**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №3**

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

на тему: «Обработка *HTTP* запросов средствами *ASP.NET Core*. Сохранение состояния. Кэширование.»

Выполнил: студент гр. ИТП-31

Демиденко Д. В.

Принял: доцент

Асенчик О. Д.

Гомель 2023

**Цель работы:** ознакомиться c методами обработкой *HTTP* средствами *ASP*.*NET* *Core*, методами сохранения состояния приложения и повышение производительности приложений путем использования разных видов кэширования.

**Задание:**

Используя ранее разработанные объектную модель для доступа к данным в заданной предметной области разработать простое *ASP*.*NET* *Core* приложение.

2.1. С использование методов *Run*, *Map* и *Use* разработать:

1. компоненты промежуточного уровня (middleware) и встроить их в конвейер обработки *HTTP* запроса с целью кэширования 20 записей из каждой таблицы базы данных заданной предметной области с помощью встроенного инструмента кэширования ­­­­­­­­­­– объекта IMemoryCache. Данные в кэше хранить неизменными в течение 2\**N+*240 секунд, где *N* – номер вашего варианта.

2. собственную систему маршрутизации входящих запросов:

• если *URL* адрес входящего запроса содержит \*info* – выводить в выходной поток для отображения браузером информацию о клиенте и выходить из конвейера обработки запроса;

• если *URL* адрес входящего запроса содержит \*table* (где *table* – имя таблицы из базы данных) – выводить в выходной поток для отображения браузером с использование метода *Response*.*WriteAsync* кэшированную информацию из соответствующей таблицы базы данных и выходить из конвейера обработки запроса;

• если *URL* адрес входящего запроса содержит \*searchform1* или \*searchform2* – выводить в выходной поток для отображения браузером с использование метода *Response*.*WriteAsync* формы для поиска информации из базы данных и выходить из конвейера обработки запроса;

форма должна содержать, как минимум: одно поле, одного поле со списком, один список, одну кнопку;

• в противном случае (*URL* адрес входящего запроса не содержит перечисленных выше элементов) – продолжать обрабатывать другие компоненты конвейера обработки запросов;

2.2. Реализовать сохранение состояния элементов одной формы одной страницы с использованием куки (\*searchform1*).

2.3. Реализовать сохранение состояния элементов одной формы одной страницы в виде одного объекта специальной структуры с использованием объекта *Session* (\*searchform2*).

2.4. Осуществить заполнение элементов формы при их загрузке данными ранее сохранненными в объекте *Session* и куки (\*searchform1*, \*searchform2*).

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

2.6. Разместить выполненный проект на *github*.

**Ход работы**

В первую очередь при выполнении лабораторной работы база данных созданная в первой лабораторной работе была перенесена в проект при помощи технологии *ENTITY FRAMEWORK*. После перенесения базы в проект были сгенерированы классы моделей и класс контекста. Далее после подключения *Entity Framework* строка подключения была перенесена в конфигурационный файл *appsetings.json* далее при помощи класса *WebApplicationBuilder* файл контекста был внедрен в при помощи *DI* в главный класс проекта.

Для кеширования таблиц базы данных был создан класс *RealEstateAgencyContext*, в котором находятся методы занесения таблиц в кэш и изъятие данных из кэша. Методы добавления данных в кэш вызывается в методе *Run* при запуске сервера.

Далее были написаны обработчики для запросов с использованием метода *Map*. Для построения *HTMI*-страницы был создан класс *TableWriter* содержащий метод *WriteTable*.

Далее после создания всех классов были установлены все связи между *url* адресами и обработчиками. Пример страницы с главным меню приложения указаны на рисунке 1.

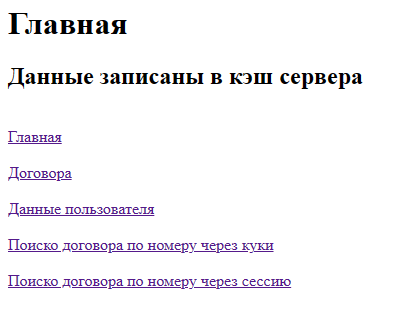


Рисунок 1 – Пример главного меню приложения

Далее был реализован метод который выводят данные из базы в виде таблицы. Пример таблицы указан на рисунке 2.

