**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №2

По дисциплине: **«Разработка приложений баз данных для информационных систем»**

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

Выполнил: студент группы ИТП-31

Пронуза М.Ю.

Проверил:

Малиновский И.Л.

Гомель 2024

**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.

⦁ 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 и передавать управление системе маршрутизации MVC фреймворка;

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

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

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

⦁ Разместить выполненный проект в репозитории на GitHub, создать README.md файл.

⦁ Используя средство GitHub Actions, написать рабочий процесс, который будет осуществлять компиляцию проекта под две разные платформы при любом изменении в репозитории.

⦁ Отредактировать 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.

**Ход выполнения результаты работы:**

Текстовый вариант задания для варианта 17 представлен на рисунке 1.

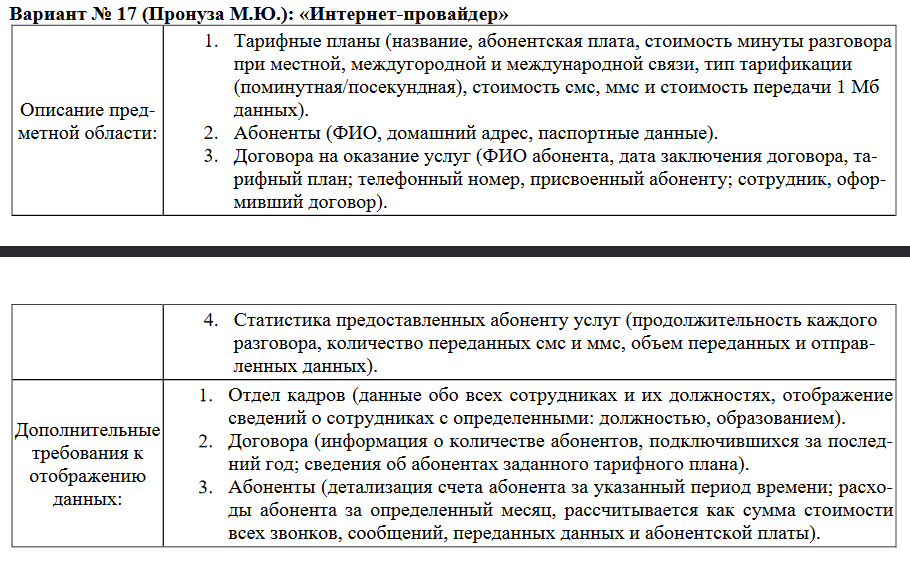


Рисунок 1 – Задание варианта 17

На рисунке 2 представлены кэшированные результаты запроса на выборку таблицы из удаленной базы данных.

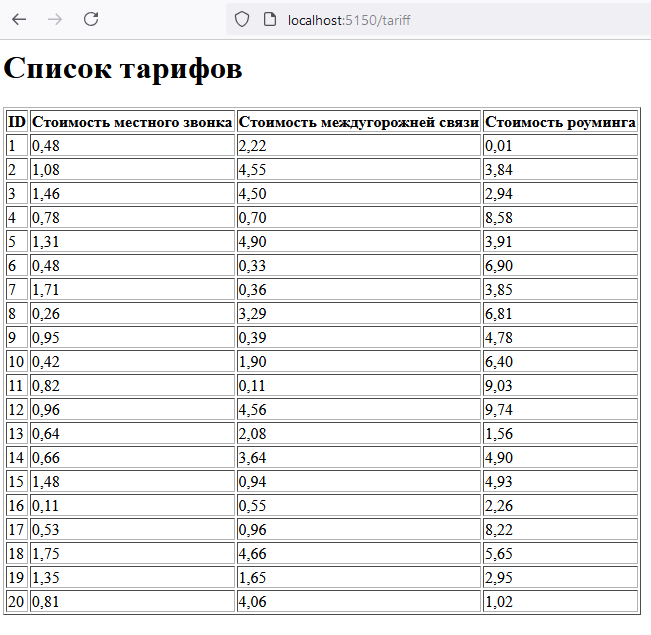


Рисунок 2 – результаты кэшированного запроса

На рисунке 3 представлен результат работы формы, результаты которой сохраняются в сессионное хранилище.

