**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №5

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 группы ИТП-31

Пронуза М.Ю.

Проверил:

Малиновский И.Л.

Гомель 2024

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

**Задание:**

1. 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):

o 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.

o Класс контекста данных для доступа к базе данных.

Создать контроллер для операций с данными таблицы, стоящей на

стороне отношения «многие».

Задокументировать API с помощью инструментов Swagger.

2. Создать проект для покрытия методов котроллера модульными тестами,

написать тесты и добиться их успешного прохождения.

3. Создать клиентское приложение в виде HTML страницы для визуального

выполнения всех операций с данными выбранной в п.2 таблицы, использующих

запросы к методам разработанного контроллера.

4. Создать и добавить в решение проект для покрытия методов котроллеров

модульными тестами, написать тесты и добиться их успешного прохождения.

5. Разместить созданное решение, содержащие проекты приложения и тестов, в

свой репозитории на GitHub, создать README.md файл для этого репозитория.

6. Используя средство GitHub Actions, написать рабочий процесс, который

будет осуществлять компиляцию решения и запуск тестов при любом

изменении в репозитории.

7. Отредактировать README.md файл опубликованного проекта, вставив в

него код для создания эмблемы состояния рабочего процесса (status badge),

показывающей, чем в данный момент завершился рабочий процесс

**Ход выполнения результаты работы:**

Текстовый вариант задания для варианта 17 представлен на рисунке 1.

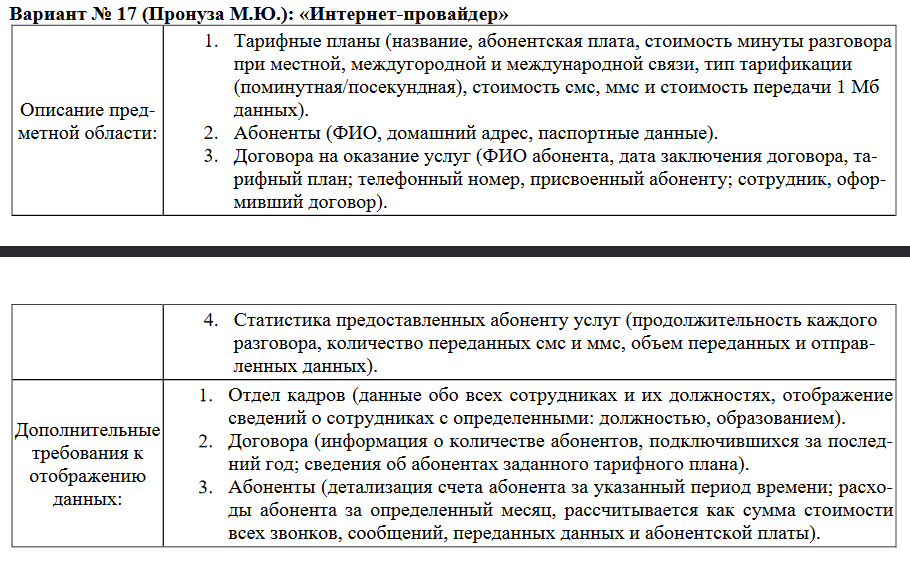


Рисунок 1 – Задание варианта 17

На рисунке 2 представлено *API* ображения к *CRUD* операциям в *Swagger*.

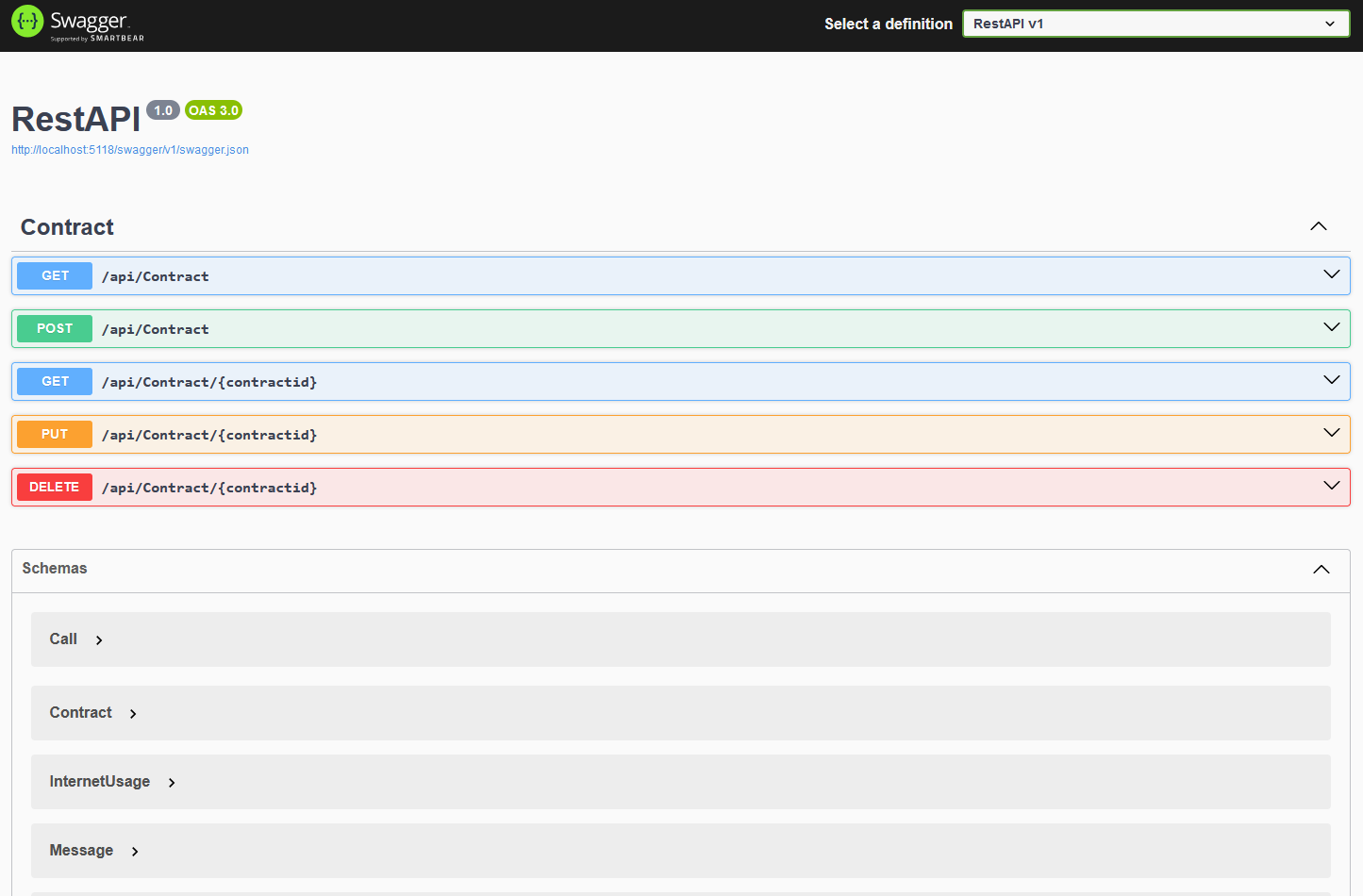


Рисунок 2 – представление *API* в *Swagger*

На рисунке 3 представлены результаты запроса на выборку таблицы контрактов из удаленной базы данных.

