**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

Бодарев Е.Ю.

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 *SubsCityContext*, 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. Пример проверки занесения таблиц в кэш находится на рисунке 1.

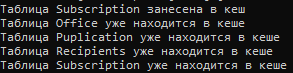


Рисунок 1 – Пример проверки занесения таблиц в кэш

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 2.

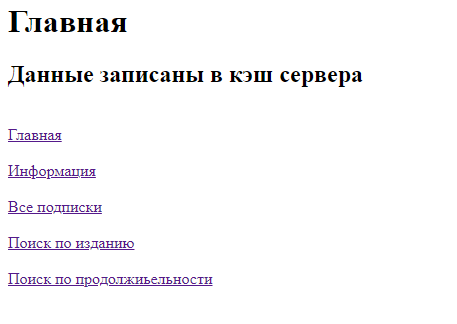


Рисунок 2 – Пример главного меню приложения

Далее была разработана страница для вывода информации о запросе пользователя. Пример этой страницы указан на рисунке 3.

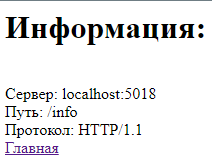


Рисунок 3 – Пример страницы с информацией о запросе клиента

Далее был реализован метод который выводят данные из базы в виде таблицы. Для сравнения ускорения запросов с кэшированием и без была замерена скорость запроса без кэширования, а потом с ним. Пример скорости запроса без кэширования указан на рисунке 4.

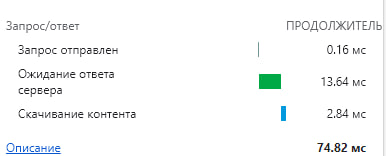


Рисунок 4 – Скорость выполнения запроса без кэширования

Далее была замерена скорость обработки запроса с кэшированием. Пример скорости запроса с кэширования указан на рисунке 5.



Рисунок 5 – Скорость выполнения запроса с кэширования

Далее после получения данных из кэша они передаются в метод для отрисовки таблицы на основе модели. Пример таблицы подписок указана на рисунке 6.

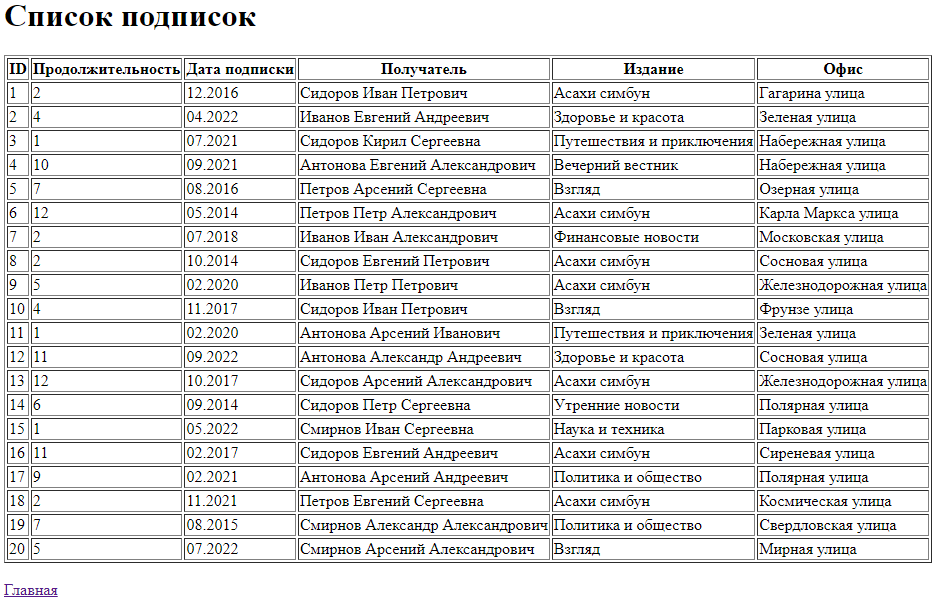


Рисунок 6 – Пример таблицы

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 7.