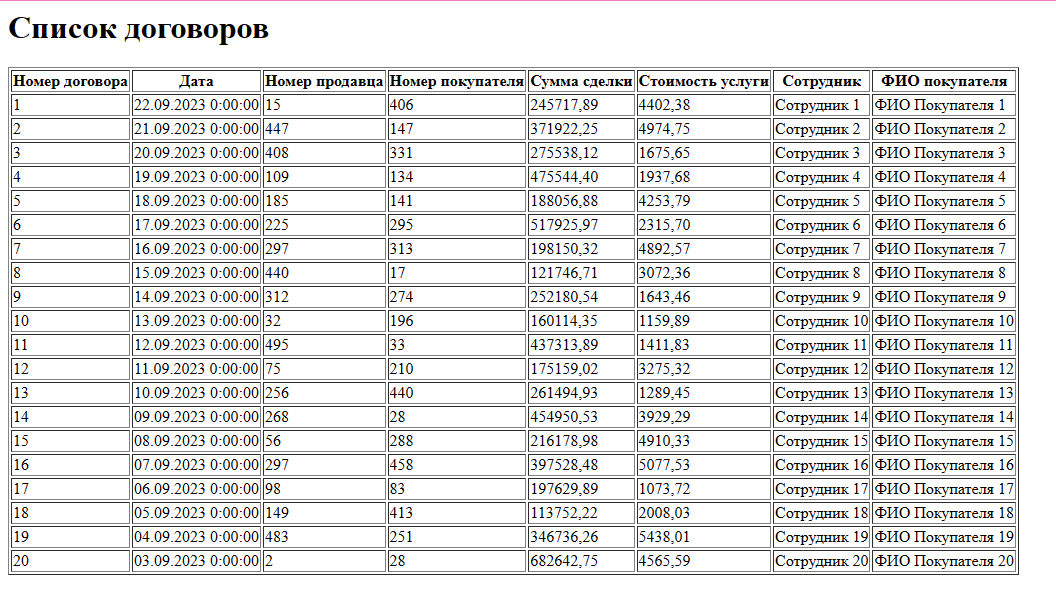


Рисунок 2 – Пример таблицы

Далее были реализованы страницы, которые реализуют поля для фильтрации данных в таблице. Поля, которые используются для ввода данных для фильтрации сохраняют данные в *Sessions.* Пример страницы с поиском указан на рисунке 3.

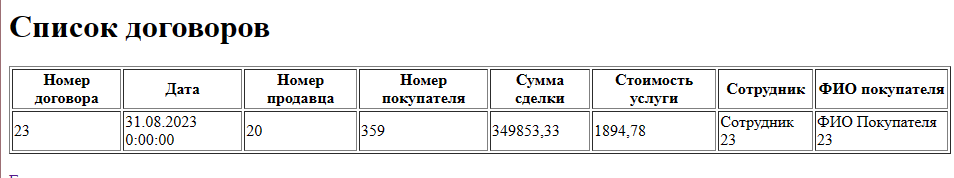


Рисунок 3 – Пример таблицы с фильтрацией с использованием *Sessions*

Далее были реализованы страницы, которые реализуют поля для фильтрации данных в таблице. Поля, которые используются для ввода данных для фильтрации сохраняют данные в *Cookies.* Пример страницы с поиском указан на рисунке 4.

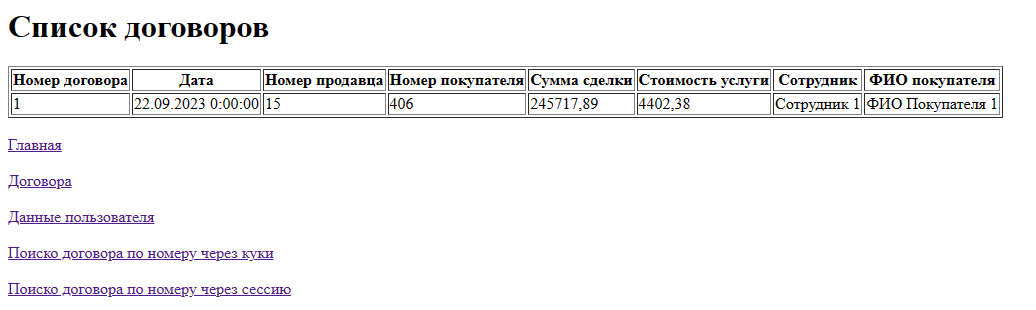


Рисунок 4 – Пример таблицы с фильтрацией с использованием *Cookies*

Пример хранения данных в *Session* и *Cookies* указаны на рисунке 5.

