

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 №5

«Разработка интерфейса приложения баз данных с использованием
аутентификации и авторизации»

ВАРИАНТ №12

Брачное агентство

Выполнил:

студент группы ИТИ-31, Зеленский К.А.

Принял:

доцент Асенчик О.Д.

Гомель 2024

Цель работы: получить навыки использования *ASP.NET MVC Core* для создания интерфейса типовых *web*-приложений для работы с информацией из реляционных баз данных с использованием аутентификации и авторизации.

Ход работы и результаты.

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

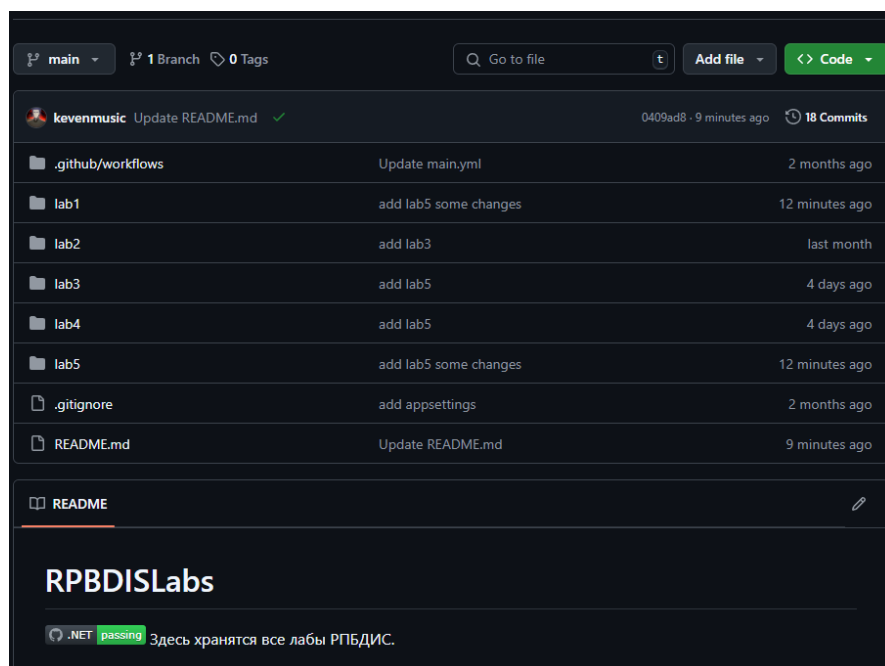


Рисунок 1 – *GitHub* Репозиторий

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

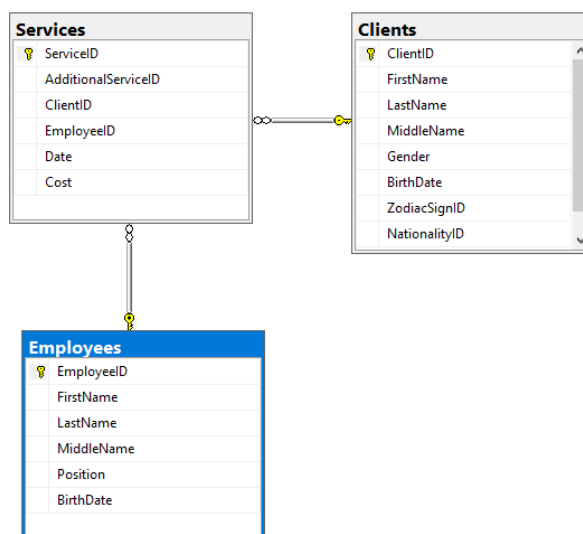


Рисунок 2 – Таблицы согласно варианту

На рисунке 3 представлен просмотр данных таблицы «Клиенты», где предусмотрено разбиение данных на страницы, фильтрация по одному или нескольким полям, а также реализовано сохранение состояния (значений) элементов представлений, используя объект *Session*.

Клиенты

Создать

Имя клиента:

Иван

Пол:

Мужской

Национальности:

Француз


Знак зодиака:

Рак

Возраст:

Хобби:

Найти

Фото	Клиент ↑	Фамилия	Отчество	Пол ↑	Дата рождения	Профессия	Телефон	Возраст ↑	Хобби ↑	Национальность ↑	Знак зодиака ↑	Действие
	Иван	Васильев	Павлович	Мужской	12.11.1968	Экономист	+72.42069e+009	52	Путешествия	Француз	Рак	<div>Редактировать</div> <div>Подробнее</div> <div>Удалить</div>

1

Рисунок 3 – Просмотр данных таблицы «Клиенты»

На рисунке 4 представлено добавление данных в таблицу «Клиенты».

Создать

Клиент

Имя Клиента

Клиент

Отчество

Отчество

Фамилия

Фамилия

Пол

Мужской

Профессия

Программист

Дата рождения

01.02.2006

Национальность

Русский

Знак зодиака


Овен

Выберите файл

Выберите файл

3.png

3.png



Создать

Вернуться к списку

Рисунок 4 – Добавление данных в таблицу «Клиенты»

На рисунке 5 представлено редактирование данных таблицы «Клиенты».

Редактировать

Клиент

Имя Клиента
Иван

Отчество
Павлович

Фамилия
Васильев

Пол
Мужской


Профессия
Экономист

Дата рождения
12.11.1968

Национальность
Француз

Знак зодиака
Рак

Front Image
Выберите файл Файл не выбран Выберите файл




Сохранить

Рисунок 5 – Редактирование данных таблицы «Клиенты»

На рисунке 6 представлено удаление данных таблицы «Клиенты».

Удаление

Вы уверены, что хотите удалить этого клиента?

Фото


Имя Клиента
Иван

Отчество
Павлович

Фамилия
Васильев

Профессия
Экономист

Дата рождения
12.11.1968

Название национальности
Француз

Название знака
Рак

Удалить Вернуться к списку

Рисунок 6 – Удаление данных таблицы «Клиенты»

На рисунке 7 представлен просмотр данных таблицы «Сотрудники», где предусмотрено разбиение данных на страницы, фильтрация по одному или нескольким полям, а также реализовано сохранение состояния (значений) элементов представлений, используя объект *Session*.

Сотрудники

Создать

Имя сотрудника:

Максим

Фамилия сотрудника:

Семёнов

Отчество сотрудника:

Дмитриевич

Дата рождения:

12.11.2006

Найти

Имя Сотрудника ↑	Фамилия ↑	Отчество ↑	Должность	Дата рождения ↑	Действие		
Максим	Семёнов	Дмитриевич	Маркетолог	12.11.2006	Редактировать	Подробнее	Удалить

1

Рисунок 7 – Просмотр данных таблицы «Сотрудники»

На рисунке 8 представлен просмотр данных таблицы «Услуги», где предусмотрено разбиение данных на страницы, фильтрация по одному или нескольким полям, а также реализовано сохранение состояния (значений) элементов представлений, используя объект *Session*.

Услуги

Создать

Имя клиента:

Имя сотрудника:

Услуга:

Выберите услугу

Мин. цена:

500

Макс. цена:

1000

Найти

Клиент ↑	Сотрудник ↑	Дополнительная услуга ↑	Дата услуги	Стоимость ↓	Действие		
Максим	Артем	Подарочные сертификаты	08.05.2024	1000,00	Редактировать	Подробнее	Удалить
Наталья	Юлия	Подарочные сертификаты	18.01.2024	1000,00	Редактировать	Подробнее	Удалить
Дмитрий	Светлана	Подарочные сертификаты	19.12.2023	1000,00	Редактировать	Подробнее	Удалить
Ольга	Анатолий	Подарочные сертификаты	18.07.2024	1000,00	Редактировать	Подробнее	Удалить
Роман	Роман	Подарочные сертификаты	16.10.2024	1000,00	Редактировать	Подробнее	Удалить
Роман	Юлия	Подарочные сертификаты	09.11.2024	1000,00	Редактировать	Подробнее	Удалить

500&MaxCost=1000

Рисунок 8 – Просмотр данных таблицы «Услуги»

На рисунке 9 представлена аутентификация администратора при помощи *ASP.NET Identity*.

Войти

Используйте локальную учетную запись для входа.

Электронная почта
admin@gmail.com

Пароль

☐ Remember me?

Войти

[Забыли пароль?](#)

[Зарегистрироваться как новый пользователь](#)

[Отправить подтверждение на почту снова](#)

Используйте другой сервис для входа.

Нет настроенных внешних сервисов для аутентификации. Посмотрите эту [статью о настройке приложения ASP.NET для входа через внешние сервисы](#).

Рисунок 9 – Аутентификация администратора

На рисунке 10 представлена реализация двух ролевых политик.

Пользователи

Добавить

Имя	Email	Дата регистрации	Роль	
admin@gmail.com	admin@gmail.com	06.11.2024	admin	Изменить Сменить пароль Удалить
kevenmusic538@gmail.com	kevenmusic538@gmail.com	06.11.2024	user	Изменить Сменить пароль Удалить

Рисунок 10 – Две ролевые политики

На рисунке 11 представлено успешно пройденное модульное тестирование.

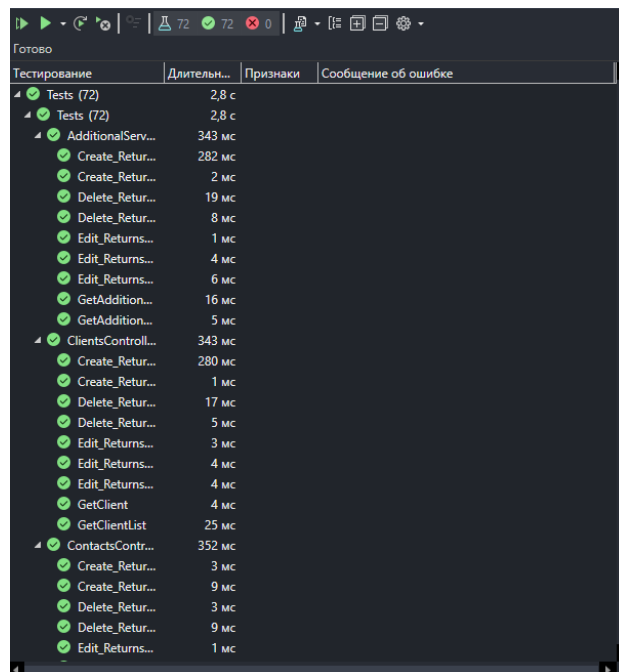


Рисунок 11 – Модульное тестирование

На рисунке 12 представлен рабочий процесс *GitHub Actions*, который осуществляет компиляцию проекта под две разные платформы при любом изменении в репозитории.

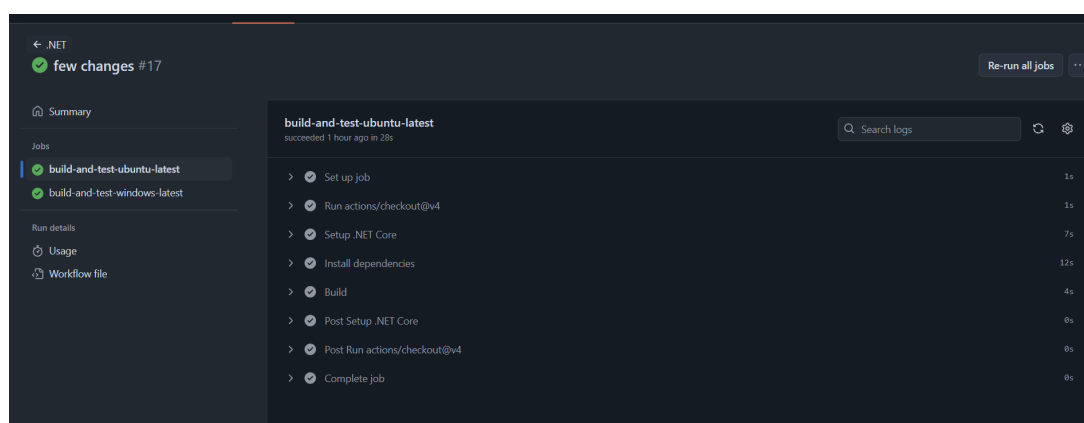


Рисунок 12 – Рабочий процесс *GitHub Actions*

На рисунке 13 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.

