**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: **Использование ENTITY FRAMEWORK и LINQ для работы с базами данных**

Выполнил: студент группы ИТП-31

Пронуза М.Ю.

Проверил:

Малиновский И.Л.

Гомель 2024

**Цель работы**: Ознакомиться с возможностями ENTITY FRAMEWORK и получить навыки написания LINQ запросов к объектам, связанным с таблицами базы данных СУБД MS SQL Server.

**Задание:**

1. Создать, с использованием .NET Entity Framework Core консольное приложение, содержащее набор классов, моделирующих соответствующей своему варианту предметную область, на основе созданной в лабораторной работе № 1 базой данных MS SQL Server.

Для этого необходимо, используя подход Database First и средство Scaffolding (Reverse Engineering) Entity Framework Core, создать классы, моделирующие таблицы базы данных, и класс контекста данных.

2. Выполнить, используя объекты Entity Framework Core и LINQ:

⦁ Выборку всех данных из таблицы, стоящей в схеме базы данных на стороне отношения «один» – 1 шт.

⦁ Выборку данных из таблицы, стоящей в схеме базы данных на стороне отношения «один», отфильтрованные по определенному условию, налагающему ограничения на одно или несколько полей – 1 шт.

⦁ Выборку данных, сгруппированных по любому из полей данных с выводом какого-либо итогового результата (min, max, avg, сount или др.) по выбранному полю из таблицы, стоящей в схеме базы данных на стороне отношения «многие» – 1 шт.

⦁ Выборку данных из двух полей двух таблиц, связанных между собой отношением «один-ко-многим» – 1 шт.

⦁ Выборку данных из двух таблиц, связанных между собой отношением «один-ко-многим» и отфильтрованным по некоторому условию, налагающему ограничения на значения одного или нескольких полей – 1 шт.

⦁ Вставку данных в таблицы, стоящей на стороне отношения «Один» – 1 шт.

⦁ Вставку данных в таблицы, стоящей на стороне отношения «Многие» – 1 шт.:

⦁ Удаление данных из таблицы, стоящей на стороне отношения «Один» – 1 шт.

⦁ Удаление данных из таблицы, стоящей на стороне отношения «Многие» – 1 шт.

⦁ Обновление удовлетворяющих определенному условию записей в любой из таблиц базы данных – 1 шт.

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

4. 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.

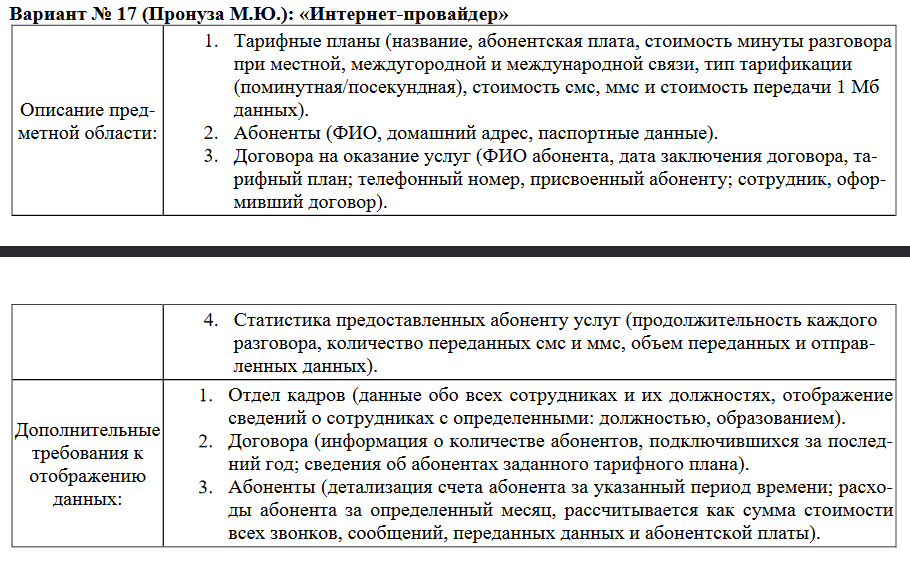


Рисунок 1 – Задание варианта 17

На рисунке 2 представлены вывод результатов работы приложения.

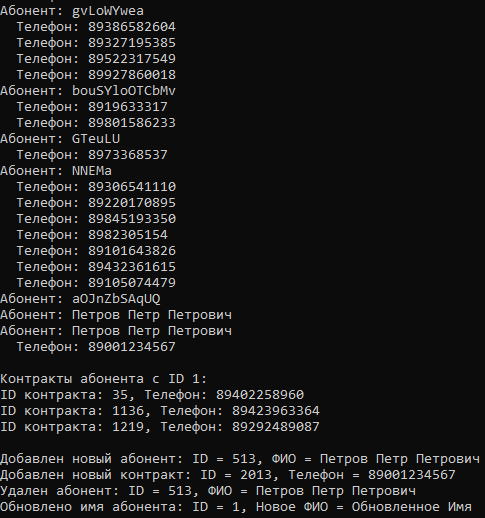


Рисунок 2 – результат работы приложения

На рисунке 3 содержание файла *README.md* на *Github*.