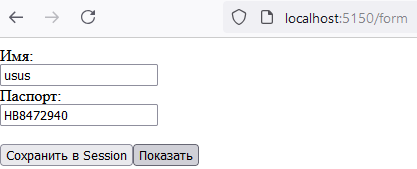


Рисунок 3 – результаты работы формы

На рисунке 4 представлен результат работы формы, результаты которой сохраняются в куки.

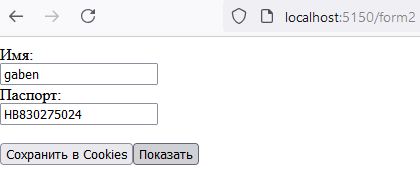


Рисунок 4 – результаты работы формы

**Вывод:** в ходе выполнения лабораторной работы ознакомились c методами обработкой HTTP средствами ASP.NET Core, методами сохранения состояния приложения и повышение производительности приложений путем использования разных видов кэширования.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Program.cs**

using Microsoft.AspNetCore.Http;

using Microsoft.EntityFrameworkCore;

using System.Threading.Tasks;

using TelecomApp.Models;

using TelecomApp.Services;

using AspWeb.Infrastructure;

using Microsoft.AspNetCore.Builder;

using Newtonsoft.Json;

namespace AspWeb

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<Db8328Context>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("TelecomDatabase")));

builder.Services.AddMemoryCache();

// Add services to the container.

builder.Services.AddControllersWithViews();

builder.Services.AddScoped<ICachedDataService, CachedDataService>();

// добавление поддержки сессии

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

}

app.UseStaticFiles();

app.UseSession();

// Вывод информации о клиенте

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

{

appBuilder.Run(async (context) =>

{

// Формирование строки для вывода

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>";

// Вывод данных

await context.Response.WriteAsync(strResponse);

});

});

//Запоминание в Session значений, введенных в форме

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

{

appBuilder.Run(async (context) =>

{

// Считывание из Session объекта User

Subscriber user = context.Session.Get<Subscriber>("user") ?? new Subscriber();

// Формирование строки для вывода динамической HTML формы

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

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

"<BODY><FORM action ='/form' method='POST'>" + // Возвращаем '/form'

"Имя:<BR><INPUT type = 'text' name = 'FullName' value = '" + user.FullName + "'>" +

"<BR>Паспорт:<BR><INPUT type = 'text' name = 'PassportData' value = '" + user.PassportData + "' >" +

"<BR><BR><INPUT type ='submit' value='Сохранить в Session'><INPUT type ='submit' value='Показать'></FORM>";

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

// Проверка на метод запроса

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

{

// Запись в Session данных объекта User

user.FullName = context.Request.Form["FullName"];

user.PassportData = context.Request.Form["PassportData"];

context.Session.Set<Subscriber>("user", user);

}

// Асинхронный вывод динамической HTML формы

await context.Response.WriteAsync(strResponse);

});

});

// Запоминание в Cookies значений, введенных в форме

app.Map("/form2", async context =>

{

// Считывание из куки объекта User

var cookieValue = context.Request.Cookies["user"];

Subscriber user = cookieValue != null ? JsonConvert.DeserializeObject<Subscriber>(cookieValue) : new Subscriber();

// Формирование строки для вывода динамической HTML формы

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

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

"<BODY><FORM action='/form2' method='POST'>" +

$"Имя:<BR><INPUT type='text' name='FullName' value='{user.FullName}'>" +

$"<BR>Паспорт:<BR><INPUT type='text' name='PassportData' value='{user.PassportData}' >" +

"<BR><BR><INPUT type='submit' value='Сохранить в Cookies'><INPUT type='submit' value='Показать'></FORM>";

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

// Проверка на метод запроса

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

{

// Запись в куки данных объекта User

user.FullName = context.Request.Form["FullName"];

user.PassportData = context.Request.Form["PassportData"];

// Сериализация объекта Subscriber в JSON и сохранение в куки

var options = new CookieOptions

{

Expires = DateTimeOffset.UtcNow.AddMinutes(30) // Устанавливаем срок действия куки

