Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

09.03.01 Информатика и вычислительная техника
код и наименование направления подготовки

ОТЧЕТ
по производственной практике

по направлению 09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) – «Программное обеспечение средств вычислительной техники и автоматизированных систем», квалификация – бакалавр,
программа академического бакалавриата,
форма обучения – очная, год начала подготовки (по учебному плану) – 2016

Выполнил:
студент гр. ИП-713
«11» июля 2020 г. _____ /Михеев Н.А./

Оценка «_____»

Руководитель практики
от университета
доцент Кафедры ПМиК
«11» июля 2020 г. _____ /Приставка П. А./

Новосибирск, 2020 г.

ПЛАН-ГРАФИК ПРОВЕДЕНИЯ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ

Тип практики: производственная практика по получению профессиональных умений и опыта профессиональной деятельности
Способ проведения практики: стационарная
Форма проведения практики: дискретно по периодам проведения практики

Тема: разработка серверной части приложения для взаимодействия с базой данных студентов

Содержание практики

Наименование видов деятельности	Дата (начало – окончание)
1. Общее ознакомление со структурным подразделением предприятия, вводный инструктаж по технике безопасности	03.02.2020–15.02.2020
2. Выдача задания на практику, деление студентов на группы (если необходимо), определение конкретной индивидуальной темы, формирование плана работ	17.02.2020–22.02.2020
3. Работа с библиотечными фондами структурного подразделения или предприятия, сбор и анализ материалов по теме практики	24.02.2020–21.03.2020
4. Выполнение работ в соответствии с составленным планом: 1) Разработка backend 2) Разработка frondend. 3) Доработка проекта.	 23.03.2020 – 11.04.2020 13.04.2020 - 8.05.2020 11.05.2020 – 30.05.2020
5. Анализ полученных результатов и произведенной работы	01.06.2020–06.06.2020
6. Составление отчета по практике, защита отчета	29.06.2020–11.07.2020

Согласовано:

Руководитель практики
от университета
доцент Кафедры ПМиК _____ /Приставка П. А./

СОДЕРЖАНИЕ

1. ЗАДАНИЕ НА ПРАКТИКУ

Реализовать клиент-серверное приложение для редактирования базы данных:

1) Пользователь должен иметь возможность видеть текущее состояние базы данных. Если нет соединения с базой данных, должно быть отображено сообщение об отсутствии соединения. Должна присутствовать кнопка обновления базы данных для получения текущего состояния базы данных. В самой базе данных должны присутствовать три поля, где будут храниться ФИО студентов, их возраст и интересы. Все данные о студентах и их информация должна храниться на сервере в базе данных.

2) На сайте пользователю должно быть предложено ввести свои данные: ФИО, возраст и интересы. Эти данные должны быть введены в отдельные помеченные поля. После ввода информации в поля пользователь должен нажать кнопку вставки данных для добавления информации в базу данных. Также на клиентской части должно быть предусмотрено удаление информации определенных пользователей с помощью отдельного поля для ввода ФИО.

2. ВВЕДЕНИЕ

Интернет распространился повсеместно и сейчас сложно найти людей, не имеющих к нему доступа. В связи с этим сегодня нужно уметь создавать даже простейшие сервисы, например, для хранения данных.

3. ОПИСАНИЕ ИЗУЧЕННЫХ ПРОГРАММНЫХ ПРОДУКТОВ И ТЕХНОЛОГИЙ ИСПОЛЬЗУЕМЫХ В РАБОТЕ ПРЕДПРИЯТИЯ

Для реализации клиентского приложения онлайн магазина будет использоваться язык Java.

На сегодняшний момент язык Java является одним из самых распространенных и популярных языков программирования. Первая версия языка появилась еще в 1996 году в недрах компании Sun Microsystems, впоследствии поглощенной компанией Oracle. Java задумывался как универсальный язык программирования, который можно применять для различного рода задач. И к настоящему времени язык Java проделал большой путь, было издано множество различных версий. Текущей версией является Java 12, которая вышла в марте 2019 года. А Java превратилась из просто универсального языка в целую платформу и экосистему, которая объединяет различные технологии, используемые в целого ряда задач: от создания десктопных приложений до написания крупных веб-порталов и сервисов. Кроме того, язык Java активно применяется для создания программного обеспечения для целого ряда устройств: обычных ПК, планшетов, смартфонов и мобильных телефонов и даже бытовой техники. Достаточно вспомнить популярность мобильной ОС Android, большинство программ для которой пишутся именно на Java.