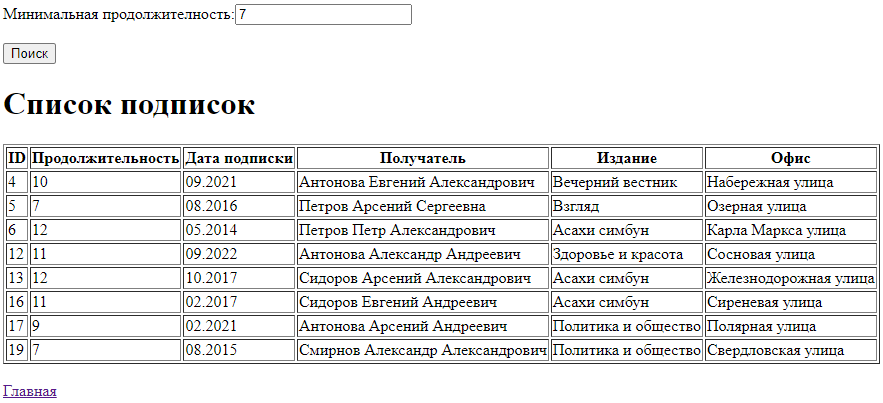


Рисунок 7 – Пример таблицы с фильтрацией с использованием *Sessions*

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 8.

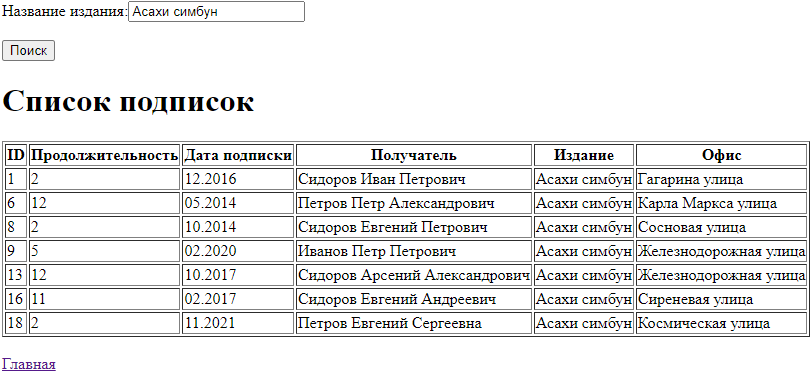


Рисунок 8 – Пример таблицы с фильтрацией с использованием *Cookies*

Пример хранения данных в *Session* и *Cookies* указаны на рисунке 9.

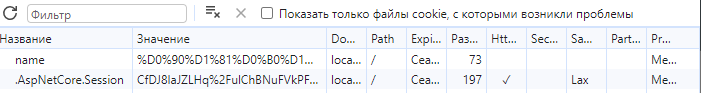


Рисунок 9 – Пример хранения данных в *Session* и *Cookies*

*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 *SubsCityContext*

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

namespace Laba3;

public partial class SubsCityContext : DbContext

{

public SubsCityContext()

{

}

public SubsCityContext(DbContextOptions<SubsCityContext> options)

: base(options)

{

}

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

public virtual DbSet<EmployeePosition> EmployeePositions { get; set; }

public virtual DbSet<Office> Offices { get; set; }

public virtual DbSet<OfficeView> OfficeViews { get; set; }

public virtual DbSet<Publication> Publications { get; set; }

public virtual DbSet<PublicationType> PublicationTypes { get; set; }

public virtual DbSet<PublicationView> PublicationViews { get; set; }

public virtual DbSet<Recipient> Recipients { get; set; }

public virtual DbSet<RecipientAddress> RecipientAddresses { get; set; }

public virtual DbSet<RecipientView> RecipientViews { get; set; }

public virtual DbSet<Subscription> Subscriptions { get; set; }

public virtual DbSet<SubscriptionView> SubscriptionViews { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Employee>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Employee\_\_3213E83FBA716D56");

entity.ToTable("Employee");

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.Middlename)

.HasMaxLength(20)

.HasColumnName("middlename");

entity.Property(e => e.Name)

.HasMaxLength(20)

.HasColumnName("name");

entity.Property(e => e.OfficeId).HasColumnName("office\_id");

entity.Property(e => e.PositionId).HasColumnName("position\_id");

entity.Property(e => e.Surname)

.HasMaxLength(20)

.HasColumnName("surname");

entity.HasOne(d => d.Office).WithMany(p => p.Employees)

.HasForeignKey(d => d.OfficeId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Employee\_\_office\_\_49C3F6B7");

