МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Разработка приложений баз данных для информационных систем»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
«Разработка моделей и контроллеров *ASP.NET MVC* приложения баз данных»

ВАРИАНТ №12
Брачное агентство

Выполнил:
студент группы ИТИ-31, Зеленский К.А.
Принял:
доцент Асенчик О.Д.

Гомель 2024

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.

Ход работы и результаты.

На рисунке 1 представлен созданный *GitHub* репозиторий. *https://github.com/kevenmusic/RPBDISLabs/tree/main*.

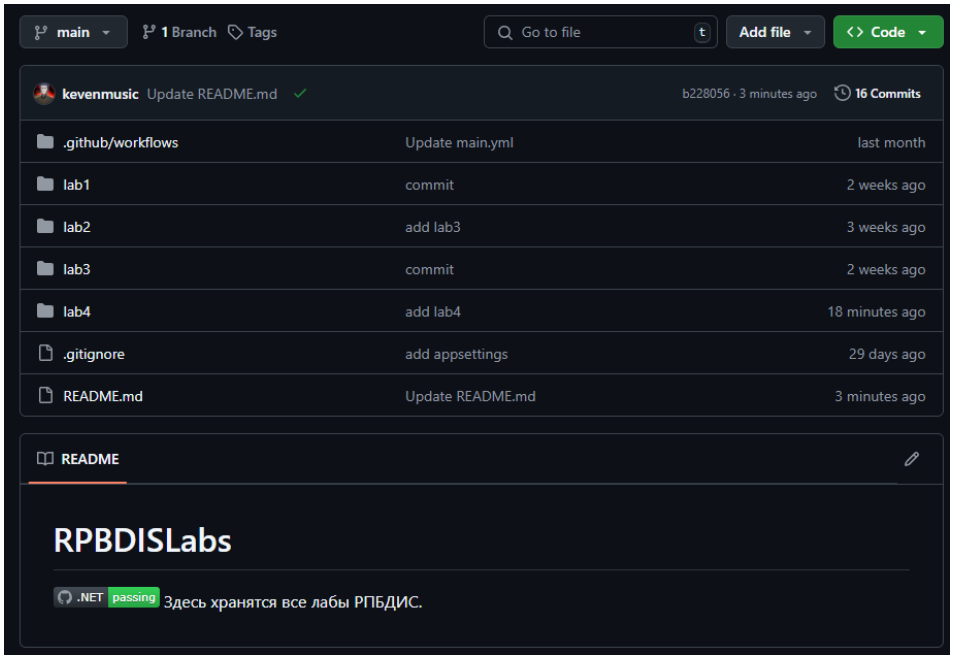

Рисунок 1 – *GitHub* Репозиторий

На рисунке 2 представлены таблицы, где одна из таблиц обязательно находится на стороне отношения «многие».

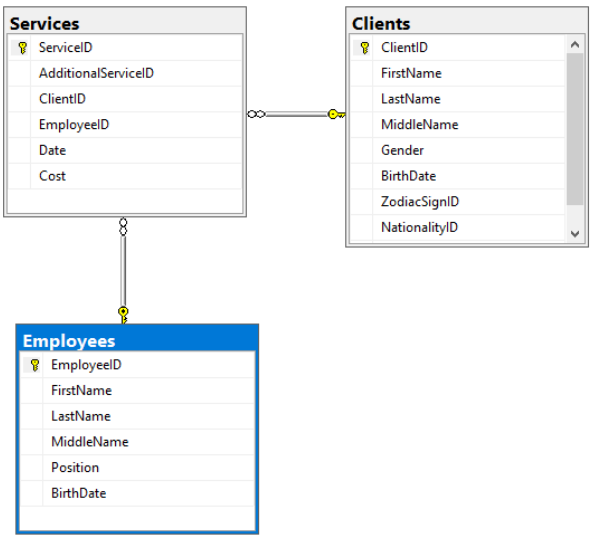

Рисунок 2 – Таблицы согласно варинту

На рисунке 3 представлено разработанное представление *Home* (Приложение Ё). для отображения данных из трёх таблиц, выбранных контроллером *HomeController* (Приложение Б).

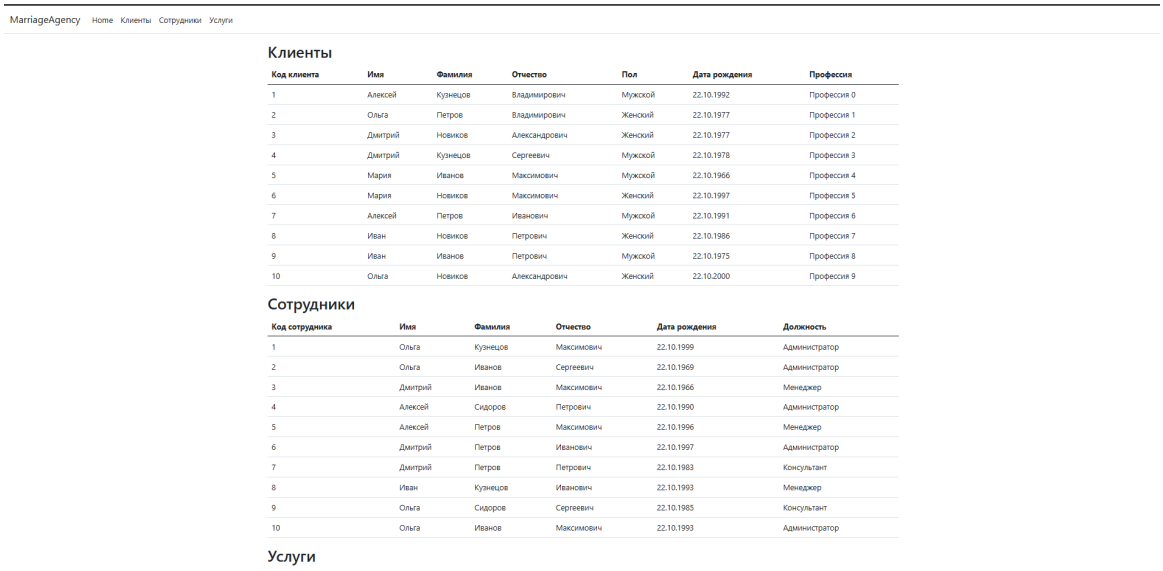

Рисунок 3 – Представление *Home*

На рисунке 4 представлено разработанное представление *Clients* (Приложение Ё), которое выводит вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».