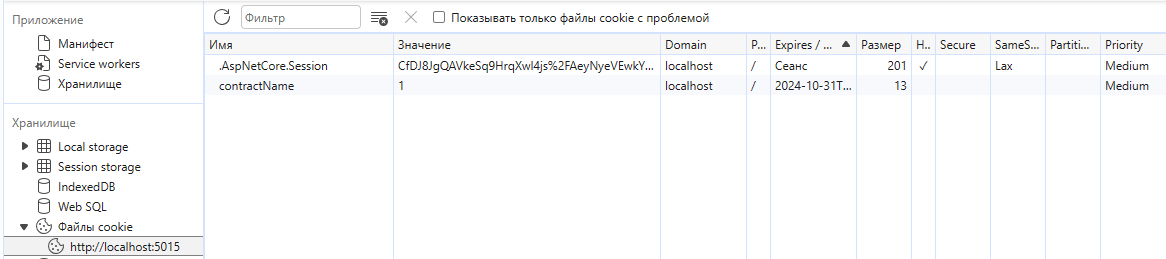


Рисунок 5 – Пример хранения данных в *Session* и *Cookies*

*GitHub*: https://github.com/EvgeniBondarev/DDBAISE/tree/main/Laba3

**Вывод**: в ходе выполнения лабораторной работы была освоена технология *ASP .NET* для создания веб-приложений. Изучен интерфейс *IMemoryCache* для кэширования данных, освоены методы обработки запросов с использованием класса *HttpContext* и способы сохранения информации во временных хранилищах с применением технологий *Sessions* и *Cookies*.

**ПРИЛОЖНИЕ А**

Листинг класса *RealEstateAgencyContext*

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

using RealEstateAgencyL3.Models;

namespace RealEstateAgencyL3.Data;

public partial class RealEstateAgencyContext : DbContext

{

public RealEstateAgencyContext()

{

}

public RealEstateAgencyContext(DbContextOptions<RealEstateAgencyContext> options)

: base(options)

{

}

public virtual DbSet<Apartment> Apartments { get; set; }

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

public virtual DbSet<ContractService> ContractServices { get; set; }

public virtual DbSet<Seller> Sellers { get; set; }

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

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

=> optionsBuilder.UseSqlServer("Data Source=DESKTOP-3T2N3HR\\SQLEXPRESS;Initial Catalog=RealEstateAgency;Integrated Security=True;Encrypt=False;TrustServerCertificate=True");

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Apartment>(entity =>

{

entity.HasKey(e => e.ApartmentId).HasName("PK\_\_Apartmen\_\_CBDF57444B3534CB");

entity.Property(e => e.ApartmentId)

.ValueGeneratedNever()

.HasColumnName("ApartmentID");

entity.Property(e => e.AdditionalPreferences).HasMaxLength(1000);

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

entity.Property(e => e.Description).HasMaxLength(1000);

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

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

});

modelBuilder.Entity<Contract>(entity =>

{

entity.HasKey(e => e.ContractId).HasName("PK\_\_Contract\_\_C90D3409C859D437");

entity.Property(e => e.ContractId)

.ValueGeneratedNever()

.HasColumnName("ContractID");

entity.Property(e => e.BuyerId).HasColumnName("BuyerID");

entity.Property(e => e.DateOfContract).HasColumnType("date");

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

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

entity.Property(e => e.Fiobuyer)

.HasMaxLength(255)

.HasColumnName("FIOBuyer");

entity.Property(e => e.SellerId).HasColumnName("SellerID");

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

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

.HasForeignKey(d => d.BuyerId)

.HasConstraintName("FK\_\_Contracts\_\_Buyer\_\_3C69FB99");

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

.HasForeignKey(d => d.SellerId)

.HasConstraintName("FK\_\_Contracts\_\_Selle\_\_3B75D760");

});

modelBuilder.Entity<ContractService>(entity =>

{

entity.HasKey(e => e.ContractServiceId).HasName("PK\_\_Contract\_\_55E1AE33F404A6CB");

entity.Property(e => e.ContractServiceId)

.ValueGeneratedNever()

.HasColumnName("ContractServiceID");

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

entity.Property(e => e.ServiceId).HasColumnName("ServiceID");