MongoDB — документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++. Используется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MEAN.

HTML (от англ. *HyperText Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Для реализации определённого функционала взаимодействия сайта с пользователем будет использоваться JavaScript. JavaScript — мультипарадигменный язык программирования. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Кроме JavaScript на клиентской части используется и технология AJAX. AJAX — это аббревиатура, которая означает Asynchronous Javascript and XML. На самом деле, AJAX не является новой технологией, так как и Javascript, и XML существуют уже довольно продолжительное время, а AJAX — это синтез обозначенных технологий. AJAX чаще всего ассоциируется с термином Web 2.0 и преподносится как новейшее Web-приложение.

При использовании AJAX нет необходимости обновлять каждый раз всю страницу, так как обновляется только ее конкретная часть. Это намного удобнее, так как не приходится долго ждать, и экономичнее, так как не все обладают безлимитным интернетом.

4. РЕАЛИЗАЦИЯ ПРОЕКТА

4.1 Реализация клиентской части

Приложение для удобства пользователя должно иметь графический интерфейс для этого используется веб-страница. В основе будет лежать окно с таблицей студентов, где будут выведены их ФИО, возраст и интересы.

```
function getInfo() {
    $.getJSON(`http://localhost:8081/info`, function (data) {
        console.log(data);
        document.getElementById("header4").style.visibility='hidden'
        var text = `<table class="table table-bordered"><tr><th>ФИО</th><th>Возраст</th><th>Интересы</th></tr>`;
        for (var el = 0; el < data.length; el++) {
            text += `<tr><td>${data[el].name}</td><td>${data[el].age}</td><td>${data[el].interests}</td></tr>`
        }
        text += '</table>';
        $(".result").html(text);
    }).fail(function () {
        var text = `<p>No connection to the server :c Try again.</p>`;
        $(".result").html(text);
    });
}
```

Рис. 1 – функция для отправки GET

Данная функция используется для отправки GET-запроса для получения данных с сервера и занесения их в базу данных и вывода на сайт без перезагрузки страницы – динамический (AJAX).

```
function insertObject() {
    if (document.getElementById("name").value == 0 || document.getElementById("age").value == 0
        || document.getElementById("interests").value == 0) {
        alert("Blank space found. Check your input fields.")
        return;
    }
    $.getJSON("http://localhost:8081/insert?name=" + document.getElementById("name").value + "&age=" +
        document.getElementById("age").value + "&interests=" + document.getElementById("interests").value, function (data) {
        console.log(data);
        alert("Successfully inserted!")
        getInfo();
    }).fail(function () {
            alert("No connection to the server :c Try again.");
    });
}
```

Рис. 2 – вставка объекта

Данная функция предназначена для добавления в базу данных элементов, она принимает на вход данные из полей ввода на веб-странице. Если одно из полей для ввода – пустое, то функция не отправляет запрос с добавлением данных. Это тоже реализовано с помощью AJAX + JavaScript.

Также есть проверка на работоспособность сервера, в случае ошибки подключения клиент не завершит работу аварийно, а лишь попросит нажать кнопку добавления ещё раз.

```
function deleteEntry() {
    if (document.getElementById("delete").value == null) {
        alert("No delete name provided. Check your input field.")
        return;
    }
    $.getJSON("http://localhost:8081/delete?name=" + document.getElementById("delete").value, function (data) {
        console.log(data)
        alert("Successfully deleted!")
        getInfo()
    }).fail(function () {
        alert("No connection to the server or wrong entry name :c Try again.");
    });
}
```

