|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **09.03.04 Программная инженерия**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 4 |

**Дисциплина:** Архитектура ЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Сучков А.Д. |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Попов А.Ю. |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Цель:** изучить взаимодействие между серверами и отправку запросов на другой сервер, а также рассмотреть передачу параметров скрипту и дочерние процессы. Реализовать программу на ЯП Prolog.

**Часть 1**

**Задание 1**

Создать сервер А. На стороне сервера хранится файл с содержимым в формате JSON. При получении запроса на /insert/record идёт добавление записи в файл. При получении запроса на /select/record идёт получение записи из файла. Каждая запись хранит информацию о машине (название и стоимость).

Создать сервер Б. На стороне сервера хранится файл с содержимым в формате JSON. Каждая запись в файле хранит информацию о складе и массиве машин, находящихся на данном складе. То есть каждая запись хранит в себе название склада (строку) и массив названий машин (массив строк). При получении запроса на /insert/record идёт добавление записи в файл. При получении запроса на /select/record идёт получение записи из файла.

Создать сервер C. Сервер выдаёт пользователю страницы с формами для ввода информации. При этом сервер взаимодействует с серверами А и Б. Реализовать для пользователя функции:

* создание нового типа машины
* получение информации о стоимости машины по её типу
* создание нового склада с находящимися в нём машинами
* получение информации о машинах на складе по названию склада

Реализовать удобный для пользователя интерфейс взаимодействия с системой (использовать поля ввода и кнопки).

**Листинг класса сервера A**

|  |
| --- |
| class ServerA {      static fs = require("fs");      static express = require("express");      constructor(port) {          this.app = ServerA.express();          this.port = port;          try {              this.app.listen(this);              console.log(` Starting server on port ${this.port}... `);          } catch (error) {              console.log(" Failure while starting server!");              throw new Error(' Port is unavalible!');          }            this.app.use(this.getHeaders);          this.app.use(ServerA.express.static(\_\_dirname + '/static'));          this.app.post('/insert/record', this.insertRecord);          this.app.post('/select/record', this.selectRecord);          console.log(" Server started succesfully!");      }      getHeaders(request, response, next) {          response.header("Cache-Control", "no-cache, no-store, must-revalidate");          response.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");          response.header("Access-Control-Allow-Origin", "\*");          next();      }      insertRecord(request, response) {          function loadBody(request, callback) {              let body = [];              request.on('data', (chunk) => {                  body.push(chunk);              }).on('end', () => {                  body = Buffer.concat(body).toString();                  callback(body);              });          }          loadBody(request, function(body) {              const obj = JSON.parse(body);              const name = obj.name;              const price = obj.price;              const storage\_path = "data/cars.json";              const fd = ServerA.fs.readFileSync(storage\_path, "utf8")              let storage = fd.length ? new Map(JSON.parse(fd)) : new Map();              const name\_exists = storage.has(name);              let added = false;              if (!name\_exists) {                  added = true;                  storage.set(name, price);                  ServerA.fs.writeFileSync(storage\_path, JSON.stringify([...storage]));              }              response.end(JSON.stringify({answer: added}));          });      }      selectRecord(request, response) {          function loadBody(request, callback) {              let body = [];              request.on('data', (chunk) => {                  body.push(chunk);              }).on('end', () => {                  body = Buffer.concat(body).toString();                  callback(body);              });          }          loadBody(request, function(body) {              const obj = JSON.parse(body);              const name = obj.name;              const storage\_path = "data/cars.json";              const fd = ServerA.fs.readFileSync(storage\_path, "utf8")              let storage = fd.length ? new Map(JSON.parse(fd)) : new Map();              let found = false;              let price;              if (storage.has(name)) {                  found = true;                  price = storage.get(name);              }              response.end(JSON.stringify({answer: found, price: price}));          });      }  } |

