг pageGames.hbs:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>GAMES</title>
  <style>
    body {
      background-color: #000;
      color: #fff;
      text-align: center;
      padding: 20px;
    }
  </style>
</head>
<body>

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

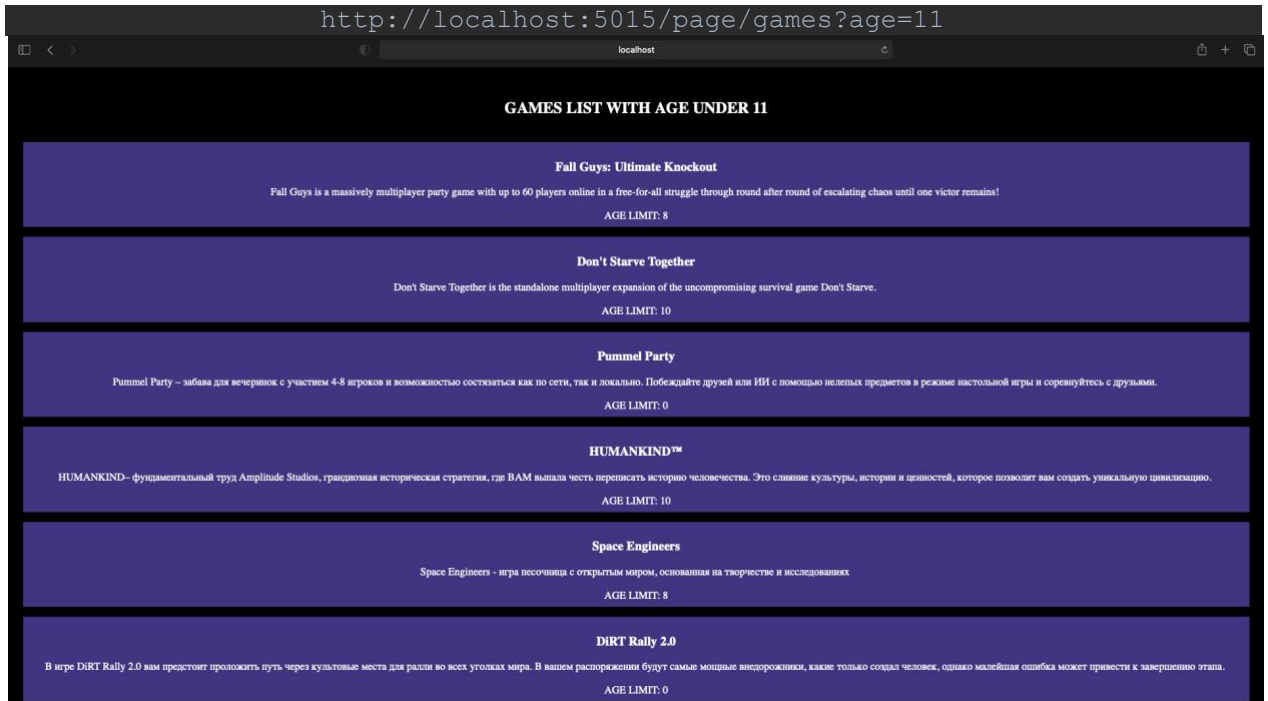
{{#each gamesArray}}
  <div style="background: darkslateblue; margin-bottom: 15px; padding: 8px;">
    <h3>{{this.name}}</h3>
    {{this.desc}}
```

```

    <br><br>
    AGE LIMIT: {{this.age}}
  </div>
</each>
</body>
</html>

```

Выведем игры для детей, которым 11 лет:



Задание 2.2

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 index.js:

```

"use strict";

// Импортируем библиотеки.
const express = require("express");
const cookieSession = require("cookie-session");
const fs = require("fs");

// Запускаем сервер.
const app = express();
const port = 5015;

```

```

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

// Активируем шаблонизатор.
app.set("view engine", "hbs");

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

// 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");
  next();
});

const pageLogin = fs.readFileSync("pageLogin.html", "utf8");

function authorize(login, password) {
  const db = JSON.parse(fs.readFileSync("db.json", "utf8"));
  let num;
  let founded = false;
  for (num = 0; num < db.length; num++) {
    if (db[num].login === login && db[num].password === password) {
      founded = true;
      break;
    }
  }
  return(JSON.stringify({exist: founded, person: db[num]}));
}

// Удалить все cookie.
app.get("/api/delete", function(request, response) {
  request.session = null;
  response.end(pageLogin);
});

app.get("/login", function(request, response) {
  const login = request.query.login;
  const password = request.query.password;
  const obj = JSON.parse(authorize(login, password));
  if (obj.exist) {
    request.session.login = login;
    request.session.password = password;
    const personObj = {
      login: obj.person.login,
      hobby: obj.person.hobby,
      age: obj.person.age
    };
    response.render("pageLk.hbs", personObj);
  }
  else {
    response.end(pageLogin);
  }
});

app.get("/lk", function(request, response) {
  if (!request.session.login || !request.session.password) {
    response.end(pageLogin);
  }
  else {
    const login = request.session.login;
    const password = request.session.password;
    const obj = JSON.parse(authorize(login, password));
    if (obj.exist) {
      const personObj = {
        login: obj.person.login,
        hobby: obj.person.hobby,
        age: obj.person.age
      };
      response.render("pageLk.hbs", personObj);
    }
    else {
      response.end(pageLogin);
    }
  }
});

```