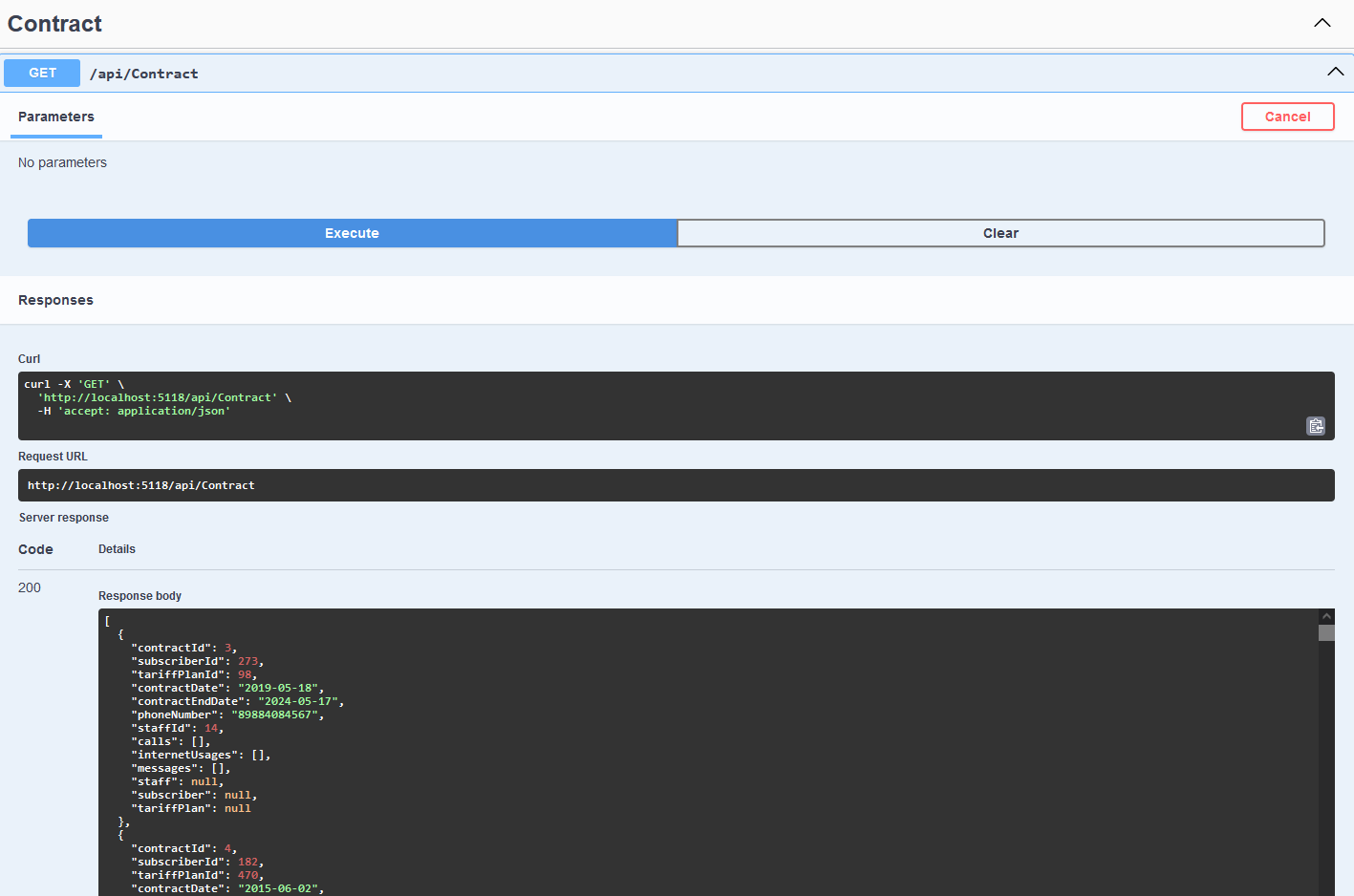


Рисунок 3 – выборка из таблицы контрактов

На рисунке 4 представлено успешное выполнение тестов.

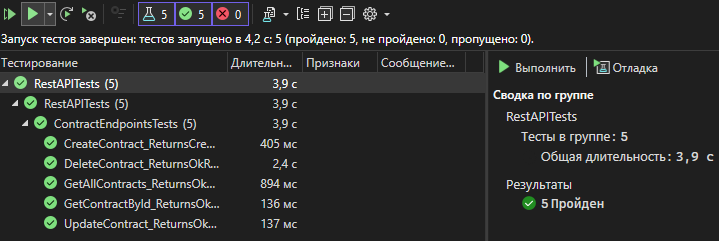


Рисунок 4 – выполнение тестов

**Вывод:** в ходе выполнения лабораторной работы ознакомились с возможностями .NET Core по получению отображению и изменение данных при помощи технологии WEB API.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Program.cs**

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.DependencyInjection;

using TelecomWeb.Middleware;

using TelecomWeb.Models;

using TelecomWeb.Services;

namespace TelecomWeb

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

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

options.UseSqlServer(builder.Configuration.GetConnectionString("TelecomDatabase")));

builder.Services.AddMemoryCache();

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession();

builder.Services.AddScoped<ICachedDataService, CachedDataService>();

builder.Services.AddIdentity<IdentityUser, IdentityRole>(options =>

{

options.Password.RequireDigit = false;

options.Password.RequiredLength = 4;

options.Password.RequireNonAlphanumeric = false;

options.Password.RequireUppercase = false;

options.Password.RequireLowercase = false;

})

.AddEntityFrameworkStores<TelecomDbContext>()

.AddDefaultUI()

.AddDefaultTokenProviders();

builder.Services.AddScoped<IdentitySeedData>();

builder.Services.AddControllersWithViews();

var app = builder.Build();

app.UseSession();

app.UseDbInitializer();

app.UseResponseCaching();

app.UseAuthentication();

app.UseAuthorization();

using (var scope = app.Services.CreateScope())

