**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №4**

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

на тему: «Разработка моделей и контроллеров *ASP.NET MVC* приложения баз данных»

Выполнил: студент гр. ИТП-31

Демиденко Д. В.

Принял: ректор

Асенчик О. Д.

Гомель 2023

**Цель работы:** ознакомиться с возможностями *ASP.NET Core MVC* и *Entity Framework Core* для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.

**Задание:**

Создать с использованием *ASP.NET Core MVC Web*-приложение, содержащее набор классов, моделирующих предметную область, и осуществляющих генерацию и заполнение тестовыми наборами записей базу данных. Разработать один компонент *middleware*, контроллеры и представления для выборки и отображения информации из не менее чем 3- таблиц базы данных с использованием механизма внедрение зависимостей.

Для выполнения задания необходимо создать:

• Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Перечень таблиц предварительно согласовывается с преподавателем. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.

• Класс контекста данных.

• Другие классы, например, классы *View Model* и т.п. (при необходимости).

• Компонент *middleware*, вызываемый в классе *Startup*, для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.

• Классы контроллеров (по одному на каждую таблицу базы данных) для обработки обращений пользователя, выборки данных из таблиц и вызова соответствующих представлений для отображения выбранных данных.

• Разработать представления для отображения данных из таблиц, выбранных контроллерами. Представления, работающими с таблицами, стоящими на стороне отношения «многие» в схеме базы данных, должны выводить вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».

• Используя предварительно созданный и сконфигурированный в классе *Startup* профиль кэширования, подключить кэширование вывода для страниц с использованием атрибута *ResponseCache* для соответствующих методов контроллера. Данные в кэше хранить неизменными в течение 2\**N*+240 секунд, где *N* – номер вашего варианта.

• С использованием средств разработчика браузера (*Chrome, Firefox*) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием атрибута *ResponseCache*.

Для проверки преподавателем следует разместить разработанный проект на *GitHub*.

**Ход работы**

В ходе выполнения лабораторной работы при помощи *Entity framework* были перенесены три модели которые были связаны между собой. Строка подключения к базе данных хранится в файле *appsettings.json*. Класс контекста был внедрен в приложение при помощи *DI*. Листинг этих моделей класса контекста.

Далее были разработаны контроллеры, которые используются для передачи данных, которые хранятся в моделях в представления. Были разработаны три контроллера *ApartnmentsController, ContractsController* и *SellersController*. Каждый контроллер используется для работы с каждой моделью. Листинг всех этих контролеров указан в приложении А.

Далее были разработаны представления выводя данных из контролера в *HTML* страницу. Пример этих представлений указан в приложении А.

Пример представления с информаций о договорах указан на рисунке 1.