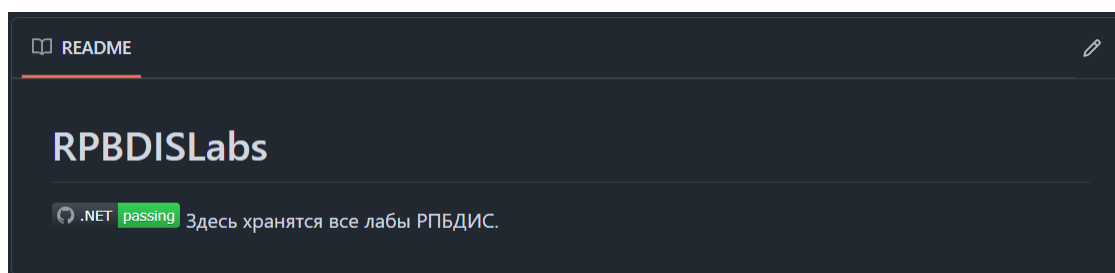


Рисунок 13 – Эмблема состояния

Выводы: были получены навыки использования *ASP.NET MVC Core* для создания интерфейса типовых *web*-приложений для работы с информацией из реляционных баз данных с использованием аутентификации и авторизации.

ПРИЛОЖЕНИЕ А

(обязательное)

Текст программы

Program.cs:

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

var builder = WebApplication.CreateBuilder(args);

// Получаем строку подключения
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection")
    ?? throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");

builder.Services.AddDbContext<MarriageAgencyContext>(options => options.UseSqlServer(connectionString, b =>
    b.MigrationsAssembly("MarriageAgency")));
string connectionUsers = builder.Configuration.GetConnectionString("IdentityConnection");
// Настройка контекста для Identity (пользователи, роли, аутентификация и т.д.)
builder.Services.AddDbContext<ApplicationDbContext>(options => options.UseSqlServer(connectionUsers));

// Настройка Identity с использованием ApplicationDbContext для аутентификации
builder.Services.AddIdentity<ApplicationUser, IdentityRole>(options => options.SignIn.RequireConfirmedAccount =
    false)
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultUI()
    .AddDefaultTokenProviders();

// Добавление RazorPages и MVC
builder.Services.AddRazorPages().AddRazorRuntimeCompilation();
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

// Добавление сервисов для Identity
builder.Services.AddTransient<IServiceService, ServiceService>();

// Настройка кэширования
builder.Services.AddMemoryCache();

// Настройка сессий
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30); // Время жизни сессии
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});

builder.Services.Configure<CookiePolicyOptions>(options =>
{
    options.CheckConsentNeeded = context => true;
    options.MinimumSameSitePolicy = SameSiteMode.None;
});

// Использование MVC
builder.Services.AddControllersWithViews(options => options.SuppressImplicitRequiredAttributeForNonNullableReferenceTypes = true);
//Использование RazorPages
```



```
builder.Services.AddRazorPages();
WebApplication app = builder.Build();

// Настройка HTTP конвейера
if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseCookiePolicy();
// Добавляем поддержку сессий
app.UseSession();

// Добавляем компонент middleware для кэширования
app.UseOperatinCache("Services 10");
// добавляем компонента miiddleware по инициализации базы данных
app.UseDbInitializer();
// Маршрутизация
app.UseRouting();

// Использование Identity
app.UseAuthentication();
app.UseAuthorization();

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

app.Run();
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Текст программы

HomeController.cs

```
using MarriageAgency.DataLayer.Data;
using MarriageAgency.DataLayer.Models;
using MarriageAgency.ViewModels;
using MarriageAgency.ViewModels.ServicesViewModel;
using Microsoft.AspNetCore.Mvc;

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

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

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

ClientsController.cs

```
using MarriageAgency.DataLayer.Data;
using MarriageAgency.DataLayer.Models;
using MarriageAgency.Infrastructure;
using MarriageAgency.Infrastructure.Filters;
using MarriageAgency.ViewModels;
using MarriageAgency.ViewModels.ClientsViewModel;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
public class ClientsController : Controller
{
    private readonly int pageSize = 10; // количество элементов на странице
    private readonly MarriageAgencyContext _context;
    private readonly IWebHostEnvironment _webHostEnvironment;

    public ClientsController(MarriageAgencyContext context, IWebHostEnvironment webHost, IConfiguration appCon-
fig = null)
    {
        _webHostEnvironment = webHost;
        _context = context;
        if (appConfig != null)
```

```

    {
        pageSize = int.Parse(appConfig["Parameters:PageSize"]);
    }
}

// GET: Clients
[ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
[SetToSession("Client")]
[Authorize]
public async Task<IActionResult> Index(FilterClientsViewModel client, SortState sortOrder = SortState.No, int
page = 1)
{
    if (client.ClientName == null && client.Gender == null && client.NationalityName == null
        && client.ZodiacSignName == null && client.Age == null && client.Hobbies == null)
    {
        if (HttpContext != null)
        {
            var sessionClient = MarriageAgency.Infrastructure.SessionExtensions.Get(HttpContext.Session, "Client");

            if (sessionClient != null)
                client = Transformations.DictionaryToObject<FilterClientsViewModel>(sessionClient);
        }
    }

    // Сортировка и фильтрация данных
    IQueryable<Client> marriageAgencyContext = _context.Clients
        .Include(c => c.Contact) // Загружаем связанные записи Contact
        .Include(c => c.Nationality) // Загружаем связанные записи Nationality
        .Include(c => c.PhysicalAttribute) // Загружаем связанные записи PhysicalAttribute
        .Include(c => c.ZodiacSign); // Загружаем связанные записи ZodiacSign

    marriageAgencyContext = Sort_Search(marriageAgencyContext, sortOrder, client.ClientName ?? "",
        client.Gender ?? "", client.NationalityName ?? "", client.ZodiacSignName ?? "",
        client.Age, client.Hobbies ?? "");

    // Разбиение на страницы
    var count = await marriageAgencyContext.CountAsync();
    marriageAgencyContext = marriageAgencyContext.Skip((page - 1) * pageSize).Take(pageSize);

    var clientsList = await marriageAgencyContext.ToListAsync();

    ViewData["Name"] = new SelectList(_context.Nationalities, "Name", "Name");
    ViewData["ZodiacSignName"] = new SelectList(_context.ZodiacSigns, "Name", "Name");

    // Формирование модели для передачи представлению
    ClientsViewModel clients = new()
    {
        Clients = clientsList,
        PageViewModel = new PageViewModel(count, page, pageSize),
        SortViewModel = new SortViewModel(sortOrder),
        FilterClientsViewModel = client,
    };

    return View(clients);
}

// GET: Clients/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var client = await _context.Clients
        .Include(c => c.Contact) // Загружаем связанные записи Contact
        .Include(c => c.Nationality) // Загружаем связанные записи Nationality

```

```

        .Include(c => c.PhysicalAttribute) // Загружаем связанные записи PhysicalAttribute
        .Include(c => c.ZodiacSign) // Загружаем связанные записи ZodiacSign
        .SingleOrDefaultAsync(m => m.ClientId == id);

    if (client == null)
    {
        return NotFound();
    }

    return View(client);
}

// GET: Clients/Create
[Authorize(Roles = "admin")]
public IActionResult Create()
{
    var nationality = _context.Nationalities;
    if (nationality != null) ViewData["NationalityId"] = new SelectList(nationality, "NationalityId", "Name");
    var zodiacSign = _context.ZodiacSigns;
    if (zodiacSign != null) ViewData["ZodiacSignId"] = new SelectList(zodiacSign, "ZodiacSignId", "Name");
    return View();
}

// POST: Clients/Create
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "admin")]
public async Task<IActionResult> Create(Client client)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }
    else
    {
        string uniqueFileName = UploadedFile(client);
        client.ClientPhoto = uniqueFileName;
        _context.Add(client);
        await _context.SaveChangesAsync();
    }

    return RedirectToAction(nameof(Index));
}

// GET: Clients/Edit/5
[Authorize(Roles = "admin")]
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var client = await _context.Clients.SingleOrDefaultAsync(m => m.ClientId == id);
    if (client == null)
    {
        return NotFound();
    }

    var zodiacSign = _context.ZodiacSigns;
    if (zodiacSign != null) ViewData["ZodiacSignId"] = new SelectList(zodiacSign, "ZodiacSignId", "Name", client.ZodiacSignId);
    var nationality = _context.Nationalities;
    if (nationality != null) ViewData["NationalityId"] = new SelectList(nationality, "NationalityId", "Name", client.NationalityId);

```

```

        return View(client);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    [Authorize(Roles = "admin")]
    public async Task<IActionResult> Edit(int id,
    [Bind("ClientId,FirstName,LastName,MiddleName,Gender,BirthDate,ZodiacSignId,NationalityId,Profession,FrontImage,ClientPhoto")] Client client)
    {
        if (id != client.ClientId)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                // Получаем существующего клиента из базы данных
                var existingClient = await _context.Clients.AsNoTracking().FirstOrDefaultAsync(c => c.ClientId == client.ClientId);

                if (existingClient == null)
                {
                    return NotFound();
                }

                // Если файл был загружен, обновляем фото
                if (client.FrontImage != null)
                {
                    client.ClientPhoto = UploadedFile(client); // Загружаем файл
                }
                else
                {
                    // Если файл не был загружен, сохраняем старое фото
                    client.ClientPhoto = existingClient.ClientPhoto;
                }

                // Обновляем информацию о клиенте
                _context.Update(client);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ClientExists(client.ClientId))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
    }

    ViewData["ZodiacSignId"] = new SelectList(_context.ZodiacSigns, "ZodiacSignId", "Name", client.ZodiacSignId);
    ViewData["NationalityId"] = new SelectList(_context.Nationalities, "NationalityId", "Name", client.NationalityId);
    return View(client);
}

// GET: Clients/Delete/5
[Authorize(Roles = "admin")]
public async Task<IActionResult> Delete(int? id)

```

```

{
    if (id == null)
    {
        return NotFound();
    }

    var client = await _context.Clients
        .Include(c => c.Contact) // Загружаем связанные записи Contact
        .Include(c => c.Nationality) // Загружаем связанные записи Nationality
        .Include(c => c.PhysicalAttribute) // Загружаем связанные записи PhysicalAttribute
        .Include(c => c.ZodiacSign) // Загружаем связанные записи ZodiacSign
        .SingleOrDefaultAsync(m => m.ClientId == id);
    if (client == null)
    {
        return NotFound();
    }

    return View(client);
}

// POST: Clients/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
[Authorize(Roles = "admin")]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    // Находим клиента вместе с связанными сущностями
    var client = await _context.Clients
        .Include(c => c.Contact) // Загружаем связанные записи Contact
        .Include(c => c.Nationality) // Загружаем связанные записи Nationality
        .Include(c => c.PhysicalAttribute) // Загружаем связанные записи PhysicalAttribute
        .Include(c => c.ZodiacSign) // Загружаем связанные записи ZodiacSign
        .SingleOrDefaultAsync(m => m.ClientId == id);

    if (client != null)
    {
        // Удаляем связанные сущности, если они существуют
        if (client.Contact != null)
        {
            _context.Contacts.Remove(client.Contact); // Удаляем запись Contact
        }
        if (client.PhysicalAttribute != null)
        {
            _context.PhysicalAttributes.Remove(client.PhysicalAttribute); // Удаляем запись PhysicalAttribute
        }
        if (client.Nationality != null)
        {
            _context.Nationalities.Remove(client.Nationality); // Удаляем запись Nationality
        }
        if (client.ZodiacSign != null)
        {
            _context.ZodiacSigns.Remove(client.ZodiacSign); // Удаляем запись ZodiacSign
        }

        // Удаляем запись клиента
        _context.Clients.Remove(client);

        // Сохраняем изменения в базе данных
        await _context.SaveChangesAsync();
    }

    // Перенаправляем на страницу со списком клиентов
    return RedirectToAction(nameof(Index));
}

```