{

var initializer = scope.ServiceProvider.GetRequiredService<IdentitySeedData>();

initializer.Initialize();

}

if (!app.Environment.IsDevelopment())

{

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

}

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>{endpoints.MapRazorPages();});

app.MapControllerRoute(

name: "default",

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

app.Run();

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**TelecomDbContext.cs**

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

namespace TelecomWeb.Models;

public partial class TelecomDbContext : DbContext

{

public TelecomDbContext()

{

}

public TelecomDbContext(DbContextOptions<TelecomDbContext> options)

: base(options)

{

}

public virtual DbSet<Call> Calls { get; set; }

public virtual DbSet<Contract> Contracts { get; set; }

public virtual DbSet<ContractCall> ContractCalls { get; set; }

public virtual DbSet<ContractInternetUsage> ContractInternetUsages { get; set; }

public virtual DbSet<ContractMessage> ContractMessages { get; set; }

public virtual DbSet<EmployeeInfo> EmployeeInfos { get; set; }

public virtual DbSet<InternetUsage> InternetUsages { get; set; }

public virtual DbSet<Message> Messages { get; set; }

public virtual DbSet<Staff> Staff { get; set; }

public virtual DbSet<StaffPosition> StaffPositions { get; set; }

public virtual DbSet<Subscriber> Subscribers { get; set; }

public virtual DbSet<SubscriberInfo> SubscriberInfos { get; set; }

public virtual DbSet<TariffPlan> TariffPlans { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

#warning To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see https://go.microsoft.com/fwlink/?LinkId=723263.

=> optionsBuilder.UseSqlServer("Server=db8328.public.databaseasp.net; Database=db8328; User Id=db8328; Password=5z?X=4Mb3m-Q; Encrypt=True; TrustServerCertificate=True; MultipleActiveResultSets=True;");

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.UseCollation("Cyrillic\_General\_CI\_AS");

modelBuilder.Entity<Call>(entity =>

{

entity.HasKey(e => e.CallId).HasName("PK\_\_Calls\_\_5180CF8A3D2D2ED2");

entity.Property(e => e.CallId).HasColumnName("CallID");

entity.Property(e => e.CallDate).HasColumnType("datetime");

entity.Property(e => e.ContractId).HasColumnName("ContractID");

entity.HasOne(d => d.Contract).WithMany(p => p.Calls)

.HasForeignKey(d => d.ContractId)

.HasConstraintName("FK\_\_Calls\_\_ContractI\_\_14B10FFA");

});

modelBuilder.Entity<Contract>(entity =>

{

entity.HasKey(e => e.ContractId).HasName("PK\_\_Contract\_\_C90D34095C346A02");

entity.Property(e => e.ContractId).HasColumnName("ContractID");

entity.Property(e => e.PhoneNumber).HasMaxLength(20);

entity.Property(e => e.StaffId).HasColumnName("StaffID");

entity.Property(e => e.SubscriberId).HasColumnName("SubscriberID");

entity.Property(e => e.TariffPlanId).HasColumnName("TariffPlanID");

entity.HasOne(d => d.Staff).WithMany(p => p.Contracts)

.HasForeignKey(d => d.StaffId)

.HasConstraintName("FK\_\_Contracts\_\_Staff\_\_11D4A34F");

entity.HasOne(d => d.Subscriber).WithMany(p => p.Contracts)

.HasForeignKey(d => d.SubscriberId)

.HasConstraintName("FK\_\_Contracts\_\_Subsc\_\_0FEC5ADD");

entity.HasOne(d => d.TariffPlan).WithMany(p => p.Contracts)

.HasForeignKey(d => d.TariffPlanId)

.HasConstraintName("FK\_\_Contracts\_\_Tarif\_\_10E07F16");

});

modelBuilder.Entity<ContractCall>(entity =>

{

entity

.HasNoKey()

.ToView("ContractCalls");

entity.Property(e => e.CallDate).HasColumnType("datetime");

entity.Property(e => e.CallId).HasColumnName("CallID");

entity.Property(e => e.ContractId).HasColumnName("ContractID");

entity.Property(e => e.PhoneNumber).HasMaxLength(20);

});

modelBuilder.Entity<ContractInternetUsage>(entity =>

{

entity

.HasNoKey()

.ToView("ContractInternetUsage");

entity.Property(e => e.ContractId).HasColumnName("ContractID");

entity.Property(e => e.DataReceivedMb)

.HasColumnType("decimal(10, 2)")

.HasColumnName("DataReceivedMB");

entity.Property(e => e.DataSentMb)

.HasColumnType("decimal(10, 2)")

.HasColumnName("DataSentMB");

entity.Property(e => e.PhoneNumber).HasMaxLength(20);

entity.Property(e => e.UsageDate).HasColumnType("datetime");

entity.Property(e => e.UsageId).HasColumnName("UsageID");

});

modelBuilder.Entity<ContractMessage>(entity =>

{

entity

.HasNoKey()

.ToView("ContractMessages");

entity.Property(e => e.ContractId).HasColumnName("ContractID");

entity.Property(e => e.IsMms).HasColumnName("IsMMS");

entity.Property(e => e.MessageDate).HasColumnType("datetime");

entity.Property(e => e.MessageId).HasColumnName("MessageID");

entity.Property(e => e.PhoneNumber).HasMaxLength(20);

});

modelBuilder.Entity<EmployeeInfo>(entity =>

{

entity

.HasNoKey()

.ToView("EmployeeInfo");

entity.Property(e => e.Education).HasMaxLength(100);

entity.Property(e => e.FullName).HasMaxLength(150);

entity.Property(e => e.PositionName).HasMaxLength(100);

entity.Property(e => e.StaffId).HasColumnName("StaffID");

});

modelBuilder.Entity<InternetUsage>(entity =>

{

entity.HasKey(e => e.UsageId).HasName("PK\_\_Internet\_\_29B197C0F4400AED");

entity.ToTable("InternetUsage");

entity.Property(e => e.UsageId).HasColumnName("UsageID");

entity.Property(e => e.ContractId).HasColumnName("ContractID");

entity.Property(e => e.DataReceivedMb)

.HasColumnType("decimal(10, 2)")

.HasColumnName("DataReceivedMB");

entity.Property(e => e.DataSentMb)

.HasColumnType("decimal(10, 2)")

.HasColumnName("DataSentMB");

entity.Property(e => e.UsageDate).HasColumnType("datetime");

entity.HasOne(d => d.Contract).WithMany(p => p.InternetUsages)

.HasForeignKey(d => d.ContractId)

.HasConstraintName("FK\_\_InternetU\_\_Contr\_\_1A69E950");

