**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №4

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

на тему: **Разработка моделей и контроллеров ASP.NET MVC приложения баз данных**

Выполнил: студент группы ИТП-31

Пронуза М.Ю.

Проверил:

Малиновский И.Л.

Гомель 2024

**Цель работы**: ознакомиться с возможностями ASP.NET Core MVC и Entity Framework Core для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.

**Задание:**

Создать с использованием ASP.NET Core MVC Web-приложение, содержащее набор классов, моделирующих предметную область, и осуществляющих генерацию и заполнение тестовыми наборами записей базу данных. Разработать один компонент middleware, контроллеры и представления для выборки и отображения информации из не менее чем 3- таблиц базы данных с использованием механизма внедрение зависимостей.

Для выполнения задания необходимо создать:

⦁ Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Перечень таблиц предварительно согласовывается с преподавателем. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.

⦁ Класс контекста данных.

⦁ Другие классы, например классы View Model и т.п. (при необходимости).

⦁ Компонент middleware для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.

⦁ Классы контроллеров (по одному на каждую таблицу базы данных) для обработки обращений пользователя, выборки данных из таблиц и вызова соответствующих представлений для отображения выбранных данных.

⦁ Разработать представления для отображения данных из таблиц, выбранных контроллерами. Представления, работающими с таблицами, стоящими на стороне отношения «многие» в схеме базы данных, должны выводить вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».

⦁ Используя предварительно созданный и сконфигурированный профиль кэширования, подключить кэширование вывода для страниц с использованием атрибута ResponseCache для соответствующих методов контроллера. Данные в кэше хранить неизменными в течение 2\*N+240 секунд, где N- номер вашего варианта.

⦁ С использованием средств разработчика браузера (Chrome, Firefox) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием атрибута ResponseCache.

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.

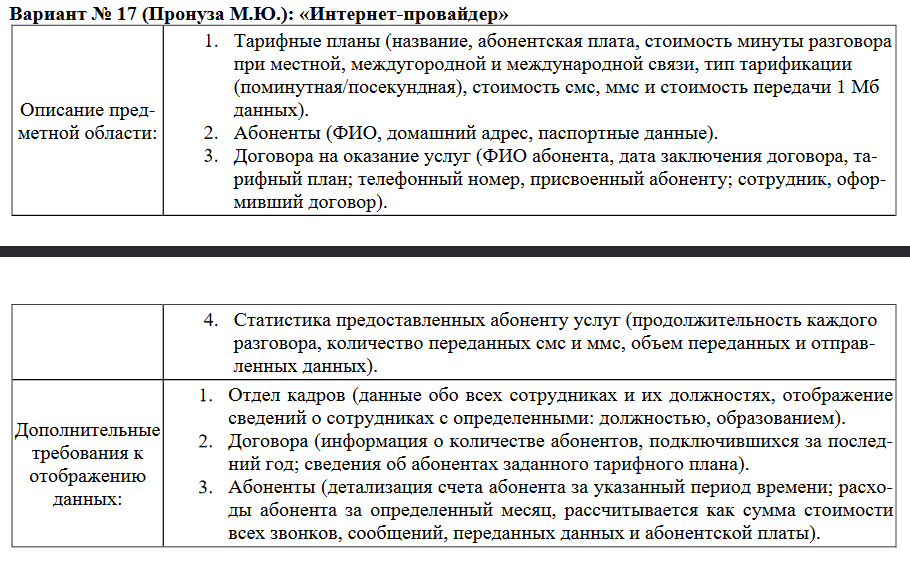


Рисунок 1 – Задание варианта 17

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

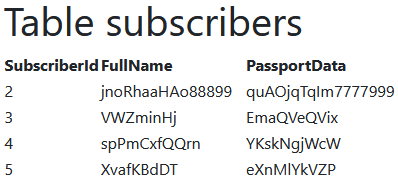


Рисунок 2 – результаты кэшированного запроса

На рисунке 3 представлены кэшированные результаты запроса на выборку таблицы на стороне многие из удаленной базы данных.

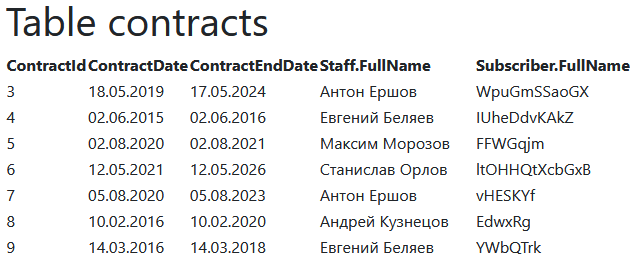


Рисунок 3 – выборка из таблицы на стороне многи

На рисунке 4 представлен результат добавления заголовков кэширования.