Рис. 3 – удаление записи

Реализация практически аналогична функции вставки.

```
<h2 id="header2">Hello, User!</h2>
<h4 id="header4">For the first time you need to push the button "Get DB info"</h4>
<div class="row">
    <div class="col-md-6">
        <p class="result">Results would be shown here!</p>
    </div>
</div>
<button type="button" class="btn btn-primary mb-1" onclick="getInfo()">Update DB</button>
<div class="w-25">
    <form>
        <div class="input-group mb-1">
            <div class="input-group-prepend">
                <span class="input-group-text" id="basic-addon1">Name</span>
            </div>
            <input id="name" type="text" class="form-control" placeholder="Input Name" aria-label="Name" aria-describedby="basic-ad
        </div>
        <div class="input-group mb-1">
            <div class="input-group-prepend">
                <span class="input-group-text" id="basic-addon2">Age</span>
            </div>
            <input id="age" type="text" class="form-control" placeholder="Input Age" aria-label="Age" aria-describedby="basic-addon
        </div>
        <div class="input-group mb-1">
            <div class="input-group-prepend">
                <span class="input-group-text" id="basic-addon3">Interests</span>
            </div>
            <input id="interests" type="text" class="form-control" placeholder="Input Interests" aria-label="Interests" aria-descri
        </div>
    </form>
</div>
<button type="button" class="btn btn-success mb-1" onclick="insertObject()">Insert</button>
<div class="input-group mb-1 w-25">
    <div class="input-group-prepend">
        <button class="btn btn-danger" type="button" onclick="deleteEntry()">Delete</button>
    </div>
    <input id="delete" type="text" class="form-control" placeholder="Insert name of entry" aria-label="" aria-describedby="basic-ad
</div>
/body>
```

Рис. 4 – элементы веб-страницы

В данном блоке кода представлен Bootstrap. Это HTML/CSS/JS-фреймворк. Имеет набор CSS-стилей и JavaScript-скриптов для быстрого создания современных адаптивных сайтов.

4.2 Реализация серверной части

Для разработки сервера для данного проекта был использован язык Java и сопутствующие библиотеки для упрощения разработки: Project Lombok, ServerSocket, MongoDB-библиотеки и т.д.

Lombok — проект по добавлению дополнительной функциональности в Java с помощью изменения исходного кода перед Java компиляцией.

По сути, проект Lombok позволяет избавиться от многословности Java в большинстве случаев и перестать писать огромные количества кода из getters, setters, equals, hashcode и toString в результате Java становиться почти такой же краткой как Kotlin, Scala или C#.

Серверная часть является многопоточной, это позволяет отправлять на сервер множество запросов одновременно. Это работает с помощью создания новых потоков при каждом запросе.

```
public static void main(String[] args) throws IOException {
    ServerSocket server = new ServerSocket(8081);
    Socket client;
    int clientsNumber = 0;
    while (clientsNumber < 15) {
        client = server.accept();
        System.out.println("New Client Connected!");
        clientsNumber++;
        new Thread(new ServerThread(client)).start();
    }
    server.close();
}
```

Рис. 5 – создание потоков

Обработка каждого потока вынесена из класса Main, реализована в классе ServerThread. В этом классе полностью прописана логика работы сервера с запросами клиента.

В методах класса реализована поддержка MongoDB, обработка приёма запросов, подключение сервера к базе данных. Запросы отправляются в формате JSON.

JSON (JavaScript Object Notation) - простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 3rd Edition - December 1999.

5. РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРАКТИКИ

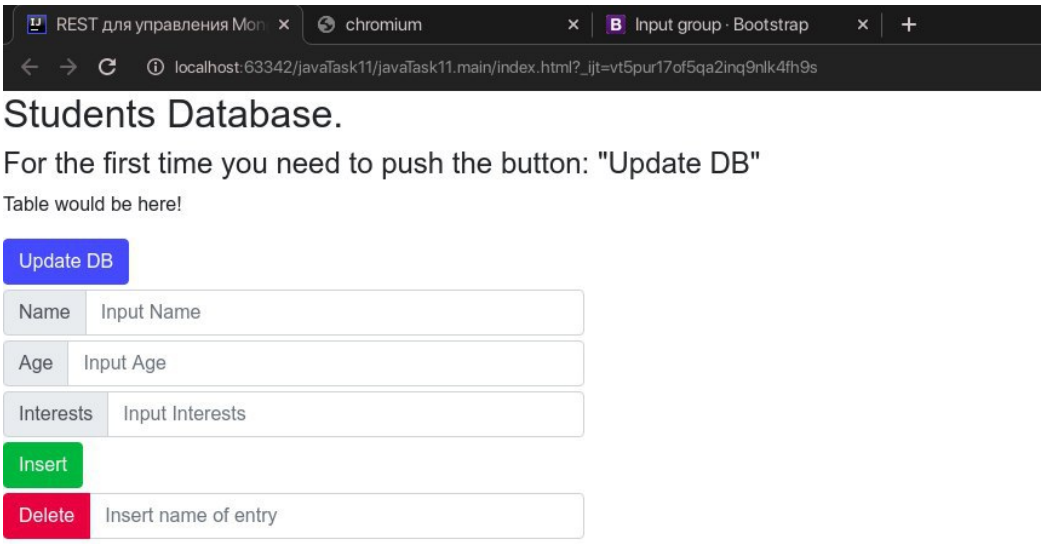

Рис. 6 – веб-страница без таблицы

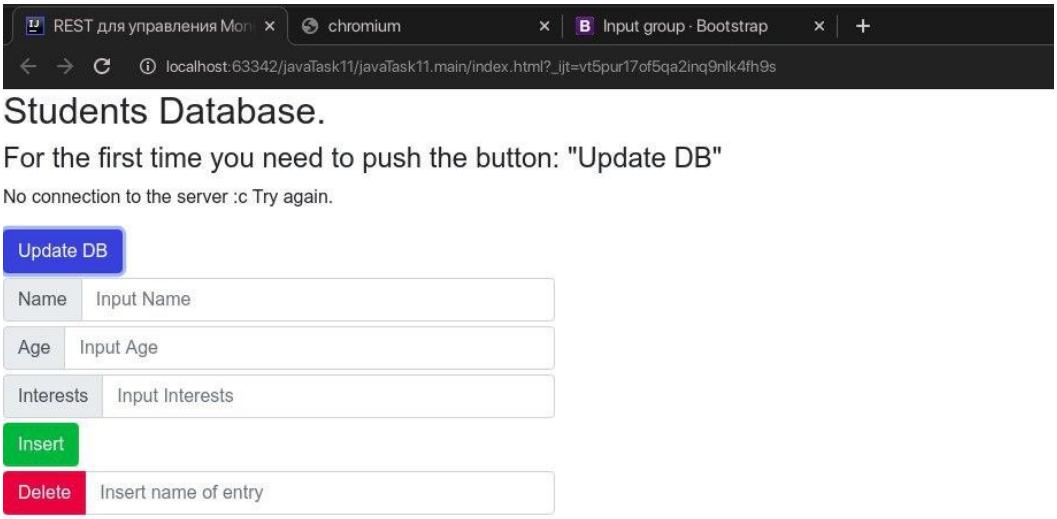

Рис. 7 – просьба нажать кнопку для обновления таблицы на вебсайте

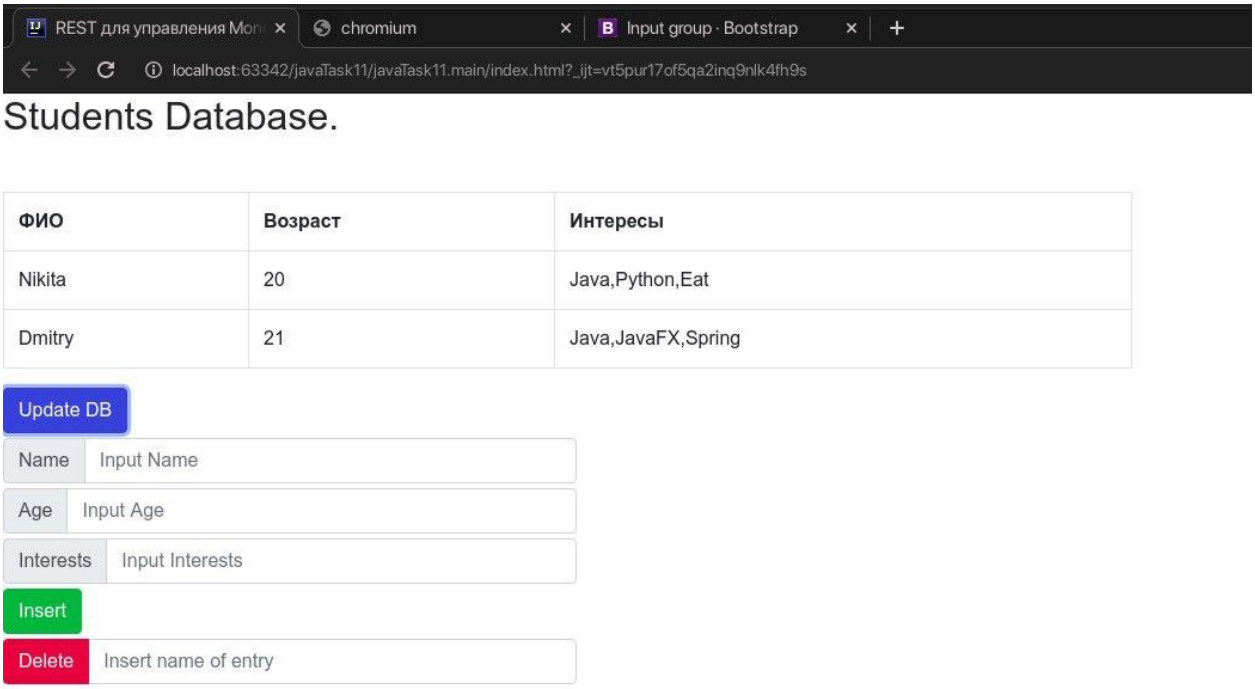

Рис. 8 – таблица студентов

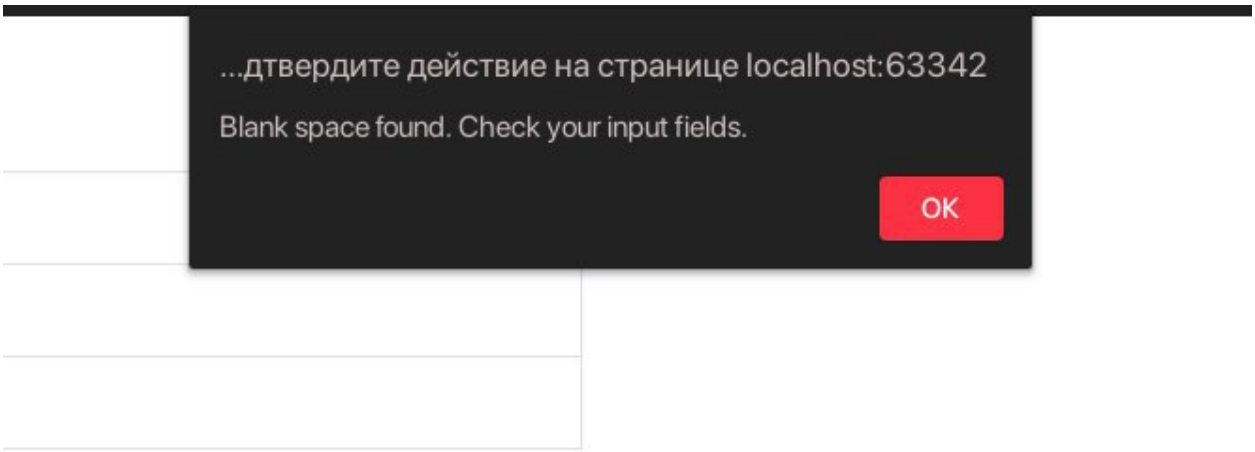