```
});
```

Листинг pageLogin.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>LOGIN</title>
  <style>
    body {
      background-color: #000;
      color: #fff;
      text-align: center;
      padding: 20px;
    }
  </style>
</head>
<body>
<h1>AUTHORIZATION</h1>
<form method="GET" action="/login">
  <p>
    <label>
      <span>LOGIN</span> <br>
      <input name="login" type="text" spellcheck="false" autocomplete="off">
    </label>
  </p>
  <p>
    <label>
      <span>PASSWORD</span> <br>
      <input name="password" type="text" spellcheck="false" autocomplete="off">
    </label>
  </p>
  <input type="submit" value="SEND">
</form>
</body>
</html>
```

Листинг pageLk.hbs:

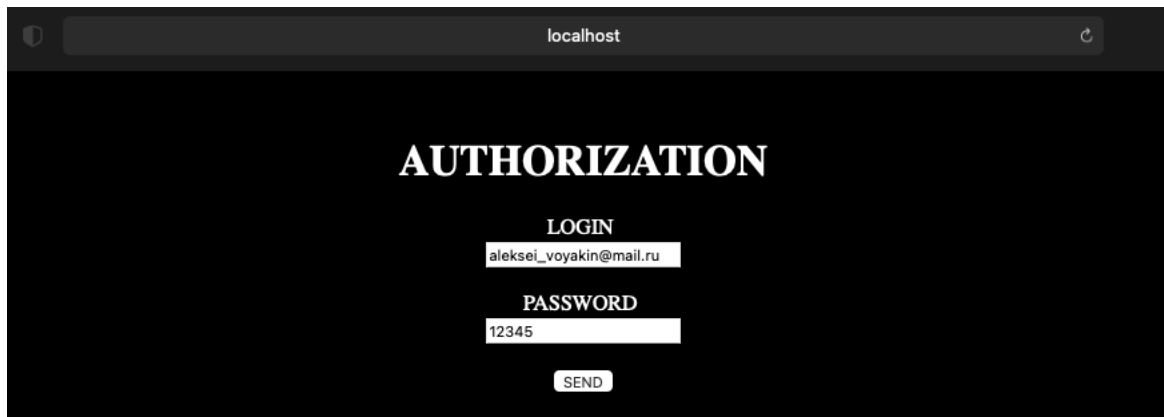
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>GAMES</title>
  <style>
    body {
      background-color: #000;
      color: #fff;
      text-align: center;
      padding: 20px;
    }
  </style>
</head>
<body>

<h2>USER INFO</h2>
<br>

LOGIN: {{login}}
<br>
HOBBY: {{hobby}}
<br>
AGE: {{age}}
<br>
<form method="GET" action="/api/delete">
  <br>
  <input type="submit" value="LOGOUT">
</form>

</body>
</html>
```

Зайдём на страницу <http://localhost:5015/lk>:



localhost

AUTHORIZATION

LOGIN

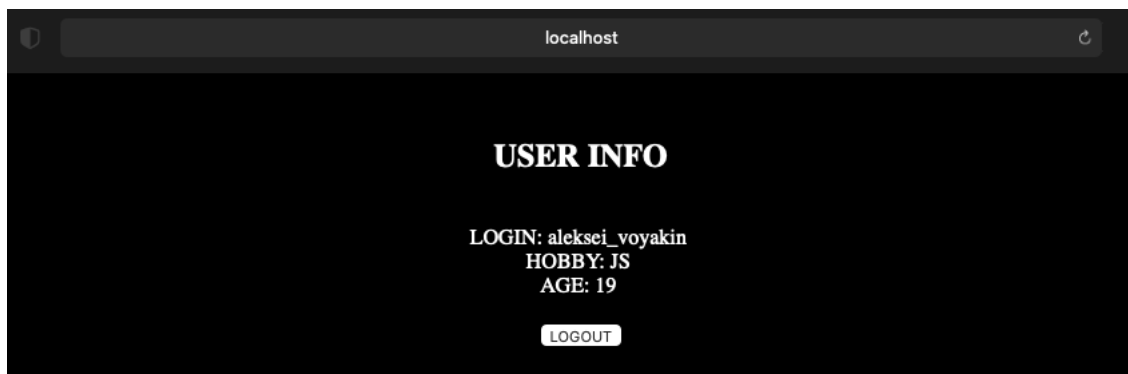
aleksei_voyakin@mail.ru

PASSWORD

12345

SEND

Мы попали на страницу входа, т.к. cookie еще не созданы. Авторизуемся:



localhost

USER INFO

LOGIN: aleksei_voyakin
HOBBY: JS
AGE: 19

LOGOUT

Мы попали на страничку личного кабинета, теперь если мы будем переходить на сайт <http://localhost:5015/lk> с новых страниц, или после закрытия браузера мы автоматически будем попадать на страницу личного кабинета до тех пор пока не нажмём кнопку “LOGOUT” или не истечёт время сессии.

Вывод: в ходе данной лабораторной работы я изучил AJAX запросы POST и GET и научился с ними работать. Во второй части ЛР я научился работать с cookie файлами (создавать удалять, проверять наличие).