**Листинг класса сервера B**

|  |
| --- |
| class ServerB {      static fs = require("fs");      static express = require("express");      constructor(port) {          this.app = ServerB.express();          this.port = port;          try {              this.app.listen(this);              console.log(` Starting server on port ${this.port}... `);          } catch (error) {              console.log(" Failure while starting server!");              throw new Error(' Port is unavalible!');          }            this.app.use(this.getHeaders);          this.app.use(ServerB.express.static(\_\_dirname + '/static'));          this.app.post('/insert/record' , this.insertRecord);          this.app.post('/select/record', this.selectRecord);          console.log(" Server started succesfully!");      }      getHeaders(request, response, next) {          response.header("Cache-Control", "no-cache, no-store, must-revalidate");          response.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");          response.header("Access-Control-Allow-Origin", "\*");          next();      }      loadBody(request, callback) {          let body = [];          request.on('data', (chunk) => {              body.push(chunk);          }).on('end', () => {              body = Buffer.concat(body).toString();              callback(body);          });      }      insertRecord(request, response) {          function loadBody(request, callback) {              let body = [];              request.on('data', (chunk) => {                  body.push(chunk);              }).on('end', () => {                  body = Buffer.concat(body).toString();                  callback(body);              });          }          console.log(1);          loadBody(request, function(body) {              const obj = JSON.parse(body);              const name = obj.name;              const cars = obj.cars;              const storage\_path = "data/storage.json";              const fd = ServerB.fs.readFileSync(storage\_path, "utf8")              let storage = fd.length ? new Map(JSON.parse(fd)) : new Map();              console.log(name, cars);              console.log(storage);              const name\_exists = storage.has(name);              let added = false;              if (!name\_exists) {                  added = true;                  storage.set(name, cars);                  ServerB.fs.writeFileSync(storage\_path, JSON.stringify([...storage]));              }              response.end(JSON.stringify({answer: added}));          });      }      selectRecord(request, response) {          function loadBody(request, callback) {              let body = [];              request.on('data', (chunk) => {                  body.push(chunk);              }).on('end', () => {                  body = Buffer.concat(body).toString();                  callback(body);              });          }          loadBody(request, function(body) {              const obj = JSON.parse(body);              const name = obj.name;              const storage\_path = "data/storage.json";              const fd = ServerB.fs.readFileSync(storage\_path, "utf8")              let storage = fd.length ? new Map(JSON.parse(fd)) : new Map();              let found = false;              let cars;              if (storage.has(name)) {                  found = true;                  cars = storage.get(name);              }              response.end(JSON.stringify({answer: found,                                           cars: cars}));          });      }  } |