});

modelBuilder.Entity<Message>(entity =>

{

entity.HasKey(e => e.MessageId).HasName("PK\_\_Messages\_\_C87C037C0167DB96");

entity.Property(e => e.MessageId).HasColumnName("MessageID");

entity.Property(e => e.ContractId).HasColumnName("ContractID");

entity.Property(e => e.IsMms).HasColumnName("IsMMS");

entity.Property(e => e.MessageDate).HasColumnType("datetime");

entity.HasOne(d => d.Contract).WithMany(p => p.Messages)

.HasForeignKey(d => d.ContractId)

.HasConstraintName("FK\_\_Messages\_\_Contra\_\_178D7CA5");

});

modelBuilder.Entity<Staff>(entity =>

{

entity.HasKey(e => e.StaffId).HasName("PK\_\_Staff\_\_96D4AAF7835BD125");

entity.Property(e => e.StaffId).HasColumnName("StaffID");

entity.Property(e => e.Education).HasMaxLength(100);

entity.Property(e => e.FullName).HasMaxLength(150);

entity.Property(e => e.PositionId).HasColumnName("PositionID");

entity.HasOne(d => d.Position).WithMany(p => p.Staff)

.HasForeignKey(d => d.PositionId)

.HasConstraintName("FK\_\_Staff\_\_PositionI\_\_093F5D4E");

});

modelBuilder.Entity<StaffPosition>(entity =>

{

entity.HasKey(e => e.PositionId).HasName("PK\_\_StaffPos\_\_60BB9A59D4D3AD5C");

entity.ToTable("StaffPosition");

entity.Property(e => e.PositionId).HasColumnName("PositionID");

entity.Property(e => e.PositionName).HasMaxLength(100);

});

modelBuilder.Entity<Subscriber>(entity =>

{

entity.HasKey(e => e.SubscriberId).HasName("PK\_\_Subscrib\_\_7DFEB634B3799F70");

entity.Property(e => e.SubscriberId).HasColumnName("SubscriberID");

entity.Property(e => e.FullName).HasMaxLength(150);

entity.Property(e => e.HomeAddress).HasMaxLength(255);

entity.Property(e => e.PassportData).HasMaxLength(100);

});

modelBuilder.Entity<SubscriberInfo>(entity =>

{

entity

.HasNoKey()

.ToView("SubscriberInfo");

entity.Property(e => e.ContractId).HasColumnName("ContractID");

entity.Property(e => e.HomeAddress).HasMaxLength(255);

entity.Property(e => e.PassportData).HasMaxLength(100);

entity.Property(e => e.PhoneNumber).HasMaxLength(20);

entity.Property(e => e.SubscriberFullName).HasMaxLength(150);

entity.Property(e => e.SubscriberId).HasColumnName("SubscriberID");

entity.Property(e => e.TariffName).HasMaxLength(100);

});

modelBuilder.Entity<TariffPlan>(entity =>

{

entity.HasKey(e => e.TariffPlanId).HasName("PK\_\_TariffPl\_\_29A9282A1C614268");

entity.Property(e => e.TariffPlanId).HasColumnName("TariffPlanID");

entity.Property(e => e.DataRatePerMb)

.HasColumnType("decimal(10, 2)")

.HasColumnName("DataRatePerMB");

entity.Property(e => e.InternationalCallRate).HasColumnType("decimal(10, 2)");

entity.Property(e => e.LocalCallRate).HasColumnType("decimal(10, 2)");

entity.Property(e => e.LongDistanceCallRate).HasColumnType("decimal(10, 2)");

entity.Property(e => e.MmsRate).HasColumnType("decimal(10, 2)");

entity.Property(e => e.SmsRate).HasColumnType("decimal(10, 2)");

entity.Property(e => e.SubscriptionFee).HasColumnType("decimal(10, 2)");

entity.Property(e => e.TariffName).HasMaxLength(100);

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

**ПРИЛОЖЕНИЕ В**

(обязательное)

**SessionExtensions.cs**

using Microsoft.AspNetCore.Http;

using Newtonsoft.Json;

namespace AspWeb.Infrastructure

{

// Методы расширения для ISession для работы с произвольными объектами

public static class SessionExtensions

{

//Запись произвольного объекта в сессию

public static void Set<T>(this ISession session, string key, T value)

{

session.SetString(key, JsonConvert.SerializeObject(value));

}

//Считывание произвольного объекта из сессии

public static T Get<T>(this ISession session, string key)

{

var value = session.GetString(key);

return value == null ? default : JsonConvert.DeserializeObject<T>(value);

}

}

}

**ПРИЛОЖЕНИЕ Г**

(обязательное)

**CachedDataService.cs**

using Microsoft.Extensions.Caching.Memory;

using TelecomApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

namespace TelecomApp.Services

{

public class CachedDataService : ICachedDataService

{

private readonly Db8328Context \_dbContext;

private readonly IMemoryCache \_memoryCache;

public CachedDataService(Db8328Context dbContext, IMemoryCache memoryCache)

{

\_dbContext = dbContext;

\_memoryCache = memoryCache;

}

public IEnumerable<Contract> GetContracts(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Contract> contracts))

{

contracts = \_dbContext.Contracts.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, contracts, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return contracts;

}

public IEnumerable<InternetUsage> GetInternetUsages(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<InternetUsage> internetUsages))

{

internetUsages = \_dbContext.InternetUsages.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, internetUsages, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return internetUsages;

}

public IEnumerable<Message> GetMessages(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Message> messages))

{

messages = \_dbContext.Messages.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, messages, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return messages;

}

public IEnumerable<Call> GetCalls(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Call> calls))

{

calls = \_dbContext.Calls.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, calls, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return calls;

}

public IEnumerable<Staff> GetStaff(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Staff> staff))

{

staff = \_dbContext.Staff.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, staff, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return staff;

}

public IEnumerable<StaffPosition> GetStaffPositions(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<StaffPosition> staffPositions))

{

staffPositions = \_dbContext.StaffPositions.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, staffPositions, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return staffPositions;

}

public IEnumerable<Subscriber> GetSubscribers(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Subscriber> subscribers))

{

subscribers = \_dbContext.Subscribers.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, subscribers, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return subscribers;

}

public IEnumerable<TariffPlan> GetTariffPlans(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<TariffPlan> tariffPlans))

{

tariffPlans = \_dbContext.TariffPlans.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, tariffPlans, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return tariffPlans;

}

}

}

**ПРИЛОЖЕНИЕ Д**

(обязательное)

**ICachedDataService.cs**

using TelecomApp.Models;

using System.Collections.Generic;

namespace TelecomApp.Services

{

public interface ICachedDataService

{

IEnumerable<Contract> GetContracts(string cacheKey, int rowsNumber = 20);