Клиенты

Имя Клиента	Фамилия	Отчество	Пол	Дата рождения	Знак зодиака	Национальность	Телефон	Возраст	Профессия
Алексей	Кузнецов	Владимирович	Мужской	22.10.1992	Близнецы	Американец	+79161234560	57	Профессия 0
Ольга	Петров	Владимирович	Женский	22.10.1977	Рыбы	Китаец	+79161234561	57	Профессия 1
Дмитрий	Новиков	Александрович	Женский	22.10.1977	Козерог	Немец	+79161234562	18	Профессия 2
Дмитрий	Кузнецов	Сергеевич	Мужской	22.10.1978	Весы	Француз	+79161234563	46	Профессия 3
Мария	Иванов	Максимович	Мужской	22.10.1966	Лев	Китаец	+79161234564	38	Профессия 4
Мария	Новиков	Максимович	Женский	22.10.1997	Рак	Русский	+79161234565	22	Профессия 5
Алексей	Петров	Иванович	Мужской	22.10.1991	Рыбы	Американец	+79161234566	34	Профессия 6
Иван	Новиков	Петрович	Женский	22.10.1986	Козерог	Американец	+79161234567	30	Профессия 7
Иван	Иванов	Петрович	Мужской	22.10.1975	Весы	Немец	+79161234568	25	Профессия 8
Ольга	Новиков	Александрович	Женский	22.10.2000	Скорпион	Индиец	+79161234569	18	Профессия 9
Алексей	Новиков	Максимович	Мужской	22.10.1971	Овен	Бразилец	+791612345610	47	Профессия 10
Ольга	Кузнецов	Петрович	Мужской	22.10.1983	Близнецы	Американец	+791612345611	28	Профессия 11
Мария	Сидоров	Петрович	Женский	22.10.1970	Стрелец	Немец	+791612345612	20	Профессия 12
Мария	Кузнецов	Дмитриевич	Мужской	22.10.1975	Водолей	Русский	+791612345613	51	Профессия 13
Дмитрий	Новиков	Сергеевич	Женский	22.10.1979	Овен	Индиец	+791612345614	35	Профессия 14
Ольга	Новиков	Петрович	Женский	22.10.1993	Рак	Немец	+791612345615	34	Профессия 15
Ольга	Иванов	Петрович	Мужской	22.10.2000	Близнецы	Русский	+791612345616	29	Профессия 16
Алексей	Сидоров	Максимович	Женский	22.10.1991	Козерог	Китаец	+791612345617	56	Профессия 17

Рисунок 4 – Представление *Clients*

На рисунке 5 представлено разработанное представление *Employees* (Приложение Ё), стоящее на стороне отношения «один».

Сотрудники

Имя Сотрудника	Фамилия	Отчество	Должность	Дата рождения
Ольга	Кузнецов	Максимович	Администратор	22.10.1999
Ольга	Иванов	Сергеевич	Администратор	22.10.1969
Дмитрий	Иванов	Максимович	Менеджер	22.10.1966
Алексей	Сидоров	Петрович	Администратор	22.10.1990
Алексей	Петров	Максимович	Менеджер	22.10.1996
Дмитрий	Петров	Иванович	Администратор	22.10.1997
Дмитрий	Петров	Петрович	Консультант	22.10.1983
Иван	Кузнецов	Иванович	Менеджер	22.10.1993
Ольга	Сидоров	Сергеевич	Консультант	22.10.1985
Ольга	Иванов	Максимович	Администратор	22.10.1993
Иван	Новиков	Александрович	Администратор	22.10.1978
Алексей	Кузнецов	Александрович	Администратор	22.10.1997
Мария	Новиков	Александрович	Администратор	22.10.1968
Ольга	Новиков	Александрович	Администратор	22.10.1967
Дмитрий	Сидоров	Максимович	Администратор	22.10.1981
Дмитрий	Новиков	Александрович	Администратор	22.10.1983
Дмитрий	Сидоров	Сергеевич	Администратор	22.10.1966
Дмитрий	Новиков	Александрович	Администратор	22.10.1966

Рисунок 5 – Представление *Employees*

На рисунке 6 представлено разработанное представление *Services* (Приложение Ё), стоящее на стороне отношения «многие» в схеме базы данных, которое выводит вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один», а также группировка и фильтрация по различным полям.

Услуги

Поиск

Имя клиента:

Имя сотрудника:

Минимальная стоимость:

Максимальная стоимость:

Дата начала: дд.мм.гггг

Дата окончания: дд.мм.гггг

Найти

Дата услуги	Стоимость	Имя Клиента	Имя Сотрудника	Название услуги
16.07.2024	201,16	Иван	Дмитрий	Доставка подарков и цветов
17.07.2024	690,04	Иван	Мария	Консультации
18.07.2024	428,12	Ольга	Дмитрий	Премиум сватовство
18.07.2024	1053,10	Мария	Иван	Консультации
21.07.2024	367,53	Алексей	Алексей	Премиум сватовство
23.07.2024	910,10	Алексей	Ольга	Организация мероприятий
27.07.2024	305,66	Мария	Дмитрий	Консультации

Рисунок 6 – Представление *Services*

На рисунке 7 представлен рабочий процесс *GitHub Actions*, который осуществляет компиляцию проекта под две разные платформы при любом изменении в репозитории.

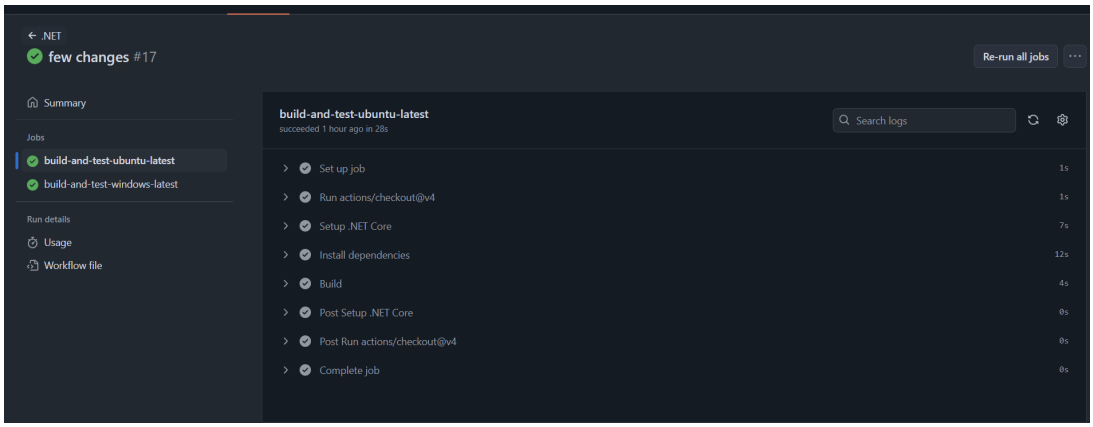

Рисунок 7 – Рабочий процесс *GitHub Actions*

На рисунке 8 представлен отредактированный *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.