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

.HasForeignKey(d => d.PositionId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Employee\_\_positi\_\_4AB81AF0");

});

modelBuilder.Entity<EmployeePosition>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Employee\_\_3213E83F167E0439");

entity.ToTable("EmployeePosition");

entity.HasIndex(e => e.Position, "UQ\_\_Employee\_\_75FE9D9930B8A4EF").IsUnique();

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.Position)

.HasMaxLength(50)

.HasColumnName("position");

});

modelBuilder.Entity<Office>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Office\_\_3213E83F15657E01");

entity.ToTable("Office");

entity.HasIndex(e => e.MobilePhone, "UQ\_\_Office\_\_3867605B3F8D03F9").IsUnique();

entity.HasIndex(e => e.Email, "UQ\_\_Office\_\_AB6E6164EC42B120").IsUnique();

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.Email)

.HasMaxLength(255)

.HasColumnName("email");

entity.Property(e => e.MobilePhone)

.HasMaxLength(20)

.HasColumnName("mobile\_phone");

entity.Property(e => e.OnwnerSurname)

.HasMaxLength(20)

.HasColumnName("onwner\_surname");

entity.Property(e => e.OwnerMiddlename)

.HasMaxLength(20)

.HasColumnName("owner\_middlename");

entity.Property(e => e.OwnerName)

.HasMaxLength(20)

.HasColumnName("owner\_name");

entity.Property(e => e.StreetName)

.HasMaxLength(50)

.HasColumnName("street\_name");

});

modelBuilder.Entity<OfficeView>(entity =>

{

entity

.HasNoKey()

.ToView("OfficeView");

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

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

entity.Property(e => e.OfficeId)

.ValueGeneratedOnAdd()

.HasColumnName("OfficeID");

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

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

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

entity.Property(e => e.StreetName).HasMaxLength(50);

});

modelBuilder.Entity<Publication>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Publicat\_\_3213E83FBFFED7F5");

entity.ToTable("Publication");

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.Name)

.HasMaxLength(70)

.HasColumnName("name");

entity.Property(e => e.Price)

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

.HasColumnName("price");

entity.Property(e => e.TypeId).HasColumnName("type\_id");

entity.HasOne(d => d.Type).WithMany(p => p.Publications)

.HasForeignKey(d => d.TypeId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Publicati\_\_type\_\_\_3E52440B");

});

modelBuilder.Entity<PublicationType>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Publicat\_\_3213E83FF4660137");

entity.ToTable("PublicationType");

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.Type)

.HasMaxLength(20)

.HasColumnName("type");

});

modelBuilder.Entity<PublicationView>(entity =>

{

entity

.HasNoKey()

.ToView("PublicationView");

entity.Property(e => e.PublicationId).HasColumnName("PublicationID");

entity.Property(e => e.PublicationName).HasMaxLength(70);

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

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

});

modelBuilder.Entity<Recipient>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Recipien\_\_3213E83F8B16D66F");

entity.ToTable("Recipient");

entity.HasIndex(e => e.MobilePhone, "UQ\_\_Recipien\_\_3867605B8CDDE220").IsUnique();

entity.HasIndex(e => e.Email, "UQ\_\_Recipien\_\_AB6E6164F25EBCDE").IsUnique();

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.AddressId).HasColumnName("address\_id");

entity.Property(e => e.Email)

.HasMaxLength(255)

.HasColumnName("email");

entity.Property(e => e.Middlename)

.HasMaxLength(20)

.HasColumnName("middlename");

entity.Property(e => e.MobilePhone)

.HasMaxLength(20)

.HasColumnName("mobile\_phone");

entity.Property(e => e.Name)

.HasMaxLength(20)

.HasColumnName("name");

entity.Property(e => e.Surname)

.HasMaxLength(20)

.HasColumnName("surname");

entity.HasOne(d => d.Address).WithMany(p => p.Recipients)

.HasForeignKey(d => d.AddressId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Recipient\_\_addre\_\_4316F928");

});

modelBuilder.Entity<RecipientAddress>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Recipien\_\_3213E83FEFE7DA13");

entity.ToTable("RecipientAddress");

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.Apartment).HasColumnName("apartment");

entity.Property(e => e.House).HasColumnName("house");

entity.Property(e => e.Street)

.HasMaxLength(50)

.HasColumnName("street");