```

private bool ClientExists(int id)

```

```

    {
        return _context.Clients.Any(e => e.ClientId == id);
    }

private string UploadedFile(Client client)
{
    string uniqueFileName = null;
    if (client.FrontImage != null)
    {
        string uploadsFolder = Path.Combine(_webHostEnvironment.WebRootPath, "images");
        uniqueFileName = Guid.NewGuid().ToString() + "_" + client.FrontImage.FileName;
        string filePath = Path.Combine(uploadsFolder, uniqueFileName);
        using (var fileStream = new FileStream(filePath, FileMode.Create))
        {
            client.FrontImage.CopyTo(fileStream);
        }
    }
    return uniqueFileName;
}

private static IQueryable<Client> Sort_Search(IQueryable<Client> clients, SortState sortOrder, string ClientName,
string Gender, string NationalityName, string ZodiacSignName, int? Age, string Hobbies)
{
    clients = clients.Where(o => o.FirstName.Contains(ClientName ?? ""));

    if (!string.IsNullOrEmpty(Gender))
    {
        clients = clients.Where(o => o.Gender.Contains(Gender ?? ""));
    }

    if (!string.IsNullOrEmpty(NationalityName))
    {
        clients = clients.Where(o => o.Nationality.Name.Contains(NationalityName ?? ""));
    }

    if (!string.IsNullOrEmpty(ZodiacSignName))
    {
        clients = clients.Where(o => o.ZodiacSign.Name.Contains(ZodiacSignName ?? ""));
    }

    if (Age.HasValue)
    {
        clients = clients.Where(o => o.PhysicalAttribute.Age == Age.Value);
    }

    if (!string.IsNullOrEmpty(Hobbies))
    {
        clients = clients.Where(o => o.PhysicalAttribute.Hobbies.Contains(Hobbies ?? ""));
    }

    switch (sortOrder)
    {
        {
            case SortState.ClientNameAsc:
                clients = clients.OrderBy(s => s.FirstName);
                break;
            case SortState.ClientNameDesc:
                clients = clients.OrderByDescending(s => s.FirstName);
                break;

            case SortState.GenderAsc:
                clients = clients.OrderBy(s => s.Gender);
                break;
            case SortState.GenderDesc:
                clients = clients.OrderByDescending(s => s.Gender);
                break;

            case SortState.NationalityAsc:

```

```

        clients = clients.OrderBy(s => s.Nationality.Name);
        break;
    case SortState.NationalityDesc:
        clients = clients.OrderByDescending(s => s.Nationality.Name);
        break;

    case SortState.ZodiacSignNameAsc:
        clients = clients.OrderBy(s => s.ZodiacSign.Name);
        break;
    case SortState.ZodiacSignNameDesc:
        clients = clients.OrderByDescending(s => s.ZodiacSign.Name);
        break;

    case SortState.AgeAsc:
        clients = clients.OrderBy(s => s.PhysicalAttribute.Age);
        break;
    case SortState.AgeDesc:
        clients = clients.OrderByDescending(s => s.PhysicalAttribute.Age);
        break;

    case SortState.HobbiesAsc:
        clients = clients.OrderBy(s => s.PhysicalAttribute.Hobbies);
        break;
    case SortState.HobbiesDesc:
        clients = clients.OrderByDescending(s => s.PhysicalAttribute.Hobbies);
        break;

    default:
        break;
}

return clients;
}
}

```

EmployeesController.cs

```

using MarriageAgency.DataLayer.Data;
using MarriageAgency.DataLayer.Models;
using MarriageAgency.Infrastructure;
using MarriageAgency.Infrastructure.Filters;
using MarriageAgency.ViewModels;
using MarriageAgency.ViewModels.EmployeesViewModel;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace MarriageAgency.Controllers
{
    public class EmployeesController : Controller
    {
        private readonly MarriageAgencyContext _context;
        private readonly int pageSize = 10; // количество элементов на странице

        public EmployeesController(MarriageAgencyContext context, IConfiguration appConfig = null)
        {
            _context = context;
            if (appConfig != null)
            {
                pageSize = int.Parse(appConfig["Parameters:PageSize"]);
            }
        }

        // GET: Employee
        [SetToSession("Employee")]
        [Authorize]
        [ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
    }
}

```



```

public async Task<IActionResult> Index(FilterEmployeesViewModel employee, SortState sortOrder = Sort-
State.No, int page = 1, DateOnly? birthDate = null)
{
    if (employee.EmployeeName == null && employee.EmployeeMiddleName == null
        && employee.EmployeeLastName == null
        && employee.BirthDate == null)
    {
        if (HttpContext != null)
        {
            var sessionEmployee = Infrastructure.SessionExtensions.Get(HttpContext.Session, "Employee");
            if (sessionEmployee != null)
            {
                employee = Transformations.DictionaryToObject<FilterEmployeesViewModel>(sessionEmployee);
            }
        }
    }

    IQueryable<Employee> marriageAgencyContext = _context.Employees;

    marriageAgencyContext = Sort_Search(marriageAgencyContext, sortOrder, employee.EmployeeName ?? "",
        employee.EmployeeLastName ?? "",
        employee.EmployeeMiddleName ?? "",
        employee.BirthDate);

    var count = await marriageAgencyContext.CountAsync();
    marriageAgencyContext = marriageAgencyContext.Skip((page - 1) * pageSize).Take(pageSize);

    EmployeesViewModel employees = new()
    {
        Employees = await marriageAgencyContext.ToListAsync(),
        PageViewModel = new PageViewModel(count, page, pageSize),
        SortViewModel = new SortViewModel(sortOrder),
        FilterEmployeesViewModel = employee,
    };

    return View(employees);
}

// GET: Employees/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var employee = await _context.Employees
        .SingleOrDefaultAsync(m => m.EmployeeId == id);
    if (employee == null)
    {
        return NotFound();
    }

    return View(employee);
}

// GET: Employees/Create
[Authorize(Roles = "admin")]
public IActionResult Create()
{
    return View();
}

// POST: Employees/Create
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "admin")]

```

```

        public async Task<IActionResult> Create([Bind("EmployeeId, FirstName, MiddleName, LastName, Position,
        BirthDate")] Employee employee)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }
            else
            {
                _context.Add(employee);
                await _context.SaveChangesAsync();
            }

            return RedirectToAction(nameof(Index));
        }

// GET: Employees/Edit/5
[Authorize(Roles = "admin")]
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var employee = await _context.Employees.SingleOrDefaultAsync(m => m.EmployeeId == id);
    if (employee == null)
    {
        return NotFound();
    }
    return View(employee);
}

// POST: Employees/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "admin")]
public async Task<IActionResult> Edit(int id, [Bind("EmployeeId, FirstName, MiddleName, LastName, Position,
        BirthDate")] Employee employee)
{
    if (id != employee.EmployeeId)
    {
        return NotFound();
    }

    if (!ModelState.IsValid)
    {
        return View(employee);
    }
    else
    {
        try
        {
            _context.Update(employee);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!EmployeeExists(employee.EmployeeId))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
    }
}

```

```

        }
        return RedirectToAction(nameof(Index));
    }
}

// GET: Employees/Delete/5
[Authorize(Roles = "admin")]
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var employee = await _context.Employees
        .SingleOrDefaultAsync(m => m.EmployeeId == id);
    if (employee == null)
    {
        return NotFound();
    }

    return View(employee);
}

// POST: Employees/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
[Authorize(Roles = "admin")]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var employee = await _context.Employees.SingleOrDefaultAsync(m => m.EmployeeId == id);
    _context.Employees.Remove(employee);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool EmployeeExists(int id)
{
    return _context.Employees.Any(e => e.EmployeeId == id);
}

private static IQueryable<Employee> Sort_Search(IQueryable<Employee> employees, SortState sortOrder, string
employeeName, string employeeLastName, string employeeMiddleName, DateOnly? birthDate)
{
    if (!string.IsNullOrEmpty(employeeName))
    {
        employees = employees.Where(e => e.FirstName.Contains(employeeName));
    }

    if (!string.IsNullOrEmpty(employeeLastName))
    {
        employees = employees.Where(e => e.LastName.Contains(employeeLastName));
    }

    if (!string.IsNullOrEmpty(employeeMiddleName))
    {
        employees = employees.Where(e => e.MiddleName.Contains(employeeMiddleName));
    }

    if (birthDate.HasValue)
    {
        employees = employees.Where(e => e.BirthDate == birthDate.Value);
    }

    switch (sortOrder)
    {
        case SortState.EmployeeNameAsc:

```

```

        employees = employees.OrderBy(s => s.FirstName);
        break;
    case SortState.EmployeeNameDesc:
        employees = employees.OrderByDescending(s => s.FirstName);
        break;
    case SortState.EmployeeLastNameAsc:
        employees = employees.OrderBy(s => s.LastName);
        break;
    case SortState.EmployeeLastNameDesc:
        employees = employees.OrderByDescending(s => s.LastName);
        break;
    case SortState.EmployeeMiddleNameAsc:
        employees = employees.OrderBy(s => s.MiddleName);
        break;
    case SortState.EmployeeMiddleNameDesc:
        employees = employees.OrderByDescending(s => s.MiddleName);
        break;
    case SortState.BirthDateAsc:
        employees = employees.OrderBy(s => s.BirthDate);
        break;
    case SortState.BirthDateDesc:
        employees = employees.OrderByDescending(s => s.BirthDate);
        break;
    default:
        break;
    }

    return employees;
}
}
}

```

ServicesController.cs

```

using MarriageAgency.DataLayer.Data;
using MarriageAgency.DataLayer.Models;
using MarriageAgency.Infrastructure;
using MarriageAgency.Infrastructure.Filters;
using MarriageAgency.ViewModels;
using MarriageAgency.ViewModels.ServicesViewModel;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;

namespace MarriageAgency.Controllers
{
    public class ServicesController : Controller
    {
        private readonly MarriageAgencyContext _context;
        private readonly int pageSize = 10; // количество элементов на странице

        public ServicesController(MarriageAgencyContext context, IConfiguration appConfig = null)
        {
            _context = context;
            if (appConfig != null)
            {
                pageSize = int.Parse(appConfig["Parameters:PageSize"]);
            }
        }

        // GET: Services
        [SetToSession("Service")]
        [Authorize]
        [ResponseCache(Location = ResponseCacheLocation.Any, Duration = 264)]
        public async Task<ActionResult> Index(FilterServicesViewModel service, SortState sortOrder = SortState.No, int
        page = 1)
    }
}

```

```

{
    if (service.ClientName == null && service.EmployeeName == null
        && service.AdditionalServiceName == null && service.MinCost == null && service.MaxCost == null)
    {
        // Считывание данных из сессии
        if (HttpContext != null)
        {
            // Считывание данных из сессии
            var sessionService = Infrastructure.SessionExtensions.Get(HttpContext.Session, "Service");
            if (sessionService != null)
                service = Transformations.DictionaryToObject<FilterServicesViewModel>(sessionService);
        }
    }

    // Сортировка и фильтрация данных
    IQueryable<Service> marriageAgencyContext = _context.Services
        .Include(o => o.Employee)
        .Include(o => o.Client)
        .Include(o => o.AdditionalService);

    marriageAgencyContext = Sort_Search(marriageAgencyContext, sortOrder, service.ClientName ?? "",
        service.EmployeeName ?? "", service.AdditionalServiceName ?? "", service.MinCost, service.MaxCost);

    // Разбиение на страницы
    var count = await marriageAgencyContext.CountAsync();
    marriageAgencyContext = marriageAgencyContext.Skip((page - 1) * pageSize).Take(pageSize);

    // Получение данных асинхронно
    var servicesList = await marriageAgencyContext.ToListAsync();
    ViewData["Name"] = new SelectList(_context.AdditionalServices, "Name", "Name");
    // Формирование модели для передачи представлению
    ServicesViewModel services = new()
    {
        Services = servicesList,
        SortViewModel = new SortViewModel(sortOrder),
        FilterServicesViewModel = service,
        PageViewModel = new PageViewModel(count, page, pageSize)
    };

    return View(services);
}

// GET: Services/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var service = await _context.Services
        .Include(o => o.Employee)
        .Include(o => o.Client)
        .Include(o => o.AdditionalService)
        .SingleOrDefaultAsync(m => m.ServiceId == id);
    if (service == null)
    {
        return NotFound();
    }

    return View(service);
}

// GET: Services/Create
[Authorize(Roles = "admin")]
public IActionResult Create()
{

```

```

        ViewData["ClientId"] = new SelectList(_context.Clients, "ClientId", "FirstName");
        ViewData["EmployeeId"] = new SelectList(_context.Employees, "EmployeeId", "FirstName");
        ViewData["AdditionalServiceId"] = new SelectList(_context.AdditionalServices, "AdditionalServiceId",
"Name");

        return View();
    }

    // POST: Services/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    [Authorize(Roles = "admin")]
    public async Task<IActionResult> Create([Bind("ServiceId,AdditionalServiceId, EmployeeId, ClientId, Date,
Cost")] Service service)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }
        else
        {
            _context.Add(service);
            await _context.SaveChangesAsync();
        }

        return RedirectToAction(nameof(Index));
    }

    // GET: Services/Edit/5
    [Authorize(Roles = "admin")]
    public async Task<IActionResult> Edit(int id)
    {
        if (id == null)
        {
            return NotFound();
        }
        var service = await _context.Services.SingleOrDefaultAsync(m => m.ServiceId == id);

        if (service == null)
        {
            return NotFound();
        }

        ViewData["ClientId"] = new SelectList(_context.Clients, "ClientId", "FirstName", service.ClientId);
        ViewData["EmployeeId"] = new SelectList(_context.Employees, "EmployeeId", "FirstName", ser-
vice.EmployeeId);
        ViewData["AdditionalServiceId"] = new SelectList(_context.AdditionalServices, "AdditionalServiceId",
"Name", service.AdditionalServiceId);
        return View(service);
    }