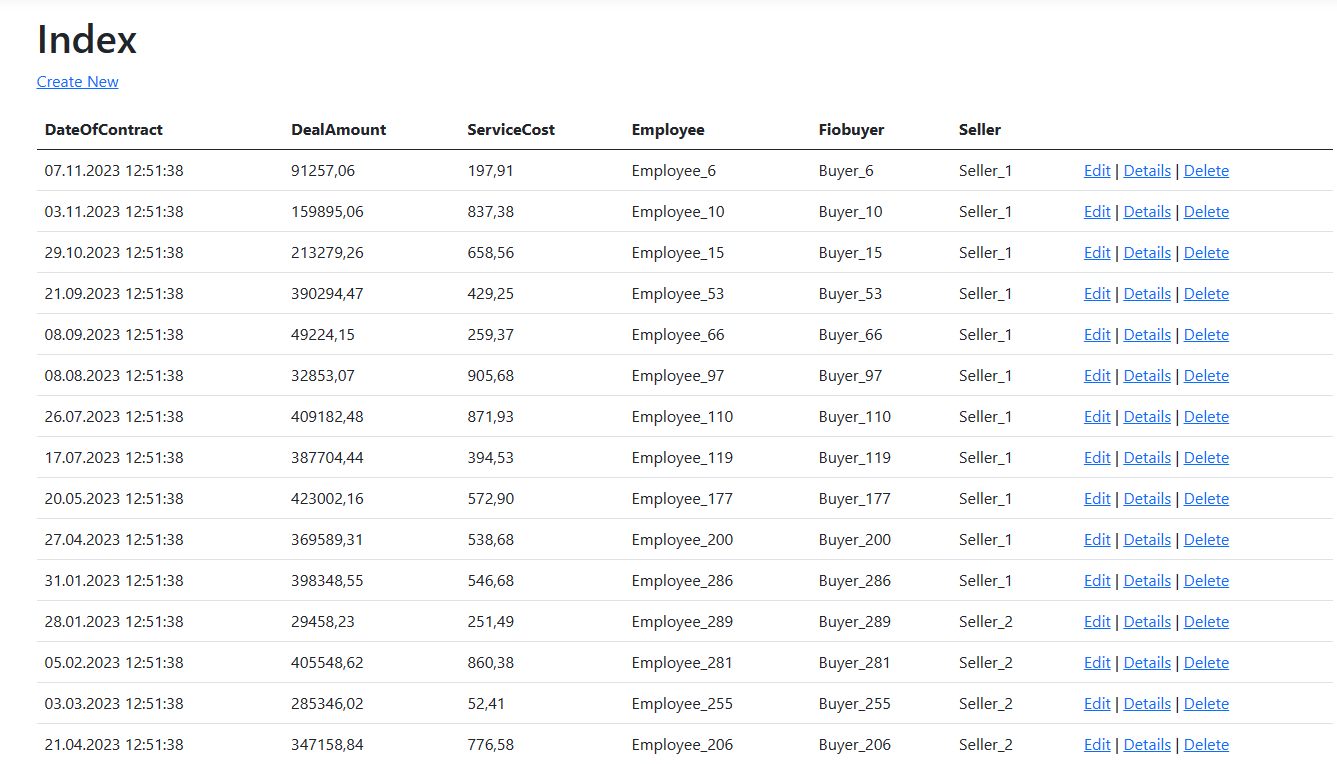


Рисунок 1 – Пример страницы с информацией о договорах

Пример представления с информаций о квартирах указан на рисунке 2.

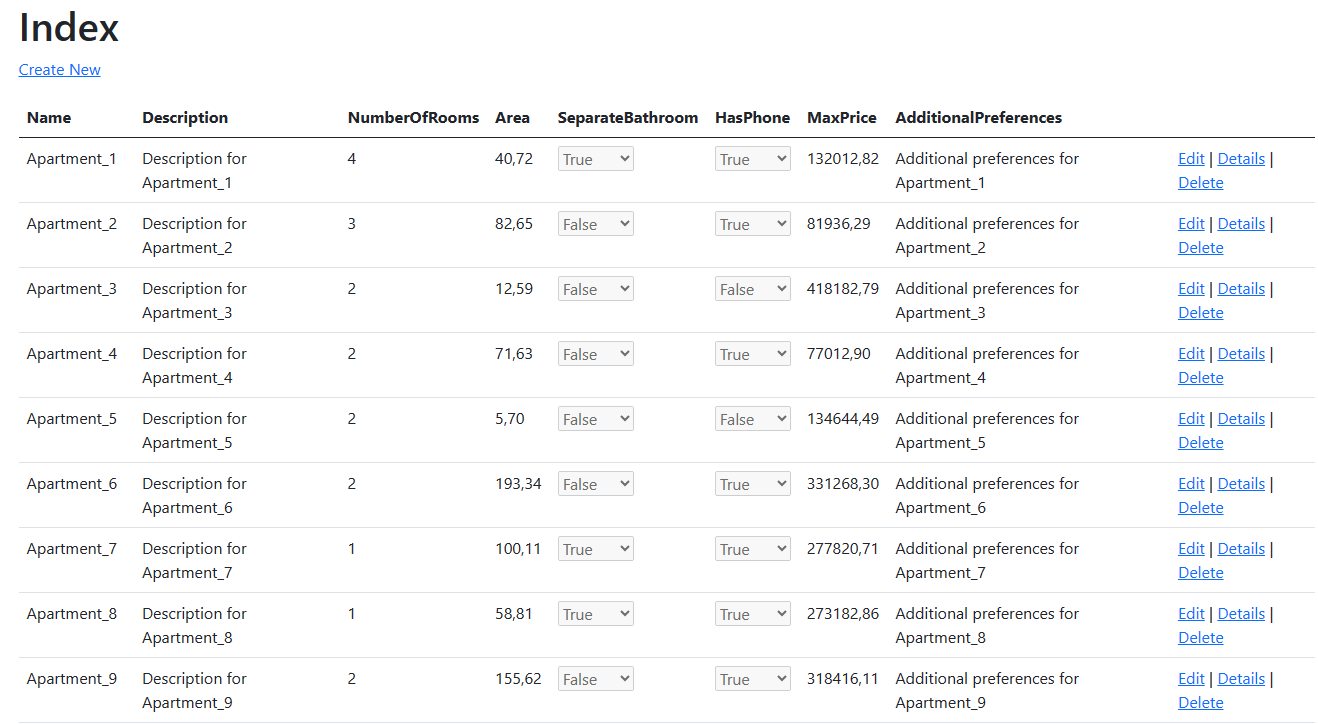


Рисунок 2 – Пример страницы с информацией о квартирах

Пример представления с информаций о продавцах указан на рисунке 3.