});

modelBuilder.Entity<RecipientView>(entity =>

{

entity

.HasNoKey()

.ToView("RecipientView");

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

entity.Property(e => e.RecipientId).HasColumnName("RecipientID");

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

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

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

entity.Property(e => e.RecipientStreet).HasMaxLength(50);

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

});

modelBuilder.Entity<Subscription>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Subscrip\_\_3213E83FEB906352");

entity.ToTable("Subscription");

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.Duration).HasColumnName("duration");

entity.Property(e => e.OfficeId).HasColumnName("office\_id");

entity.Property(e => e.PublicationId).HasColumnName("publication\_id");

entity.Property(e => e.RecipientId).HasColumnName("recipient\_id");

entity.Property(e => e.SubscriptionStartDate)

.HasMaxLength(7)

.HasColumnName("subscription\_start\_date");

entity.HasOne(d => d.Office).WithMany(p => p.Subscriptions)

.HasForeignKey(d => d.OfficeId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Subscript\_\_offic\_\_4D94879B");

entity.HasOne(d => d.Publication).WithMany(p => p.Subscriptions)

.HasForeignKey(d => d.PublicationId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Subscript\_\_publi\_\_4E88ABD4");

entity.HasOne(d => d.Recipient).WithMany(p => p.Subscriptions)

.HasForeignKey(d => d.RecipientId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Subscript\_\_recip\_\_4F7CD00D");

});

modelBuilder.Entity<SubscriptionView>(entity =>

{

entity

.HasNoKey()

.ToView("SubscriptionView");

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

entity.Property(e => e.PublicationName).HasMaxLength(70);

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

entity.Property(e => e.SubscriptionId).HasColumnName("SubscriptionID");

entity.Property(e => e.SubscriptionStartDate).HasMaxLength(7);

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

Листинг класса *ICachedSubsCityDb*

namespace Laba3.Services

{

public interface ICachedSubsCityDb

{

void AddOfficeToCache(CacheKey key, int rowsNumber = 100);

IEnumerable<Office> GetOffice(CacheKey key, int rowsNumber = 100);

void AddPuplicationToCache(CacheKey key, int rowsNumber = 100);

IEnumerable<Publication> GetPublication(CacheKey key, int rowsNumber = 100);

void AddRecipientToCache(CacheKey key, int rowsNumber = 100);

IEnumerable<Recipient> GetRecipient(CacheKey key, int rowsNumber = 100);

void AddSubscriptionToCache(CacheKey key, int rowsNumber = 20);

IEnumerable<Subscription> GetSubscription(CacheKey key, int rowsNumber = 20);

}

Листинг класса *CachedSubsCityDb*

using Microsoft.Extensions.Caching.Memory;

using System.Threading.Tasks;

namespace Laba3.Services

{

public class CachedSubsCityDb

{

private readonly SubsCityContext \_dbContext;

private readonly IMemoryCache \_memoryCache;

private readonly int \_saveTime;

public CachedSubsCityDb(SubsCityContext dbContext, IMemoryCache memoryCache)

{

\_dbContext = dbContext;

\_memoryCache = memoryCache;

\_saveTime = 2 \* 2 + 240;

}

public void AddOfficeToCache(CacheKey key, int rowsNumber = 100)

{

if (!\_memoryCache.TryGetValue(key, out IEnumerable<Office> cachedOffice))

{

cachedOffice = \_dbContext.Offices.Take(rowsNumber).ToList();

if (cachedOffice != null)

{

\_memoryCache.Set(key, cachedOffice, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(\_saveTime)

});

}

Console.WriteLine("Таблица Office занесена в кеш");

}

else

{

Console.WriteLine("Таблица Office уже находится в кеше");

}

}

public IEnumerable<Office> GetOffice(CacheKey key, int rowsNumber = 100)

{

IEnumerable<Office> offices;

if (!\_memoryCache.TryGetValue(key, out offices))

{

offices = \_dbContext.Offices.Take(rowsNumber).ToList();

if (offices != null)

{

\_memoryCache.Set(key, offices,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(\_saveTime)));

}

}

return offices;

}

public void AddPuplicationToCache(CacheKey key, int rowsNumber = 100)

{

if (!\_memoryCache.TryGetValue(key, out IEnumerable<Publication> cachedPublication))