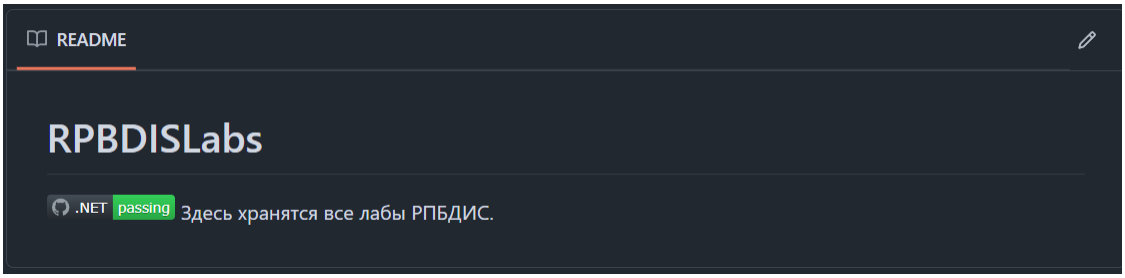

Рисунок 8 – Эмблема состояния

Выводы: был ознакомлен 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.

ПРИЛОЖЕНИЕ А
(обязательное)

Текст программы

Program.cs:

```
using MarriageAgency.Data;
using MarriageAgency.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using MarriageAgency.Services;
using MarriageAgency.Middleware;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");
builder.Services.AddDbContext<MarriageAgencyContext>(options =>
    options.UseSqlServer(connectionString));
IServiceCollection services = builder.Services;
services.AddRazorPages().AddRazorRuntimeCompilation();
services.AddDatabaseDeveloperPageExceptionFilter();

services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<MarriageAgencyContext>();

services.AddTransient<IServiceService, ServiceService>();
// добавление кэширования
services.AddMemoryCache();
// добавление поддержки сессии
services.AddDistributedMemoryCache();
services.AddSession();

//Использование MVC
services.AddControllersWithViews();
WebApplication app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    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();

// добавляем поддержку сессий
app.UseSession();

// добавляем компонент middleware по инициализации базы данных и производим инициализацию базы
app.UseDbInitializer();

// добавляем компонент middleware для реализации кэширования и записывем данные в кэш
app.UseOperatinCache("Services 10");

//Маршрутизация
app.UseRouting();
```

```
app.UseAuthorization();

// устанавливаем сопоставление маршрутов с контроллерами
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.MapRazorPages();

                                                    app.Run();
```

ПРИЛОЖЕНИЕ Б
(обязательное)

Текст программы

HomeController.cs

```
using MarriageAgency.Data;
using MarriageAgency.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

public class ClientsController : Controller
{
   private readonly MarriageAgencyContext _context;

   public ClientsController(MarriageAgencyContext context)
   {
      _context = context;
   }

   // GET: Clients
   [ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
   public async Task<IActionResult> Index()
   {
      // Включение всех зависимостей
      var clients = await _context.Clients
         .Include(c => c.Contact)
         .Include(c => c.Nationality)
         .Include(c => c.PhysicalAttribute)
         .Include(c => c.ZodiacSign)
         .Include(c => c.Photo) // Если есть такая связь
         .ToListAsync();

      return View(clients);
   }
}
using Microsoft.AspNetCore.Mvc;
using MarriageAgency.Data;
using MarriageAgency.Models;
using MarriageAgency.ViewModels;
using System.Linq;
using Microsoft.Identity.Client.Extensions.Msal;
using System.Threading.Tasks;

namespace MarriageAgency.Controllers
{
   public class HomeController(MarriageAgencyContext db) : Controller
   {
      private readonly MarriageAgencyContext _db = db;

      public IActionResult Index()
      {
         int numberRows = 10;
         List<Client> clients = [.. _db.Clients.Take(numberRows)];
         List<Employee> employees = [.. _db.Employees.Take(numberRows)];
         List<ServiceViewModel> services = [.. _db.Services
            .OrderByDescending(d => d.Date)
            .Select(s => new ServiceViewModel {
               ServiceId = s.ServiceId,
               Employee = s.Employee,
               Client = s.Client,
               AdditionalService = s.AdditionalService,
               Date = s.Date,
               Cost = s.Cost,
            })
```

```
            .Take(numberRows)];

        HomeViewModel homeViewModel = new() { Clients = clients, Employees = employees, Services = services };
        return View(homeViewModel);
    }
  }
}
```

ClientsController.cs

```
using MarriageAgency.Data;
using MarriageAgency.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

public class ClientsController : Controller
{
  private readonly MarriageAgencyContext _context;

  public ClientsController(MarriageAgencyContext context)
  {
    _context = context;
  }

  // GET: Clients
  [ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
  public async Task<IActionResult> Index()
  {
    // Включение всех зависимостей
    var clients = await _context.Clients
      .Include(c => c.Contact)
      .Include(c => c.Nationality)
      .Include(c => c.PhysicalAttribute)
      .Include(c => c.ZodiacSign)
      .Include(c => c.Photo) // Если есть такая связь
      .ToListAsync();

    return View(clients);
  }
}
```

EmployeesController.cs

```
using MarriageAgency.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace MarriageAgency.Controllers
{
  public class EmployeesController(MarriageAgencyContext context) : Controller
  {
    private readonly MarriageAgencyContext _context = context;

    // GET: Employees
    [ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
    public async Task<IActionResult> Index()
    {
      return View(await _context.Employees.ToListAsync());
    }
  }
}
```

ServicesController.cs

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
```

```
using MarriageAgency.Data;
using MarriageAgency.ViewModels;
using MarriageAgency.Models;

namespace MarriageAgency.Controllers
{
   public class ServicesController : Controller
   {
      private readonly MarriageAgencyContext _context;

      public ServicesController(MarriageAgencyContext context)
      {
         _context = context;
      }

      [ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
      public async Task<IActionResult> Index(
         string clientsSearch,
         string employeesSearch,
         DateTime? startDate,
         DateTime? endDate,
         decimal? minCost,
         decimal? maxCost,
         int? page,
         SortState sortOrder
       )