IEnumerable<InternetUsage> GetInternetUsages(string cacheKey, int rowsNumber = 20);

IEnumerable<Message> GetMessages(string cacheKey, int rowsNumber = 20);

IEnumerable<Call> GetCalls(string cacheKey, int rowsNumber = 20);

IEnumerable<Staff> GetStaff(string cacheKey, int rowsNumber = 20);

IEnumerable<StaffPosition> GetStaffPositions(string cacheKey, int rowsNumber = 20);

IEnumerable<Subscriber> GetSubscribers(string cacheKey, int rowsNumber = 20);

IEnumerable<TariffPlan> GetTariffPlans(string cacheKey, int rowsNumber = 20);

}

}

**ПРИЛОЖЕНИЕ Е**

(обязательное)

**DbInitializerMiddleware.cs**

using TelecomWeb.Data;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using TelecomWeb.Models;

namespace TelecomWeb.Middleware

{

public class DbInitializerMiddleware

{

private readonly RequestDelegate \_next;

public DbInitializerMiddleware(RequestDelegate next)

{

\_next = next;

}

public async Task Invoke(HttpContext context, TelecomDbContext dbContext)

{

var hasSubscribers = dbContext.Subscribers.Any();

if (!hasSubscribers)

{

DbInitializer.Initialize(dbContext);

}

await \_next(context);

}

}

public static class DbInitializerExtensions

{

public static IApplicationBuilder UseDbInitializer(this IApplicationBuilder builder)

{

return builder.UseMiddleware<DbInitializerMiddleware>();

}

}

}

**ПРИЛОЖЕНИЕ Ё**

(обязательное)

**DbInitializer.cs**

using TelecomWeb.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Linq;

namespace TelecomWeb.Data

{

public static class DbInitializer

{

public static void Initialize(TelecomDbContext db)

{

db.Database.EnsureCreated();

string sqlFilePath = "sql/insert.sql";

if (File.Exists(sqlFilePath))

{

string sql = File.ReadAllText(sqlFilePath);

db.Database.ExecuteSqlRaw(sql);

}

else

{

Console.WriteLine($"Файл не найден: {sqlFilePath}");

}

}

}

}

**ПРИЛОЖЕНИЕ Ж**

(обязательное)

**Contracts table**

**Контроллер:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using TelecomWeb.Models;

namespace TelecomWeb.Controllers.tables

{

[Authorize]

public class ContractsController : Controller

{

private readonly TelecomDbContext \_context;

public ContractsController(TelecomDbContext context)

{

\_context = context;

}

// GET: Contracts

public async Task<IActionResult> Index(int pageNumber = 1, string searchSubscriber = "", string searchTariff = "", string searchStaff = "", string searchPhone = "")

{

int pageSize = 100;

var telecomDbContext = \_context.Contracts

.Include(c => c.Staff)

.Include(c => c.Subscriber)

.Include(c => c.TariffPlan)

.AsQueryable();

if (!string.IsNullOrEmpty(searchSubscriber))

{

telecomDbContext = telecomDbContext.Where(c => c.Subscriber.FullName.Contains(searchSubscriber));

}

if (!string.IsNullOrEmpty(searchTariff))

{

telecomDbContext = telecomDbContext.Where(c => c.TariffPlan.TariffName.Contains(searchTariff));

}

if (!string.IsNullOrEmpty(searchStaff))

{

telecomDbContext = telecomDbContext.Where(c => c.Staff.FullName.Contains(searchStaff));

}

if (!string.IsNullOrEmpty(searchPhone))

{

telecomDbContext = telecomDbContext.Where(c => c.PhoneNumber.Contains(searchPhone));

}

var totalContracts = await telecomDbContext.CountAsync();

var contracts = await telecomDbContext

.OrderBy(c => c.ContractId)

.Skip((pageNumber - 1) \* pageSize)

.Take(pageSize)

.ToListAsync();

var viewModel = new PaginatedList<Contract>(contracts, totalContracts, pageNumber, pageSize);

ViewData["searchSubscriber"] = searchSubscriber;

ViewData["searchTariff"] = searchTariff;

ViewData["searchStaff"] = searchStaff;

ViewData["searchPhone"] = searchPhone;

return View(viewModel);

}

// GET: Contracts/Details/5

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

{

if (id == null)

{

return NotFound();

}

var contract = await \_context.Contracts

.Include(c => c.Staff)

.Include(c => c.Subscriber)

.Include(c => c.TariffPlan)

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

if (contract == null)

{

return NotFound();

}

return View(contract);

}

// GET: Contracts/Create

public IActionResult Create()

{

ViewData["StaffId"] = new SelectList(\_context.Staff, "StaffId", "FullName");

ViewData["SubscriberId"] = new SelectList(\_context.Subscribers, "SubscriberId", "FullName");

ViewData["TariffPlanId"] = new SelectList(\_context.TariffPlans, "TariffPlanId", "TariffName");

return View();

}

// POST: Contracts/Create

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

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("ContractId,SubscriberId,TariffPlanId,ContractDate,ContractEndDate,PhoneNumber,StaffId")] Contract contract)

{

if (ModelState.IsValid)

{

\_context.Add(contract);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["StaffId"] = new SelectList(\_context.Staff, "StaffId", "StaffId", contract.StaffId);

ViewData["SubscriberId"] = new SelectList(\_context.Subscribers, "SubscriberId", "SubscriberId", contract.SubscriberId);

ViewData["TariffPlanId"] = new SelectList(\_context.TariffPlans, "TariffPlanId", "TariffPlanId", contract.TariffPlanId);

return View(contract);

}

// GET: Contracts/Edit/5

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

{

if (id == null)

{

return NotFound();

}

var contract = await \_context.Contracts.FindAsync(id);

if (contract == null)

{

return NotFound();

}

ViewData["StaffName"] = new SelectList(\_context.Staff, "StaffId", "FullName", contract.StaffId);

ViewData["SubscriberName"] = new SelectList(\_context.Subscribers, "SubscriberId", "FullName", contract.SubscriberId);

ViewData["TariffPlanName"] = new SelectList(\_context.TariffPlans, "TariffPlanId", "TariffName", contract.TariffPlanId);

return View(contract);

}

// POST: Contracts/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("ContractId,SubscriberId,TariffPlanId,ContractDate,ContractEndDate,PhoneNumber,StaffId")] Contract contract)

{

if (id != contract.ContractId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(contract);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

catch (DbUpdateConcurrencyException)

{

if (!ContractExists(contract.ContractId))

{

return NotFound();

}

else

{

throw;

}

}

}

ViewData["StaffName"] = new SelectList(\_context.Staff, "StaffId", "FullName", contract.StaffId);