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

.HasForeignKey(d => d.ContractId)

.HasConstraintName("FK\_\_ContractS\_\_Contr\_\_412EB0B6");

entity.HasOne(d => d.Service).WithMany(p => p.ContractServices)

.HasForeignKey(d => d.ServiceId)

.HasConstraintName("FK\_\_ContractS\_\_Servi\_\_4222D4EF");

});

modelBuilder.Entity<Seller>(entity =>

{

entity.HasKey(e => e.SellerId).HasName("PK\_\_Sellers\_\_7FE3DBA12300466D");

entity.Property(e => e.SellerId)

.ValueGeneratedNever()

.HasColumnName("SellerID");

entity.Property(e => e.AdditionalInformation).HasMaxLength(1000);

entity.Property(e => e.Address).HasMaxLength(500);

entity.Property(e => e.ApartmentAddress).HasMaxLength(500);

entity.Property(e => e.ApartmentId).HasColumnName("ApartmentID");

entity.Property(e => e.DateOfBirth).HasColumnType("date");

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

entity.Property(e => e.Gender).HasMaxLength(10);

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

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

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

entity.HasOne(d => d.Apartment).WithMany(p => p.Sellers)

.HasForeignKey(d => d.ApartmentId)

.HasConstraintName("FK\_\_Sellers\_\_Apartme\_\_38996AB5");

});

modelBuilder.Entity<Service>(entity =>

{

entity.HasKey(e => e.ServiceId).HasName("PK\_\_Services\_\_C51BB0EAFDD30054");

entity.Property(e => e.ServiceId)

.ValueGeneratedNever()

.HasColumnName("ServiceID");

entity.Property(e => e.Description).HasMaxLength(1000);

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

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

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

Листинг класса *ICacheContractsService*

using System.Threading.Tasks;

using RealEstateAgencyL3.Models;

namespace RealEstateAgencyL3.Services

{

public interface ICacheContractsService

{

public IEnumerable<Contract> GetContracts(int rowsNumber = 20);

public void AddContracts(string cacheKey, int rowsNumber = 20);

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

public IEnumerable<Contract> GetContractById(int? contractId, int rowsNumber = 20);

}

}

Листинг класса *CacheContractsService*

using Microsoft.Extensions.Caching.Memory;

using RealEstateAgencyL3.Data;

using Microsoft.EntityFrameworkCore;

using System.Threading.Tasks;

using RealEstateAgencyL3.Models;

using System.Linq;

using System.Net;

namespace RealEstateAgencyL3.Services

{

public class CacheContractsService : ICacheContractsService

{

private readonly RealEstateAgencyContext \_dbContext;

private readonly IMemoryCache \_memoryCache;

private int time = 2 \* 9 \* 240;

public CacheContractsService(RealEstateAgencyContext dbContext, IMemoryCache memoryCache)

{

\_dbContext = dbContext;

\_memoryCache = memoryCache;

}

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

{

return \_dbContext.Contracts.Take(rowsNumber).ToList();

}

public void AddContracts(string cacheKey, int rowsNumber = 20)

{

IEnumerable<Contract> Contracts = \_dbContext.Contracts.Take(rowsNumber).ToList();

if (Contracts != null)

{

\_memoryCache.Set(cacheKey, Contracts, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(time)

});

}

}

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

{

IEnumerable<Contract> Contracts;

if (!\_memoryCache.TryGetValue(cacheKey, out Contracts))

{

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

if (Contracts != null)

{

\_memoryCache.Set(cacheKey, Contracts,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(time)));

}

}

return Contracts;

}

public IEnumerable<Contract> GetContractById(int? ContractId, int rowsNumber = 20)

{

return \_dbContext.Contracts

.Where(contract => contract.ContractId == ContractId)

.ToList();

}

}

}

Листинг класса *Program*

using Microsoft.AspNetCore.Http;

using Microsoft.EntityFrameworkCore;

using RealEstateAgencyL3.Data;

using System.Threading.Tasks;

using RealEstateAgencyL3.Services;

using RealEstateAgencyL3.Models;

using System.Text.Json;

internal class Program

{

private static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

var services = builder.Services;