**Листинг класса сервера C**

|  |
| --- |
| class ServerC {      static fs = require("fs");      static express = require("express");      constructor(port) {          this.app = ServerC.express();          this.port = port;          try {              this.app.listen(this);              console.log(` Starting server on port ${this.port}... `);          } catch (error) {              console.log(" Failure while starting server!");              throw new Error(' Port is unavalible!');          }            this.app.use(this.getHeaders);          this.app.use(ServerC.express.static(\_\_dirname + '/static'));          this.app.post('/add\_car', this.addCar);          this.app.get('/get\_car', this.getCar);          this.app.post('/add\_storage', this.addStorage);          this.app.get('/get\_storage', this.getStorage);          console.log(" Server started succesfully!");      }      getHeaders(request, response, next) {          response.header("Cache-Control", "no-cache, no-store, must-revalidate");          response.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");          response.header("Access-Control-Allow-Origin", "\*");          next();      }      addCar(request, response) {          const name = request.query.name;          const price = request.query.price;          function sendPost(url, body, callback) {              const headers = {};              const requests = require("request");              headers["Cache-Control"] = "no-cache, no-store, must-revalidate";              headers["Connection"] = "close";              requests.post({                  url: url,                  body: body,                  headers: headers              }, function(error, response, body) {                  if (error) {                      callback(null);                  } else {                      callback(body);                  }              });          }          sendPost("http://localhost:5015/insert/record",                    JSON.stringify({name: name,                                    price: price      }), function(answerString) {              response.end(answerString);          });      }      getCar(request, response) {          const name = request.query.name;          function sendPost(url, body, callback) {              const headers = {};              const requests = require("request");              headers["Cache-Control"] = "no-cache, no-store, must-revalidate";              headers["Connection"] = "close";              requests.post({                  url: url,                  body: body,                  headers: headers              }, function(error, response, body) {                  if (error) {                      callback(null);                  } else {                      callback(body);                  }              });          }          sendPost("http://localhost:5015/select/record",                    JSON.stringify({name: name}),          function(answerString) {              response.end(answerString);          });      }      addStorage(request, response) {          const name = request.query.name;          const cars = request.query.cars;          function sendPost(url, body, callback) {              const headers = {};              const requests = require("request");              headers["Cache-Control"] = "no-cache, no-store, must-revalidate";              headers["Connection"] = "close";              requests.post({                  url: url,                  body: body,                  headers: headers              }, function(error, response, body) {                  if (error) {                      callback(null);                  } else {                      callback(body);                  }              });          }          sendPost("http://localhost:5020/insert/record",                    JSON.stringify({name: name,                                    cars: cars}),          function(answerString) {              response.end(answerString);          });      }      getStorage(request, response) {          const name = request.query.name;          const requests = require("request");          function sendPost(url, body, callback) {              const headers = {};              headers["Cache-Control"] = "no-cache, no-store, must-revalidate";              headers["Connection"] = "close";              requests.post({                  url: url,                  body: body,                  headers: headers              }, function(error, response, body) {                  if (error) {                      callback(null);                  } else {                      callback(body);                  }              });          }          sendPost("http://localhost:5020/select/record",                    JSON.stringify({name: name}),          function(answerString) {              response.end(answerString);          });      }  } |

**Листинг HTML страницы для добавления машин**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset="UTF-8">      <title>Добавить машину</title>      <link rel="stylesheet" type="text/css" href="      /stylesheets/style.css" />  </head>  <body>      <h1>Добавление машины</h1>      <p>Введите название машины</p>      <input id="name\_input" type="text" spellcheck="false" autocomplete="off">      <p>Введите стоимость машины</p>      <input id="price\_input" type="text" spellcheck="false" autocomplete="off">      <br>      <br>      <div id="add\_btn" class="btn-class">Добавить машину</div>      <br>      <br>      <h5 id="result\_label"></h5>      <script src="/scripts/add\_car.js"></script>  </body>  </html> |

**Листинг скрипта страницы для добавления машин**

|  |
| --- |
| "use strict";  window.onload = function() {      const name\_input = document.getElementById("name\_input");      const price\_input = document.getElementById("price\_input");      const btn = document.getElementById("add\_btn");      const label = document.getElementById("result\_label");      function ajaxGet(urlString, callback) {          let request = new XMLHttpRequest();          request.open("POST", urlString, true);          request.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");          request.send(null);          request.onload = function() {              callback(request.response);          };      };      btn.onclick = function() {          const name = name\_input.value;          const price = price\_input.value;          const url = `/add\_car?name=${name}&price=${price}`;          ajaxGet(url, function(stringAnswer) {              const objectAnswer = JSON.parse(stringAnswer);              const added = objectAnswer.answer;              label.innerHTML = added ? `Машина <font color="red">${name}</font> с ценой <font color="green">${price}</font> добавлена!` :                                        `Машина <font color="red">${name}</font> уже существует в базе!`;              });      };  }; |

**Листинг HTML страницы для добавления склада**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset="UTF-8">      <title>Добавить склад</title>      <link rel="stylesheet" type="text/css" href="      /stylesheets/style.css" />  </head>  <body>      <h1>Добавление склада</h1>      <p>Введите название склада</p>      <input id="name\_input" type="text" spellcheck="false" autocomplete="off">      <p>Введите список машин</p>      <input id="cars\_input" type="text" spellcheck="false" autocomplete="off">      <br>      <br>      <div id="add\_btn" class="btn-class">Добавить склад</div>      <br>      <br>      <h5 id="result\_label"></h5>      <script src="/scripts/add\_storage.js"></script>  </body>  </html> |