ViewData["SubscriberName"] = new SelectList(\_context.Subscribers, "SubscriberId", "FullName", contract.SubscriberId);

ViewData["TariffPlanName"] = new SelectList(\_context.TariffPlans, "TariffPlanId", "TariffName", contract.TariffPlanId);

return View(contract);

}

// GET: Contracts/Delete/5

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

{

if (id == null)

{

return NotFound();

}

var contract = await \_context.Contracts

.Include(c => c.Staff)

.Include(c => c.Subscriber)

.Include(c => c.TariffPlan)

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

if (contract == null)

{

return NotFound();

}

return View(contract);

}

// POST: Contracts/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var contract = await \_context.Contracts.FindAsync(id);

if (contract != null)

{

\_context.Contracts.Remove(contract);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool ContractExists(int id)

{

return \_context.Contracts.Any(e => e.ContractId == id);

}

}

}

**Create.cshtml:**

@model TelecomWeb.Models.Contract

@{

ViewData["Title"] = "Create";

}

<h1>Create</h1>

<h4>Contract</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Create">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label asp-for="SubscriberId" class="control-label"></label>

<select asp-for="SubscriberId" class="form-control" asp-items="ViewBag.SubscriberId"></select>

</div>

<div class="form-group">

<label asp-for="TariffPlanId" class="control-label"></label>

<select asp-for="TariffPlanId" class ="form-control" asp-items="ViewBag.TariffPlanId"></select>

</div>

<div class="form-group">

<label asp-for="ContractDate" class="control-label"></label>

<input asp-for="ContractDate" class="form-control" />

<span asp-validation-for="ContractDate" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="ContractEndDate" class="control-label"></label>

<input asp-for="ContractEndDate" class="form-control" />

<span asp-validation-for="ContractEndDate" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="PhoneNumber" class="control-label"></label>

<input asp-for="PhoneNumber" class="form-control" />

<span asp-validation-for="PhoneNumber" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="StaffId" class="control-label"></label>

<select asp-for="StaffId" class="form-control" asp-items="ViewBag.StaffId"></select>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Back to List</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Delete.cshtml:**

@model TelecomWeb.Models.Contract

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Contract</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.ContractDate)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.ContractDate)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.ContractEndDate)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.ContractEndDate)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.PhoneNumber)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.PhoneNumber)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Staff)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Staff.FullName)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Subscriber)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Subscriber.FullName)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.TariffPlan)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.TariffPlan.TariffName)

</dd>

</dl>

<form asp-action="Delete">

<input type="hidden" asp-for="ContractId" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<a asp-action="Index">Back to List</a>

</form>

</div>

**Details.cshtml:**

@model TelecomWeb.Models.Contract

@{

ViewData["Title"] = "Details";

}

<h1>Details</h1>

<div>

<h4>Contract</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.ContractDate)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.ContractDate)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.ContractEndDate)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.ContractEndDate)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.PhoneNumber)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.PhoneNumber)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Staff)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Staff.FullName)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Subscriber)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Subscriber.FullName)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.TariffPlan)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.TariffPlan.TariffName)

</dd>

</dl>

</div>

<div>

<a asp-action="Edit" asp-route-id="@Model?.ContractId">Edit</a> |

<a asp-action="Index">Back to List</a>

</div>

**Edit.cshtml:**

@model TelecomWeb.Models.Contract

@{

ViewData["Title"] = "Edit";

}

<h1>Edit</h1>

<h4>Contract</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Edit" method="post">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<input type="hidden" asp-for="ContractId" />

<div class="form-group">

<label asp-for="SubscriberId" class="control-label"></label>

<select asp-for="SubscriberId" class="form-control" asp-items="ViewBag.SubscriberName"></select>

<span asp-validation-for="SubscriberId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="TariffPlanId" class="control-label"></label>

<select asp-for="TariffPlanId" class="form-control" asp-items="ViewBag.TariffPlanName"></select>

<span asp-validation-for="TariffPlanId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="ContractDate" class="control-label"></label>

<input asp-for="ContractDate" class="form-control" type="date" />

<span asp-validation-for="ContractDate" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="ContractEndDate" class="control-label"></label>

<input asp-for="ContractEndDate" class="form-control" type="date" />

<span asp-validation-for="ContractEndDate" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="PhoneNumber" class="control-label"></label>

<input asp-for="PhoneNumber" class="form-control" />

<span asp-validation-for="PhoneNumber" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="StaffId" class="control-label"></label>

<select asp-for="StaffId" class="form-control" asp-items="ViewBag.StaffName"></select>

<span asp-validation-for="StaffId" class="text-danger"></span>

</div>

<div asp-validation-summary="All" class="text-danger"></div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Back to List</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Index.cshtml:**

@model PaginatedList<TelecomWeb.Models.Contract>

@{

ViewData["Title"] = "Index";

}

<h1>Contracts</h1>

<p>

<a asp-action="Create">Create New</a>

</p>

<form method="get" class="form-inline mb-3">

<div class="form-group">

<input type="text" name="searchSubscriber" class="form-control" placeholder="Subscriber" value="@ViewData["searchSubscriber"]" />

</div>

<div class="form-group mx-sm-2">

<input type="text" name="searchTariff" class="form-control" placeholder="Tariff" value="@ViewData["searchTariff"]" />

</div>

<div class="form-group mx-sm-2">

<input type="text" name="searchStaff" class="form-control" placeholder="Staff" value="@ViewData["searchStaff"]" />

</div>

<div class="form-group mx-sm-2">

<input type="text" name="searchPhone" class="form-control" placeholder="Phone Number" value="@ViewData["searchPhone"]" />

</div>

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

</form>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.First().Subscriber.FullName)

</th>

<th>

@Html.DisplayNameFor(model => model.First().TariffPlan.TariffName)

</th>

<th>

@Html.DisplayNameFor(model => model.First().Staff.FullName)

</th>

<th>

@Html.DisplayNameFor(model => model.First().ContractDate)

</th>

<th>

@Html.DisplayNameFor(model => model.First().ContractEndDate)

</th>

<th>

@Html.DisplayNameFor(model => model.First().PhoneNumber)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Subscriber.FullName)

</td>

<td>

@Html.DisplayFor(modelItem => item.TariffPlan.TariffName)

</td>

<td>

@Html.DisplayFor(modelItem => item.Staff.FullName)

</td>

<td>

@Html.DisplayFor(modelItem => item.ContractDate)

</td>

<td>

@Html.DisplayFor(modelItem => item.ContractEndDate)

</td>

<td>

@Html.DisplayFor(modelItem => item.PhoneNumber)