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

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

services.AddMemoryCache();

services.AddDistributedMemoryCache();

services.AddSession();

services.AddScoped<ICacheContractsService, CacheContractsService>();

var app = builder.Build();

app.UseStaticFiles();

app.UseSession();

app.Map("/info", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

string strResponse = "<HTML><HEAD><TITLE>Информация</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Информация:</H1>";

strResponse += "<BR> Сервер: " + context.Request.Host;

strResponse += "<BR> Путь: " + context.Request.PathBase;

strResponse += "<BR> Протокол: " + context.Request.Protocol;

strResponse += "<BR><A href='/'>Главная</A></BODY></HTML>";

await context.Response.WriteAsync(strResponse);

});

});

app.Map("/contracts", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

ICacheContractsService cachedContractsService = context.RequestServices.GetService<ICacheContractsService>();

IEnumerable<Contract> Contracts = cachedContractsService.GetContracts("Contracts20");

string HtmlString = "<HTML><HEAD><TITLE>Договора</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Список договоров</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Номер договора</TH>";

HtmlString += "<TH>Дата</TH>";

HtmlString += "<TH>Номер продавца</TH>";

HtmlString += "<TH>Номер покупателя</TH>";

HtmlString += "<TH>Сумма сделки</TH>";

HtmlString += "<TH>Стоимость услуги</TH>";

HtmlString += "<TH>Сотрудник</TH>";

HtmlString += "<TH>ФИО покупателя</TH>";

HtmlString += "</TR>";

foreach (var Contract in Contracts)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + Contract.ContractId + "</TD>";

HtmlString += "<TD>" + Contract.DateOfContract + "</TD>";

HtmlString += "<TD>" + Contract.SellerId + "</TD>";

HtmlString += "<TD>" + Contract.BuyerId + "</TD>";

HtmlString += "<TD>" + Contract.DealAmount + "</TD>";

HtmlString += "<TD>" + Contract.ServiceCost + "</TD>";

HtmlString += "<TD>" + Contract.Employee + "</TD>";

HtmlString += "<TD>" + Contract.Fiobuyer + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "<BR><A href='/'>Главная</A></BR>";

HtmlString += "<BR><A href='/Contracts'>Договора</A></BR>";

HtmlString += "<BR><A href='/form'>Данные пользователя</A></BR>";

HtmlString += "<BR><A href='/searchForm1'>Поиско договора по номеру через куки</A></BR>";

HtmlString += "<BR><A href='/SeacrhForm2'>Поиско договора по номеру через сессию</A></BR>";

HtmlString += "</BODY></HTML>";

await context.Response.WriteAsync(HtmlString);

});

});

app.Map("/searchForm1", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

ICacheContractsService cachedContractsService = context.RequestServices.GetService<ICacheContractsService>();

IEnumerable<Contract> Contracts = cachedContractsService.GetContracts("Contracts20");

string HtmlString = "<HTML><HEAD><TITLE>Договора</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<form method = 'get' action='/SearchContract1'>";

if (context.Request.Cookies.TryGetValue("ContractName", out string savedContract))

{

HtmlString += "<input type='text' name='ContractName' placeholder='Введите номер договора' value='" + savedContract + "' />";

}

else

{

HtmlString += "<input type='text' name='ContractName' placeholder='Введите номер договора' />";

}

HtmlString += "<button type='submit'>Поиск</button>" +

"</form>" +

"<BODY><H1>Список договоров</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Номер договора</TH>";

HtmlString += "<TH>Дата</TH>";

HtmlString += "<TH>Номер продавца</TH>";

HtmlString += "<TH>Номер покупателя</TH>";

HtmlString += "<TH>Сумма сделки</TH>";

HtmlString += "<TH>Стоимость услуги</TH>";

HtmlString += "<TH>Сотрудник</TH>";

HtmlString += "<TH>ФИО покупателя</TH>";

HtmlString += "</TR>";

foreach (var Contract in Contracts)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + Contract.ContractId + "</TD>";

HtmlString += "<TD>" + Contract.DateOfContract + "</TD>";

HtmlString += "<TD>" + Contract.SellerId + "</TD>";

HtmlString += "<TD>" + Contract.BuyerId + "</TD>";

HtmlString += "<TD>" + Contract.DealAmount + "</TD>";

HtmlString += "<TD>" + Contract.ServiceCost + "</TD>";

HtmlString += "<TD>" + Contract.Employee + "</TD>";

HtmlString += "<TD>" + Contract.Fiobuyer + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "<BR><A href='/'>Главная</A></BR>";