Рис. 9 – ошибка при добавлении пустой записи

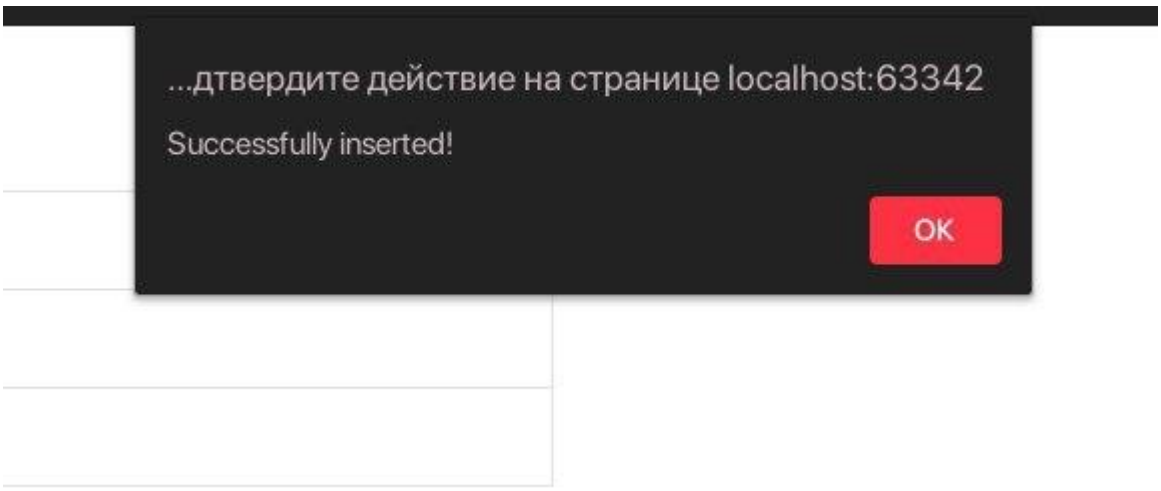

Рис. 10 – сообщение при успешном отправлении

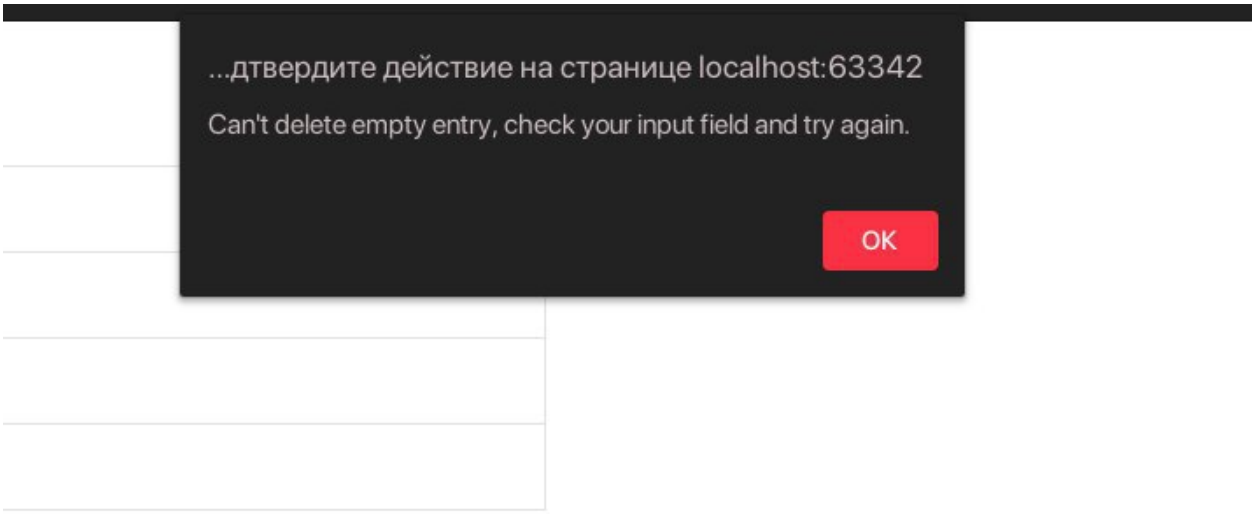

Рис. 11 – ошибка при удалении студента без заполнения имени в поле

```
New Client Connected!
Client request: GET /info HTTP/1.1
Formed response: HTTP/1.1 200 OK
Content-type: application/json
Access-Control-Allow-Origin: *
```

Рис. 12 – обработка запроса сервером

6. ЗАКЛЮЧЕНИЕ

В результате производственной практики был создан проект имеющий клиентскую и серверную части объединённые тесным взаимодействием. Кроме того, был получен опыт разработки при использовании технологий, использующихся в современных бизнес-приложениях.

7. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

Docs.oracle: сайт. – URL: https://docs.oracle.com/en/ (дата обращения 13.03.2020). – Текст: электронный.

Mongodb: сайт. – URL: https://www.mongodb.com/ (Дата обращения: 15.04.2020) – Текст: электронный.

8. ПРИЛОЖЕНИЕ

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>REST для управления MongoDB</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>
  </head>
  <body>
    <script>
      function getInfo() {
        $.getJSON(`http://localhost:8081/info`, function (data) {
          console.log(data);
          document.getElementById("header4").style.visibility='hidden'
          var text = `<table class="table table-
bordered"><tr><th>ФИО</th><th>Возраст</th><th>Интересы</th></tr>`;
          for (var el = 0; el < data.length; el++) {
            text +=
`<tr><td>${data[el].name}</td><td>${data[el].age}</td><td>${data[el].interests}</td></tr>`
          }
          text += '</table>';
          $(".result").html(text);
        }).fail(function () {
          var text = `<p>No connection to the server :c Try again.</p>`;
          $(".result").html(text);
        });
      }
```

```
        function insertObject() {
            if (document.getElementById("name").value == 0 || document.getElementById("age").value == 0
                || document.getElementById("interests").value == 0) {
                alert("Blank space found. Check your input fields.")
                return;
            }
            $.getJSON("http://localhost:8081/insert?name=" + document.getElementById("name").value + "&age=" +
                document.getElementById("age").value + "&interests=" + document.getElementById("interests").value,
function (data) {
                console.log(data);
                alert("Successfully inserted!")
                document.getElementById("name").value = 0
                document.getElementById("age").value = 0
                document.getElementById("interests").value = 0
                getInfo();
            }).fail(function () {
                    alert("No connection to the server :c Try again.");
            });
        }

        function deleteEntry() {
            if (document.getElementById("delete").value == null) {
                alert("No delete name provided. Check your input field.")
                return;
            }
            $.getJSON("http://localhost:8081/delete?name=" + document.getElementById("delete").value, function
(data) {
                console.log(data)
                alert("Successfully deleted!")
                getInfo()
            }).fail(function () {
                alert("Can't delete empty entry, check your input field and try again.");
            });
        }
    </script>
    <h2 id="header2">Students Database.</h2>
    <h4 id="header4">For the first time you need to push the button: "Update DB"</h4>
    <div class="row">
        <div class="col-md-6">
            <p class="result">Table would be here!</p>
        </div>
    </div>
    <button type="button" class="btn btn-primary mb-1" onclick="getInfo()">Update DB</button>
    <div class="w-25">
        <form>
            <div class="input-group mb-1">
                <div class="input-group-prepend">
                    <span class="input-group-text" id="basic-addon1">Name</span>
                </div>
                <input id="name" type="text" class="form-control" placeholder="Input Name" aria-label="Name" aria-
describedby="basic-addon1">
            </div>
            <div class="input-group mb-1">
                <div class="input-group-prepend">
                    <span class="input-group-text" id="basic-addon2">Age</span>
                </div>
                <input id="age" type="text" class="form-control" placeholder="Input Age" aria-label="Age" aria-
describedby="basic-addon2">
            </div>
            <div class="input-group mb-1">
                <div class="input-group-prepend">
```