      {
         // Получаем все услуги с клиентами и сотрудниками
         var servicesQuery = _context.Services
            .Include(s => s.Client)
            .Include(s => s.Employee)
             .Include(s => s.AdditionalService)
            .AsQueryable();

         // Преобразуем в ServiceViewModel
         var serviceViewModels = await servicesQuery.Select(s => new ServiceViewModel
         {
            Date = s.Date,
            Cost = s.Cost,
            Client = s.Client,
            Employee = s.Employee,
            AdditionalService = s.AdditionalService
         }).ToListAsync();

         // Применяем фильтры
         var filteredServices = FilterServices(serviceViewModels.AsQueryable(), clientsSearch, employeesSearch, min-
Cost, maxCost, startDate, endDate);

         // Применяем сортировку
         var sortedServices = SortServices(filteredServices, sortOrder);

         // Пагинация
         int pageSize = 10;
         int pageNumber = page ?? 1;
         int totalCount = sortedServices.Count();
         var items = sortedServices
            .Skip((pageNumber - 1) * pageSize)
            .Take(pageSize)
            .ToList();

         // Формируем ViewModel
         var viewModel = new ServicesViewModel
         {
            Services = items,
            PageViewModel = new PageViewModel(totalCount, pageNumber, pageSize),
            ServiceViewModel = new ServiceViewModel
```

```
            {
                SortViewModel = new SortViewModel(sortOrder),
            }
        };

        return View(viewModel);
    }

    // Фильтрация для ServiceViewModel
    [ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
    private IQueryable<ServiceViewModel> FilterServices(
        IQueryable<ServiceViewModel> services,
        string clientsSearch,
        string employeesSearch, // Новый параметр для фильтрации по имени сотрудника
        decimal? minCost,
        decimal? maxCost,
        DateTime? startDate,
        DateTime? endDate)
    {
        // Фильтрация по имени клиента
        if (!string.IsNullOrEmpty(clientsSearch))
        {
            services = services.Where(s => s.Client.FirstName.ToLower().Contains(clientsSearch.ToLower()));
        }

        // Фильтрация по имени сотрудника
        if (!string.IsNullOrEmpty(employeesSearch))
        {
            services = services.Where(s => s.Employee.FirstName.ToLower().Contains(employeesSearch.ToLower()));
        }

        // Фильтрация по минимальной стоимости
        if (minCost.HasValue)
        {
            services = services.Where(s => s.Cost >= minCost.Value);
        }

        // Фильтрация по максимальной стоимости
        if (maxCost.HasValue)
        {
            services = services.Where(s => s.Cost <= maxCost.Value);
        }

        // Фильтрация по дате начала
        if (startDate.HasValue)
        {
            DateOnly start = DateOnly.FromDateTime(startDate.Value);
            services = services.Where(s => s.Date >= start);
        }

        // Фильтрация по дате окончания
        if (endDate.HasValue)
        {
            DateOnly end = DateOnly.FromDateTime(endDate.Value);
            services = services.Where(s => s.Date <= end);
        }

        return services;
    }

    // Сортировка для ServiceViewModel
    [ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
    private IQueryable<ServiceViewModel> SortServices(IQueryable<ServiceViewModel> services, SortState
sortOrder)
    {
        return sortOrder switch
        {
```

```
                SortState.CostAsc => services.OrderBy(s => s.Cost),
                SortState.CostDesc => services.OrderByDescending(s => s.Cost),
                SortState.ClientNameAsc => services.OrderBy(s => s.Client.FirstName),
                SortState.ClientNameDesc => services.OrderByDescending(s => s.Client.FirstName),
                SortState.EmployeeNameAsc => services.OrderBy(s => s.Employee.FirstName),
                SortState.EmployeeNameDesc => services.OrderByDescending(s => s.Employee.FirstName),
                _ => services.OrderBy(s => s.Date),
            };
        }
    }
}
```

ПРИЛОЖЕНИЕ В
(обязательное)

Текст программы

DbInitializer.cs

```
using MarriageAgency.Models;
using System;
using System.Linq;

namespace MarriageAgency.Data
{
   public static class DbInitializer
   {
      public static void Initialize(MarriageAgencyContext db)
      {
         db.Database.EnsureCreated();

         // Проверка, есть ли клиенты в базе
         if (db.Clients.Any())
         {
            return; // База данных уже инициализирована
         }

         Random rand = new(1);

         // Вставка данных для Знаков Зодиака
         string[] zodiacSigns = { "Овен", "Телец", "Близнецы", "Рак", "Лев", "Дева", "Весы", "Скорпион", "Стрелец", "Козерог", "Водолей", "Рыбы" };
         foreach (var sign in zodiacSigns)
         {
            db.ZodiacSigns.Add(new ZodiacSign { Name = sign, Description = "Описание для " + sign });
         }
         db.SaveChanges();

         // Вставка данных для Национальностей
         string[] nationalities = { "Американец", "Француз", "Немец", "Русский", "Китаец", "Индиец", "Бразилец" };
         foreach (var nationality in nationalities)
         {
            db.Nationalities.Add(new Nationality { Name = nationality, Notes = "Гражданин " + nationality });
         }
         db.SaveChanges();

         // Массивы имен и отчеств
         string[] firstNames = { "Иван", "Мария", "Алексей", "Ольга", "Дмитрий" };
         string[] lastNames = { "Иванов", "Петров", "Сидоров", "Кузнецов", "Новиков" };
         string[] middleNames = { "Иванович", "Петрович", "Сергеевич", "Александрович", "Владимирович", "Дмитриевич", "Максимович" };

         // Вставка данных для Клиентов
         for (int i = 0; i < 50; i++)
         {
            var zodiacSignId = rand.Next(1, zodiacSigns.Length + 1);
            var nationalityId = rand.Next(1, nationalities.Length + 1);

            var client = new Client
            {
               FirstName = firstNames[rand.Next(firstNames.Length)],
               LastName = lastNames[rand.Next(lastNames.Length)],
               MiddleName = middleNames[rand.Next(middleNames.Length)],
               Gender = rand.Next(2) == 0 ? "Мужской" : "Женский",
               BirthDate = DateOnly.FromDateTime(DateTime.Today.AddYears(-rand.Next(18, 60))),
               ZodiacSignId = zodiacSignId,
               NationalityId = nationalityId,
               Profession = "Профессия " + i
```

```
            };

            db.Clients.Add(client);
            db.SaveChanges();