**Листинг скрипта страницы для добавления машин**

|  |
| --- |
| "use strict";  window.onload = function() {      const name\_input = document.getElementById("name\_input");      const cars\_input = document.getElementById("cars\_input");      const btn = document.getElementById("add\_btn");      const label = document.getElementById("result\_label");      function ajaxGet(urlString, callback) {          let request = new XMLHttpRequest();          request.open("POST", urlString, true);          request.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");          request.send(null);          request.onload = function() {              callback(request.response);          };      };      btn.onclick = function() {          const name = name\_input.value;          const cars = cars\_input.value.split(',');          const url = `/add\_storage?name=${name}&cars=${cars}`;          ajaxGet(url, function(stringAnswer) {              const objectAnswer = JSON.parse(stringAnswer);              const added = objectAnswer.answer;              label.innerHTML = added ? `Склад <font color="red">${name}</font> с машинами <font color="green">${cars}</font> добавлена!` :                                        `Склад <font color="red">${name}</font> уже существует в базе!`;              });      };  }; |

**Листинг HTML страницы для получения машин**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset="UTF-8">      <title>Найти машину</title>      <link rel="stylesheet" type="text/css" href="      /stylesheets/style.css" />  </head>  <body>      <h1>Получение машины</h1>      <p>Введите название машины</p>      <input id="name\_input" type="text" spellcheck="false" autocomplete="off">      <br>      <br>      <div id="add\_btn" class="btn-class">Найти машину</div>      <br>      <br>      <h5 id="result\_label"></h5>      <script src="/scripts/get\_car.js"></script>  </body>  </html> |

**Листинг скрипта страницы для получения машин**

|  |
| --- |
| "use strict";  window.onload = function() {      const name\_input = document.getElementById("name\_input");      const btn = document.getElementById("add\_btn");      const label = document.getElementById("result\_label");      function ajaxGet(urlString, callback) {          let request = new XMLHttpRequest();          request.open("GET", urlString, true);          request.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");          request.send(null);          request.onload = function() {              callback(request.response);          };      };      btn.onclick = function() {          const name = name\_input.value;          const url = `/get\_car?name=${name}`;          ajaxGet(url, function(stringAnswer) {              const objectAnswer = JSON.parse(stringAnswer);              const found = objectAnswer.answer;              const price = objectAnswer.price;              label.innerHTML = found ? `Машина <font color="red">${name}</font> с ценой <font color="green">${price}</font> найдена в базе!` :                                        `Машина <font color="red">${name}</font> не найдена в базе!`;              });      };  }; |

**Листинг HTML страницы для получения склада**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset="UTF-8">      <title>Найти склад</title>      <link rel="stylesheet" type="text/css" href="      /stylesheets/style.css" />  </head>  <body>      <h1>Получение склада</h1>      <p>Введите название склада</p>      <input id="name\_input" type="text" spellcheck="false" autocomplete="off">      <br>      <br>      <div id="add\_btn" class="btn-class">Найти склад</div>      <br>      <br>      <h5 id="result\_label"></h5>      <script src="/scripts/get\_storage.js"></script>  </body>  </html> |

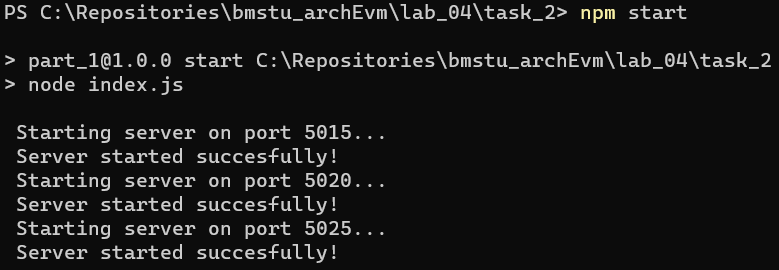
**Листинг скрипта страницы для получения склада**