    // POST: Services/Edit/5
    [HttpPost]
    [ValidateAntiForgeryToken]
    [Authorize(Roles = "admin")]
    public async Task<IActionResult> Edit(int id,
[Bind("ServiceId,AdditionalServiceId,ClientId,EmployeeId,Date,Cost")] Service service)
    {
        if (id != service.ServiceId)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {

```

```

        try
        {
            _context.Update(service);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ServiceExists(service.ServiceId))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }

    ViewData["ClientId"] = new SelectList(_context.Clients, "ClientId", "FirstName", service.ClientId);
    ViewData["EmployeeId"] = new SelectList(_context.Employees, "EmployeeId", "FirstName", service.EmployeeId);
    ViewData["AdditionalServiceId"] = new SelectList(_context.AdditionalServices, "AdditionalServiceId", "Name", service.AdditionalServiceId);
    return View(service);
}

// GET: Services/Delete/5
[Authorize(Roles = "admin")]
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var service = await _context.Services
        .Include(o => o.Employee)
        .Include(o => o.Client)
        .Include(o => o.AdditionalService)
        .SingleOrDefaultAsync(m => m.ServiceId == id);
    if (service == null)
    {
        return NotFound();
    }

    return View(service);
}

// POST: Services/Delete/5
[HttpPost, ActionName("Delete")]
[Authorize(Roles = "admin")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var service = await _context.Services.SingleOrDefaultAsync(m => m.ServiceId == id);
    _context.Services.Remove(service);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool ServiceExists(int id)
{
    return _context.Services.Any(e => e.ServiceId == id);
}

private static IQueryable<Service> Sort_Search(

```

```

IQueryable<Service> services,
SortState sortOrder,
string searchClientName,
string searchEmployeeName,
string additionalServiceName,
decimal? minCost,
decimal? maxCost)
{
    switch (sortOrder)
    {
        case SortState.ClientNameAsc:
            services = services.OrderBy(s => s.Client.FirstName);
            break;
        case SortState.ClientNameDesc:
            services = services.OrderByDescending(s => s.Client.FirstName);
            break;
        case SortState.EmployeeNameAsc:
            services = services.OrderBy(s => s.Employee.FirstName);
            break;
        case SortState.EmployeeNameDesc:
            services = services.OrderByDescending(s => s.Employee.FirstName);
            break;
        case SortState.AdditionalNameAsc:
            services = services.OrderBy(s => s.AdditionalService.Name);
            break;
        case SortState.AdditionalNameDesc:
            services = services.OrderByDescending(s => s.AdditionalService.Name);
            break;
        case SortState.CostAsc:
            services = services.OrderBy(s => s.Cost);
            break;
        case SortState.CostDesc:
            services = services.OrderByDescending(s => s.Cost);
            break;
    }

    services = services.Include(o => o.Client)
        .Include(o => o.Employee)
        .Include(o => o.AdditionalService)
        .Where(o => (string.IsNullOrEmpty(searchClientName) || o.Client.FirstName.Contains(searchClientName))
            && (string.IsNullOrEmpty(searchEmployeeName) ||
o.Employee.FirstName.Contains(searchEmployeeName))
            && (!minCost.HasValue || o.Cost >= minCost.Value)
            && (!maxCost.HasValue || o.Cost <= maxCost.Value)
            && (string.IsNullOrEmpty(additionalServiceName) ||
o.AdditionalService.Name.Contains(additionalServiceName)));

    return services;
}
}
}

```


ПРИЛОЖЕНИЕ В

(обязательное)

Текст программы

DbUserInitializer.cs

```
using MarriageAgency.Models;
using Microsoft.AspNetCore.Identity;

namespace MarriageAgency.Data
{
    //Инициализация базы данных первой учетной записью и двумя ролями admin и user
    public static class DbUserInitializer
    {
        public static async Task Initialize(HttpContext context)
        {
            UserManager<ApplicationUser> userManager = context.RequestServices.GetRequiredService<UserManager<ApplicationUser>>();
            RoleManager<IdentityRole> roleManager = context.RequestServices.GetRequiredService<RoleManager<IdentityRole>>();
            string adminEmail = "admin@gmail.com";
            string adminName = "admin@gmail.com";

            string password = "_Aa123456";
            if (await roleManager.FindByNameAsync("admin") == null)
            {
                await roleManager.CreateAsync(new IdentityRole("admin"));
            }
            if (await roleManager.FindByNameAsync("user") == null)
            {
                await roleManager.CreateAsync(new IdentityRole("user"));
            }
            if (await userManager.FindByNameAsync(adminEmail) == null)
            {
                ApplicationUser admin = new()
                {
                    Email = adminEmail,
                    UserName = adminName,
                    RegistrationDate = DateTime.Now
                };
                IdentityResult result = await userManager.CreateAsync(admin, password);
                if (result.Succeeded)
                {
                    await userManager.AddToRoleAsync(admin, "admin");
                }
            }
        }
    }
}
```

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.DataLayer.Data;
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")))
    {
        DbUserInitializer.Initialize(context).Wait();
        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 numberOfRows = 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 numberOfRows = 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 MarriageAgency/MarriageAgency/MarriageAgency.csproj

- name: Build

run: dotnet build MarriageAgency/MarriageAgency/MarriageAgency.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=true",
    "IdentityConnection": "Server=HOME-PC;Database=Users;Trusted_Connection=True;TrustServerCertificate=True;MultipleActiveResultSets=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "Parameters": {
    "PageSize": "10"
  }
}
```

ПРИЛОЖЕНИЕ Ё

(обязательное)

Текст программы

Clients

Index.cshtml

```
@using MarriageAgency.ViewModels.ClientsViewModel;
```

```
@model ClientsViewModel;
```

```
@{  
    ViewData["Title"] = "Клиенты";  
    Layout = "~/Views/Shared/_Layout.cshtml";  
    string ClientName = Model.FilterClientsViewModel.ClientName;  
    string Gender = Model.FilterClientsViewModel.Gender;  
    string NationalityName = Model.FilterClientsViewModel.NationalityName;  
    string ZodiacSignName = Model.FilterClientsViewModel.ZodiacSignName;  
    int? Age = Model.FilterClientsViewModel.Age;  
    string Hobbies = Model.FilterClientsViewModel.Hobbies;  
}
```

```
<h2 class="mb-4">Клиенты</h2>
```

```
<div class="mb-3">
```

```
    <a asp-action="Create" class="btn btn-primary">Создать</a>
```

```
</div>
```

```
@using (Html.BeginForm("Index", "Clients", FormMethod.Get, new { @class = "form-inline mb-3" }))  
{
```

```
    <fieldset class="form-group">
```

```
        <label for="ClientName" class="mr-2">Имя клиента:</label>
```

```
        @Html.TextBox("ClientName", ClientName, new { @class = "form-control mr-2", @id = "ClientName" })
```

```
    </fieldset>
```

```
    <fieldset class="form-group">
```

```
        <label for="Gender" class="mr-2">Пол:</label>
```

```
        @Html.DropDownList("Gender", new List<SelectListItem>
```

```
        {
```

```
            new SelectListItem { Text = "Не выбрано", Value = "" },
```

```
            new SelectListItem { Text = "Мужской", Value = "Мужской" },
```

```
            new SelectListItem { Text = "Женский", Value = "Женский" }
```

```
        }, new { @class = "form-control mr-2", @id = "Gender" })
```

```
    </fieldset>
```

```
    <fieldset class="form-group">
```

```
        <label for="NationalityName" class="mr-2">Национальность:</label>
```

```
        @Html.DropDownList("NationalityName", (SelectList)ViewData["Name"], "Выберите национальность", new {  
@class = "form-control mr-2", @id = "NationalityId" })
```

```
    </fieldset>
```

```
    <fieldset class="form-group">
```

```
        <label for="ZodiacSignName" class="mr-2">Знак зодиака:</label>
```

```
        @Html.DropDownList("ZodiacSignName", (SelectList)ViewData["ZodiacSignName"], "Выберите знак  
зодиака", new { @class = "form-control mr-2", @id = "ZodiacSignId" })
```

```
    </fieldset>
```

```
    <fieldset class="form-group">
```

```
        <label for="Age" class="mr-2">Возраст:</label>
```

```
        @Html.TextBox("Age", Age?.ToString(), new { @class = "form-control mr-2", @id = "Age" })
```

```
    </fieldset>
```

```
    <fieldset class="form-group">
```

```
        <label for="Hobbies" class="mr-2">Хобби:</label>
```

```
        @Html.TextBox("Hobbies", Hobbies, new { @class = "form-control mr-2", @id = "Hobbies" })
```

```

</fieldset>

<input type="submit" value="Найти" class="btn btn-outline-success mt-1" />
}

<table class="table table-striped table-bordered">
  <thead class="thead-light">
    <tr>
      <th style="width: 150px;">@Html.DisplayNameFor(model => model.FilterClientsViewModel.ClientPhoto)</th>
      <th style="width: 200px;">
        <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.ClientNameSort" asp-route-ClientName="@ClientName" asp-route-Gender="@Gender"
          asp-route-NationalityName="@NationalityName" asp-route-ZodiacSignName="@ZodiacSignName" asp-route-Age="@Age" asp-route-Hobbies="@Hobbies"
          style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
            @Html.DisplayNameFor(model => model.FilterClientsViewModel.ClientName)
            @if (Model.SortViewModel.ClientNameSort == SortState.ClientNameAsc)
            {
              <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
            }
            else if (Model.SortViewModel.ClientNameSort == SortState.ClientNameDesc)
            {
              <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
            }
          </a>
      </th>
      <th style="width: 150px;">@Html.DisplayNameFor(model => model.FilterClientsViewModel.LastName)</th>
      <th style="width: 150px;">@Html.DisplayNameFor(model => model.FilterClientsViewModel.MiddleName)</th>
      <th style="width: 100px;">
        @Html.DisplayNameFor(model => model.FilterClientsViewModel.Gender)
        <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.GenderSort" asp-route-ClientName="@ClientName" asp-route-Gender="@Gender"
          asp-route-NationalityName="@NationalityName" asp-route-ZodiacSignName="@ZodiacSignName" asp-route-Age="@Age" asp-route-Hobbies="@Hobbies"
          style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
            @if (Model.SortViewModel.GenderSort == SortState.GenderAsc)
            {
              <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
            }
            else if (Model.SortViewModel.GenderSort == SortState.GenderDesc)
            {
              <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
            }
          </a>
      </th>
      <th style="width: 150px;">@Html.DisplayNameFor(model => model.FilterClientsViewModel.BirthDate)</th>
      <th style="width: 150px;">@Html.DisplayNameFor(model => model.FilterClientsViewModel.Profession)</th>
      <th style="width: 150px;">@Html.DisplayNameFor(model => model.FilterClientsViewModel.Contact.Phone)</th>
      <th style="width: 150px;">
        @Html.DisplayNameFor(model => model.FilterClientsViewModel.Age)
        <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.AgeSort" asp-route-ClientName="@ClientName" asp-route-Gender="@Gender"
          asp-route-NationalityName="@NationalityName" asp-route-ZodiacSignName="@ZodiacSignName" asp-route-Age="@Age" asp-route-Hobbies="@Hobbies"
          style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
            @if (Model.SortViewModel.AgeSort == SortState.AgeAsc)
            {
              <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
            }
            else if (Model.SortViewModel.AgeSort == SortState.AgeDesc)
            {
              <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
            }
          </a>
      </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.ClientPhoto)</td>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.ClientName)</td>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.LastName)</td>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.MiddleName)</td>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.Gender)</td>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.BirthDate)</td>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.Profession)</td>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.Contact.Phone)</td>
      <td>@Html.DisplayFor(model => model.FilterClientsViewModel.Age)</td>
    </tr>
  </tbody>
</table>

