МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8

дисциплина «Программирование сетевых приложений»

на тему «Разработка HTTP­сервисов»

Выполнил: студент гр. ИТП-41

Болотникова Е.В.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2022

**Задание:** изучить основные принципы разработки http-сервисов и принципы создания архитектуры веб приложения.

.

**Задание:**

Разработать *HTTP*­сервис по выбранной теме ­ реализовать *CRUD* (*Create Read Update Delete*) операции.

*GET* - для получения данных

*POST* - для создания ресурса (сохранения данных)

*PUT* - для обновления ресурса (обновление данных)

*DELETE* - для удаления ресурса (удаления строк)

(рекомендуется использовать .*Net*, *Java* или *Python*, можно использовать другие языки программирования по согласованию с преподавателем). Приложение должно поставлять и получать данные в одном из следующих форматов: *XML*, *JSON*, *YAML*.

На сервере использовать любую БД для хранения данных, рекомендуется использовать *MySQL*, *PostgreSQL*, *MS SQL Server*. Приветствуется использование не реляционных СУБД (например, ***MongoDB***или *Casandra*)

Клиентское приложение: это *html*­страница, где запросы (вызовы сервиса) осуществляются с использованием *AJAX* (рекомендуется использовать *jQuery*). Для студентов претендующих на оценку 9­10 следует использовать любой из перечисленных *JavaScript* *framework*: *ReactJS*, *AngularJS*, *Backbone*, *KnockoutJS*, *AmberJS* или разработать мобильный клиент для *Android*, *iOS* или *Windows Phone*.

Индивидуалное задание: служащий с полями имя, возраст, стаж, место работы.

Результат работы приложения приведен на рисунке 1.