HtmlString += "<BR><A href='/Contracts'>Договора</A></BR>";

HtmlString += "<BR><A href='/form'>Данные пользователя</A></BR>";

HtmlString += "<BR><A href='/searchForm1'>Поиско договора по номеру через куки</A></BR>";

HtmlString += "<BR><A href='/SeacrhForm2'>Поиско договора по номеру через сессию</A></BR>";

HtmlString += "</BODY></HTML>";

await context.Response.WriteAsync(HtmlString);

});

});

app.Map("/SearchContract1", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

string contractName = context.Request.Query["contractName"];

Console.WriteLine(contractName);

var cookieOptions = new CookieOptions

{

Expires = DateTimeOffset.UtcNow.AddYears(1),

IsEssential = true

};

context.Response.Cookies.Append("contractName", contractName, cookieOptions);

ICacheContractsService cachedContractsService = context.RequestServices.GetService<ICacheContractsService>();

IEnumerable<Contract> Contracts = cachedContractsService.GetContractById(int.Parse(contractName));

string HtmlString = "<HTML><HEAD><TITLE>Договора</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Список договоров</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Номер договора</TH>";

HtmlString += "<TH>Дата</TH>";

HtmlString += "<TH>Номер продавца</TH>";

HtmlString += "<TH>Номер покупателя</TH>";

HtmlString += "<TH>Сумма сделки</TH>";

HtmlString += "<TH>Стоимость услуги</TH>";

HtmlString += "<TH>Сотрудник</TH>";

HtmlString += "<TH>ФИО покупателя</TH>";

HtmlString += "</TR>";

foreach (var Contract in Contracts)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + Contract.ContractId + "</TD>";

HtmlString += "<TD>" + Contract.DateOfContract + "</TD>";

HtmlString += "<TD>" + Contract.SellerId + "</TD>";

HtmlString += "<TD>" + Contract.BuyerId + "</TD>";

HtmlString += "<TD>" + Contract.DealAmount + "</TD>";

HtmlString += "<TD>" + Contract.ServiceCost + "</TD>";

HtmlString += "<TD>" + Contract.Employee + "</TD>";

HtmlString += "<TD>" + Contract.Fiobuyer + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "<BR><A href='/'>Главная</A></BR>";

HtmlString += "<BR><A href='/Contracts'>Договора</A></BR>";

HtmlString += "<BR><A href='/form'>Данные пользователя</A></BR>";

HtmlString += "<BR><A href='/searchForm1'>Поиско договора по номеру через куки</A></BR>";

HtmlString += "<BR><A href='/SeacrhForm2'>Поиско договора по номеру через сессию</A></BR>";

HtmlString += "</BODY></HTML>";

await context.Response.WriteAsync(HtmlString);

});

});

app.Map("/SeacrhForm2", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

ICacheContractsService cachedContractsService = context.RequestServices.GetService<ICacheContractsService>();

IEnumerable<Contract> Contracts = cachedContractsService.GetContracts("Contracts20");

string json = File.ReadAllText("session.json");

string HtmlString = "<HTML><HEAD><TITLE>Учители</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<form method='get' action='/SearchContract2'>";

if (context.Session.TryGetValue("SearchFormState", out byte[] searchFormStateBytes))

{

var searchFormState = JsonSerializer.Deserialize<SearchFormState>(json);

Console.WriteLine(searchFormState.ContactId);

HtmlString += "<input type='text' name='ContractName' placeholder='Введите имя учителя' value='" + searchFormState.ContactId + "' />";

}

else

{

HtmlString += "<input type='text' name='ContractName' placeholder='Введите номер договора' />";

}

HtmlString += "<button type='submit'>Поиск</button>" +

"</form>" +

"<BODY><H1>Список договоров</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Номер договора</TH>";

HtmlString += "<TH>Дата</TH>";

HtmlString += "<TH>Номер продавца</TH>";

HtmlString += "<TH>Номер покупателя</TH>";

HtmlString += "<TH>Сумма сделки</TH>";

HtmlString += "<TH>Стоимость услуги</TH>";

HtmlString += "<TH>Сотрудник</TH>";

HtmlString += "<TH>ФИО покупателя</TH>";

HtmlString += "</TR>";

foreach (var Contract in Contracts)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + Contract.ContractId + "</TD>";

HtmlString += "<TD>" + Contract.DateOfContract + "</TD>";