            // Добавление Контакта для каждого клиента
            db.Contacts.Add(new Contact
            {
                ClientId = client.ClientId,
                Address = "Адрес " + i,
                Phone = "+7916123456" + i,
                PassportData = "Паспортные данные " + i
            });

            // Добавление Физических характеристик для каждого клиента
            db.PhysicalAttributes.Add(new PhysicalAttribute
            {
                ClientId = client.ClientId, // Связываем с клиентом по ID
                Age = rand.Next(18, 60),
                Height = (decimal)(150 + rand.NextDouble() * 50),
                Weight = (decimal)(50 + rand.NextDouble() * 50),
                ChildrenCount = rand.Next(0, 5),
                MaritalStatus = rand.Next(2) == 0 ? "Холост" : "Женат",
                BadHabits = rand.Next(2) == 0 ? "Нет" : "Курение",
                Hobbies = "Хобби " + i
            });
        }
        db.SaveChanges();

        // Вставка данных для Сотрудников
        string[] employeePositions = { "Менеджер", "Консультант", "Администратор" };
        for (int i = 0; i < 20; i++)
        {
            db.Employees.Add(new Employee
            {
                FirstName = firstNames[rand.Next(firstNames.Length)],
                LastName = lastNames[rand.Next(lastNames.Length)],
                MiddleName = middleNames[rand.Next(middleNames.Length)],
                Position = employeePositions[rand.Next(employeePositions.Length)],
                BirthDate = DateOnly.FromDateTime(DateTime.Today.AddYears(-rand.Next(25, 60)))
            });
        }
        db.SaveChanges();

        // Вставка данных для Дополнительных Услуг
        string[] additionalServices = { "Премиум сватовство", "Консультации", "Организация мероприятий", "До-
ставка подарков и цветов" };
        foreach (var serviceName in additionalServices)
        {
            db.AdditionalServices.Add(new AdditionalService
            {
                Name = serviceName,
                Description = "Описание услуги: " + serviceName,
                Price = (decimal)(100 + rand.NextDouble() * 500)
            });
        }
        db.SaveChanges();

        // Вставка данных для Услуг
        for (int i = 0; i < 100; i++)
        {
            db.Services.Add(new Service
            {
                ClientId = rand.Next(1, 51),
                EmployeeId = rand.Next(1, 21),
                AdditionalServiceId = rand.Next(1, additionalServices.Length + 1),
                Date = DateOnly.FromDateTime(DateTime.Today.AddDays(-rand.Next(1, 100))),
```

```
                Cost = (decimal)(200 + rand.NextDouble() * 1000)
            });
        }
        db.SaveChanges();
    }
  }
}
```

DbInitializerMiddleware.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MarriageAgency.Models;
using MarriageAgency.Data;

namespace MarriageAgency.Middleware
{
  public class DbInitializerMiddleware
  {
    private readonly RequestDelegate _next;

    public DbInitializerMiddleware(RequestDelegate next)
    {
      _next = next;
    }

    public Task Invoke(HttpContext context, IServiceProvider serviceProvider, MarriageAgencyContext db)
    {
      if (!(context.Session.Keys.Contains("starting")))
      {
        DbInitializer.Initialize(db);
        context.Session.SetString("starting", "Yes");
      }

      return _next.Invoke(context);
    }
  }

  public static class DbInitializerExtensions
  {
    public static IApplicationBuilder UseDbInitializer(this IApplicationBuilder builder)
    {
      return builder.UseMiddleware<DbInitializerMiddleware>();
    }

  }
}
```

DbCacheMiddleware.cs

```
using MarriageAgency.ViewModels;
using Microsoft.Extensions.Caching.Memory;
using MarriageAgency.Services;

namespace MarriageAgency.Middleware
{
  public class DbCacheMiddleware(RequestDelegate next, IMemoryCache memoryCache, string cacheKey = "Services 10")
  {
    private readonly RequestDelegate _next = next;
    private readonly IMemoryCache _memoryCache = memoryCache;
    private readonly string _cacheKey = cacheKey;

    public Task Invoke(HttpContext httpContext, IServiceService operationService)
```

```
        {
            // пытаемся получить элемент из кэша
            if (!_memoryCache.TryGetValue(_cacheKey, out HomeViewModel homeViewModel))
            {
                // если в кэше не найден элемент, получаем его от сервиса
                homeViewModel = operationService.GetHomeViewModel();
                // и сохраняем в кэше
                _memoryCache.Set(_cacheKey, homeViewModel,
                    new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(1)));

            }

            return _next(httpContext);
        }
    }

    // Метод расширения, используемый для добавления промежуточного программного обеспечения в конвейер
HTTP-запроса.
    public static class DbCacheMiddlewareExtensions
    {
        public static IApplicationBuilder UseOperatinCache(this IApplicationBuilder builder, string cacheKey)
        {
            return builder.UseMiddleware<DbCacheMiddleware>(cacheKey);
        }
    }
}
```

ПРИЛОЖЕНИЕ Г
(обязательное)

Текст программы

IServiceService.cs
```
using MarriageAgency.ViewModels;

namespace MarriageAgency.Services
{
   public interface IServiceService
   {
      HomeViewModel GetHomeViewModel(int numberRows = 10);
   }
}
```

ServiceService.cs
```
using MarriageAgency.ViewModels;
using MarriageAgency.Data;

namespace MarriageAgency.Services
{
   // Класс выборки 10 записей из всех таблиц
   public class ServiceService(MarriageAgencyContext context) : IServiceService
   {
      private readonly MarriageAgencyContext _context = context;

      public HomeViewModel GetHomeViewModel(int numberRows = 10)
      {
         var clients = _context.Clients.Take(numberRows).ToList();
         var employees = _context.Employees.Take(numberRows).ToList();
         List<ServiceViewModel> services = [.. _context.Services
            .OrderByDescending(d => d.Date)
            .Select(s => new ServiceViewModel
            {
               ServiceId = s.ServiceId,
               Date = s.Date,
               Cost = s.Cost
            })
            .Take(numberRows)];

         HomeViewModel homeViewModel = new()
         {
            Clients = clients,
            Employees = employees,
            Services = services
         };
         return homeViewModel;
      }
   }
}
```

ПРИЛОЖЕНИЕ Д
(обязательное)