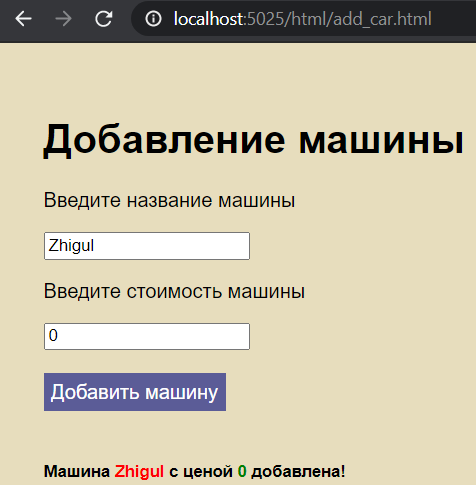
|  |
| --- |
| "use strict";  window.onload = function() {      const name\_input = document.getElementById("name\_input");      const btn = document.getElementById("add\_btn");      const label = document.getElementById("result\_label");      function ajaxGet(urlString, callback) {          let request = new XMLHttpRequest();          request.open("GET", urlString, true);          request.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");          request.send(null);          request.onload = function() {              callback(request.response);          };      };      btn.onclick = function() {          const name = name\_input.value;          const url = `/get\_storage?name=${name}`;          ajaxGet(url, function(stringAnswer) {              const objectAnswer = JSON.parse(stringAnswer);              const found = objectAnswer.answer;              const cars = objectAnswer.cars;              label.innerHTML = found ? `Склад <font color="red">${name}</font> с машинами <font color="green">${cars}</font> найден в базе!` :                                        `Склад <font color="red">${name}</font> не найден в базе!`;              });      };  }; |

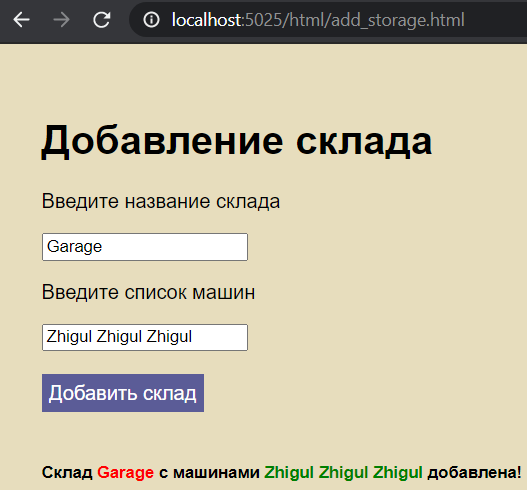
**Листинг CSS файла для страниц**

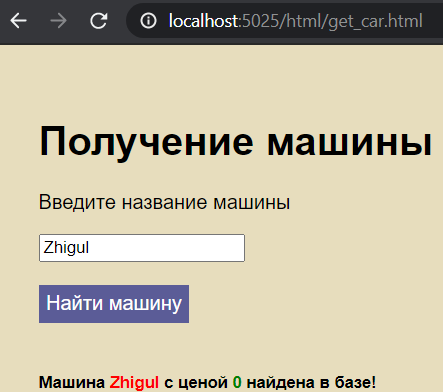
|  |
| --- |
| body {      padding: 30px;      background: rgb(231, 221, 189);      font-family: Geneva, Arial, Helvetica, sans-serif;  }  .btn-class {      padding: 6px;      background: rgb(91, 92, 151);      color: white;      cursor: pointer;      display: inline-block;  } |