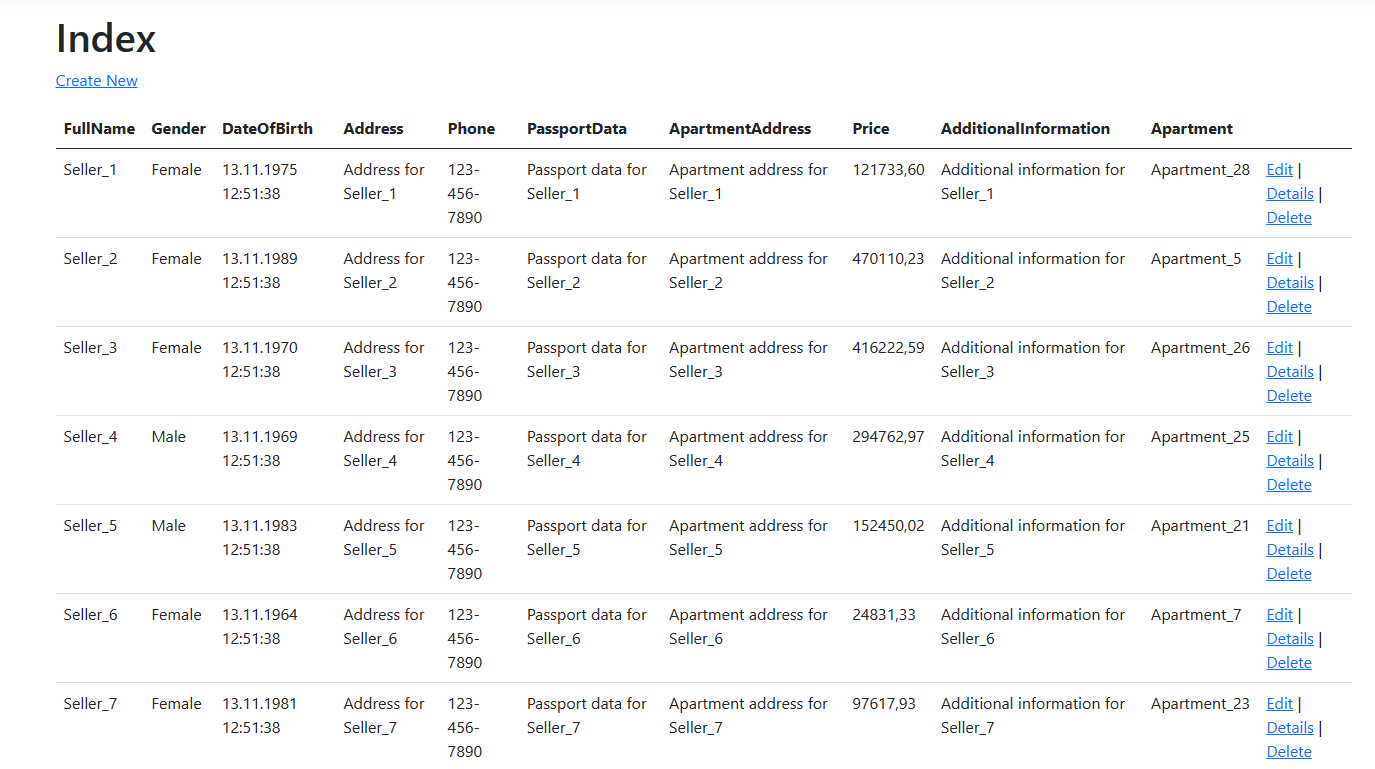


Рисунок 3 – Пример страницы с информацией о продавцах

После выполнения лабораторной работы созданные проект был добавлен в локальный *git* репозиторий а потом перенесен в *GitHub* репозиторий своего аккаунта. Чтобы ознакомится с созданным проектом можно по ссылке *GitHub*: [ddemidenko/RPBDIS: Лабораторные работы РПБДТС (github.com)](https://github.com/ddemidenko/RPBDIS).