```
                <span class="input-group-text" id="basic-addon3">Interests</span>
            </div>
            <input id="interests" type="text" class="form-control" placeholder="Input Interests" aria-
label="Interests" aria-describedby="basic-addon3">
        </div>
      </form>
    </div>
    <button type="button" class="btn btn-success mb-1" onclick="insertObject()">Insert</button>
    <div class="input-group mb-1 w-25">
      <div class="input-group-prepend">
        <button class="btn btn-danger" type="button" onclick="deleteEntry()">Delete</button>
      </div>
      <input id="delete" type="text" class="form-control" placeholder="Insert name of entry" aria-label="" aria-
describedby="basic-addon1">
    </div>
  </body>
</html>
```

Main.java

```
package ru.sibsutis;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;


public class Main {

    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(8081);
        Socket client;
        int clientsNumber = 0;
        while (clientsNumber < 15) {
            client = server.accept();
            System.out.println("New Client Connected!");
            clientsNumber++;
            new Thread(new ServerThread(client)).start();
        }
        server.close();
    }
}
```

ServerThread.java

```
package ru.sibsutis;

import com.mongodb.BasicDBObject;
import com.mongodb.client.*;
import lombok.*;
import org.bson.Document;

import java.io.*;
import java.net.Socket;
import java.util.*;



@Getter @Setter
```

```
@NoArgsConstructor
@AllArgsConstructor
class ServerThread implements Runnable {
   /** Client connection socket */
   Socket client;

   /**
    * New client connection handling.
    */
   public void _runThread() throws IOException {
      MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
      MongoDatabase db = mongoClient.getDatabase("user");
      MongoCollection<Document> collection = db.getCollection("profile");
      StringBuilder response = new StringBuilder();
      String JSON = "";

      Scanner in = new Scanner(client.getInputStream());
      String str = in.nextLine();
      System.out.println("Client request: " + str);

      str = str.substring(5);
      str = str.substring(0, str.length() - 9);
      str = str.replace("%20", "");

      int requestType = 0;
      if(str.contains("info"))
         requestType = 1;
      else if (str.contains("insert"))
         requestType = 2;
      else if (str.contains("delete"))
         requestType = 3;

      switch (requestType) {
         case 1:
            for (Document doc : collection.find()) {
               response.append(doc.toJson());
               response.append(",");
            }
            JSON += response.substring(0, response.length() - 1);
            sendResponse(JSON);
            break;
         case 2:
            String[] args = str.split("[?&]");
            String[] recordName = args[1].split("[=]");
            String[] recordAge = args[2].split("[=]");
            String[] recordInterest = args[3].split("[=]");
            List<String> allInterests = Arrays.asList(recordInterest[1].split("[,]"));
            Document record = new Document("name", recordName[1])
                  .append("age", recordAge[1])
                  .append("interests", allInterests);
            collection.insertOne(record);
            for (Document doc : collection.find()) {
               response.append(doc.toJson());
               response.append(",");
            }
            JSON += response.toString().substring(0, response.length() - 1);
            sendResponse(JSON);
            break;
         case 3:
            String[] arg = str.split("[?&]");
            String[] deleteName = arg[1].split("[=]");
            if (deleteName.length < 2) {
```

```
                System.out.println("Empty delete request");
                break;
            }
            BasicDBObject searchObject = new BasicDBObject();
            searchObject.append("name", deleteName[1]);
            collection.deleteOne(searchObject);
            for (Document doc : collection.find()) {
                response.append(doc.toJson());
                response.append(",");
            }
            JSON += response.toString().substring(0, response.length() - 1);
            sendResponse(JSON);
            break;
        default:
            JSON += "{\"Status\": \"ok\"}";
            sendResponse(JSON);
    }
    mongoClient.close();
    client.close();
  }

  private void sendResponse(String json) throws IOException {
    String response = "HTTP/1.1 200 OK\n" +
            "Content-type: application/json\n" +
            "Access-Control-Allow-Origin: *\n" +
            "\n" +
            "[" + json + "]";
    System.out.println("Formed response: " + response);
    PrintWriter out = new PrintWriter(client.getOutputStream());
    out.print(response);
    out.flush();
  }

  /**
   * Started thread.
   */
  @Override
  public void run() {
    try {
        _runThread();
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```