Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра ЭВМ

Отчет по лабораторной работе №15

на тему

«Создание клиентского веб-приложения»

по дисциплине  
**«Визуальное программирование»**

Выполнили:

Студенты группы №140

Бригада 5

Сафаров Д. А.

Тимохин Е. С.

Проверили:  
ст. преп. Хизриева Н.И.

ст. преп. Бастрычкин А.С.

**Цель работы**

Получение навыков работы с библиотекой Xamarin.Essentials.

**Задание**

Доработать клиентское веб-приложение в соответствии со следующими требованиями:

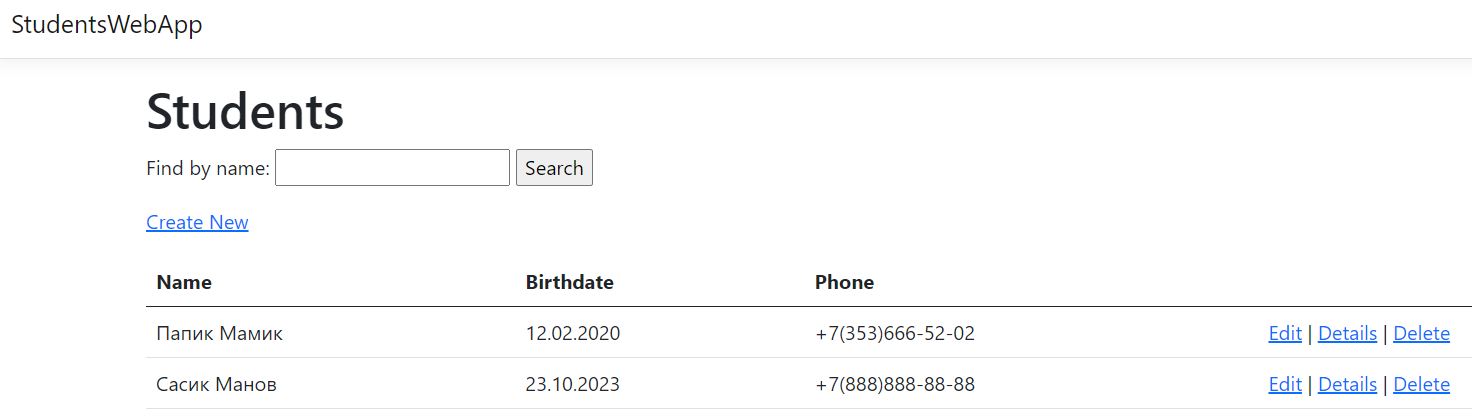
1) Необходимо убрать ненужные веб-страницы. При запуске приложения должна открываться страница со списком студентов (Students.Index).

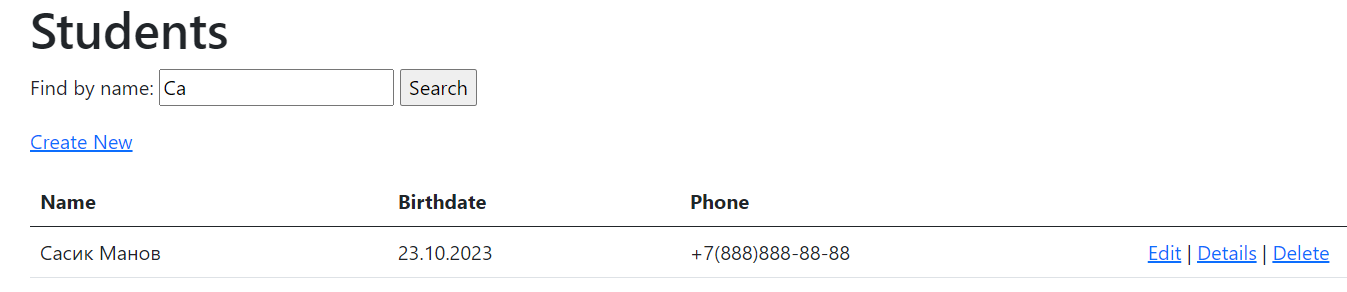
2) Добавить возможность поиска студента по ФИО.

