**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №6

**по дисциплине: «Разработка приложений баз данных для информационных систем»**

на тему: Создание Web-сервиса с использованием WEB API ASP.NET MVC (ASP.NET Core MVC) для работы с информацией из базы данных

Выполнил: студент гр. ИТП-31

Солодков М.А

Принял: доцент

Асенчик О.Д.

Гомель 2019

**Цель работы:** ознакомиться с возможностями .NET Core по получению, отображению и изменение данных при помощи технологии WEB API.

**Задание**

Создать с использованием технологии WEB API Web-приложение, в котором организовать получение, отображение и изменение данных.

Для этого необходимо:

Создать (использовать ранее разработанные в предыдущих лабораторных работах):

* 1. Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.
  2. Класс контекста данных для доступа к базе данных.

Создать контроллер для операций с данными таблицы, стоящей на стороне отношения «многие».

Клиентское приложение в виде HTML страницы для визуального выполнения всех операций с данными выбранной в п.2 таблицы, использующих запросы к методам разработанного контроллера.

**Ход работы**

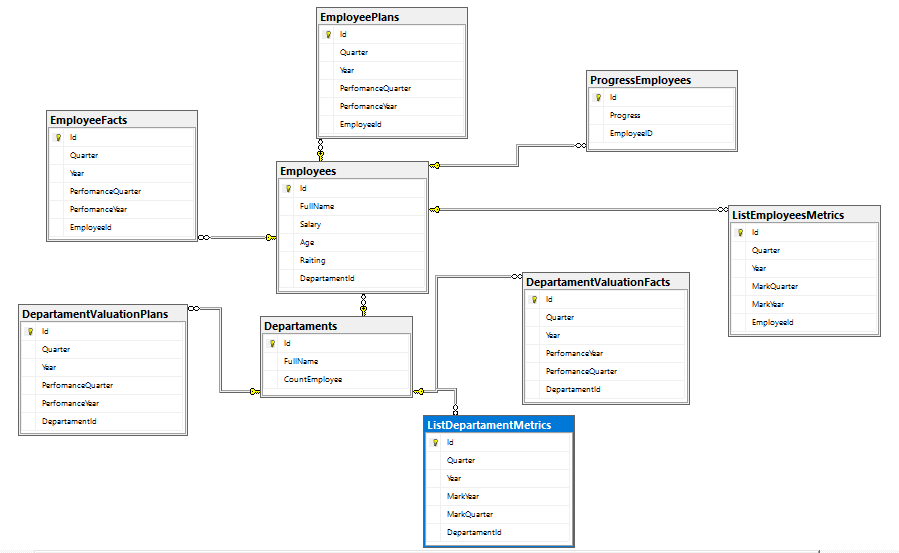


Рисунок 1 – Диаграмма базы данных

Исходный код классов представлен в приложении А.