Текст программы

Main.yml

```
name: .NET

on:
  push:
  pull_request:
    branches: [ main ]
    paths:
    - '**.cs'
    - '**.csproj'

env:
  DOTNET_VERSION: '8.0' # The .NET SDK version to use

jobs:
  build-and-test:
    name: build-and-test-${{matrix.os}}
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-latest, windows-latest]

    steps:
    - uses: actions/checkout@v4

    - name: Setup .NET Core
      uses: actions/setup-dotnet@v4
      with:
        dotnet-version: ${{ env.DOTNET_VERSION }}

    - name: Install dependencies
      run: dotnet restore lab2/EFCoreLINQ/EFCoreLINQ/EFCoreLINQ.csproj

    - name: Build
      run: dotnet build lab2/EFCoreLINQ/EFCoreLINQ/EFCoreLINQ.csproj --configuration Release --no-restore
```

ПРИЛОЖЕНИЕ Е
(обязательное)

Текст программы

appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=HOME-
PC;Database=MarriageAgencyDB;Trusted_Connection=True;TrustServerCertificate=True;MultipleActiveResultSets=t
rue"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

ПРИЛОЖЕНИЕ Ё
(обязательное)

Текст программы

Clients
Index.cshtml

```
@model IEnumerable<MarriageAgency.Models.Client>

@{
    ViewData["Title"] = "Клиенты";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Клиенты</h2>

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.FirstName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.LastName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.MiddleName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Gender)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.BirthDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ZodiacSign.Name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Nationality.Name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Contact.Phone)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.PhysicalAttribute.Age)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Profession)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var client in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => client.FirstName)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => client.LastName)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => client.MiddleName)
                </td>
```

```
            <td>
                @Html.DisplayFor(modelItem => client.Gender)
            </td>
            <td>
                @Html.DisplayFor(modelItem => client.BirthDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => client.ZodiacSign.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => client.Nationality.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => client.Contact.Phone)
            </td>
            <td>
                @Html.DisplayFor(modelItem => client.PhysicalAttribute.Age)
            </td>
            <td>
                @Html.DisplayFor(modelItem => client.Profession)
            </td>
        </tr>
    }
  </tbody>
</table>
```

Employees
Index.cshtml

```
@model IEnumerable<MarriageAgency.Models.Employee>

@{
  ViewData["Title"] = "Сотрудники";
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Сотрудники</h2>

<table class="table">
  <thead>
    <tr>
      <th>
        @Html.DisplayNameFor(model => model.FirstName)
      </th>
      <th>
        @Html.DisplayNameFor(model => model.LastName)
      </th>
      <th>
        @Html.DisplayNameFor(model => model.MiddleName)
      </th>
      <th>
        @Html.DisplayNameFor(model => model.Position)
      </th>
      <th>
        @Html.DisplayNameFor(model => model.BirthDate)
      </th>
    </tr>
  </thead>
  <tbody>
    @foreach (var employee in Model)
    {
      <tr>
        <td>
          @Html.DisplayFor(modelItem => employee.FirstName)
        </td>
        <td>
          @Html.DisplayFor(modelItem => employee.LastName)
```

```
            </td>
            <td>
                @Html.DisplayFor(modelItem => employee.MiddleName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => employee.Position)
            </td>
            <td>
                @Html.DisplayFor(modelItem => employee.BirthDate)
            </td>
        </tr>
    }
  </tbody>
</table>
```

Home
Index.cshtml

```
@model MarriageAgency.ViewModels.HomeViewModel
@{
  ViewData["Title"] = "Информация о брачном агентстве";
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Клиенты</h2>
<table class="table">
  <thead>
    <tr>
      <th>Код клиента</th>
      <th>Имя</th>
      <th>Фамилия</th>
      <th>Отчество</th>
      <th>Пол</th>
      <th>Дата рождения</th>
      <th>Профессия</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var client in Model.Clients)
    {
      <tr>
        <td>@client.ClientId</td>
        <td>@client.FirstName</td>
        <td>@client.LastName</td>
        <td>@client.MiddleName</td>
        <td>@client.Gender</td>
        <td>@client.BirthDate</td>
        <td>@client.Profession</td>
      </tr>
    }
  </tbody>
</table>

<h2>Сотрудники</h2>
<table class="table">
  <thead>
    <tr>
      <th>Код сотрудника</th>
      <th>Имя</th>
      <th>Фамилия</th>
      <th>Отчество</th>
      <th>Дата рождения</th>
      <th>Должность</th>
    </tr>
  </thead>
  <tbody>
```

```
        @foreach (var employee in Model.Employees)
        {
            <tr>
                <td>@employee.EmployeeId</td>
                <td>@employee.FirstName</td>
                <td>@employee.LastName</td>
                <td>@employee.MiddleName</td>
                <td>@employee.BirthDate</td>
                <td>@employee.Position</td>
            </tr>
        }
    </tbody>
</table>

<h2>Услуги</h2>
<table class="table">
    <thead>
        <tr>
            <th>Код услуги</th>
            <th>Доп.Услуга</th>
            <th>Сотрудник</th>
            <th>Клиент</th>
            <th>Дата</th>
            <th>Стоимость</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var service in Model.Services)
        {
            <tr>
                <td>@service.ServiceId</td>
                <td>@service.AdditionalService.Name</td>
                <td>@service.Employee.FirstName</td>
                <td>@service.Client.FirstName</td>
                <td>@service.Date.ToString("dd/MM/yyyy")</td>
                <td>@service.Cost</td>
            </tr>
        }
    </tbody>
</table>
```