```



```

</th>
<th style="width: 150px;">
    @Html.DisplayNameFor(model => model.FilterClientsViewModel.Hobbies)
    <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.HobbiesSort" asp-route-
ClientName="@ClientName" asp-route-Gender="@Gender"
    asp-route-NationalityName="@NationalityName" asp-route-ZodiacSignName="@ZodiacSignName" asp-
route-Age="@Age" asp-route-Hobbies="@Hobbies"
    style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
        @if (Model.SortViewModel.HobbiesSort == SortState.HobbiesAsc)
        {
            <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
        }
        else if (Model.SortViewModel.HobbiesSort == SortState.HobbiesDesc)
        {
            <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
        }
    </a>
</th>
<th style="width: 150px;">
    @Html.DisplayNameFor(model => model.FilterClientsViewModel.NationalityName)
    <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.NationalitySort" asp-route-
ClientName="@ClientName" asp-route-Gender="@Gender"
    asp-route-NationalityName="@NationalityName" asp-route-ZodiacSignName="@ZodiacSignName" asp-
route-Age="@Age" asp-route-Hobbies="@Hobbies"
    style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
        @if (Model.SortViewModel.NationalitySort == SortState.NationalityAsc)
        {
            <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
        }
        else if (Model.SortViewModel.NationalitySort == SortState.NationalityDesc)
        {
            <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
        }
    </a>
</th>
<th style="width: 150px;">
    @Html.DisplayNameFor(model => model.FilterClientsViewModel.ZodiacSignName)
    <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.ZodiacSignNameSort" asp-route-
ClientName="@ClientName" asp-route-Gender="@Gender"
    asp-route-NationalityName="@NationalityName" asp-route-ZodiacSignName="@ZodiacSignName" asp-
route-Age="@Age" asp-route-Hobbies="@Hobbies"
    style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
        @if (Model.SortViewModel.ZodiacSignNameSort == SortState.ZodiacSignNameAsc)
        {
            <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
        }
        else if (Model.SortViewModel.ZodiacSignNameSort == SortState.ZodiacSignNameDesc)
        {
            <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
        }
    </a>
</th>
<th style="width: 150px;" class="text-center">Действие</th>
</tr>
</thead>
<tbody>
    @foreach (var item in Model.Clients)
    {
        <tr>
            <td>
                
            </td>
            <td>@Html.DisplayFor(modelItem => item.FirstName)</td>
            <td>@Html.DisplayFor(modelItem => item.LastName)</td>
            <td>@Html.DisplayFor(modelItem => item.MiddleName)</td>
            <td>@Html.DisplayFor(modelItem => item.Gender)</td>
            <td>@Html.DisplayFor(modelItem => item.BirthDate)</td>
        </tr>
    }
</tbody>

```

```

        <td>@Html.DisplayFor(modelItem => item.Profession)</td>
        <td>@Html.DisplayFor(modelItem => item.Contact.Phone)</td>
        <td>@Html.DisplayFor(modelItem => item.PhysicalAttribute.Age)</td>
        <td>@Html.DisplayFor(modelItem => item.PhysicalAttribute.Hobbies)</td>
        <td>@Html.DisplayFor(modelItem => item.Nationality.Name)</td>
        <td>@Html.DisplayFor(modelItem => item.ZodiacSign.Name)</td>
        <td>
            <div class="btn-group d-flex" role="group" aria-label="Basic example">
                <a asp-action="Edit" asp-route-id="@item.ClientId" class="btn btn-sm btn-primary me-1">Редактировать</a>
                <a asp-action="Details" asp-route-id="@item.ClientId" class="btn btn-sm btn-secondary me-1">Подробнее</a>
                <a asp-action="Delete" asp-route-id="@item.ClientId" class="btn btn-sm btn-danger">Удалить</a>
            </div>
        </td>
    </tr>
</tbody>
</table>

<page-link page-model="Model.PageViewModel" page-action="Index"
    page-url-ClientName="@ClientName"
    page-url-sortOrder="@((Model.SortViewModel.CurrentState))"
    page-url-Gender="@Gender"
    page-url-NationalityName="@NationalityName"
    page-url-ZodiacSignName="@ZodiacSignName"
    page-url-Age="@Age"
    page-url-Hobbies="@Hobbies"></page-link>

```

Edit.cshtml

@model Client

```

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

```

<h2 class="text-center mb-4">Редактировать</h2>

```

<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card shadow-sm">
                <div class="card-body">
                    <h4 class="card-title text-center mb-4">КЛИЕНТ</h4>
                    <form enctype="multipart/form-data" asp-action="Edit">
                        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                        <input type="hidden" asp-for="ClientId">
                        <div class="form-group">
                            <label asp-for="FirstName" class="control-label"></label>
                            <input asp-for="FirstName" class="form-control" />
                            <span asp-validation-for="FirstName" class="text-danger"></span>
                        </div>
                        <div class="form-group">
                            <label asp-for="MiddleName" class="control-label"></label>
                            <input asp-for="MiddleName" class="form-control" />
                            <span asp-validation-for="MiddleName" class="text-danger"></span>
                        </div>
                        <div class="form-group">
                            <label asp-for="LastName" class="control-label"></label>
                            <input asp-for="LastName" class="form-control" />
                            <span asp-validation-for="LastName" class="text-danger"></span>
                        </div>
                        <div class="form-group">
                            <label asp-for="Gender" class="control-label"></label>
                            <select asp-for="Gender" class="form-control">

```

```

        <option value="Мужской">Мужской</option>
        <option value="Женский">Женский</option>
    </select>
    <span asp-validation-for="Gender" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="Profession" class="control-label"></label>
    <input asp-for="Profession" class="form-control" />
    <span asp-validation-for="Profession" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="BirthDate" class="control-label"></label>
    <input asp-for="BirthDate" class="form-control" type="date" />
    <span asp-validation-for="BirthDate" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="NationalityId" class="control-label"></label>
    <select asp-for="NationalityId" class="form-control" asp-items="ViewBag.NationalityId"></select>
    <span asp-validation-for="NationalityId" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="ZodiacSignId" class="control-label"></label>
    <select asp-for="ZodiacSignId" class="form-control" asp-items="ViewBag.ZodiacSignId"></select>
    <span asp-validation-for="ZodiacSignId" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="FrontImage" class="control-label"></label>
    <div class="custom-file">
        <input asp-for="FrontImage" type="file" class="custom-file-input" id="FLFrontImage" on-
change="document.getElementById('FrontImagePrv').src = window.URL.createObjectURL(this.files[0])" />
        <label class="custom-file-label" for="FLFrontImage">Выберите файл</label>
    </div>
    <span asp-validation-for="FrontImage" class="text-danger"></span>
    <!-- Превью изображения -->
    <td>
        
    </td>
</div>

<div class="form-group text-center mt-4">
    <input type="submit" value="Сохранить" class="btn btn-primary" />
</div>
</form>
</div>
</div>
</div>
</div>
</div>
</div>

<div class="text-center mt-3">
    <a asp-action="Index" class="btn btn-link">Вернуться к списку</a>
</div>

@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
    <script type="text/javascript">
        $( ".custom-file-input" ).on( "change", function () {
            var fileName = $(this).val().split("\\").pop();
            console.log("Файл выбран: " + fileName); // Добавляем лог для отслеживания
            $(this).siblings( ".custom-file-label" ).addClass( "selected" ).html( fileName );
        });
    </script>

```

```
}
```

Details.cshtml

@model Client

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

```
<h2 class="text-center mb-4">Детально</h2>
```

```
<div class="container">
    <div class="card shadow-sm">
        <div class="card-body">
            <h4 class="card-title text-center mb-4">Клиент</h4>
            <dl class="row">
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.ClientPhoto)
                </dt>
                <dd class="col-sm-8">
                    
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.FirstName)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.FirstName)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.MiddleName)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.MiddleName)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.LastName)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.LastName)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Profession)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Profession)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.BirthDate)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.BirthDate)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Nationality.Name)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Nationality.Name)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.ZodiacSign.Name)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.ZodiacSign.Name)
                </dd>
            </dl>
        </div>
    </div>
</div>
```

```

        </div>
    </div>
</div>

<div class="text-center mt-3">
    <a asp-action="Edit" asp-route-id="@Model.ClientId" class="btn btn-primary mx-2">Редактировать</a>
    <a asp-action="Index" class="btn btn-secondary mx-2">Вернуться к списку</a>
</div>

```

Delete.cshtml

@model Client

```

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

```

<h2 class="text-center mb-4">Удаление</h2>

```

<div class="container">
    <div class="card shadow-sm">
        <div class="card-body">
            <h4 class="card-title text-center mb-4">Вы уверены, что хотите удалить этого клиента?</h4>

            <dl class="row">
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.ClientPhoto)
                </dt>
                <dd class="col-sm-8">
                    
                </dd>
                <dt class="col-sm-3">
                    @Html.DisplayNameFor(model => model.FirstName)
                </dt>
                <dd class="col-sm-9">
                    @Html.DisplayFor(model => model.FirstName)
                </dd>

                <dt class="col-sm-3">
                    @Html.DisplayNameFor(model => model.MiddleName)
                </dt>
                <dd class="col-sm-9">
                    @Html.DisplayFor(model => model.MiddleName)
                </dd>

                <dt class="col-sm-3">
                    @Html.DisplayNameFor(model => model.LastName)
                </dt>
                <dd class="col-sm-9">
                    @Html.DisplayFor(model => model.LastName)
                </dd>

                <dt class="col-sm-3">
                    @Html.DisplayNameFor(model => model.Profession)
                </dt>
                <dd class="col-sm-9">
                    @Html.DisplayFor(model => model.Profession)
                </dd>

                <dt class="col-sm-3">
                    @Html.DisplayNameFor(model => model.BirthDate)
                </dt>
                <dd class="col-sm-9">
                    @Html.DisplayFor(model => model.BirthDate)
                </dd>
            </dl>

```

```

<dt class="col-sm-3">
    @Html.DisplayNameFor(model => model.Nationality.Name)
</dt>
<dd class="col-sm-9">
    @Html.DisplayFor(model => model.Nationality.Name)
</dd>

<dt class="col-sm-3">
    @Html.DisplayNameFor(model => model.ZodiacSign.Name)
</dt>
<dd class="col-sm-9">
    @Html.DisplayFor(model => model.ZodiacSign.Name)
</dd>
</dl>

<form asp-action="Delete" class="text-center mt-4">
    <input type="hidden" asp-for="ClientId" />
    <button type="submit" class="btn btn-danger">Удалить</button>
    <a asp-action="Index" class="btn btn-secondary ml-2">Вернуться к списку</a>
</form>
</div>
</div>
</div>

```

Create.cshtml

@model Client

```

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

```

<h2 class="text-center mb-4">Создать</h2>

```

<div class="container">
    <div class="card shadow-sm">
        <div class="card-body">
            <h4 class="card-title text-center mb-4">Клиент</h4>
            <form enctype="multipart/form-data" asp-action="Create">
                <div asp-validation-summary="ModelOnly" class="text-danger"></div>

                <div class="form-group">
                    <label asp-for="FirstName" class="control-label"></label>
                    <input asp-for="FirstName" class="form-control" data-validate="no-numbers-required" required/>
                    <span asp-validation-for="FirstName" class="text-danger"></span>
                </div>

                <div class="form-group">
                    <label asp-for="MiddleName" class="control-label"></label>
                    <input asp-for="MiddleName" class="form-control" data-validate="no-numbers-required" required />
                    <span asp-validation-for="MiddleName" class="text-danger"></span>
                </div>

                <div class="form-group">
                    <label asp-for="LastName" class="control-label"></label>
                    <input asp-for="LastName" class="form-control" data-validate="no-numbers-required" required/>
                    <span asp-validation-for="LastName" class="text-danger"></span>
                </div>

                <div class="form-group">
                    <label asp-for="Gender" class="control-label"></label>
                    <select asp-for="Gender" class="form-control">
                        <option value="Мужской">Мужской</option>
                        <option value="Женский">Женский</option>
                    </select>
                    <span asp-validation-for="Gender" class="text-danger"></span>
                </div>

                <div class="form-group">
                    <label asp-for="Profession" class="control-label"></label>

```

```

        <input asp-for="Profession" class="form-control" data-validate="no-numbers-required" required/>
        <span asp-validation-for="Profession" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="BirthDate" class="control-label"></label>
        <input asp-for="BirthDate" class="form-control" type="date" data-validate="no-numbers-required" re-
quired/>
        <span asp-validation-for="BirthDate" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="NationalityId" class="control-label"></label>
        <select asp-for="NationalityId" class="form-control" asp-items="ViewBag.NationalityId"></select>
        <span asp-validation-for="NationalityId" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="ZodiacSignId" class="control-label"></label>
        <select asp-for="ZodiacSignId" class="form-control" asp-items="ViewBag.ZodiacSignId"></select>
        <span asp-validation-for="ZodiacSignId" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label for="FrontImage" class="control-label">Выберите файл</label>
        <div class="custom-file">
            <input type="file" class="custom-file-input" id="FrontImage" on-
change="document.getElementById('FrontImagePrv').src = window.URL.createObjectURL(this.files[0])" />
            <label class="custom-file-label" for="FrontImage">Выберите файл</label>
        </div>
        <span asp-validation-for="FrontImage" class="text-danger"></span>

        <img id="FrontImagePrv" src="" alt="Изображение" width="150" height="150" style="border:1px solid
#ddd; margin-top:20px; margin-left:120px;" />
    </div>

    <div class="form-group text-center mt-4">
        <input type="submit" value="Создать" class="btn btn-primary" />
        <a asp-action="Index" class="btn btn-secondary ml-2">Вернуться к списку</a>
    </div>
</form>
</div>
</div>
</div>