};

context.Response.Cookies.Append("user", JsonConvert.SerializeObject(user), options);

}

// Асинхронный вывод динамической HTML формы

await context.Response.WriteAsync(strResponse);

});

// Вывод кэшированной информации из таблицы базы данных

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

{

appBuilder.Run(async (context) =>

{

var cachedDataService = context.RequestServices.GetService<ICachedDataService>();

IEnumerable<TariffPlan> tariff = cachedDataService.GetTariffPlans("Tariff20");

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

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

"<BODY><H1>Список тарифов</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

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

HtmlString += "<TH>Стоимость местного звонка</TH>";

HtmlString += "<TH>Стоимость междугорожней связи</TH>";

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

HtmlString += "</TR>";

foreach (var tar in tariff)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + tar.TariffPlanId + "</TD>";

HtmlString += "<TD>" + tar.LocalCallRate + "</TD>";

HtmlString += "<TD>" + tar.LongDistanceCallRate + "</TD>";

HtmlString += "<TD>" + tar.InternationalCallRate + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

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

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

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

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

// Вывод данных

await context.Response.WriteAsync(HtmlString);

});

});

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.Run();

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Db8328Context.cs**

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

namespace TelecomApp.Models;

public partial class Db8328Context : DbContext

{

public Db8328Context()

{

}

public Db8328Context(DbContextOptions<Db8328Context> options)

: base(options)

{

}

public virtual DbSet<Call> Calls { get; set; }

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

public virtual DbSet<ContractCall> ContractCalls { get; set; }

public virtual DbSet<ContractInternetUsage> ContractInternetUsages { get; set; }

public virtual DbSet<ContractMessage> ContractMessages { get; set; }

public virtual DbSet<EmployeeInfo> EmployeeInfos { get; set; }

public virtual DbSet<InternetUsage> InternetUsages { get; set; }

public virtual DbSet<Message> Messages { get; set; }

public virtual DbSet<Staff> Staff { get; set; }

public virtual DbSet<StaffPosition> StaffPositions { get; set; }

public virtual DbSet<Subscriber> Subscribers { get; set; }

public virtual DbSet<SubscriberInfo> SubscriberInfos { get; set; }

public virtual DbSet<TariffPlan> TariffPlans { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

// Загрузка конфигурации из файла appsettings.json

var configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build();

// Получение строки подключения

var connectionString = configuration.GetConnectionString("TelecomDatabase");

optionsBuilder.UseSqlServer(connectionString);

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.UseCollation("Cyrillic\_General\_CI\_AS");

modelBuilder.Entity<Call>(entity =>

{

entity.HasKey(e => e.CallId).HasName("PK\_\_Calls\_\_5180CF8A4C1A2D2C");

entity.Property(e => e.CallId).HasColumnName("CallID");

entity.Property(e => e.CallDate).HasColumnType("datetime");

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

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

.HasForeignKey(d => d.ContractId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Calls\_\_ContractI\_\_3A4CA8FD");

});

modelBuilder.Entity<Contract>(entity =>

{

entity.HasKey(e => e.ContractId).HasName("PK\_\_Contract\_\_C90D34098518ECB7");

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

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

entity.Property(e => e.StaffId).HasColumnName("StaffID");

entity.Property(e => e.SubscriberId).HasColumnName("SubscriberID");

entity.Property(e => e.TariffPlanId).HasColumnName("TariffPlanID");

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

.HasForeignKey(d => d.StaffId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Contracts\_\_Staff\_\_37703C52");

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

.HasForeignKey(d => d.SubscriberId)

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK\_\_Contracts\_\_Subsc\_\_3587F3E0");

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

.HasForeignKey(d => d.TariffPlanId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Contracts\_\_Tarif\_\_367C1819");

});

modelBuilder.Entity<ContractCall>(entity =>

{

entity

.HasNoKey()

.ToView("ContractCalls");

entity.Property(e => e.CallDate).HasColumnType("datetime");

entity.Property(e => e.CallId).HasColumnName("CallID");