**Вывод**: в ходе выполнения лабораторной работы была изучена такая технология *ASP .NET* *Core MVC*. Было разработаны классы моделей и контекста, предназначенные для работы с данными. Классы контроллера для связи моделей с представлениями. Классы представления, предназначенные для вывода результата.

**ПРИЛОЖНИЕ А**

Листинг класса *Program*

using Microsoft.EntityFrameworkCore;

using RealEstateAgency4.Middleware;

using RealEstateAgency4.Models;

namespace RealEstateAgency4

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

string connection = builder.Configuration.GetConnectionString("SqlServerConnection")!;

builder.Services.AddDbContext<RealEstateAgencyContext>(options => options.UseSqlServer(connection));

// Add services to the container.

builder.Services.AddControllersWithViews();

builder.Services.AddSession();

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.UseSession();

app.UseDBInitializer();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.Run();

}

}

}

Листинг класса *ApartmentsController*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using RealEstateAgency4.Models;

namespace RealEstateAgency4.Controllers

{

public class ApartmentsController : Controller

{

private readonly RealEstateAgencyContext \_context;

public ApartmentsController(RealEstateAgencyContext context)

{

\_context = context;

}

// GET: Apartments

public async Task<IActionResult> Index()

{

return \_context.Apartments != null ?

View(await \_context.Apartments.ToListAsync()) :

Problem("Entity set 'RealEstateAgencyContext.Apartments' is null.");

}

// GET: Apartments/Details/5

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

{

if (id == null || \_context.Apartments == null)

{

return NotFound();

}

var apartment = await \_context.Apartments

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

if (apartment == null)

{

return NotFound();

}

return View(apartment);

}

// GET: Apartments/Create

public IActionResult Create()

{

return View();

}

// POST: Apartments/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("ApartmentId,Name,Description,NumberOfRooms,Area,SeparateBathroom,HasPhone,MaxPrice,AdditionalPreferences")] Apartment apartment)

{

if (ModelState.IsValid)

{

\_context.Add(apartment);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(apartment);

}

// GET: Apartments/Edit/5

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

{

if (id == null || \_context.Apartments == null)

{

return NotFound();

}

var apartment = await \_context.Apartments.FindAsync(id);

if (apartment == null)

{

return NotFound();

}

return View(apartment);

}

// POST: Apartments/Edit/5

// 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> Edit(int id, [Bind("ApartmentId,Name,Description,NumberOfRooms,Area,SeparateBathroom,HasPhone,MaxPrice,AdditionalPreferences")] Apartment apartment)

{

if (id != apartment.ApartmentId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(apartment);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ApartmentExists(apartment.ApartmentId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(apartment);

}

// GET: Apartments/Delete/5

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

{

if (id == null || \_context.Apartments == null)

{

return NotFound();

}

var apartment = await \_context.Apartments

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

if (apartment == null)

{

return NotFound();

}

return View(apartment);

}

// POST: Apartments/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

if (\_context.Apartments == null)

{

return Problem("Entity set 'RealEstateAgencyContext.Apartments' is null.");

}

var apartment = await \_context.Apartments.FindAsync(id);

if (apartment != null)

{

\_context.Apartments.Remove(apartment);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool ApartmentExists(int id)

{

return (\_context.Apartments?.Any(e => e.ApartmentId == id)).GetValueOrDefault();

}

}

}

Листинг класса *ContractsController*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using RealEstateAgency4.Models;

namespace RealEstateAgency4.Controllers

{

public class ContractsController : Controller

{

private readonly RealEstateAgencyContext \_context;

public ContractsController(RealEstateAgencyContext context)

{

\_context = context;

}

// GET: Contracts

public async Task<IActionResult> Index()

{

var realEstateAgencyContext = \_context.Contracts.Include(c => c.Seller);

return View(await realEstateAgencyContext.ToListAsync());

}

// GET: Contracts/Details/5

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

{

if (id == null || \_context.Contracts == null)

{

return NotFound();

}

var contract = await \_context.Contracts

.Include(c => c.Seller)

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

if (contract == null)

{

return NotFound();

}

return View(contract);

}

// GET: Contracts/Create

public IActionResult Create()