```

```

@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
</script>
    document.addEventListener("DOMContentLoaded", function () {
        const textInputs = document.querySelectorAll("input[data-validate='no-numbers-required']");

        textInputs.forEach(input => {
            input.addEventListener("input", function () {
                // Проверка на числа
                if (/^d/.test(this.value)) {
                    this.setCustomValidity("Числа не допускаются.");
                } else if (this.value.trim() === "") { // Проверка на пустоту
                    this.setCustomValidity("Поле не должно быть пустым.");
                } else {
                    this.setCustomValidity("");
                }
            });
        });
    });
</script>
<script type="text/javascript">
    $(".custom-file-input").on("change", function () {

```

```

        var fileName = $(this).val().split("\\").pop();
        $(this).siblings(".custom-file-label").addClass("selected").html(fileName);
    });
    document.addEventListener("DOMContentLoaded", function () {
        // Добавим обработчик на отправку формы
        const form = document.querySelector('form');
        form.addEventListener('submit', function (event) {
            const fileInput = document.getElementById('FrontImage');
            if (fileInput.files.length === 0) {
                event.preventDefault(); // Предотвратить отправку формы
                alert("Пожалуйста, выберите файл.");
            }
        });
    });
});
</script>
}

```

Employees

Index.cshtml

```

@using MarriageAgency.ViewModels.EmployeesViewModel;
@model EmployeesViewModel;

@{
    ViewData["Title"] = "Клиенты";
    Layout = "~/Views/Shared/_Layout.cshtml";
    string EmployeeName = Model.FilterEmployeesViewModel.EmployeeName;
    string EmployeeLastName = Model.FilterEmployeesViewModel.EmployeeLastName;
    string EmployeeMiddleName = Model.FilterEmployeesViewModel.EmployeeMiddleName;
    DateOnly? BirthDate = Model.FilterEmployeesViewModel.BirthDate;
}

<h2 class="mb-4">Сотрудники</h2>

<div class="mb-3">
    <a asp-action="Create" class="btn btn-primary">Создать</a>
</div>

@using (Html.BeginForm("Index", "Employees", FormMethod.Get, new { @class = "form-inline mb-3" }))
{
    <fieldset class="form-group">
        <label for="EmployeeName" class="mr-2">Имя сотрудника:</label>
        @Html.TextBox("EmployeeName", EmployeeName, new { @class = "form-control mr-2", @id = "EmployeeName" })
    </fieldset>

    <fieldset class="form-group">
        <label for="EmployeeLastName" class="mr-2">Фамилия сотрудника:</label>
        @Html.TextBox("EmployeeLastName", EmployeeLastName, new { @class = "form-control mr-2", @id = "EmployeeLastName" })
    </fieldset>

    <fieldset class="form-group">
        <label for="EmployeeMiddleName" class="mr-2">Отчество сотрудника:</label>
        @Html.TextBox("EmployeeMiddleName", EmployeeMiddleName, new { @class = "form-control mr-2", @id = "EmployeeMiddleName" })
    </fieldset>

    <fieldset class="form-group">
        <label for="BirthDate" class="mr-2">Дата рождения:</label>
        @Html.TextBox("BirthDate", BirthDate?.ToString("yyyy-MM-dd"), new { @class = "form-control mr-2", @id = "BirthDate", type = "date" })
    </fieldset>

    <input type="submit" value="Найти" class="btn btn-outline-success mt-1" />
}

```



```

<table class="table table-striped table-bordered">
  <thead class="thead-light">
    <tr>
      <th style="width: 150px;">
        <a asp-action="Index"
          asp-route-sortOrder="@Model.SortViewModel.EmployeeNameSort"
          asp-route-EmployeeName="@EmployeeName"
          asp-route-EmployeeLastName="@EmployeeLastName"
          asp-route-EmployeeMiddleName="@EmployeeMiddleName"
          asp-route-BirthDate="@BirthDate?.ToString("yyyy-MM-dd")"
          style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
            @Html.DisplayNameFor(model => model.FilterEmployeesViewModel.EmployeeName)

            @if (Model.SortViewModel.EmployeeNameSort == SortState.EmployeeNameAsc)
            {
              <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
            }
            else if (Model.SortViewModel.EmployeeNameSort == SortState.EmployeeNameDesc)
            {
              <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
            }
          </a>
      </th>
      <th style="width: 150px;">
        <a asp-action="Index"
          asp-route-sortOrder="@Model.SortViewModel.EmployeeLastNameSort"
          asp-route-EmployeeName="@EmployeeName"
          asp-route-EmployeeLastName="@EmployeeLastName"
          asp-route-EmployeeMiddleName="@EmployeeMiddleName"
          asp-route-BirthDate="@BirthDate?.ToString("yyyy-MM-dd")"
          style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
            @Html.DisplayNameFor(model => model.FilterEmployeesViewModel.EmployeeLastName)

            @if (Model.SortViewModel.EmployeeLastNameSort == SortState.EmployeeLastNameAsc)
            {
              <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
            }
            else if (Model.SortViewModel.EmployeeLastNameSort == SortState.EmployeeLastNameDesc)
            {
              <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
            }
          </a>
      </th>
      <th style="width: 100px;">
        <a asp-action="Index"
          asp-route-sortOrder="@Model.SortViewModel.EmployeeMiddleNameSort"
          asp-route-EmployeeName="@EmployeeName"
          asp-route-EmployeeLastName="@EmployeeLastName"
          asp-route-EmployeeMiddleName="@EmployeeMiddleName"
          asp-route-BirthDate="@BirthDate?.ToString("yyyy-MM-dd")"
          style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
            @Html.DisplayNameFor(model => model.FilterEmployeesViewModel.EmployeeMiddleName)

            @if (Model.SortViewModel.EmployeeMiddleNameSort == SortState.EmployeeMiddleNameAsc)
            {
              <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
            }
            else if (Model.SortViewModel.EmployeeMiddleNameSort == SortState.EmployeeMiddleNameDesc)
            {
              <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
            }
          </a>
      </th>
      <th style="width: 150px;">
        @Html.DisplayNameFor(model => model.FilterEmployeesViewModel.Position)
      </th>
    </tr>
  </thead>
</table>

```

```

<th style="width: 150px;">
  <a asp-action="Index"
    asp-route-sortOrder="@Model.SortViewModel.BirthDateSort"
    asp-route-EmployeeName="@EmployeeName"
    asp-route-EmployeeLastName="@EmployeeLastName"
    asp-route-EmployeeMiddleName="@EmployeeMiddleName"
    asp-route-BirthDate="@BirthDate?.ToString("yyyy-MM-dd")"
    style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
    @Html.DisplayNameFor(model => model.FilterEmployeesViewModel.BirthDate)

    @if (Model.SortViewModel.BirthDateSort == SortState.BirthDateAsc)
    {
      <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
    }
    else if (Model.SortViewModel.BirthDateSort == SortState.BirthDateDesc)
    {
      <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
    }
  </a>
</th>
<th style="width: 150px;" class="text-center">Действие</th>
</tr>
</thead>
<tbody>
  @foreach (var item in Model.Employees)
  {
    <tr>
      <td>@Html.DisplayFor(modelItem => item.FirstName)</td>
      <td>@Html.DisplayFor(modelItem => item.LastName)</td>
      <td>@Html.DisplayFor(modelItem => item.MiddleName)</td>
      <td>@Html.DisplayFor(modelItem => item.Position)</td>
      <td>@Html.DisplayFor(modelItem => item.BirthDate)</td>
      <td>
        <div class="btn-group d-flex" role="group" aria-label="Basic example">
          <a asp-action="Edit" asp-route-id="@item.EmployeeId" class="btn btn-sm btn-primary me-1">Редактировать</a>
          <a asp-action="Details" asp-route-id="@item.EmployeeId" class="btn btn-sm btn-secondary me-1">Подробно</a>
          <a asp-action="Delete" asp-route-id="@item.EmployeeId" class="btn btn-sm btn-danger">Удалить</a>
        </div>
      </td>
    </tr>
  }
</tbody>
</table>

<page-link page-model="Model.PageViewModel" page-action="Index"
  page-url-EmployeeName="@EmployeeName"
  page-url-EmployeeLastName="@EmployeeLastName"
  page-url-EmployeeMiddleName="@EmployeeMiddleName"
  page-url-sortOrder="@((Model.SortViewModel.CurrentState))"
  page-url-BirthDate="@BirthDate?.ToString("yyyy-MM-dd")"></page-link>

```

Edit.cshtml

@model Employee

```

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

```

<h2 class="text-center mb-4">Редактировать</h2>

```

<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-6">

```

```

<div class="card shadow-sm">
  <div class="card-body">
    <h4 class="card-title text-center mb-4">Сотрудник</h4>
    <form asp-action="Edit">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <input type="hidden" asp-for="EmployeeId">
      <div class="form-group">
        <label asp-for="FirstName" class="control-label"></label>
        <input asp-for="FirstName" class="form-control" />
        <span asp-validation-for="FirstName" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="LastName" class="control-label"></label>
        <input asp-for="LastName" class="form-control" />
        <span asp-validation-for="LastName" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="MiddleName" class="control-label"></label>
        <input asp-for="MiddleName" class="form-control" />
        <span asp-validation-for="MiddleName" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="Position" class="control-label"></label>
        <input asp-for="Position" class="form-control" />
        <span asp-validation-for="Position" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="BirthDate" class="control-label"></label>
        <input asp-for="BirthDate" class="form-control" type="date" />
        <span asp-validation-for="BirthDate" class="text-danger"></span>
      </div>
      <div class="form-group text-center mt-4">
        <input type="submit" value="Сохранить" class="btn btn-primary" />
      </div>
    </form>
  </div>
</div>

```

```

<div class="text-center mt-3">
  <a asp-action="Index" class="btn btn-link">Вернуться к списку</a>
</div>

```

```

@section Scripts {
  @{
    await Html.RenderPartialAsync("_ValidationScriptsPartial");
  }
  <script>
    document.addEventListener("DOMContentLoaded", function () {
      const textInputs = document.querySelectorAll("input[data-validate='no-numbers']");

      textInputs.forEach(input => {
        input.addEventListener("input", function () {
          // Проверка на числа
          if (/^d/.test(this.value)) {
            this.setCustomValidity("Числа не допускаются.");
          } else if (this.value.trim() === "") { // Проверка на пустоту
            this.setCustomValidity("Поле не должно быть пустым.");
          } else {
            this.setCustomValidity("");
          }
        });
      });
    });
  </script>

```

```
}
```

Details.cshtml

```
@model Employee
```

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

```
<h2 class="text-center mb-4">Детально</h2>
```

```
<div class="container">  
    <div class="card shadow-sm">  
        <div class="card-body">  
            <h4 class="card-title text-center mb-4">Сотрудник</h4>  
            <dl class="row">  
                <dt class="col-sm-4">  
                    @Html.DisplayNameFor(model => model.FirstName)  
                </dt>  
                <dd class="col-sm-8">  
                    @Html.DisplayFor(model => model.FirstName)  
                </dd>  
                <dt class="col-sm-4">  
                    @Html.DisplayNameFor(model => model.LastName)  
                </dt>  
                <dd class="col-sm-8">  
                    @Html.DisplayFor(model => model.LastName)  
                </dd>  
                <dt class="col-sm-4">  
                    @Html.DisplayNameFor(model => model.MiddleName)  
                </dt>  
                <dd class="col-sm-8">  
                    @Html.DisplayFor(model => model.MiddleName)  
                </dd>  
                <dt class="col-sm-4">  
                    @Html.DisplayNameFor(model => model.Position)  
                </dt>  
                <dd class="col-sm-8">  
                    @Html.DisplayFor(model => model.Position)  
                </dd>  
                <dt class="col-sm-4">  
                    @Html.DisplayNameFor(model => model.BirthDate)  
                </dt>  
                <dd class="col-sm-8">  
                    @Html.DisplayFor(model => model.BirthDate)  
                </dd>  
            </dl>  
        </div>  
    </div>  
</div>  
  
<div class="text-center mt-3">  
    <a asp-action="Edit" asp-route-id="@Model.EmployeeId" class="btn btn-primary mx-2">Редактировать</a>  
    <a asp-action="Index" class="btn btn-secondary mx-2">Вернуться к списку</a>  
</div>
```