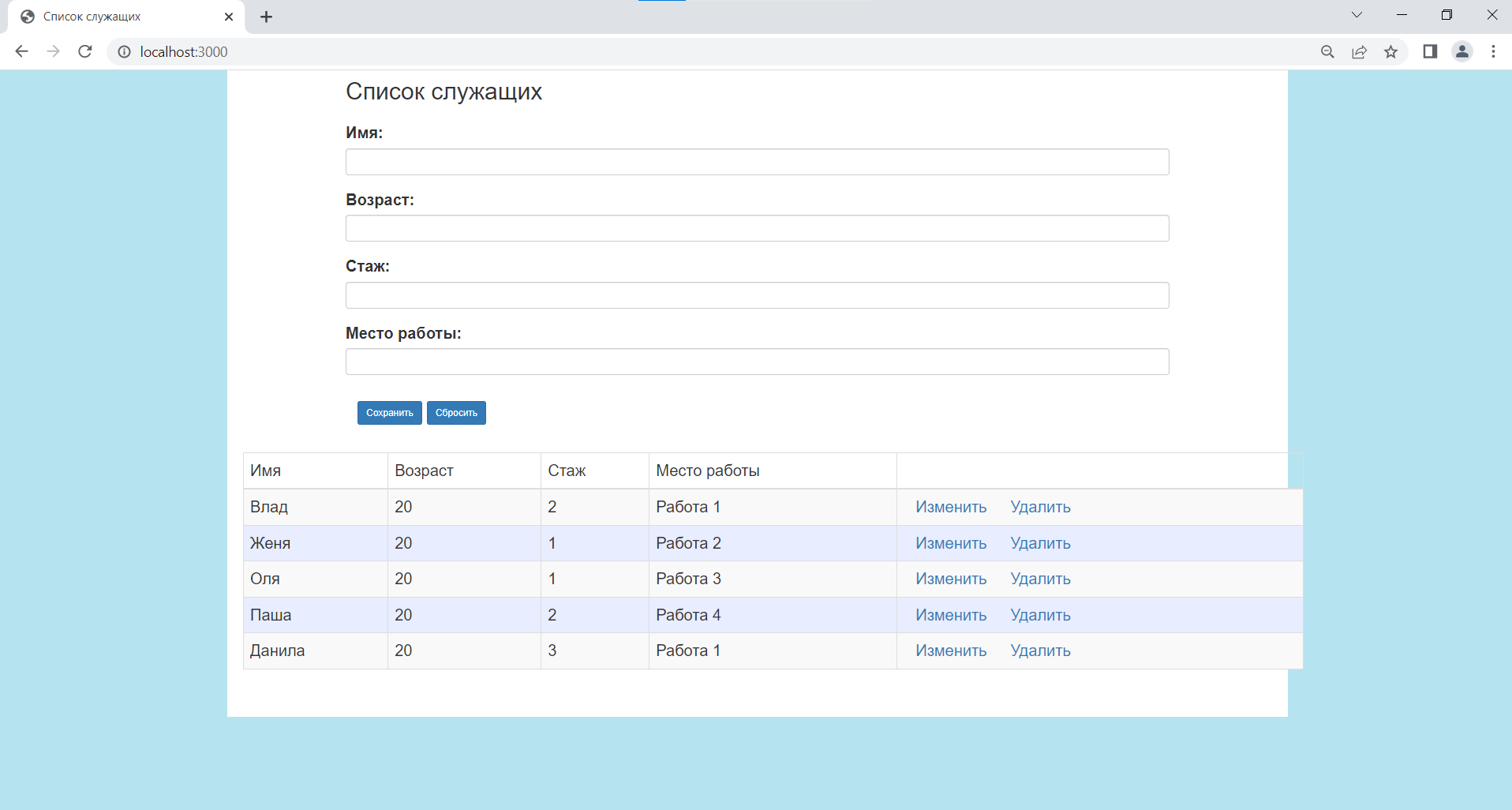


Рисунок 1 – Результат работы приложения

Результат создания записей в не реляционной СУБД *MongoDB* приведены на рисунке 2.

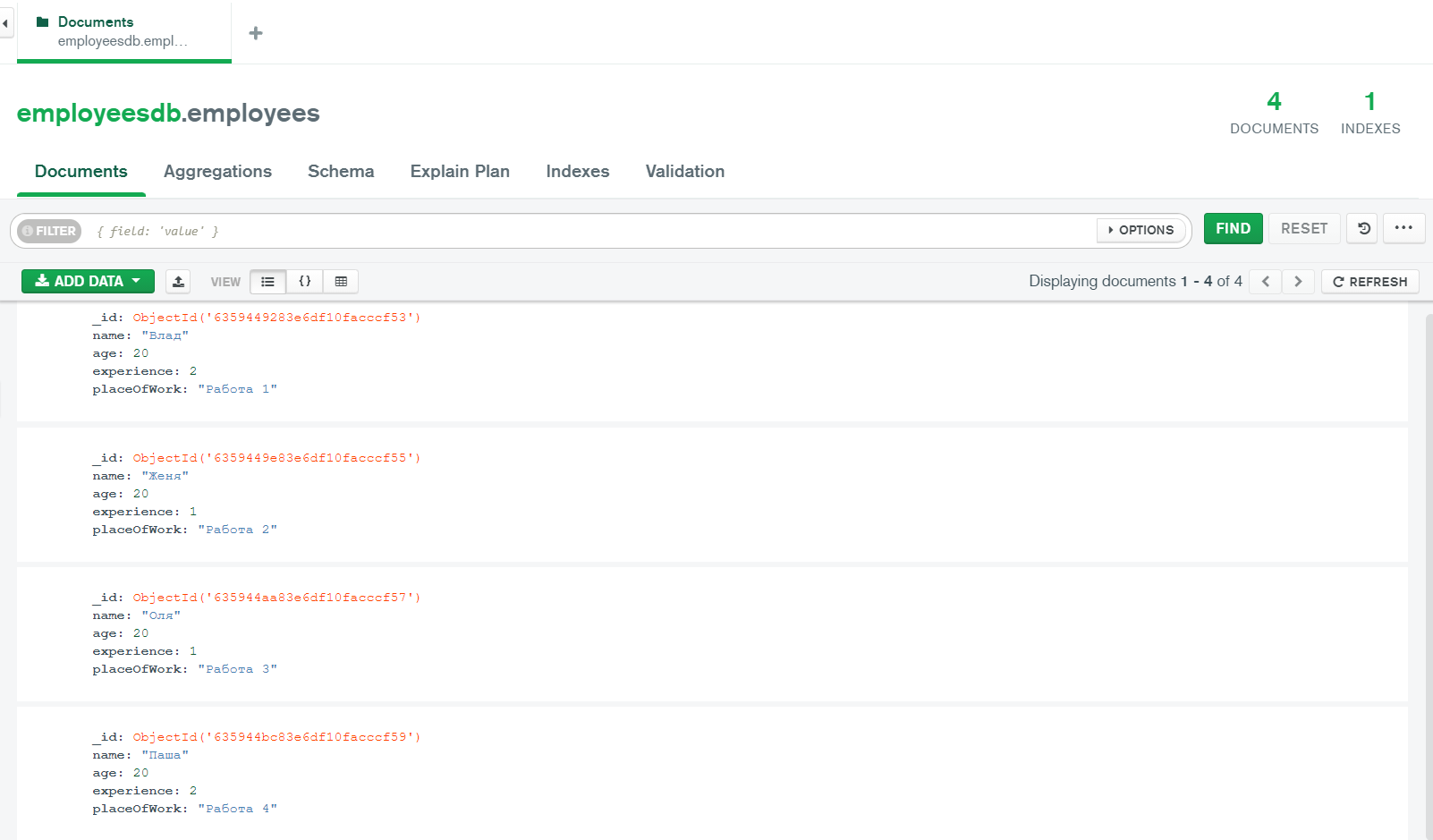


Рисунок 2 – Результат создания записей в не реляционной СУБД *MongoDB*

Листинг приложения приведён в приложении А.