{

cachedPublication = \_dbContext.Publications.Take(rowsNumber).ToList();

if (cachedPublication != null)

{

\_memoryCache.Set(key, cachedPublication, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(\_saveTime)

});

}

Console.WriteLine("Таблица Puplication занесена в кеш");

}

else

{

Console.WriteLine("Таблица Puplication уже находится в кеше");

}

}

public IEnumerable<Publication> GetPublication(CacheKey key, int rowsNumber = 100)

{

IEnumerable<Publication> publications;

if (!\_memoryCache.TryGetValue(key, out publications))

{

publications = \_dbContext.Publications.Take(rowsNumber).ToList();

if (publications != null)

{

\_memoryCache.Set(key, publications,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(\_saveTime)));

}

}

return publications;

}

public void AddRecipientToCache(CacheKey key, int rowsNumber = 100)

{

if (!\_memoryCache.TryGetValue(key, out IEnumerable<Recipient> cachedRecipient))

{

cachedRecipient = \_dbContext.Recipients.Take(rowsNumber).ToList();

if (cachedRecipient != null)

{

\_memoryCache.Set(key, cachedRecipient, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(\_saveTime)

});

}

Console.WriteLine("Таблица Recipients занесена в кеш");

}

else

{

Console.WriteLine("Таблица Recipients уже находится в кеше");

}

}

public IEnumerable<Recipient> GetRecipient(CacheKey key, int rowsNumber = 100)

{

IEnumerable<Recipient> recipients;

if (!\_memoryCache.TryGetValue(key, out recipients))

{

recipients = \_dbContext.Recipients.Take(rowsNumber).ToList();

if (recipients != null)

{

\_memoryCache.Set(key, recipients,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(\_saveTime)));

}

}

return recipients;

}

public void AddSubscriptionToCache(CacheKey key, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(key, out IEnumerable<Subscription> cachedSubscription))

{

cachedSubscription = \_dbContext.Subscriptions.Take(rowsNumber).ToList();

if (cachedSubscription != null)

{

\_memoryCache.Set(key, cachedSubscription, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(\_saveTime)

});

}

Console.WriteLine("Таблица Subscription занесена в кеш");

}

else

{

Console.WriteLine("Таблица Subscription уже находится в кеше");

}

}

public IEnumerable<Subscription> GetSubscription(CacheKey key, int rowsNumber = 20)

{

IEnumerable<Subscription> subscriptions;

if (!\_memoryCache.TryGetValue(key, out subscriptions))

{

subscriptions = \_dbContext.Subscriptions.Take(rowsNumber).ToList();

if (subscriptions != null)

{

\_memoryCache.Set(key, subscriptions,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(\_saveTime)));

}

}

return subscriptions;

}

}

}

Листинг класса *TableWriter*

namespace Laba3.Services

{

public class TableWriter

{

public string WriteTable(IEnumerable<Subscription> subscriptions, params object[] addons)

{

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

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

foreach (string addon in addons)

{

HtmlString += addon;

}

HtmlString +="<BODY><H1>Список подписок</H1><TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>ID</TH>";

HtmlString += "<TH>Продолжительность</TH>";

HtmlString += "<TH>Дата подписки</TH>";

HtmlString += "<TH>Получатель</TH>";

HtmlString += "<TH>Издание</TH>";

HtmlString += "<TH>Офис</TH>";

HtmlString += "</TR>";

foreach (var subscription in subscriptions)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + subscription.Id + "</TD>";

HtmlString += "<TD>" + subscription.Duration + "</TD>";

HtmlString += "<TD>" + subscription.SubscriptionStartDate + "</TD>";

HtmlString += "<TD>" + subscription.Recipient.ToString() + "</TD>";

HtmlString += "<TD>" + subscription.Publication.Name + "</TD>";

HtmlString += "<TD>" + subscription.Office.StreetName + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

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

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

return HtmlString;

}

}

}

Листинг класса *Program*

using Laba3;

using Laba3.Services;

using Microsoft.EntityFrameworkCore;

using System.Threading.Tasks;

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<SubsCityContext>(options => options.UseSqlServer(connection));

services.AddMemoryCache();

services.AddDistributedMemoryCache();

services.AddScoped<CachedSubsCityDb>();

services.AddSession();