Рисунок 4 – заголовки кэшировани

**Вывод:** в ходе выполнения лабораторной работы ознакомились с возможностями ASP.NET Core MVC и Entity Framework Core для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Program.cs**

using Microsoft.EntityFrameworkCore;

using TelecomWeb.Middleware;

using TelecomWeb.Models;

using TelecomWeb.Services;

namespace TelecomWeb

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

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

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

builder.Services.AddMemoryCache();

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

// добавление поддержки сессии

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession();

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

// Add services to the container.

builder.Services.AddControllersWithViews();

var app = builder.Build();

app.UseSession();

app.UseDbInitializer();

app.UseResponseCaching();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

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

}

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.Run();

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**TelecomDbContext.cs**

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

namespace TelecomWeb.Models;

public partial class TelecomDbContext : DbContext

{

public TelecomDbContext()

{

}

public TelecomDbContext(DbContextOptions<TelecomDbContext> 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)

#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 https://go.microsoft.com/fwlink/?LinkId=723263.

=> optionsBuilder.UseSqlServer("Server=db8328.public.databaseasp.net; Database=db8328; User Id=db8328; Password=5z?X=4Mb3m-Q; Encrypt=True; TrustServerCertificate=True; MultipleActiveResultSets=True;");

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

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

modelBuilder.Entity<Call>(entity =>

{

entity.HasKey(e => e.CallId).HasName("PK\_\_Calls\_\_5180CF8A3D2D2ED2");

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)

.HasConstraintName("FK\_\_Calls\_\_ContractI\_\_14B10FFA");

});

modelBuilder.Entity<Contract>(entity =>

{

entity.HasKey(e => e.ContractId).HasName("PK\_\_Contract\_\_C90D34095C346A02");

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)

.HasConstraintName("FK\_\_Contracts\_\_Staff\_\_11D4A34F");

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

.HasForeignKey(d => d.SubscriberId)

.HasConstraintName("FK\_\_Contracts\_\_Subsc\_\_0FEC5ADD");

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

.HasForeignKey(d => d.TariffPlanId)

.HasConstraintName("FK\_\_Contracts\_\_Tarif\_\_10E07F16");

});

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\_\_29B197C0F4400AED");

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)

.HasConstraintName("FK\_\_InternetU\_\_Contr\_\_1A69E950");

});

modelBuilder.Entity<Message>(entity =>

{

entity.HasKey(e => e.MessageId).HasName("PK\_\_Messages\_\_C87C037C0167DB96");

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)

.HasConstraintName("FK\_\_Messages\_\_Contra\_\_178D7CA5");

});

modelBuilder.Entity<Staff>(entity =>

{

entity.HasKey(e => e.StaffId).HasName("PK\_\_Staff\_\_96D4AAF7835BD125");

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)

.HasConstraintName("FK\_\_Staff\_\_PositionI\_\_093F5D4E");

});

modelBuilder.Entity<StaffPosition>(entity =>

{

entity.HasKey(e => e.PositionId).HasName("PK\_\_StaffPos\_\_60BB9A59D4D3AD5C");

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\_\_7DFEB634B3799F70");

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\_\_29A9282A1C614268");

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);

}

}

**ПРИЛОЖЕНИЕ Е**

(обязательное)

**DbInitializerMiddleware.cs**

using TelecomWeb.Data;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using TelecomWeb.Models;

namespace TelecomWeb.Middleware

{

public class DbInitializerMiddleware

{

private readonly RequestDelegate \_next;

public DbInitializerMiddleware(RequestDelegate next)

{

\_next = next;

}

public async Task Invoke(HttpContext context, TelecomDbContext dbContext)

{

var hasSubscribers = dbContext.Subscribers.Any();

if (!hasSubscribers)

{

DbInitializer.Initialize(dbContext);

}

await \_next(context);

}

}

public static class DbInitializerExtensions

{

public static IApplicationBuilder UseDbInitializer(this IApplicationBuilder builder)

{

return builder.UseMiddleware<DbInitializerMiddleware>();

}

}

}

**ПРИЛОЖЕНИЕ Ё**

(обязательное)

**DbInitializer.cs**

using TelecomWeb.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Linq;

namespace TelecomWeb.Data

{

public static class DbInitializer

{

public static void Initialize(TelecomDbContext db)

{

db.Database.EnsureCreated();

string sqlFilePath = "sql/insert.sql";

if (File.Exists(sqlFilePath))

{

string sql = File.ReadAllText(sqlFilePath);

db.Database.ExecuteSqlRaw(sql);

}

else

{

Console.WriteLine($"Файл не найден: {sqlFilePath}");

}

}

}

}