**Вывод**: в процессе выполнения данной лабораторной работы были изучены основные принципы разработки клиент серверных приложений с использованием *JS* и базы данных *MongoDB*. А также полученные знания были применены на практике для разработки серверного приложения для обработки пользовательских запросов.

Приложение А

app.js:

const mongoose = require("mongoose");

const express = require("express");

const Schema = mongoose.Schema;

const app = express();

const jsonParser = express.json();

const employeeScheme = new Schema({name: String, age: Number, experience: Number, placeOfWork: String}, {versionKey: false});

const Employee = mongoose.model("Employee", employeeScheme);

app.use(express.static(\_\_dirname + "/public"));

mongoose.connect("mongodb://localhost:27017/employeesdb", { useUnifiedTopology: true, useNewUrlParser: true}, function(err){

    if(err) return console.log(err);

    app.listen(3000, function(){

        console.log("Сервер ожидает подключения...");

    });

});

app.get("/api/employees", function(req, res){

    Employee.find({}, function(err, employees){

        if(err) return console.log(err);

        res.send(employees)

    });

});

app.get("/api/employees/:id", function(req, res){

    const id = req.params.id;

    Employee.findOne({\_id: id}, function(err, employee){

        if(err) return console.log(err);

        res.send(employee);

    });

});

app.post("/api/employees", jsonParser, function (req, res) {

    if(!req.body) return res.sendStatus(400);

    const employeeName = req.body.name;

    const employeeAge = req.body.age;

    const employeeExperience = req.body.experience;

    const employeePlaceOfWork = req.body.placeOfWork;

    const employee = new Employee({name:  employeeName, age: employeeAge, experience: employeeExperience, placeOfWork: employeePlaceOfWork});

    employee.save(function(err){

        if(err) return console.log(err);

        res.send(employee);

    });

});

app.delete("/api/employees/:id", function(req, res){

    const id = req.params.id;

    Employee.findByIdAndDelete(id, function(err, employee){

        if(err) return console.log(err);

        res.send(employee);

    });

});

app.put("/api/employees", jsonParser, function(req, res){

    if(!req.body) return res.sendStatus(400);

    const id = req.body.id;

    const employeeName = req.body.name;

    const employeeAge = req.body.age;

    const employeeExperience = req.body.experience;

    const employeePlaceOfWork = req.body.placeOfWork;

    const newEmployee = {name:  employeeName, age: employeeAge, experience: employeeExperience, placeOfWork: employeePlaceOfWork};

    Employee.findOneAndUpdate({\_id: id}, newEmployee, {new: true}, function(err, employee){

        if(err) return console.log(err);

        res.send(employee);

    });

});

index.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width" />

    <title>Список пользователей</title>

    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet" />

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <header>

        <section>

          <div class="border-box"></div>

              <div id="label">Лабораторная работа №8</label>

          <div class="border-box"></div>

         </section>

    </header>

    <div id="content">

        <form name="employeeForm">

            <h2>Список служащих</h2>

            <input type="hidden" name="id" value="0" />

            <div class="form-group">

                <label for="name">Имя:</label>

                <input class="form-control" name="name" />

            </div>

            <div class="form-group">

                <label for="age">Возраст:</label>

                <input class="form-control" name="age" />

            </div>

            <div class="form-group">

                <label for="experience">Стаж:</label>

                <input class="form-control" name="experience" />

            </div>

            <div class="form-group">

                <label for="placeOfWork">Место работы:</label>

                <input class="form-control" name="placeOfWork" />

            </div>

            <div class="panel-body">

                <button type="submit" class="btn btn-sm btn-primary">Сохранить</button>

                <a id="reset" class="btn btn-sm btn-primary">Сбросить</a>

            </div>

        </form>

        <table class="table  table-striped table-bordered" >

            <thead><tr><th>Имя</th><th>Возраст</th><th>Стаж</th><th>Место работы</th><th></th></tr></thead>

            <tbody>

            </tbody>

        </table>

   </div>

    <script>

    // Получение всех пользователей

        async function GetEmployees() {

            // отправляет запрос и получаем ответ

            const response = await fetch("/api/employees", {

                method: "GET",

                headers: { "Accept": "application/json" }

            });

            // если запрос прошел нормально

            if (response.ok === true) {

                // получаем данные

                const employees = await response.json();

                let rows = document.querySelector("tbody");

                employees.forEach(employee => {

                    // добавляем полученные элементы в таблицу

                    rows.append(row(employee));