3) Добавить проверку номера телефона на соответствие шаблону: +7(xxx)xxx-xx-xx

**Практическая часть**

Код программы представлен в приложении А. Результат её работы представлен на рисунке 1.





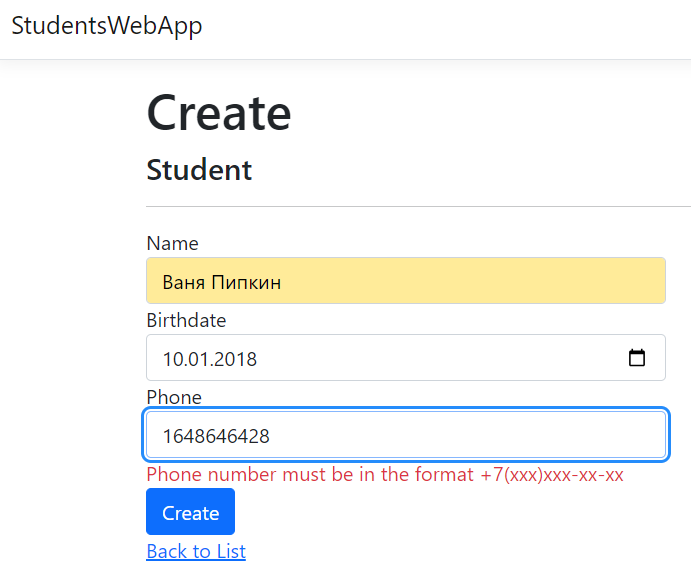


Рисунок 1 – Результат работы программы

Приложение А. Текст программы

**Student.cs**

using System.ComponentModel.DataAnnotations;

namespace StudentsWebApp.Models

{

public class Student

{

[Key]

public int Id { get; set; }

[Required]

[StringLength(100)]

public string Name { get; set; }

[Required]

[DataType(DataType.Date)]

public DateTime Birthdate { get; set; }

[Required]

[RegularExpression(@"\+7\(\d{3}\)\d{3}-\d{2}-\d{2}", ErrorMessage = "Phone number must be in the format +7(xxx)xxx-xx-xx")]

public string Phone { get; set; }

}

}

**StudentsController.cs**

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using StudentsWebApp.Data;

using StudentsWebApp.Models;

namespace StudentsWebApp.Controllers

{

public class StudentsController : Controller

{

private readonly StudentsWebAppContext \_context;

public StudentsController(StudentsWebAppContext context)

{

\_context = context;

}

// GET: Students

public async Task<IActionResult> Index(string searchString)

{

ViewData["CurrentFilter"] = searchString;

var students = from s in \_context.Student

select s;

if (!string.IsNullOrEmpty(searchString))

{

students = students.Where(s => s.Name.Contains(searchString));

}

return View(await students.ToListAsync());

}

// GET: Students/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var student = await \_context.Student

.FirstOrDefaultAsync(m => m.Id == id);

if (student == null)

{

return NotFound();

}

return View(student);

}

// GET: Students/Create

public IActionResult Create()

{

return View();

}

// POST: Students/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,Name,Birthdate,Phone")] Student student)

{

if (ModelState.IsValid)

{

\_context.Add(student);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(student);

}

// GET: Students/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var student = await \_context.Student.FindAsync(id);

if (student == null)

{

return NotFound();

}

return View(student);

}

// POST: Students/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Birthdate,Phone")] Student student)

{

if (id != student.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(student);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!StudentExists(student.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(student);

}

// GET: Students/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var student = await \_context.Student

.FirstOrDefaultAsync(m => m.Id == id);

if (student == null)

{

return NotFound();

}

return View(student);

}

// POST: Students/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var student = await \_context.Student.FindAsync(id);

if (student != null)

{

\_context.Student.Remove(student);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool StudentExists(int id)

{

return \_context.Student.Any(e => e.Id == id);

}

}

}

**Program.cs**

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.DependencyInjection;

using StudentsWebApp.Data;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<StudentsWebAppContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("StudentsWebAppContext") ?? throw new InvalidOperationException("Connection string 'StudentsWebAppContext' not found.")));

// Add services to the container.

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Students}/{action=Index}/{id?}");

app.Run();