</td>

<td>

<a asp-action="Edit" asp-route-id="@item.ContractId">Edit</a> |

<a asp-action="Details" asp-route-id="@item.ContractId">Details</a> |

<a asp-action="Delete" asp-route-id="@item.ContractId">Delete</a>

</td>

</tr>

}

</tbody>

</table>

<div class="text-center">

<ul class="pagination">

<li class="page-item @(Model.HasPreviousPage ? "" : "disabled")">

<a class="page-link" asp-action="Index" asp-route-pageNumber="@(Model.PageIndex - 1)" asp-route-searchSubscriber="@ViewData["searchSubscriber"]" asp-route-searchTariff="@ViewData["searchTariff"]" asp-route-searchStaff="@ViewData["searchStaff"]" asp-route-searchPhone="@ViewData["searchPhone"]">Previous</a>

</li>

@for (int i = 1; i <= Model.TotalPages; i++)

{

<li class="page-item @(i == Model.PageIndex ? "active" : "")">

<a class="page-link" asp-action="Index" asp-route-pageNumber="@i" asp-route-searchSubscriber="@ViewData["searchSubscriber"]" asp-route-searchTariff="@ViewData["searchTariff"]" asp-route-searchStaff="@ViewData["searchStaff"]" asp-route-searchPhone="@ViewData["searchPhone"]">@i</a>

</li>

}

<li class="page-item @(Model.HasNextPage ? "" : "disabled")">

<a class="page-link" asp-action="Index" asp-route-pageNumber="@(Model.PageIndex + 1)" asp-route-searchSubscriber="@ViewData["searchSubscriber"]" asp-route-searchTariff="@ViewData["searchTariff"]" asp-route-searchStaff="@ViewData["searchStaff"]" asp-route-searchPhone="@ViewData["searchPhone"]">Next</a>

</li>

</ul>

</div>

**ПРИЛОЖЕНИЕ З**

(обязательное)

**Users table**

**Контроллер:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Linq;

using System.Threading.Tasks;

using TelecomWeb.Models;

namespace TelecomWeb.Controllers

{

[Authorize(Roles = "admin")]

public class UsersController : Controller

{

private readonly UserManager<IdentityUser> \_userManager;

public UsersController(UserManager<IdentityUser> userManager)

{

\_userManager = userManager;

}

// GET: Users

public async Task<IActionResult> Index()

{

var users = await \_userManager.Users.ToListAsync();

return View(users);

}

// GET: Users/Details/5

public async Task<IActionResult> Details(string id)

{

if (id == null)

{

return NotFound();

}

var user = await \_userManager.FindByIdAsync(id);

if (user == null)

{

return NotFound();

}

return View(user);

}

// GET: Users/Create

public IActionResult Create()

{

return View();

}

// POST: Users/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("UserName,Email,Password")] CreateUserViewModel model)

{

if (ModelState.IsValid)

{

var user = new IdentityUser { UserName = model.UserName, Email = model.Email };

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

return RedirectToAction(nameof(Index));

}

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

return View(model);

}

// GET: Users/Edit/5

public async Task<IActionResult> Edit(string id)

{

if (id == null)

{

return NotFound();

}

var user = await \_userManager.FindByIdAsync(id);

if (user == null)

{

return NotFound();

}

var model = new EditUserViewModel { Id = user.Id, UserName = user.UserName, Email = user.Email };

return View(model);

}

// POST: Users/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(string id, [Bind("Id,UserName,Email")] EditUserViewModel model)

{

if (id != model.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

var user = await \_userManager.FindByIdAsync(id);

if (user != null)

{

user.UserName = model.UserName;

user.Email = model.Email;

var result = await \_userManager.UpdateAsync(user);

if (result.Succeeded)

{

return RedirectToAction(nameof(Index));

}

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

// GET: Users/Delete/5

public async Task<IActionResult> Delete(string id)

{

if (id == null)

{

return NotFound();

}

var user = await \_userManager.FindByIdAsync(id);

if (user == null)

{

return NotFound();

}

return View(user);

}

// POST: Users/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(string id)

{

var user = await \_userManager.FindByIdAsync(id);

if (user != null)

{

await \_userManager.DeleteAsync(user);

}

return RedirectToAction(nameof(Index));

}

}

}

**Create.cshtml:**

@using TelecomWeb.Models

@model CreateUserViewModel

@{

ViewData["Title"] = "Create User";

}

<h1>Create User</h1>

<form asp-action="Create" method="post">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label asp-for="UserName" class="control-label"></label>

<input asp-for="UserName" class="form-control" />

<span asp-validation-for="UserName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Email" class="control-label"></label>

<input asp-for="Email" class="form-control" />

<span asp-validation-for="Email" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Password" class="control-label"></label>

<input asp-for="Password" type="password" class="form-control" />

<span asp-validation-for="Password" class="text-danger"></span>

</div>

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

</form>

<a asp-action="Index">Back to List</a>

**Delete.cshtml:**

@using Microsoft.AspNetCore.Identity

@model IdentityUser

@{

ViewData["Title"] = "Delete User";

}

<h1>Delete User</h1>

<h3>Are you sure you want to delete this user?</h3>

<div>

<h4>User</h4>

<hr />

<dl class="row">

<dt class="col-sm-2">UserName</dt>

<dd class="col-sm-10">@Model.UserName</dd>

<dt class="col-sm-2">Email</dt>

<dd class="col-sm-10">@Model.Email</dd>

</dl>

</div>

<form asp-action="Delete" method="post">

<input type="hidden" asp-for="Id" />

<button type="submit" class="btn btn-danger">Delete</button> |

<a asp-action="Index">Back to List</a>

</form>

**Details.cshtml:**

@using Microsoft.AspNetCore.Identity

@model IdentityUser

@{

ViewData["Title"] = "User Details";

}

<h1>User Details</h1>

<div>

<h4>User</h4>

<hr />

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.UserName)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.UserName)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Email)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Email)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Id)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Id)

</dd>

</dl>

</div>

<div>

<a asp-action="Edit" asp-route-id="@Model.Id" class="btn btn-primary">Edit</a> |

<a asp-action="Index" class="btn btn-secondary">Back to List</a>

</div>

<a asp-action="Edit" asp-route-id="@Model?.ContractId">Edit</a> |

<a asp-action="Index">Back to List</a>

</div>

**Edit.cshtml:**

@using TelecomWeb.Models

@model EditUserViewModel

@{

ViewData["Title"] = "Edit User";

}

<h1>Edit User</h1>

<form asp-action="Edit" method="post">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<input type="hidden" asp-for="Id" />

<div class="form-group">

<label asp-for="UserName" class="control-label"></label>

<input asp-for="UserName" class="form-control" />

<span asp-validation-for="UserName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Email" class="control-label"></label>