Services
Index.cshtml

```
@model MarriageAgency.ViewModels.ServicesViewModel

@{
    ViewData["Title"] = "Услуги";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Услуги</h2>

<!-- Форма поиска с фильтрацией по имени клиента, дате и стоимости -->
@using (Html.BeginForm("Index", "Services", FormMethod.Get))
{
    <fieldset>
        <legend>Поиск</legend>
        <div>
            <label for="clientsSearch">Имя клиента:</label>
            @Html.TextBox("clientsSearch", Model.ServiceViewModel.Client, new { @class = "form-control" })
        </div>
        <div>
            <label for="employeesSearch">Имя сотрудника:</label>
            @Html.TextBox("employeesSearch", Model.ServiceViewModel.Employee, new { @class = "form-control" })
        </div>
        <div>
```

```
            <label for="minCost">Минимальная стоимость:</label>
            @Html.TextBox("minCost", Model.ServiceViewModel.SortViewModel.CurrentState, new { @class = "form-
control", type = "number", step = "0.01" })
        </div>
        <div>
            <label for="maxCost">Максимальная стоимость:</label>
            @Html.TextBox("maxCost", Model.ServiceViewModel.SortViewModel.CurrentState, new { @class = "form-
control", type = "number", step = "0.01" })
        </div>
        <div>
            <label for="startDate">Дата начала:</label>
            @Html.TextBox("startDate", Model.ServiceViewModel.SortViewModel.CurrentState, new { @class = "form-
control", type = "date" })
        </div>
        <div>
            <label for="endDate">Дата окончания:</label>
            @Html.TextBox("endDate", Model.ServiceViewModel.SortViewModel.CurrentState, new { @class = "form-
control", type = "date" })
        </div>
        <input type="submit" value="Найти" class="btn btn-primary" />
    </fieldset>
}

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.ServiceViewModel.Date)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ServiceViewModel.Cost)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ServiceViewModel.Client.FirstName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ServiceViewModel.Employee.FirstName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ServiceViewModel.AdditionalService.Name)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var service in Model.Services)
        {
            <tr>
                <td>@Html.DisplayFor(modelItem => service.Date)</td>
                <td>@Html.DisplayFor(modelItem => service.Cost)</td>
                <td>@Html.DisplayFor(modelItem => service.Client.FirstName)</td>
                <td>@Html.DisplayFor(modelItem => service.Employee.FirstName)</td>
                <td>@Html.DisplayFor(modelItem => service.AdditionalService.Name)</td>
            </tr>
        }
    </tbody>
</table>

<div>
    Страница @Model.PageViewModel.PageNumber из @Model.PageViewModel.TotalPages
    @if (Model.PageViewModel.HasPreviousPage)
    {
        <a asp-action="Index" asp-route-page="@(Model.PageViewModel.PageNumber - 1)">Предыдущая</a>
    }
    @if (Model.PageViewModel.HasNextPage)
    {
        <a asp-action="Index" asp-route-page="@(Model.PageViewModel.PageNumber + 1)">Следующая</a>
    }
```

ПРИЛОЖЕНИЕ Ж
(обязательное)

Текст программы

MarriageAgencyContext.cs
```
using Microsoft.EntityFrameworkCore;
using MarriageAgency.Models;

namespace MarriageAgency.Data;

public partial class MarriageAgencyContext(DbContextOptions<MarriageAgencyContext> options) : DbContext(options)
{
    public virtual DbSet<AdditionalService> AdditionalServices { get; set; }

    public virtual DbSet<Client> Clients { get; set; }

    public virtual DbSet<Contact> Contacts { get; set; }

    public virtual DbSet<Employee> Employees { get; set; }

    public virtual DbSet<Nationality> Nationalities { get; set; }

    public virtual DbSet<Photo> Photos { get; set; }

    public virtual DbSet<PhysicalAttribute> PhysicalAttributes { get; set; }

    public virtual DbSet<Service> Services { get; set; }

    public virtual DbSet<ZodiacSign> ZodiacSigns { get; set; }
}
```

AdditionalService.cs
```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class AdditionalService
    {
        [Display(Name = "Код доп. услуги")]
        public int AdditionalServiceId { get; set; }

        [Display(Name = "Название услуги")]
        public string Name { get; set; } = null!;

        [Display(Name = "Описание услуги")]
        public string? Description { get; set; }

        [Display(Name = "Цена услуги")]
        public decimal Price { get; set; }

        public virtual ICollection<Service> Services { get; set; } = new List<Service>();
    }
}
```

Clients.cs
```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class Client
    {
        //Id Клиента
```

```
        [Display(Name = "Код клиента")]
        public int ClientId { get; set; }

        [Display(Name = "Имя Клиента")]
        public string FirstName { get; set; } = "";

        [Display(Name = "Фамилия")]
        public string LastName { get; set; } = null!;

        [Display(Name = "Отчество")]
        public string? MiddleName { get; set; }

        [Display(Name = "Пол")]
        public string Gender { get; set; } = null!;

        [Display(Name = "Дата рождения")]
        public DateOnly? BirthDate { get; set; }

        [Display(Name = "Знак зодиака")]
        public int? ZodiacSignId { get; set; }

        [Display(Name = "Национальность")]
        public int? NationalityId { get; set; }

        [Display(Name = "Профессия")]
        public string? Profession { get; set; }

        [Display(Name = "Контактные данные")]
        public virtual Contact? Contact { get; set; }

        [Display(Name = "Национальность")]
        public virtual Nationality? Nationality { get; set; }

        [Display(Name = "Фото")]
        public virtual Photo? Photo { get; set; }

        [Display(Name = "Физические данные")]
        public virtual PhysicalAttribute? PhysicalAttribute { get; set; }

        [Display(Name = "Знак зодиака")]
        public virtual ZodiacSign? ZodiacSign { get; set; }

        [Display(Name = "Услуги")]
        public virtual ICollection<Service> Services { get; set; } = new List<Service>();
    }
}
```

Contact.cs

```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class Contact
    {
        [Key]
        public int ClientId { get; set; }

        [Display(Name = "Адрес")]
        public string? Address { get; set; }

        [Display(Name = "Телефон")]
        public string? Phone { get; set; }

        [Display(Name = "Паспортные данные")]
        public string? PassportData { get; set; }
```

```
        [Display(Name = "Клиент")]
        public virtual Client Client { get; set; } = null!;
    }
}
```

Employee.cs

```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class Employee
    {
        [Display(Name = "Код сотрудника")]
        public int EmployeeId { get; set; }

        [Display(Name = "Имя Сотрудника")]
        public string FirstName { get; set; } = null!;

        [Display(Name = "Фамилия")]
        public string LastName { get; set; } = null!;

        [Display(Name = "Отчество")]
        public string MiddleName { get; set; } = null!;

        [Display(Name = "Должность")]
        public string? Position { get; set; }

        [Display(Name = "Дата рождения")]
        public DateOnly? BirthDate { get; set; }

        [Display(Name = "Услуги")]
        public virtual ICollection<Service> Services { get; set; } = new List<Service>();
    }
}
```

Nationality.cs

```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class Nationality
    {
        [Display(Name = "Код национальности")]
        public int NationalityId { get; set; }

        [Display(Name = "Национальность")]
        public string Name { get; set; } = null!;

        [Display(Name = "Примечания")]
        public string? Notes { get; set; }

        [Display(Name = "Клиенты")]
        public virtual ICollection<Client> Clients { get; set; } = new List<Client>();
    }
}
```

Photo.cs

```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class Photo
    {
        [Key]
        [Display(Name = "Код клиента")]
        public int ClientId { get; set; }
```

```
        [Display(Name = "Фото клиента")]
        public string ClientPhoto { get; set; } = null!;

        [Display(Name = "Клиент")]
        public virtual Client Client { get; set; } = null!;
    }
}
```