                });

            }

        }

        // Получение одного пользователя

        async function GetEmployee(id) {

            const response = await fetch("/api/employees/" + id, {

                method: "GET",

                headers: { "Accept": "application/json" }

            });

            if (response.ok === true) {

                const employee = await response.json();

                const form = document.forms["employeeForm"];

                form.elements["id"].value = employee.\_id;

                form.elements["name"].value = employee.name;

                form.elements["age"].value = employee.age;

                form.elements["experience"].value = employee.experience;

                form.elements["placeOfWork"].value = employee.placeOfWork;

            }

        }

        // Добавление пользователя

        async function CreateEmployee(employeeName, employeeAge, employeeExperience, employeePlaceOfWork) {

            const response = await fetch("api/employees", {

                method: "POST",

                headers: { "Accept": "application/json", "Content-Type": "application/json" },

                body: JSON.stringify({

                    name: employeeName,

                    age: parseInt(employeeAge, 10),

                    experience: parseInt(employeeExperience, 10),

                    placeOfWork: employeePlaceOfWork

                })

            });

            if (response.ok === true) {

                const employee = await response.json();

                reset();

                document.querySelector("tbody").append(row(employee));

            }

        }

        // Изменение пользователя

        async function EditEmployee(employeeId, employeeName, employeeAge, employeeExperience, employeePlaceOfWork) {

            const response = await fetch("api/employees", {

                method: "PUT",

                headers: { "Accept": "application/json", "Content-Type": "application/json" },

                body: JSON.stringify({

                    id: employeeId,

                    name: employeeName,

                    age: parseInt(employeeAge, 10),

                    experience: parseInt(employeeExperience, 10),

                    placeOfWork: employeePlaceOfWork

                })

            });

            if (response.ok === true) {

                const employee = await response.json();

                reset();

                document.querySelector("tr[data-rowid='" + employee.\_id + "']").replaceWith(row(employee));

            }

        }

        // Удаление пользователя

        async function DeleteEmployee(id) {

            const response = await fetch("/api/employees/" + id, {

                method: "DELETE",

                headers: { "Accept": "application/json" }

            });

            if (response.ok === true) {

                const employee = await response.json();

                document.querySelector("tr[data-rowid='" + employee.\_id + "']").remove();

            }

        }

        // сброс формы

        function reset() {

            const form = document.forms["employeeForm"];

            form.reset();

            form.elements["id"].value = 0;

        }

        // создание строки для таблицы

        function row(employee) {

            const tr = document.createElement("tr");

            tr.setAttribute("data-rowid", employee.\_id);

            const nameTd = document.createElement("td");

            nameTd.append(employee.name);

            tr.append(nameTd);

            const ageTd = document.createElement("td");

            ageTd.append(employee.age);

            tr.append(ageTd);

            const experienceTd = document.createElement("td");

            experienceTd.append(employee.experience);

            tr.append(experienceTd);

            const placeOfWorkTd = document.createElement("td");

            placeOfWorkTd.append(employee.placeOfWork);

            tr.append(placeOfWorkTd);

            const linksTd = document.createElement("td");

            const editLink = document.createElement("a");

            editLink.setAttribute("data-id", employee.\_id);

            editLink.setAttribute("style", "cursor:pointer;padding:15px;");

            editLink.append("Изменить");

            editLink.addEventListener("click", e => {

                e.preventDefault();

                GetEmployee(employee.\_id);

            });

            linksTd.append(editLink);

            const removeLink = document.createElement("a");

            removeLink.setAttribute("data-id", employee.\_id);

            removeLink.setAttribute("style", "cursor:pointer;padding:15px;");

            removeLink.append("Удалить");

            removeLink.addEventListener("click", e => {

                e.preventDefault();

                DeleteEmployee(employee.\_id);

            });

            linksTd.append(removeLink);

            tr.appendChild(linksTd);

            return tr;

        }

        // сброс значений формы

        document.getElementById("reset").click(function (e) {

            e.preventDefault();

            reset();

        })

        // отправка формы

        document.forms["employeeForm"].addEventListener("submit", e => {

            e.preventDefault();

            const form = document.forms["employeeForm"];

            const id = form.elements["id"].value;

            const name = form.elements["name"].value;

            const age = form.elements["age"].value;

            const experience = form.elements["experience"].value;

            const placeOfWork = form.elements["placeOfWork"].value;

            if (id == 0)

                CreateEmployee(name, age, experience, placeOfWork);

            else

                EditEmployee(id, name, age, experience, placeOfWork);

        });

        // загрузка пользователей

        GetEmployees();

    </script>

</body>

</html>