{

ViewData["SellerId"] = new SelectList(\_context.Sellers, "SellerId", "FullName");

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,DateOfContract,SellerId,BuyerId,DealAmount,ServiceCost,Employee,Fiobuyer")] Contract contract)

{

if (ModelState.IsValid)

{

\_context.Add(contract);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["SellerId"] = new SelectList(\_context.Sellers, "SellerId", "FullName", contract.SellerId);

return View(contract);

}

// GET: Contracts/Edit/5

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

{

if (id == null || \_context.Contracts == null)

{

return NotFound();

}

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

if (contract == null)

{

return NotFound();

}

ViewData["SellerId"] = new SelectList(\_context.Sellers, "SellerId", "FullName", contract.SellerId);

return View(contract);

}

// POST: Contracts/Edit/5

// 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> Edit(int id, [Bind("ContractId,DateOfContract,SellerId,BuyerId,DealAmount,ServiceCost,Employee,Fiobuyer")] Contract contract)

{

if (id != contract.ContractId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(contract);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ContractExists(contract.ContractId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["SellerId"] = new SelectList(\_context.Sellers, "SellerId", "FullName", contract.SellerId);

return View(contract);

}

// GET: Contracts/Delete/5

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

{

if (id == null || \_context.Contracts == null)

{

return NotFound();

}

var contract = await \_context.Contracts

.Include(c => c.Seller)

.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)

{

if (\_context.Contracts == null)

{

return Problem("Entity set 'RealEstateAgencyContext.Contracts' is null.");

}

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)).GetValueOrDefault();

}

}

}

Листинг класса *SellersController*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using RealEstateAgency4.Models;

namespace RealEstateAgency4.Controllers

{

public class SellersController : Controller

{

private readonly RealEstateAgencyContext \_context;

public SellersController(RealEstateAgencyContext context)

{

\_context = context;

}

// GET: Sellers

public async Task<IActionResult> Index()

{

var realEstateAgencyContext = \_context.Sellers.Include(s => s.Apartment);

return View(await realEstateAgencyContext.ToListAsync());

}

// GET: Sellers/Details/5

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

{

if (id == null || \_context.Sellers == null)

{

return NotFound();

}

var seller = await \_context.Sellers

.Include(s => s.Apartment)

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

if (seller == null)

{

return NotFound();

}

return View(seller);

}

// GET: Sellers/Create

public IActionResult Create()

{

ViewData["ApartmentId"] = new SelectList(\_context.Apartments, "ApartmentId", "Name");

return View();

}

// POST: Sellers/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("SellerId,FullName,Gender,DateOfBirth,Address,Phone,PassportData,ApartmentId,ApartmentAddress,Price,AdditionalInformation")] Seller seller)

{

if (ModelState.IsValid)

{

\_context.Add(seller);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["ApartmentId"] = new SelectList(\_context.Apartments, "ApartmentId", "Name", seller.ApartmentId);

return View(seller);

}

// GET: Sellers/Edit/5

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

{

if (id == null || \_context.Sellers == null)

{

return NotFound();

}

var seller = await \_context.Sellers.FindAsync(id);

if (seller == null)

{

return NotFound();

}

ViewData["ApartmentId"] = new SelectList(\_context.Apartments, "ApartmentId", "Name", seller.ApartmentId);

return View(seller);

}

// POST: Sellers/Edit/5

// 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> Edit(int id, [Bind("SellerId,FullName,Gender,DateOfBirth,Address,Phone,PassportData,ApartmentId,ApartmentAddress,Price,AdditionalInformation")] Seller seller)

{

if (id != seller.SellerId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(seller);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!SellerExists(seller.SellerId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["ApartmentId"] = new SelectList(\_context.Apartments, "ApartmentId", "Name", seller.ApartmentId);

return View(seller);

}

// GET: Sellers/Delete/5

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

{

if (id == null || \_context.Sellers == null)

{

return NotFound();

}

var seller = await \_context.Sellers

.Include(s => s.Apartment)

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

if (seller == null)

{

return NotFound();

}

return View(seller);

}

// POST: Sellers/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

if (\_context.Sellers == null)

{

return Problem("Entity set 'RealEstateAgencyContext.Sellers' is null.");

}

var seller = await \_context.Sellers.FindAsync(id);

if (seller != null)

{

\_context.Sellers.Remove(seller);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool SellerExists(int id)

{

return (\_context.Sellers?.Any(e => e.SellerId == id)).GetValueOrDefault();