**Результат выполнения:**

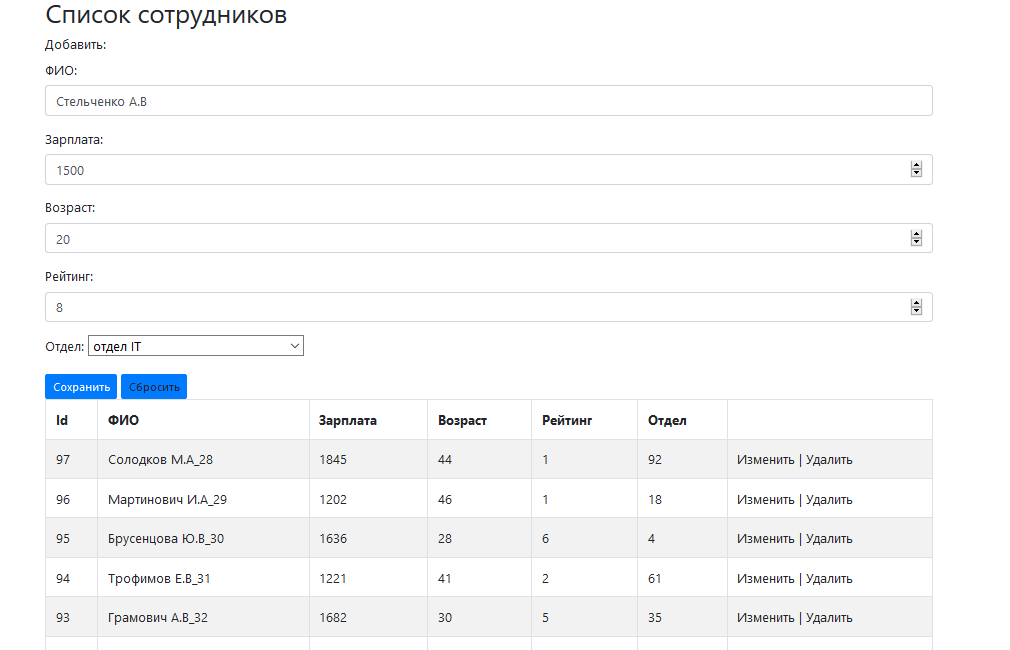


Рисунок 2 – Главная страница сайта

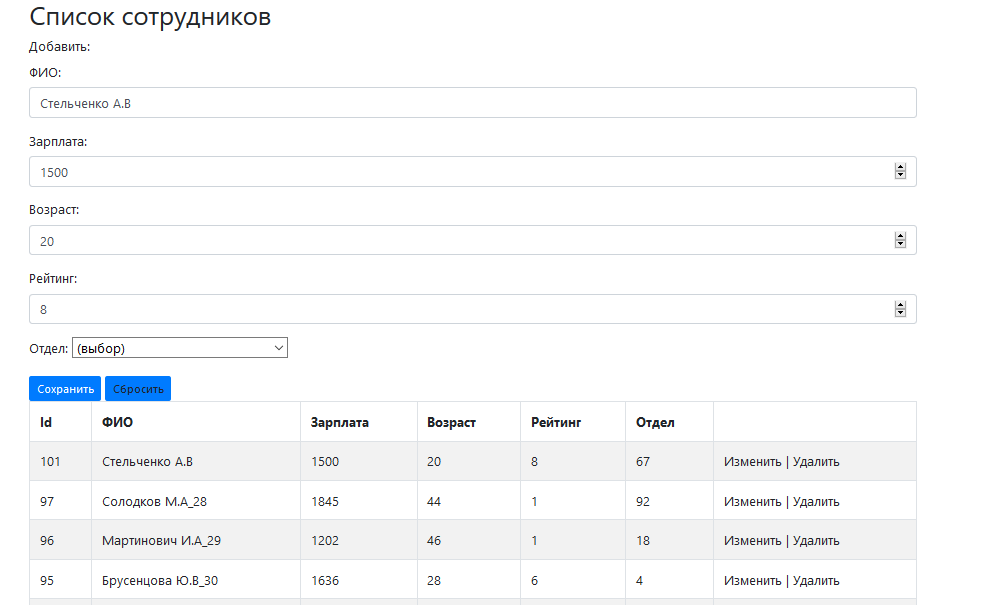


Рисунок 3 – Результат добавления

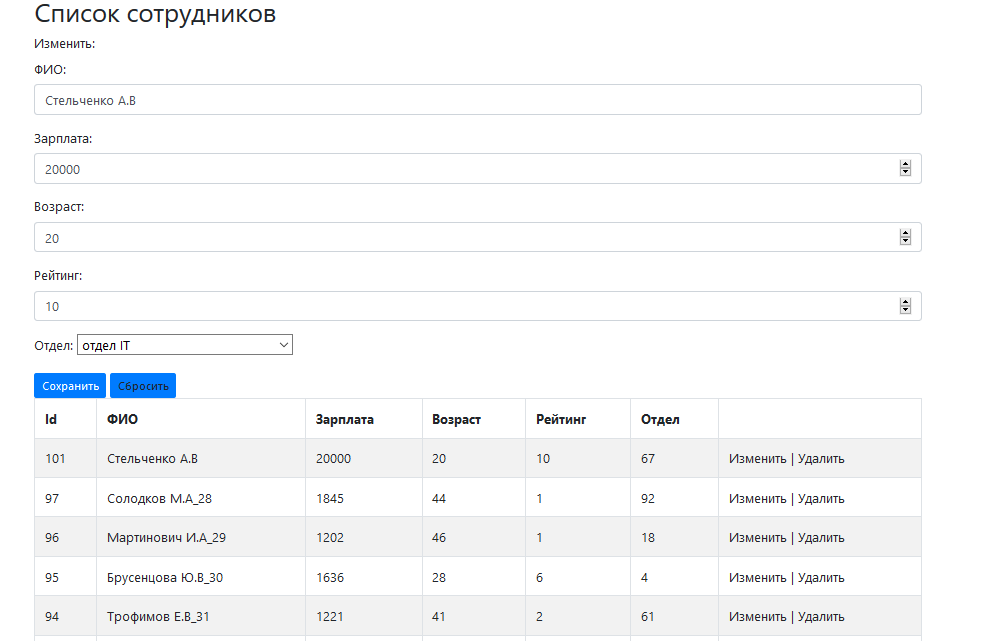


Рисунок 4 – Изменения информации

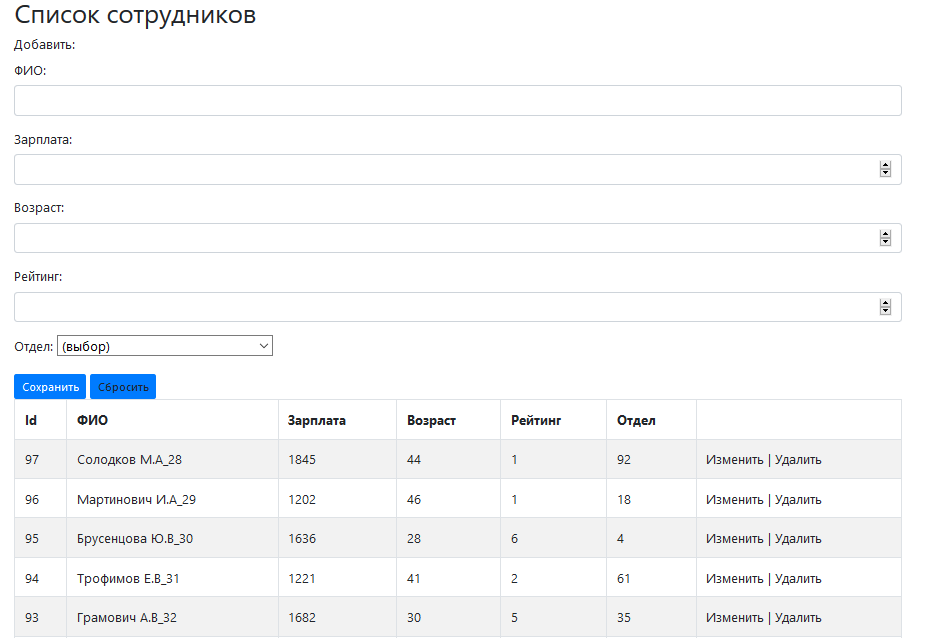


Рисунок 5 – Результат после изменения и удаления Стельченко А.В.

**Вывод:** ознакомились с возможностями .NET Core по получению, отображению и изменение данных при помощи технологии WEB API

**ПРИЛОЖЕНИЕ А**

**Код программы**

**Листинг программы Startup.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.SqlServer;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using RepairServiceCenterASP.Data;

namespace WebAPI

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

string connection = Configuration.GetConnectionString("DefaultConnection");

services.AddDbContext<RepairServiceCenterContext>(options => options.UseSqlServer(connection));

services.AddControllers();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseRouting();

app.UseAuthorization();

app.UseDefaultFiles();

app.UseStaticFiles();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

**Листинг программы EmployeeController.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using RepairServiceCenterASP.Data;

using RepairServiceCenterASP.Models;

using WebAPI.ViewModels;

namespace WebAPI.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class EmployeesController : ControllerBase

{

private readonly RepairServiceCenterContext \_context;

public EmployeesController(RepairServiceCenterContext context)

{