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

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

});

modelBuilder.Entity<ContractInternetUsage>(entity =>

{

entity

.HasNoKey()

.ToView("ContractInternetUsage");

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

entity.Property(e => e.DataReceivedMb)

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

.HasColumnName("DataReceivedMB");

entity.Property(e => e.DataSentMb)

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

.HasColumnName("DataSentMB");

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

entity.Property(e => e.UsageDate).HasColumnType("datetime");

entity.Property(e => e.UsageId).HasColumnName("UsageID");

});

modelBuilder.Entity<ContractMessage>(entity =>

{

entity

.HasNoKey()

.ToView("ContractMessages");

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

entity.Property(e => e.IsMms).HasColumnName("IsMMS");

entity.Property(e => e.MessageDate).HasColumnType("datetime");

entity.Property(e => e.MessageId).HasColumnName("MessageID");

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

});

modelBuilder.Entity<EmployeeInfo>(entity =>

{

entity

.HasNoKey()

.ToView("EmployeeInfo");

entity.Property(e => e.Education).HasMaxLength(100);

entity.Property(e => e.FullName).HasMaxLength(150);

entity.Property(e => e.PositionName).HasMaxLength(100);

entity.Property(e => e.StaffId).HasColumnName("StaffID");

});

modelBuilder.Entity<InternetUsage>(entity =>

{

entity.HasKey(e => e.UsageId).HasName("PK\_\_Internet\_\_29B197C071A2003F");

entity.ToTable("InternetUsage");

entity.Property(e => e.UsageId).HasColumnName("UsageID");

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

entity.Property(e => e.DataReceivedMb)

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

.HasColumnName("DataReceivedMB");

entity.Property(e => e.DataSentMb)

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

.HasColumnName("DataSentMB");

entity.Property(e => e.UsageDate).HasColumnType("datetime");

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

.HasForeignKey(d => d.ContractId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_InternetU\_\_Contr\_\_40058253");

});

modelBuilder.Entity<Message>(entity =>

{

entity.HasKey(e => e.MessageId).HasName("PK\_\_Messages\_\_C87C037CEF80E5DC");

entity.Property(e => e.MessageId).HasColumnName("MessageID");

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

entity.Property(e => e.IsMms).HasColumnName("IsMMS");

entity.Property(e => e.MessageDate).HasColumnType("datetime");

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

.HasForeignKey(d => d.ContractId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Messages\_\_Contra\_\_3D2915A8");

});

modelBuilder.Entity<Staff>(entity =>

{

entity.HasKey(e => e.StaffId).HasName("PK\_\_Staff\_\_96D4AAF7C1830A57");

entity.Property(e => e.StaffId).HasColumnName("StaffID");

entity.Property(e => e.Education).HasMaxLength(100);

entity.Property(e => e.FullName).HasMaxLength(150);

entity.Property(e => e.PositionId).HasColumnName("PositionID");

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

.HasForeignKey(d => d.PositionId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK\_\_Staff\_\_PositionI\_\_2EDAF651");

});

modelBuilder.Entity<StaffPosition>(entity =>

{

entity.HasKey(e => e.PositionId).HasName("PK\_\_StaffPos\_\_60BB9A59C4E44263");

entity.ToTable("StaffPosition");

entity.Property(e => e.PositionId).HasColumnName("PositionID");

entity.Property(e => e.PositionName).HasMaxLength(100);

});

modelBuilder.Entity<Subscriber>(entity =>

{

entity.HasKey(e => e.SubscriberId).HasName("PK\_\_Subscrib\_\_7DFEB634E4D583A6");

entity.Property(e => e.SubscriberId).HasColumnName("SubscriberID");

entity.Property(e => e.FullName).HasMaxLength(150);

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

entity.Property(e => e.PassportData).HasMaxLength(100);

});

modelBuilder.Entity<SubscriberInfo>(entity =>

{

entity

.HasNoKey()

.ToView("SubscriberInfo");

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

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

entity.Property(e => e.PassportData).HasMaxLength(100);