<input asp-for="Email" class="form-control" />

<span asp-validation-for="Email" class="text-danger"></span>

</div>

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

</form>

<a asp-action="Index">Back to List</a>

**Index.cshtml:**

@using Microsoft.AspNetCore.Identity

@model IEnumerable<IdentityUser>

@{

ViewData["Title"] = "Users";

}

<h1>Users</h1>

<p>

<a asp-action="Create">Create New User</a>

</p>

<table class="table">

<thead>

<tr>

<th>UserName</th>

<th>Email</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var user in Model)

{

<tr>

<td>@user.UserName</td>

<td>@user.Email</td>

<td>

<a asp-action="Edit" asp-route-id="@user.Id">Edit</a> |

<a asp-action="Details" asp-route-id="@user.Id">Details</a> |

<a asp-action="Delete" asp-route-id="@user.Id">Delete</a>

</td>

</tr>

}

</tbody>

</table>

**ПРИЛОЖЕНИЕ И**

(обязательное)

**ContractEndpoints.cs**

using Microsoft.AspNetCore.Http.HttpResults;

using Microsoft.EntityFrameworkCore;

using TelecomWeb.Models;

namespace RestAPI.Controllers;

public static class ContractEndpoints

{

public static void MapContractEndpoints(this IEndpointRouteBuilder routes)

{

var group = routes.MapGroup("/api/Contract").WithTags(nameof(Contract));

group.MapGet("/", async (TelecomDbContext db) =>

{

return await db.Contracts.ToListAsync();

})

.WithName("GetAllContracts")

.WithOpenApi();

group.MapGet("/{contractid}", async Task<Results<Ok<Contract>, NotFound>> (int contractid, TelecomDbContext db) =>

{

return await db.Contracts.AsNoTracking()

.FirstOrDefaultAsync(model => model.ContractId == contractid)

is Contract model

? TypedResults.Ok(model)

: TypedResults.NotFound();

})

.WithName("GetContractById")

.WithOpenApi();

group.MapPut("/{contractid}", async Task<Results<Ok, NotFound>> (int contractid, Contract contract, TelecomDbContext db) =>

{

var affected = await db.Contracts

.Where(model => model.ContractId == contractid)

.ExecuteUpdateAsync(setters => setters

.SetProperty(m => m.SubscriberId, contract.SubscriberId)

.SetProperty(m => m.TariffPlanId, contract.TariffPlanId)

.SetProperty(m => m.ContractDate, contract.ContractDate)

.SetProperty(m => m.ContractEndDate, contract.ContractEndDate)

.SetProperty(m => m.PhoneNumber, contract.PhoneNumber)

.SetProperty(m => m.StaffId, contract.StaffId)

);

return affected == 1 ? TypedResults.Ok() : TypedResults.NotFound();

})

.WithName("UpdateContract")

.WithOpenApi();

group.MapPost("/", async (Contract contract, TelecomDbContext db) =>

{

db.Contracts.Add(contract);

await db.SaveChangesAsync();

return TypedResults.Created($"/api/Contract/{contract.ContractId}", contract);

})

.WithName("CreateContract")

.WithOpenApi();

group.MapDelete("/{contractid}", async Task<Results<Ok, NotFound>> (int contractid, TelecomDbContext db) =>

{

var affected = await db.Contracts

.Where(model => model.ContractId == contractid)

.ExecuteDeleteAsync();

return affected == 1 ? TypedResults.Ok() : TypedResults.NotFound();

})

.WithName("DeleteContract")

.WithOpenApi();

}

}

**ПРИЛОЖЕНИЕ Й**

(обязательное)

**UnitTest.cs**

using Microsoft.AspNetCore.Mvc.Testing;

using Microsoft.EntityFrameworkCore;

using TelecomWeb.Models;

using System.Net.Http;

using System.Net.Http.Json;

using System.Text.Json;

using Xunit;

using RestAPI;

using Microsoft.VisualStudio.TestPlatform.Utilities;

using Xunit.Abstractions;

namespace RestAPITests

{

public class ContractEndpointsTests : IClassFixture<WebApplicationFactory<Program>>

{

private readonly HttpClient \_client;

private readonly WebApplicationFactory<Program> \_factory;

private readonly ITestOutputHelper \_output;

public ContractEndpointsTests(WebApplicationFactory<Program> factory, ITestOutputHelper output)

{

\_factory = factory;

\_client = \_factory.CreateClient();

\_output = output;

}

[Fact]

public async Task GetAllContracts\_ReturnsOkResult()

{

var response = await \_client.GetAsync("/api/Contract");

response.EnsureSuccessStatusCode();

var contracts = await response.Content.ReadFromJsonAsync<List<Contract>>();

Assert.NotNull(contracts);

}

[Fact]

public async Task GetContractById\_ReturnsOkResult()

{

var contractId = 1039;

\_output.WriteLine($"/api/Contract/{contractId}/");

var response = await \_client.GetAsync($"/api/Contract/{contractId}");

response.EnsureSuccessStatusCode();

var contract = await response.Content.ReadFromJsonAsync<Contract>();

Assert.NotNull(contract);

Assert.Equal(contractId, contract.ContractId);

}

[Fact]

public async Task CreateContract\_ReturnsCreatedResult()

{

var newContract = new Contract

{

SubscriberId = 4,

TariffPlanId = 4,

ContractDate = DateOnly.FromDateTime(DateTime.UtcNow),

ContractEndDate = DateOnly.FromDateTime(DateTime.UtcNow.AddYears(1)),

PhoneNumber = "123456789",

StaffId = 4

};

var response = await \_client.PostAsJsonAsync("/api/Contract", newContract);

response.EnsureSuccessStatusCode();

var createdContract = await response.Content.ReadFromJsonAsync<Contract>();

Assert.NotNull(createdContract);

Assert.Equal(newContract.PhoneNumber, createdContract.PhoneNumber);

}

[Fact]

public async Task UpdateContract\_ReturnsOkResult()

{

var contractId = 1039;

var updatedContract = new Contract

{

SubscriberId = 3,

TariffPlanId = 3,

ContractDate = DateOnly.FromDateTime(DateTime.UtcNow),

ContractEndDate = DateOnly.FromDateTime(DateTime.UtcNow.AddYears(1)),

PhoneNumber = "987654321",

StaffId = 3

};

var response = await \_client.PutAsJsonAsync($"/api/Contract/{contractId}", updatedContract);

response.EnsureSuccessStatusCode();

}

[Fact]

public async Task DeleteContract\_ReturnsOkResult()

{

var contractId = 1010;

var response = await \_client.DeleteAsync($"/api/Contract/{contractId}");

response.EnsureSuccessStatusCode();

}

}

}