PhysicalAttributes.cs

```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class PhysicalAttribute
    {
        [Key]
        [Display(Name = "Код клиента")]
        public int ClientId { get; set; }

        [Display(Name = "Возраст")]
        public int? Age { get; set; }

        [Display(Name = "Рост")]
        public decimal? Height { get; set; }

        [Display(Name = "Вес")]
        public decimal? Weight { get; set; }

        [Display(Name = "Количество детей")]
        public int? ChildrenCount { get; set; }

        [Display(Name = "Семейное положение")]
        public string? MaritalStatus { get; set; }

        [Display(Name = "Вредные привычки")]
        public string? BadHabits { get; set; }

        [Display(Name = "Хобби")]
        public string? Hobbies { get; set; }

        [Display(Name = "Клиент")]
        public virtual Client Client { get; set; } = null!;
    }
}
```

Service.cs

```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class Service
    {
        [Display(Name = "Код услуги")]
        public int ServiceId { get; set; }

        [Display(Name = "Код дополнительной услуги")]
        public int AdditionalServiceId { get; set; }

        [Display(Name = "Код клиента")]
        public int ClientId { get; set; }

        [Display(Name = "Код сотрудника")]
        public int EmployeeId { get; set; }

        [Display(Name = "Дата услуги")]
```

```
        public DateOnly Date { get; set; }

        [Display(Name = "Стоимость")]
        public decimal Cost { get; set; }

        [Display(Name = "Дополнительная услуга")]
        public virtual AdditionalService AdditionalService { get; set; } = null!;

        [Display(Name = "Клиент")]
        public virtual Client Client { get; set; } = null!;

        [Display(Name = "Сотрудник")]
        public virtual Employee Employee { get; set; } = null!;
    }
}
```

ZodiacSign.cs

```
using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.Models
{
    public partial class ZodiacSign
    {
        [Display(Name = "Код знака зодиака")]
        public int ZodiacSignId { get; set; }

        [Display(Name = "Знак зодиака")]
        public string Name { get; set; } = null!;

        [Display(Name = "Описание")]
        public string? Description { get; set; }

        [Display(Name = "Клиенты")]
        public virtual ICollection<Client> Clients { get; set; } = new List<Client>();
    }
}
```

ПРИЛОЖЕНИЕ 3
(обязательное)

Текст программы

HomeViewModel.cs
```
using System.Threading.Tasks;
using MarriageAgency.Models;
using Microsoft.Identity.Client.Extensions.Msal;

namespace MarriageAgency.ViewModels
{
   public class HomeViewModel
   {
      public IEnumerable<Client> Clients { get; set; }
      public IEnumerable<Employee> Employees { get; set; }
      public IEnumerable<ServiceViewModel> Services { get; set; }
   }
}
```

PageViewModel.cs
```
namespace MarriageAgency.ViewModels
{
   //Класс для хранения информации о страницах разбиения
   public class PageViewModel(int count, int pageNumber, int pageSize)
   {
      public int PageNumber { get; private set; } = pageNumber;
      public int TotalPages { get; private set; } = (int)Math.Ceiling(count / (double)pageSize);

      public bool HasPreviousPage
      {
         get
         {
            return (PageNumber > 1);
         }
      }

      public bool HasNextPage
      {
         get
         {
            return (PageNumber < TotalPages);
         }
      }
   }
}
```

ServicesViewModel.cs
```
using Microsoft.AspNetCore.Mvc.Rendering;
using System.Collections.Generic;
using MarriageAgency.Models;

namespace MarriageAgency.ViewModels
{
   public class ServicesViewModel
   {
      public IEnumerable<ServiceViewModel> Services { get; set; }

      //Свойство для фильтрации
      public ServiceViewModel ServiceViewModel { get; set; }
      //Свойство для навигации по страницам
      public PageViewModel PageViewModel { get; set; }
   }
}
```

ServiceViewModel.cs

```
using System.ComponentModel.DataAnnotations;
using MarriageAgency.Models;

namespace MarriageAgency.ViewModels
{
    public class ServiceViewModel
    {
        [Display(Name = "Код услуги")]
        public int ServiceId { get; set; }

        [Display(Name = "Дата услуги")]
        [DataType(DataType.Date)]
        [Required(ErrorMessage = "Дата услуги обязательна.")]
        public DateOnly Date { get; set; }

        [Display(Name = "Стоимость")]
        [Range(0.01, double.MaxValue, ErrorMessage = "Стоимость должна быть положительным числом.")]
        [Required(ErrorMessage = "Стоимость обязательна.")]
        public decimal Cost { get; set; }

        [Display(Name = "Дополнительная услуга")]
        public virtual AdditionalService AdditionalService { get; set; } = null!;

        [Display(Name = "Клиент")]
        public virtual Client Client { get; set; } = null!;

        [Display(Name = "Сотрудник")]
        public virtual Employee Employee { get; set; } = null!;

        public SortViewModel SortViewModel { get; set; }
    }
}
```

SortViewModel.cs

```
namespace MarriageAgency.ViewModels
{
    public enum SortState
    {
        No,             // не сортировать
        CostAsc,        // по стоимости по возрастанию
        CostDesc,       // по стоимости по убыванию
        ClientNameAsc,  // по имени клиента по возрастанию
        ClientNameDesc, // по имени клиента по убыванию
        EmployeeNameAsc, // по имени сотрудника по возрастанию
        EmployeeNameDesc // по имени сотрудника по убыванию
    }

    public class SortViewModel
    {
        public SortState CostSort { get; set; }       // Сортировка по стоимости
        public SortState ClientNameSort { get; set; } // Сортировка по имени клиента
        public SortState EmployeeNameSort { get; set; } // Сортировка по имени сотрудника
        public SortState CurrentState { get; set; }     // Текущее состояние сортировки

        public SortViewModel(SortState sortOrder)
        {

            // Установка сортировки для стоимости
            CostSort = sortOrder == SortState.CostAsc ? SortState.CostDesc : SortState.CostAsc;

            // Установка сортировки для имени клиента
            ClientNameSort = sortOrder == SortState.ClientNameAsc ? SortState.ClientNameDesc : Sort-
State.ClientNameAsc;

            // Установка сортировки для имени сотрудника
```

```
            EmployeeNameSort = sortOrder == SortState.EmployeeNameAsc ? SortState.EmployeeNameDesc : Sort-
State.EmployeeNameAsc;

            // Установка текущего состояния сортировки
            CurrentState = sortOrder;
        }
    }
}
```