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

entity.Property(e => e.SubscriberFullName).HasMaxLength(150);

entity.Property(e => e.SubscriberId).HasColumnName("SubscriberID");

entity.Property(e => e.TariffName).HasMaxLength(100);

});

modelBuilder.Entity<TariffPlan>(entity =>

{

entity.HasKey(e => e.TariffPlanId).HasName("PK\_\_TariffPl\_\_29A9282AA46A8AB6");

entity.Property(e => e.TariffPlanId).HasColumnName("TariffPlanID");

entity.Property(e => e.DataRatePerMb)

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

.HasColumnName("DataRatePerMB");

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

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

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

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

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

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

entity.Property(e => e.TariffName).HasMaxLength(100);

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

**ПРИЛОЖЕНИЕ В**

(обязательное)

**SessionExtensions.cs**

using Microsoft.AspNetCore.Http;

using Newtonsoft.Json;

namespace AspWeb.Infrastructure

{

// Методы расширения для ISession для работы с произвольными объектами

public static class SessionExtensions

{

//Запись произвольного объекта в сессию

public static void Set<T>(this ISession session, string key, T value)

{

session.SetString(key, JsonConvert.SerializeObject(value));

}

//Считывание произвольного 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);

}

}

}

**ПРИЛОЖЕНИЕ Г**

(обязательное)

**CachedDataService.cs**

using Microsoft.Extensions.Caching.Memory;

using TelecomApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

namespace TelecomApp.Services

{

public class CachedDataService : ICachedDataService

{

private readonly Db8328Context \_dbContext;

private readonly IMemoryCache \_memoryCache;

public CachedDataService(Db8328Context dbContext, IMemoryCache memoryCache)

{

\_dbContext = dbContext;

\_memoryCache = memoryCache;

}

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

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Contract> contracts))

{

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

\_memoryCache.Set(cacheKey, contracts, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return contracts;

}

public IEnumerable<InternetUsage> GetInternetUsages(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<InternetUsage> internetUsages))

{

internetUsages = \_dbContext.InternetUsages.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, internetUsages, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return internetUsages;

}

public IEnumerable<Message> GetMessages(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Message> messages))

{

messages = \_dbContext.Messages.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, messages, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return messages;

}

public IEnumerable<Call> GetCalls(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Call> calls))

{

calls = \_dbContext.Calls.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, calls, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return calls;

}

public IEnumerable<Staff> GetStaff(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Staff> staff))

{

staff = \_dbContext.Staff.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, staff, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return staff;

}

public IEnumerable<StaffPosition> GetStaffPositions(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<StaffPosition> staffPositions))

{

staffPositions = \_dbContext.StaffPositions.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, staffPositions, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return staffPositions;

}

public IEnumerable<Subscriber> GetSubscribers(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Subscriber> subscribers))

{

subscribers = \_dbContext.Subscribers.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, subscribers, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return subscribers;

}

public IEnumerable<TariffPlan> GetTariffPlans(string cacheKey, int rowsNumber = 20)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<TariffPlan> tariffPlans))

{

tariffPlans = \_dbContext.TariffPlans.Take(rowsNumber).ToList();

\_memoryCache.Set(cacheKey, tariffPlans, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(282)

});

}

return tariffPlans;

}

}

}

**ПРИЛОЖЕНИЕ Д**

(обязательное)

**ICachedDataService.cs**

using TelecomApp.Models;

using System.Collections.Generic;

namespace TelecomApp.Services

{

public interface ICachedDataService

{

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

IEnumerable<InternetUsage> GetInternetUsages(string cacheKey, int rowsNumber = 20);

IEnumerable<Message> GetMessages(string cacheKey, int rowsNumber = 20);

IEnumerable<Call> GetCalls(string cacheKey, int rowsNumber = 20);

IEnumerable<Staff> GetStaff(string cacheKey, int rowsNumber = 20);

IEnumerable<StaffPosition> GetStaffPositions(string cacheKey, int rowsNumber = 20);

IEnumerable<Subscriber> GetSubscribers(string cacheKey, int rowsNumber = 20);

IEnumerable<TariffPlan> GetTariffPlans(string cacheKey, int rowsNumber = 20);

}

}