\_context = context;

}

// GET: api/Employees

[HttpGet]

public async Task<ActionResult<IEnumerable<EmployeeViewModel>>> GetEmployees()

{

var employees = \_context.Employees.Include(e => e.Post).Select(e =>

new EmployeeViewModel()

{

EmployeeId = e.EmployeeId,

FullName = e.FullName,

Experience = e.Experience.Value,

PostId = e.PostId.Value,

Post = e.Post.Name

}

);

return await employees.OrderByDescending(e => e.EmployeeId).ToListAsync();

}

[HttpGet("posts")]

[Produces("application/json")]

public async Task<ActionResult<IEnumerable<Post>>> GetPosts()

{

return await \_context.Posts.ToListAsync();

}

// GET: api/Employees/5

[HttpGet("{id}")]

public async Task<ActionResult<EmployeeViewModel>> GetEmployee(int id)

{

var employee = await \_context.Employees.Include(e => e.Post)

.Where(e => e.EmployeeId == id)

.FirstOrDefaultAsync();

if (employee == null)

{

return NotFound();

}

return new EmployeeViewModel()

{

FullName = employee.FullName,

Experience = employee.Experience.Value,

Post = employee.Post.Name,

PostId = employee.PostId.Value

};

}

// PUT: api/Employees/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://aka.ms/RazorPagesCRUD.

[HttpPut("{id}")]

public async Task<IActionResult> PutEmployee(int id, Employee employee)

{

if (id != employee.EmployeeId)

{

return BadRequest();

}

\_context.Entry(employee).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!EmployeeExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return Ok(employee);

}

// POST: api/Employees

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://aka.ms/RazorPagesCRUD.