HtmlString += "<TD>" + Contract.SellerId + "</TD>";

HtmlString += "<TD>" + Contract.BuyerId + "</TD>";

HtmlString += "<TD>" + Contract.DealAmount + "</TD>";

HtmlString += "<TD>" + Contract.ServiceCost + "</TD>";

HtmlString += "<TD>" + Contract.Employee + "</TD>";

HtmlString += "<TD>" + Contract.Fiobuyer + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "<BR><A href='/'>Главная</A></BR>";

HtmlString += "<BR><A href='/Contracts'>Договора</A></BR>";

HtmlString += "<BR><A href='/form'>Данные пользователя</A></BR>";

HtmlString += "<BR><A href='/searchForm1'>Поиско договора по номеру через куки</A></BR>";

HtmlString += "<BR><A href='/SeacrhForm2'>Поиско договора по номеру через сессию</A></BR>";

HtmlString += "</BODY></HTML>";

await context.Response.WriteAsync(HtmlString);

});

});

app.Map("/SearchContract2", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

string ContractName = context.Request.Query["ContractName"];

var searchFormState = new SearchFormState

{

ContactId = int.Parse(ContractName)

};

string json = JsonSerializer.Serialize(searchFormState);

File.WriteAllText("session.json", json);

var searchFormStateBytes = JsonSerializer.SerializeToUtf8Bytes(searchFormState);

context.Session.Set("SearchFormState", searchFormStateBytes);

ICacheContractsService cacheContractsService = context.RequestServices.GetService<ICacheContractsService>();

IEnumerable<Contract> Contracts = cacheContractsService.GetContractById(int.Parse(ContractName));

string HtmlString = "<HTML><HEAD><TITLE>Договора</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Список договоров</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Номер договора</TH>";

HtmlString += "<TH>Дата</TH>";

HtmlString += "<TH>Номер продавца</TH>";

HtmlString += "<TH>Номер покупателя</TH>";

HtmlString += "<TH>Сумма сделки</TH>";

HtmlString += "<TH>Стоимость услуги</TH>";

HtmlString += "<TH>Сотрудник</TH>";

HtmlString += "<TH>ФИО покупателя</TH>";

HtmlString += "</TR>";

foreach (var Contract in Contracts)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + Contract.ContractId + "</TD>";

HtmlString += "<TD>" + Contract.DateOfContract + "</TD>";

HtmlString += "<TD>" + Contract.SellerId + "</TD>";

HtmlString += "<TD>" + Contract.BuyerId + "</TD>";

HtmlString += "<TD>" + Contract.DealAmount + "</TD>";

HtmlString += "<TD>" + Contract.ServiceCost + "</TD>";

HtmlString += "<TD>" + Contract.Employee + "</TD>";

HtmlString += "<TD>" + Contract.Fiobuyer + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "<BR><A href='/'>Главная</A></BR>";

HtmlString += "<BR><A href='/Contracts'>Договора</A></BR>";

HtmlString += "<BR><A href='/form'>Данные пользователя</A></BR>";

HtmlString += "<BR><A href='/searchForm1'>Поиско договора по номеру через куки</A></BR>";

HtmlString += "<BR><A href='/SeacrhForm2'>Поиско договора по номеру через сессию</A></BR>";

HtmlString += "</BODY></HTML>";

await context.Response.WriteAsync(HtmlString);

});

});

app.Run((context) =>

{

ICacheContractsService cachedContractsService = context.RequestServices.GetService<ICacheContractsService>();

cachedContractsService.AddContracts("Contracts20");

string HtmlString = "<HTML><HEAD><TITLE>Договора</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Главная</H1>";

HtmlString += "<H2>Данные записаны в кэш сервера</H2>";

HtmlString += "<BR><A href='/'>Главная</A></BR>";

HtmlString += "<BR><A href='/Contracts'>Договора</A></BR>";

HtmlString += "<BR><A href='/form'>Данные пользователя</A></BR>";

HtmlString += "<BR><A href='/searchForm1'>Поиско договора по номеру через куки</A></BR>";

HtmlString += "<BR><A href='/SeacrhForm2'>Поиско договора по номеру через сессию</A></BR>";

HtmlString += "</BODY></HTML>";

return context.Response.WriteAsync(HtmlString);

});

app.Run();

}

public class SearchFormState

{

public int ContactId { get; set; }

}

}