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession();

var app = builder.Build();

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("/subscription", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

CachedSubsCityDb cachedSubsCityDb = context.RequestServices.GetService<CachedSubsCityDb>();

TableWriter tableWriter = new TableWriter();

IEnumerable<Subscription> subscriptions = cachedSubsCityDb.GetSubscription(CacheKey.Subscription);

string HtmlString = tableWriter.WriteTable(subscriptions);

Console.WriteLine(HtmlString == null);

await context.Response.WriteAsync(HtmlString);

});

});

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

{

appBuilder.Run(async (context) =>

{

CachedSubsCityDb cachedSubsCityDb = context.RequestServices.GetService<CachedSubsCityDb>();

IEnumerable<Subscription> subscriptions = cachedSubsCityDb.GetSubscription(CacheKey.Subscription);

TableWriter tableWriter = new TableWriter();

string formHtml = "<form method='post' action='/searchPublicationName'>" +

"<label for='name'>Название издания:</label>";

if (context.Request.Cookies.TryGetValue("name", out var input\_value))

{

formHtml += $"<input type='text' name='name' value='{input\_value}'><br><br>" +

"<input type='submit' value='Поиск'>" +

"</form>";

}

else

{

formHtml += "<input type='text' name='name'><br><br>" +

"<input type='submit' value='Поиск'>" +

"</form>";

}

if (context.Request.Method == "POST")

{

string name = context.Request.Form["name"];

context.Response.Cookies.Append("name", name);

IEnumerable<Subscription> subscriptionsByPublications = subscriptions.Where(s => s.Publication.Name == name);

string HtmlString = tableWriter.WriteTable(subscriptionsByPublications, formHtml);

await context.Response.WriteAsync(HtmlString);

}

else

{

string HtmlString = tableWriter.WriteTable(subscriptions, formHtml);

await context.Response.WriteAsync(HtmlString);

}

});

});

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

{

appBuilder.Run(async (context) =>

{

CachedSubsCityDb cachedSubsCityDb = context.RequestServices.GetService<CachedSubsCityDb>();

IEnumerable<Subscription> subscriptions = cachedSubsCityDb.GetSubscription(CacheKey.Subscription);

TableWriter tableWriter = new TableWriter();

string formHtml = "<form method='post' action='/searchDuration'>" +

"<label for='name'>Минимальная продолжителность:</label>";

if (context.Session.Keys.Contains("duration"))

{

int duration = Int32.Parse(context.Session.GetString("duration"));

formHtml += $"<input type='number' name='duration' value='{duration}'><br><br>" +

"<input type='submit' value='Поиск'>" +

"</form>";

}

else

{

formHtml += "<input type='number' name='duration'><br><br>" +

"<input type='submit' value='Поиск'>" +

"</form>";

}

if (context.Request.Method == "POST")

{

int duration = Int32.Parse(context.Request.Form["duration"]);

context.Session.SetString("duration", duration.ToString());

IEnumerable<Subscription> subscriptionsByPublications = subscriptions.Where(s => s.Duration >= duration);

string HtmlString = tableWriter.WriteTable(subscriptionsByPublications, formHtml);

await context.Response.WriteAsync(HtmlString);

}

else

{

string HtmlString = tableWriter.WriteTable(subscriptions, formHtml);

await context.Response.WriteAsync(HtmlString);

}

});

});

app.Run((context) =>

{

CachedSubsCityDb cachedSubsCityDb = context.RequestServices.GetService<CachedSubsCityDb>();

cachedSubsCityDb.AddOfficeToCache(CacheKey.Office);

cachedSubsCityDb.AddPuplicationToCache(CacheKey.Publication);

cachedSubsCityDb.AddRecipientToCache(CacheKey.Recipient);

cachedSubsCityDb.AddSubscriptionToCache(CacheKey.Subscription);

string HtmlString = "<HTML><HEAD><TITLE>Емкости</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='/info'>Информация</A></BR>";

HtmlString += "<BR><A href='/subscription'>Все подписки</A></BR>";

HtmlString += "<BR><A href='/searchPublicationName'>Поиск по изданию</A></BR>";

HtmlString += "<BR><A href='/searchDuration'>Поиск по продолжиьельности</A></BR>";

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

return context.Response.WriteAsync(HtmlString);

});

app.Run();

}

}