[HttpPost]

public async Task<ActionResult<Employee>> PostEmployee(Employee employee)

{

\_context.Employees.Add(employee);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetEmployee", new { id = employee.EmployeeId }, employee);

}

// DELETE: api/Employees/5

[HttpDelete("{id}")]

public async Task<ActionResult<Employee>> DeleteEmployee(int id)

{

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

\_context.Employees.Remove(employee);

await \_context.SaveChangesAsync();

return employee;

}

private bool EmployeeExists(int id)

{

return \_context.Employees.Any(e => e.EmployeeId == id);

}

}

}

**Листинг страницы employees.html:**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

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

<title>Операции</title>

<link href="lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />

<script src="lib/jquery/dist/jquery.js"></script>

<link href="lib/jquery-ui/themes/base/jquery-ui.min.css" rel="stylesheet" />

<script src="lib/jquery-ui/jquery-ui.js"></script>

<script src="lib/jquery-ui/ui/i18n/datepicker-ru.js"></script>

</head>

<body>

<div class="container">

<h2>Список сотрудников</h2>

<form name="employeeForm">

<label for="employeeId" id="typeCRUD" name="typeCRUD">Добавить:</label>

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

<div class="form-group">

<label for="fullName">ФИО:</label>

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

</div>

<div class="form-group">

<label for="experience">Опыт работы:</label>

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

</div>

<div class="form-group">

<label Добавить>Должность:</label>

<select id="postId" name="postId"></select>

</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-condensed table-striped table-bordered">

<thead><tr><th>Id</th><th>ФИО</th><th>Опыт работы</th><th>Должность</th><th></th></tr></thead>

<tbody></tbody>

</table>

</div>

<script>

function getEmployees() {

$.ajax({

url: '/api/Employees',

type: 'GET',

contentType: "application/json",

success: function (employees) {

var rows = "";

$.each(employees, function (index, employee) {

rows += row(employee);

})

$("table tbody").append(rows);

}

});

}

function getEmployee(id) {

$.ajax({

url: '/api/employees/' + id,

type: 'GET',

contentType: "aplication/json",

success: function (employee) {

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

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

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

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

form.elements["postId"].selectedIndex = employee.postId;

}

});

};

function createEmployee(fullName, experience, postId) {

$.ajax({

url: "/api/employees",

contentType: "application/json",

method: "POST",

data: JSON.stringify({

fullName: fullName,

experience: experience,

postId: postId

}),

success: function (employee) {

reset();

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

employee.post = form.elements["postId"].options[postId].text;

$("table tbody").append(row(employee));

}

})

}

//editEmployee(9, "Стольный С.В.", 5, 2)

function editEmployee(employeeId, fullName, experience, postId) {

$.ajax({

url: "/api/employees/" + employeeId,

contentType: "application/json",

method: "PUT",

data: JSON.stringify({

employeeId: employeeId,

fullName: fullName,

experience: experience,

postId: postId

}),

success: function (employee) {

reset();

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

employee.post = form.elements["postId"].options[postId].text;

$("tr[data-rowid='" + employee.employeeId + "']").replaceWith(row(employee));

}

})

}

// сброс формы

function reset() {

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

form.reset();

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

$("#typeCRUD").html("Добавить:");

}

function deleteEmpolyee(id) {

$.ajax({

url: "/api/Employees/" + id,

contentType: "application/json",

method: "DELETE",

success: function (employee) {

$("tr[data-rowid='" + employee.employeeId + "']").remove();

}

})

}

var row = function (employee) {

return "<tr data-rowid='" + employee.employeeId + "'><td>" + employee.employeeId + "</td>" +

"<td>" + employee.fullName + "</td>" +

"<td>" + employee.experience + "</td>" +

"<td>" + employee.post + "</td>" +

"<td><a class='editLink' data-id='" + employee.employeeId + "'>Изменить</a> | " +

"<a class='removeLink' data-id='" + employee.employeeId + "'>Удалить</a></td></tr>";

}

function getPosts() {

var listItems = "";

$.ajax({

url: '/api/employees/posts',

type: 'GET',

contentType: "application/json",

success: function (posts) {

listItems = listItems + "<option value=0 selected>(выбор)</option>";

$.each(posts, function (index, post) {

listItems = listItems + "<option value='" + post.postId + "'>" + post.name + "</option>";

});

$("#postId").html(listItems);

}

});

}

$("#reset").click(function (e) {

e.preventDefault();

reset();

})

$("form").submit(function (e) {

e.preventDefault();

var employeeId = parseInt(this.elements["employeeId"].value);

var fullName = this.elements["fullName"].value;

var experience = parseInt(this.elements["experience"].value);

var postId = this.elements["postId"].selectedIndex;

if (employeeId == 0)

createEmployee(fullName, experience, postId);

else

editEmployee(employeeId, fullName, experience, postId);

});

// нажатие на ссылку Изменить

$("body").on("click", ".editLink", function () {

var id = $(this).data("id");

$("#typeCRUD").html("Изменить:");

getEmployee(id);

});

// нажатие на ссылку Удалить

$("body").on("click", ".removeLink", function () {

var id = $(this).data("id");

deleteEmpolyee(id);

reset();

});

getEmployees();

getPosts();

</script>

</body>

</html>