}

}

}

Листинг класса *DataInintializer*

using RealEstateAgency4.Models;

using System;

using System.Linq;

namespace RealEstateAgency4.Middleware

{

public class DataInitializer

{

public static void Initialize(RealEstateAgencyContext dbContext)

{

dbContext.Database.EnsureCreated();

if (dbContext.Apartments.Any())

{

return;

}

SeedApartments(dbContext);

SeedSellers(dbContext);

SeedServices(dbContext);

SeedContracts(dbContext);

SeedContractServices(dbContext);

dbContext.SaveChanges();

}

private static void SeedApartments(RealEstateAgencyContext dbContext)

{

for (int apartmentId = 1; apartmentId <= 35; apartmentId++)

{

string apartmentName = "Apartment\_" + apartmentId.ToString();

string apartmentDescription = "Description for " + apartmentName;

int numberOfRooms = new Random().Next(1, 5);

decimal area = (decimal)(new Random().NextDouble() \* 200);

bool separateBathroom = new Random().Next(2) == 0;

bool hasPhone = new Random().Next(2) == 0;

decimal maxPrice = (decimal)(new Random().NextDouble() \* 500000);

string additionalPreferences = "Additional preferences for " + apartmentName;

dbContext.Apartments.Add(new Apartment

{

Name = apartmentName,

Description = apartmentDescription,

NumberOfRooms = numberOfRooms,

Area = area,

SeparateBathroom = separateBathroom,

HasPhone = hasPhone,

MaxPrice = maxPrice,

AdditionalPreferences = additionalPreferences

});

}

dbContext.SaveChanges();

}

private static void SeedSellers(RealEstateAgencyContext dbContext)

{

for (int sellerId = 1; sellerId <= 35; sellerId++)

{

string fullName = "Seller\_" + sellerId.ToString();

string gender = new Random().Next(2) == 0 ? "Male" : "Female";

DateTime dateOfBirth = DateTime.Now.AddYears(-new Random().Next(30, 60));

string address = "Address for " + fullName;

string phone = "123-456-7890";

string passportData = "Passport data for " + fullName;

int apartmentId = new Random().Next(1, 35);

string apartmentAddress = "Apartment address for " + fullName;

decimal price = (decimal)(new Random().NextDouble() \* 500000);

string additionalInformation = "Additional information for " + fullName;

dbContext.Sellers.Add(new Seller

{

FullName = fullName,

Gender = gender,

DateOfBirth = dateOfBirth,

Address = address,

Phone = phone,

PassportData = passportData,

ApartmentId = apartmentId,

ApartmentAddress = apartmentAddress,

Price = price,

AdditionalInformation = additionalInformation

});

}

dbContext.SaveChanges();

}

private static void SeedServices(RealEstateAgencyContext dbContext)

{

for (int i = 1; i <= 1000; i++)

{

string serviceName = "Service\_" + i;

string description = $"Description for {serviceName}";

decimal price = (decimal)(new Random().NextDouble() \* 100);

dbContext.Services.Add(new Service

{

Name = serviceName,

Description = description,

Price = price

});

}

dbContext.SaveChanges();

}

private static void SeedContracts(RealEstateAgencyContext dbContext)

{

for (int contractId = 1; contractId <= 300; contractId++)

{

int sellerId = new Random().Next(1, 35);

decimal dealAmount = (decimal)(new Random().NextDouble() \* 500000);

decimal serviceCost = (decimal)(new Random().NextDouble() \* 1000);

string employee = "Employee\_" + contractId.ToString();

string fioBuyer = "Buyer\_" + contractId.ToString();

dbContext.Contracts.Add(new Contract

{

DateOfContract = DateTime.Now.AddDays(-contractId),

SellerId = sellerId,

DealAmount = dealAmount,

ServiceCost = serviceCost,

Employee = employee,

Fiobuyer = fioBuyer

});

}

dbContext.SaveChanges();

}

private static void SeedContractServices(RealEstateAgencyContext dbContext)

{

for (int contractId = 1; contractId <= 300; contractId++)

{

int serviceId = new Random().Next(1, 1000);

var existingService = dbContext.Services.Find(serviceId);

if (existingService != null)

{

var contractService = new ContractService

{

ContractId = contractId,

ServiceId = serviceId

};

dbContext.ContractServices.Add(contractService);

}

}

dbContext.SaveChanges();

}

}

}