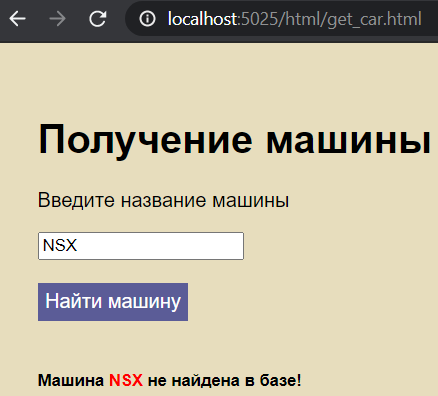
**Тесты**

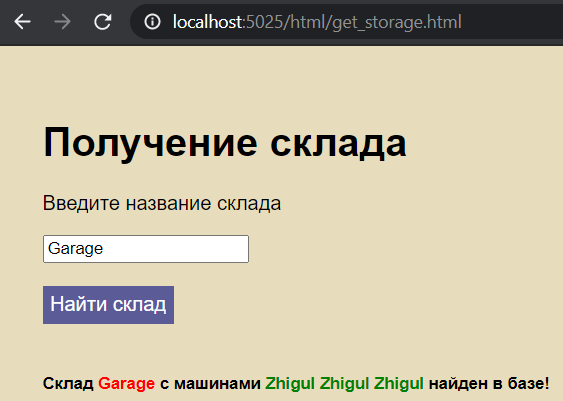


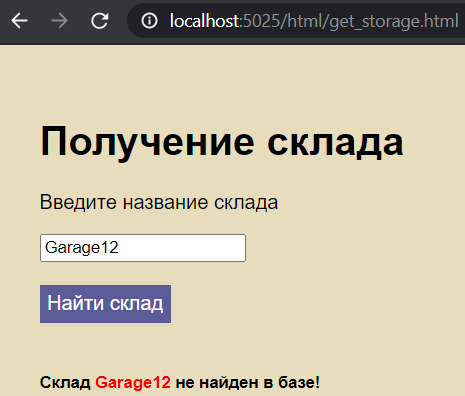












**Задание 2**

Написать скрипт, который принимает на вход число и считает его факториал. Скрипт должен получать параметр через **process.argv**.

Написать скрипт, который принимает на вход массив чисел и выводит на экран факториал каждого числа из массива. Скрипт принимает параметры через **process.argv**.

При решении задачи вызывать скрипт вычисления факториала через **execSync**.

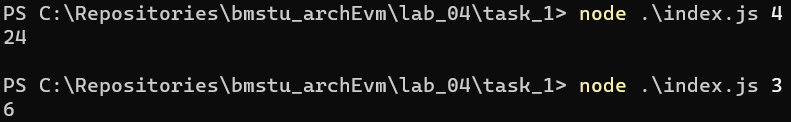
**Листинг index.js**

|  |
| --- |
| "use strict";  const execSync = require('child\_process').execSync;  function useCmd(s) {      const options = {encoding: 'utf8'};      const cmd = s.toString();      const answer = execSync(cmd, options);      return answer.toString();  }  let string = '';  for (let i = 2; i < process.argv.length; i++) {      const fact\_command = `node factorial.js ${process.argv[i]}`;      const fact = useCmd(fact\_command);      string += fact;  }  console.log(string); |

**Листинг factorial.js**

|  |
| --- |
| "use strict";  function factorial() {      let value = process.argv[2];      let result = value;      if (parseInt(value)&& value > 0) {          while (value > 2) {              value -= 1;              result \*= value;          }          console.log(result);      }  }  factorial(); |

**Тесты**



**Часть 2 Prolog**

**Задание 1**

С клавиатуры считываются числа **A** и **B**. Необходимо вывести на экран все **числа Фибоначчи**, которые принадлежат отрезку от **A** до **B**.

**Листинг**

|  |
| --- |
| ok.  input(A, B) :- read(A), read(B); ok.  printA(A, S) :- A >= S, write(A), nl; ok.  cicle(A, B, S, F) :- C is (A + B), printA(A, S), B =< F, cicle(B, C, S, F); ok.  start :- input(A, B), cicle(1, 1, A, B); ok. |

**Тесты**

****

**Вывод:** в ходе выполнения лабораторной работы было изучено взаимодействия между серверами и реализована отправка запросов на другой сервер. Также изучена и реализована передача параметров скрипту и вызов дочернего процесса. Реализована программа на ЯП Prolog.