Delete.cshtml

```
@model Employee
```

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

```

<h2 class="text-center mb-4">Удаление</h2>

<div class="container">
  <div class="card shadow-sm">
    <div class="card-body">
      <h4 class="card-title text-center mb-4">Вы уверены, что хотите удалить этого клиента?</h4>
      <dl class="row">
        <dt class="col-sm-4">
          @Html.DisplayNameFor(model => model.FirstName)
        </dt>
        <dd class="col-sm-8">
          @Html.DisplayFor(model => model.FirstName)
        </dd>
        <dt class="col-sm-4">
          @Html.DisplayNameFor(model => model.LastName)
        </dt>
        <dd class="col-sm-8">
          @Html.DisplayFor(model => model.LastName)
        </dd>
        <dt class="col-sm-4">
          @Html.DisplayNameFor(model => model.MiddleName)
        </dt>
        <dd class="col-sm-8">
          @Html.DisplayFor(model => model.MiddleName)
        </dd>
        <dt class="col-sm-4">
          @Html.DisplayNameFor(model => model.Position)
        </dt>
        <dd class="col-sm-8">
          @Html.DisplayFor(model => model.Position)
        </dd>
        <dt class="col-sm-4">
          @Html.DisplayNameFor(model => model.BirthDate)
        </dt>
        <dd class="col-sm-8">
          @Html.DisplayFor(model => model.BirthDate)
        </dd>
      </dl>
      <form asp-action="Delete" class="text-center mt-4">
        <input type="hidden" asp-for="EmployeeId" />
        <button type="submit" class="btn btn-danger">Удалить</button>
        <a asp-action="Index" class="btn btn-secondary ml-2">Вернуться к списку</a>
      </form>
    </div>
  </div>
</div>

```

Create.cshtml

@model Employee

```

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

```

```

<h2 class="text-center mb-4">Создать</h2>

<div class="container">
  <div class="card shadow-sm">
    <div class="card-body">
      <h4 class="card-title text-center mb-4">Сотрудник</h4>
      <form asp-action="Create">
        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
        <div class="form-group">
          <label asp-for="FirstName" class="control-label"></label>
          <input asp-for="FirstName" class="form-control" data-validate="no-numbers" required/>

```

```

        <span asp-validation-for="FirstName" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="LastName" class="control-label"></label>
        <input asp-for="LastName" class="form-control" data-validate="no-numbers" required/>
        <span asp-validation-for="LastName" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="MiddleName" class="control-label"></label>
        <input asp-for="MiddleName" class="form-control" data-validate="no-numbers" required/>
        <span asp-validation-for="MiddleName" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Position" class="control-label"></label>
        <input asp-for="Position" class="form-control" data-validate="no-numbers" required/>
        <span asp-validation-for="Position" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="BirthDate" class="control-label"></label>
        <input asp-for="BirthDate" class="form-control" type="date" required/>
        <span asp-validation-for="BirthDate" class="text-danger"></span>
    </div>
    <div class="form-group text-center mt-4">
        <input type="submit" value="Создать" class="btn btn-primary" />
        <a asp-action="Index" class="btn btn-secondary ml-2">Вернуться к списку</a>
    </div>
</form>
</div>
</div>
</div>

```

```

@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
    <script>
        document.addEventListener("DOMContentLoaded", function () {
            const textInputs = document.querySelectorAll("input[data-validate='no-numbers']");

            textInputs.forEach(input => {
                input.addEventListener("input", function () {
                    // Проверка на числа
                    if (/^d/.test(this.value)) {
                        this.setCustomValidity("Числа не допускаются.");
                    } else if (this.value.trim() === "") { // Проверка на пустоту
                        this.setCustomValidity("Поле не должно быть пустым.");
                    } else {
                        this.setCustomValidity("");
                    }
                });
            });
        });
    </script>
}

```

Services

Index.cshtml

```

@using MarriageAgency.ViewModels.ServicesViewModel;
@model ServicesViewModel

```

```

@{
    ViewData["Title"] = "КЛИЕНТЫ";
    Layout = "~/Views/Shared/_Layout.cshtml";
    string ClientName = Model.FilterServicesViewModel.ClientName;
    string EmployeeName = Model.FilterServicesViewModel.EmployeeName;
    string AdditionalServiceName = Model.FilterServicesViewModel.AdditionalServiceName;
}

```

```

        decimal? MinCost = Model.FilterServicesViewModel.MinCost;
        decimal? MaxCost = Model.FilterServicesViewModel.MaxCost;
    }

<h2 class="mb-4">Услуги</h2>

<div class="mb-3">
    <a asp-action="Create" class="btn btn-primary">Создать</a>
</div>

@using (Html.BeginForm("Index", "Services", FormMethod.Get, new { @class = "form-inline mb-3" }))
{
    <fieldset class="form-group">
        <label for="ClientName" class="mr-2">Имя клиента:</label>
        @Html.TextBox("ClientName", ClientName, new { @class = "form-control mr-2", @id = "ClientName" })

        <label for="EmployeeName" class="mr-2">Имя сотрудника:</label>
        @Html.TextBox("EmployeeName", EmployeeName, new { @class = "form-control mr-2", @id = "EmployeeName" })

        <label for="AdditionalServiceName" class="mr-2">Услуга:</label>
        @Html.DropDownList("AdditionalServiceName", (SelectList)ViewData["Name"], "Выберите услугу", new { @class = "form-control mr-2", @id = "AdditionalServiceId" })

        <label for="MinCost" class="mr-2">Мин. цена:</label>
        @Html.TextBox("MinCost", MinCost, new { @class = "form-control mr-2", @id = "MinCost", type = "number", step = "0.01" })

        <label for="MaxCost" class="mr-2">Макс. цена:</label>
        @Html.TextBox("MaxCost", MaxCost, new { @class = "form-control mr-2", @id = "MaxCost", type = "number", step = "0.01" })
    </fieldset>
    <input type="submit" value="Найти" class="btn btn-outline-success mt-1" />
}

<table class="table table-striped table-bordered">
    <thead class="thead-light">
        <tr>
            <th style="width: 150px;">
                <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.ClientNameSort" asp-route-ClientName="@@ClientName" asp-route-EmployeeName="@@EmployeeName" asp-route-AdditionalServiceName="@@AdditionalServiceName" asp-route-MinCost="@@MinCost" asp-route-MaxCost="@@MaxCost" style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
                    @Html.DisplayNameFor(model => model.FilterServicesViewModel.ClientName)

                    @if (Model.SortViewModel.ClientNameSort == SortState.ClientNameAsc)
                    {
                        <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
                    }
                    else if (Model.SortViewModel.ClientNameSort == SortState.ClientNameDesc)
                    {
                        <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
                    }
                </a>
            </th>

            <th style="width: 150px;">
                <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.EmployeeNameSort" asp-route-ClientName="@@ClientName" asp-route-EmployeeName="@@EmployeeName" asp-route-AdditionalServiceName="@@AdditionalServiceName" asp-route-MinCost="@@MinCost" asp-route-MaxCost="@@MaxCost" style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
                    @Html.DisplayNameFor(model => model.FilterServicesViewModel.EmployeeName)

                    @if (Model.SortViewModel.EmployeeNameSort == SortState.EmployeeNameAsc)
                    {
                        <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
                    }
                    else if (Model.SortViewModel.EmployeeNameSort == SortState.EmployeeNameDesc)
                    {
                        <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
                    }
                </a>
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var service in Model.FilterServicesViewModel.Services)
        {
            <tr>
                <td>@service.Name</td>
                <td>@service.EmployeeName</td>
                <td>@service.MinCost</td>
                <td>@service.MaxCost</td>
            </tr>
        }
    </tbody>
</table>

```

```

    }
    else if (Model.SortViewModel.EmployeeNameSort == SortState.EmployeeNameDesc)
    {
        <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
    }
</a>
</th>

<th style="width: 150px;">
    <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.AdditionalNameSort" asp-route-
ClientName="@ClientName" asp-route-EmployeeName="@EmployeeName" asp-route-
AdditionalServiceName="@AdditionalServiceName" asp-route-MinCost="@MinCost" asp-route-
MaxCost="@MaxCost" style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
        @Html.DisplayNameFor(model => model.FilterServicesViewModel.AdditionalServiceName)

        @if (Model.SortViewModel.AdditionalNameSort == SortState.AdditionalNameAsc)
        {
            <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
        }
        else if (Model.SortViewModel.AdditionalNameSort == SortState.AdditionalNameDesc)
        {
            <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
        }
    </a>
</th>

<th style="width: 150px;">
    @Html.DisplayNameFor(model => model.FilterServicesViewModel.Date)
</th>

<th style="width: 150px;">
    <a asp-action="Index" asp-route-sortOrder="@Model.SortViewModel.CostSort" asp-route-
ClientName="@ClientName" asp-route-EmployeeName="@EmployeeName" asp-route-
AdditionalServiceName="@AdditionalServiceName" asp-route-MinCost="@MinCost" asp-route-
MaxCost="@MaxCost" style="color: inherit; text-decoration: none; display: inline-flex; align-items: center;">
        @Html.DisplayNameFor(model => model.FilterServicesViewModel.Cost)

        @if (Model.SortViewModel.CostSort == SortState.CostAsc)
        {
            <span class="bi bi-arrow-up" style="margin-left: 5px;"></span>
        }
        else if (Model.SortViewModel.CostSort == SortState.CostDesc)
        {
            <span class="bi bi-arrow-down" style="margin-left: 5px;"></span>
        }
    </a>
</th>
<th style="width: 150px;" class="text-center">Действие</th>
</tr>
</thead>
<tbody>
    @foreach (var item in Model.Services)
    {
        <tr>
            <td>@Html.DisplayFor(modelItem => item.Client.FirstName)</td>
            <td>@Html.DisplayFor(modelItem => item.Employee.FirstName)</td>
            <td>@Html.DisplayFor(modelItem => item.AdditionalService.Name)</td>
            <td>@Html.DisplayFor(modelItem => item.Date)</td>
            <td>@Html.DisplayFor(modelItem => item.Cost)</td>
            <td>
                <div class="btn-group d-flex" role="group" aria-label="Basic example">
                    <a asp-action="Edit" asp-route-id="@item.ServiceId" class="btn btn-sm btn-primary me-
1">Редактировать</a>
                    <a asp-action="Details" asp-route-id="@item.ServiceId" class="btn btn-sm btn-secondary me-
1">Подробнее</a>
                    <a asp-action="Delete" asp-route-id="@item.ServiceId" class="btn btn-sm btn-danger">Удалить</a>
                </div>
            </td>
        </tr>
    }
</tbody>
</table>

```



```

        </td>
    </tr>
}
</tbody>
</table>

```

```

<page-link page-model="Model.PageViewModel" page-action="Index"
    page-url-ClientName="@ClientName"
    page-url-EmployeeName="@EmployeeName"
    page-url-AdditionalServiceName="@AdditionalServiceName"
    page-url-MinCost="@MinCost"
    page-url-MaxCost="@MaxCost"
    page-url-sortOrder="@((Model.SortViewModel.CurrentState))"></page-link>

```

Edit.cshtml

@model Service

```

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

```

```

<h2 class="text-center mb-4">Редактировать</h2>

```

```

<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card shadow-sm">
                <div class="card-body">
                    <h4 class="card-title text-center mb-4">Услуга</h4>
                    <form asp-action="Edit" method="post">
                        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                        <input type="hidden" asp-for="ServiceId" />
                        <div class="form-group">
                            <label asp-for="Client.LastName" class="control-label"></label>
                            <select asp-for="ClientId" class="form-control" asp-items="ViewBag.ClientId"></select>
                            <span asp-validation-for="ClientId" class="text-danger"></span>
                        </div>
                        <div class="form-group">
                            <label asp-for="Employee.FirstName" class="control-label"></label>
                            <select asp-for="EmployeeId" class="form-control" asp-items="ViewBag.EmployeeId"></select>
                            <span asp-validation-for="EmployeeId" class="text-danger"></span>
                        </div>
                        <div class="form-group">
                            <label asp-for="AdditionalService.Name" class="control-label"></label>
                            <select asp-for="AdditionalServiceId" class="form-control" asp-
items="ViewBag.AdditionalServiceId"></select>
                            <span asp-validation-for="AdditionalServiceId" class="text-danger"></span>
                        </div>
                        <div class="form-group">
                            <label asp-for="Date" class="control-label"></label>
                            <input asp-for="Date" class="form-control" type="date" />
                            <span asp-validation-for="Date" class="text-danger"></span>
                        </div>
                        <div class="form-group">
                            <label asp-for="Cost" class="control-label"></label>
                            <input asp-for="Cost" class="form-control" type="number" step="0.01" />
                            <span asp-validation-for="Cost" class="text-danger"></span>
                        </div>

                        <div class="form-group text-center mt-4">
                            <input type="submit" value="Сохранить" class="btn btn-primary" />
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

```

```

        </div>
    </div>
</div>

<div class="text-center mt-3">
    <a asp-action="Index" class="btn btn-link">Вернуться к списку</a>
</div>

```

```

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

```

Details.cshtml

```
@model Service
```

```

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

```

```
<h2 class="text-center mb-4">Детально</h2>
```

```

<div class="container">
    <div class="card shadow-sm">
        <div class="card-body">
            <h4 class="card-title text-center mb-4">Услуга</h4>
            <dl class="row">
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.AdditionalService.Name)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.AdditionalService.Name)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Employee.FirstName)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Employee.FirstName)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Client.FirstName)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Client.FirstName)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Date)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Date)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Cost)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Cost)
                </dd>
            </dl>
        </div>
    </div>
</div>

```

```

<div class="text-center mt-3">
    <a asp-action="Edit" asp-route-id="@Model.ServiceId" class="btn btn-primary mx-2">Редактировать</a>

```

```
<a asp-action="Index" class="btn btn-secondary mx-2">Вернуться к списку</a>
</div>
```

Delete.cshtml

@model Service

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

```
<h2 class="text-center mb-4">Удаление</h2>
```

```
<div class="container">
    <div class="card shadow-sm">
        <div class="card-body">
            <h4 class="card-title text-center mb-4">Вы уверены, что хотите удалить эту услугу?</h4>
            <dl class="row">
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.AdditionalService.Name)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.AdditionalService.Name)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Employee.FirstName)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Employee.FirstName)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Client.FirstName)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Client.FirstName)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Date)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Date)
                </dd>
                <dt class="col-sm-4">
                    @Html.DisplayNameFor(model => model.Cost)
                </dt>
                <dd class="col-sm-8">
                    @Html.DisplayFor(model => model.Cost)
                </dd>
            </dl>

            <form asp-action="Delete" class="text-center mt-4">
                <input type="hidden" asp-for="ServiceId" />
                <button type="submit" class="btn btn-danger">Удалить</button>
                <a asp-action="Index" class="btn btn-secondary ml-2">Вернуться к списку</a>
            </form>
        </div>
    </div>
</div>
```

Create.cshtml

@model Service

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

```
<h2 class="text-center mb-4">Создать</h2>
```

```
<div class="container">
  <div class="card shadow-sm">
    <div class="card-body">
      <h4 class="card-title text-center mb-4">Услуга</h4>
      <form asp-action="Create">
        <div asp-validation-summary="ModelOnly" class="text-danger"></div>

        <div class="form-group">
          <label asp-for="Client.LastName" class="control-label"></label>
          <select asp-for="ClientId" class="form-control" asp-items="ViewBag.ClientId"></select>
          <span asp-validation-for="ClientId" class="text-danger"></span>
        </div>
        <div class="form-group">
          <label asp-for="Employee.FirstName" class="control-label"></label>
          <select asp-for="EmployeeId" class="form-control" asp-items="ViewBag.EmployeeId"></select>
          <span asp-validation-for="EmployeeId" class="text-danger"></span>
        </div>
        <div class="form-group">
          <label asp-for="AdditionalService.Name" class="control-label"></label>
          <select asp-for="AdditionalServiceId" class="form-control" asp-
items="ViewBag.AdditionalServiceId"></select>
          <span asp-validation-for="AdditionalServiceId" class="text-danger"></span>
        </div>
        <div class="form-group">
          <label asp-for="Date" class="control-label"></label>
          <input asp-for="Date" class="form-control" type="date" />
          <span asp-validation-for="Date" class="text-danger"></span>
        </div>
        <div class="form-group">
          <label asp-for="Cost" class="control-label"></label>
          <input asp-for="Cost" class="form-control" type="number" step="0.01" />
          <span asp-validation-for="Cost" class="text-danger"></span>
        </div>
        <div class="form-group text-center mt-4">
          <input type="submit" value="Создать" class="btn btn-primary" />
          <a asp-action="Index" class="btn btn-secondary ml-2">Вернуться к списку</a>
        </div>
      </form>
    </div>
  </div>
</div>
```

```
@section Scripts {
  @{
    await Html.RenderPartialAsync("_ValidationScriptsPartial");
  }
}
```

ПРИЛОЖЕНИЕ Ж

(обязательное)

Текст программы

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);
            }
        }
    }
}
```

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
        EmployeeLastNameAsc, // по фамилии сотрудника по возрастанию
        EmployeeLastNameDesc, // по фамилии сотрудника по убыванию
        EmployeeMiddleNameAsc, // по отчеству сотрудника по возрастанию
        EmployeeMiddleNameDesc, // по отчеству сотрудника по убыванию
        AdditionalNameAsc, // по имени дополнительной услуги по возрастанию
        AdditionalNameDesc, // по имени дополнительной услуги по убыванию
        BirthDateAsc, // по дате рождения по возрастанию
        BirthDateDesc, // по дате рождения по убыванию
        GenderAsc, // по полу по возрастанию
        GenderDesc, // по полу по убыванию
        NationalityAsc, // по национальности по возрастанию
        NationalityDesc, // по национальности по убыванию
        NationalityNameAsc, // по имени национальности по возрастанию
        NationalityNameDesc, // по имени национальности по убыванию
        ZodiacSignNameAsc, // по знаку зодиака по возрастанию
        ZodiacSignNameDesc, // по знаку зодиака по убыванию
        AgeAsc, // по возрасту по возрастанию
        AgeDesc, // по возрасту по убыванию
        HobbiesAsc, // по хобби по возрастанию
        HobbiesDesc, // по хобби по убыванию
        ContactAddressAsc,
```

```

        ContactAddressDesc,
    }

    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 EmployeeLastNameSort { get; set; } // Сортировка по фамилии сотрудника
        public SortState EmployeeMiddleNameSort { get; set; } // Сортировка по отчеству сотрудника
        public SortState AdditionalNameSort { get; set; } // Сортировка по имени дополнительной услуги
        public SortState BirthDateSort { get; set; } // Сортировка по дате рождения
        public SortState GenderSort { get; set; } // Сортировка по полу
        public SortState NationalitySort { get; set; } // Сортировка по национальности
        public SortState NationalityNameSort { get; set; } // Сортировка по имени национальности
        public SortState ZodiacSignNameSort { get; set; } // Сортировка по знаку зодиака
        public SortState AgeSort { get; set; } // Сортировка по возрасту
        public SortState HobbiesSort { get; set; } // Сортировка по хобби
        public SortState ContactAddressSort { get; set; } // Сортировка по хобби
        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 : SortState.ClientNameAsc;

            // Установка сортировки для имени сотрудника
            EmployeeNameSort = sortOrder == SortState.EmployeeNameAsc ? SortState.EmployeeNameDesc : SortState.EmployeeNameAsc;

            // Установка сортировки для фамилии сотрудника
            EmployeeLastNameSort = sortOrder == SortState.EmployeeLastNameAsc ? SortState.EmployeeLastNameDesc : SortState.EmployeeLastNameAsc;

            // Установка сортировки для отчества сотрудника
            EmployeeMiddleNameSort = sortOrder == SortState.EmployeeMiddleNameAsc ? SortState.EmployeeMiddleNameDesc : SortState.EmployeeMiddleNameAsc;

            // Установка сортировки для имени дополнительной услуги
            AdditionalNameSort = sortOrder == SortState.AdditionalNameAsc ? SortState.AdditionalNameDesc : SortState.AdditionalNameAsc;

            // Установка сортировки для даты рождения
            BirthDateSort = sortOrder == SortState.BirthDateAsc ? SortState.BirthDateDesc : SortState.BirthDateAsc;

            // Установка сортировки для пола
            GenderSort = sortOrder == SortState.GenderAsc ? SortState.GenderDesc : SortState.GenderAsc;

            // Установка сортировки для национальности
            NationalitySort = sortOrder == SortState.NationalityAsc ? SortState.NationalityDesc : SortState.NationalityAsc;

            // Установка сортировки для знака зодиака
            ZodiacSignNameSort = sortOrder == SortState.ZodiacSignNameAsc ? SortState.ZodiacSignNameDesc : SortState.ZodiacSignNameAsc;

            // Установка сортировки для возраста
            AgeSort = sortOrder == SortState.AgeAsc ? SortState.AgeDesc : SortState.AgeAsc;

            // Установка сортировки для хобби
            HobbiesSort = sortOrder == SortState.HobbiesAsc ? SortState.HobbiesDesc : SortState.HobbiesAsc;

            NationalityNameSort = sortOrder == SortState.NationalityNameAsc ? SortState.NationalityNameDesc : SortState.NationalityNameAsc;
        }
    }

```

```

        ContactAddressSort = sortOrder == SortState.ContactAddressAsc ? SortState.ContactAddressDesc : Sort-
State.ContactAddressAsc;

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

```

ClientsViewModel.cs

```

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

namespace MarriageAgency.ViewModels.ClientsViewModel
{
    public class ClientsViewModel
    {
        public IEnumerable<Client> Clients { get; set; }

        public FilterClientsViewModel FilterClientsViewModel { get; set; }

        //Свойство для навигации по страницам
        public PageViewModel PageViewModel { get; set; }
        // Порядок сортировки
        public SortViewModel SortViewModel { get; set; }
    }
}

```

FilterClientsViewModel.cs

```

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

namespace MarriageAgency.ViewModels.ClientsViewModel
{
    public class FilterClientsViewModel
    {
        [Display(Name = "Клиент")]
        public string ClientName { get; set; } = null!;

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

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

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

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

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

```

```

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

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

```

EmployeesViewModel.cs

```

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

namespace MarriageAgency.ViewModels.EmployeesViewModel
{
    public class EmployeesViewModel
    {
        public IEnumerable<Employee> Employees { get; set; }

        public FilterEmployeesViewModel FilterEmployeesViewModel { get; set; }

        //Свойство для навигации по страницам
        public PageViewModel PageViewModel { get; set; }
        // Порядок сортировки
        public SortViewModel SortViewModel { get; set; }
    }
}

```

FilterEmployeesViewModel.cs

```

using System.ComponentModel.DataAnnotations;

namespace MarriageAgency.ViewModels.EmployeesViewModel
{
    public class FilterEmployeesViewModel
    {
        [Display(Name = "Имя Сотрудника")]
        public string EmployeeName { get; set; } = null!;

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

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

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

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

```

ServicesViewModel.cs

```

using Microsoft.AspNetCore.Mvc.Rendering;
using System.Collections.Generic;
using MarriageAgency.DataLayer.Models;

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

        //Свойство для фильтрации
        public FilterServicesViewModel FilterServicesViewModel { get; set; }
        //Свойство для навигации по страницам
    }
}

```



```

        public PageViewModel PageViewModel { get; set; }

        public SortViewModel SortViewModel { get; set; }
    }
}

```

FilterServicesViewModel.cs

```

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

```

```

namespace MarriageAgency.ViewModels.ServicesViewModel
{
    public class FilterServicesViewModel
    {
        [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; }
        public decimal? MinCost { get; set; }
        public decimal? MaxCost { get; set; }

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

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

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

```

ПРИЛОЖЕНИЕ 3

(обязательное)

Текст программы

SessionExtensions.cs

```
using Microsoft.AspNetCore.Http;
using Newtonsoft.Json;
using System.Collections.Generic;

namespace MarriageAgency.Infrastructure
{
    // Методы расширения для ISession
    public static class SessionExtensions
    {
        //Запись параметризованного объекта в сессию
        public static void Set<T>(this ISession session, string key, T value)
        {
            session.SetString(key, JsonConvert.SerializeObject(value));
        }
        //Считывание параметризованного 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);
        }
        //Запись объекта типа Dictionary<string, string> в сессию
        public static void Set(this ISession session, string key, Dictionary<string, string> dictionary)
        {
            session.SetString(key, JsonConvert.SerializeObject(dictionary));
        }
        //Считывание объекта типа Dictionary<string, string> из сессии
        public static Dictionary<string, string> Get(this ISession session, string key)
        {
            var value = session.GetString(key);
            return value == null ? default : JsonConvert.DeserializeObject<Dictionary<string, string>>(value);
        }
    }
}
```