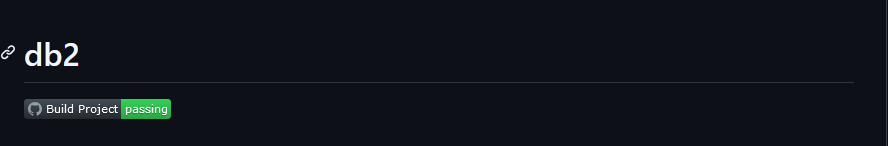


Рисунок 3 – Описание на *Github*

На рисунке 4 представлены результаты работы *github actions*.

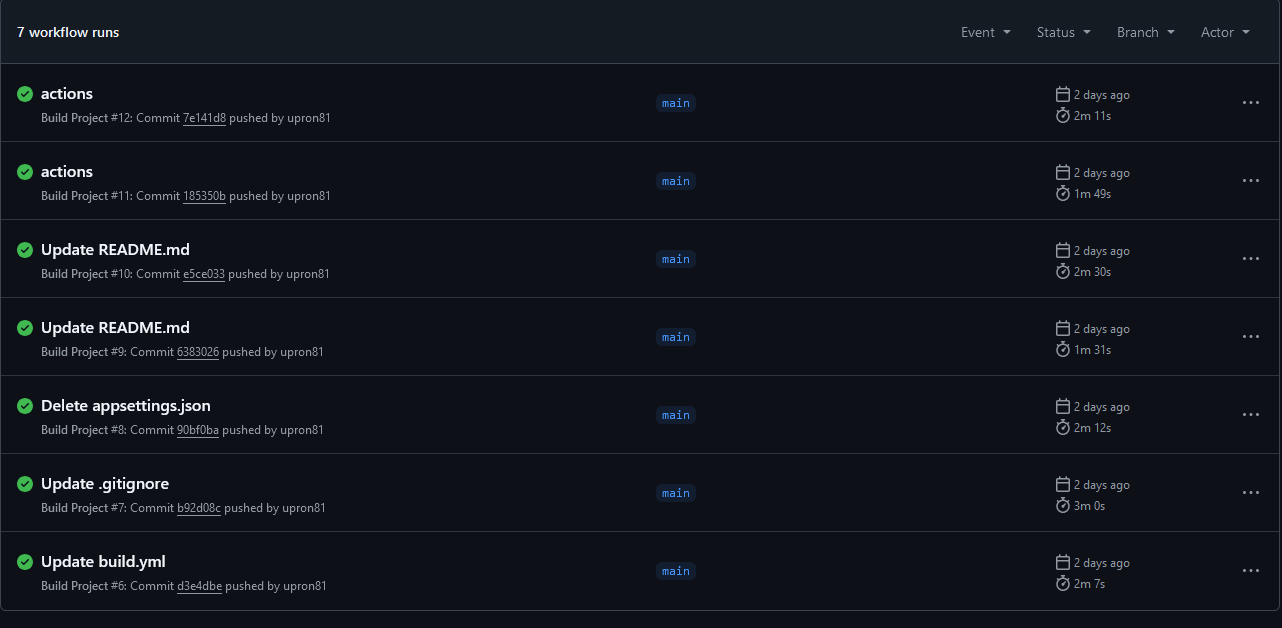


Рисунок 4 – Результаты *github actions*

**Вывод:** в ходе выполнения лабораторной работы научились работать с возможностями ENTITY FRAMEWORK и получить навыки написания LINQ запросов к объектам, связанным с таблицами базы данных СУБД MS SQL Server.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Program.cs**

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using TelecomApp.Models;

using System;

using System.Linq;

class Program

{

static void Main(string[] args)

{

var configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build();

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

var optionsBuilder = new DbContextOptionsBuilder<Db8328Context>();

optionsBuilder.UseSqlServer(connectionString);

using (var context = new Db8328Context(optionsBuilder.Options))

{

var subscribers = context.Subscribers.ToList();

Console.WriteLine("Все абоненты:");

foreach (var subscriber in subscribers)

{

Console.WriteLine($"ID: {subscriber.SubscriberId}, ФИО: {subscriber.FullName}");

}

var filteredSubscribers = context.Subscribers

.Where(s => s.HomeAddress.Contains("Москва"))

.ToList();

Console.WriteLine("\nАбоненты с адресом, содержащим 'Москва':");

foreach (var subscriber in filteredSubscribers)

{

Console.WriteLine($"ID: {subscriber.SubscriberId}, ФИО: {subscriber.FullName}, Адрес: {subscriber.HomeAddress}");

}

var groupedSubscribers = context.Contracts

.GroupBy(c => c.TariffPlanId)

.Select(g => new

{

TariffPlanId = g.Key,

Count = g.Count()

}).ToList();

Console.WriteLine("\nКоличество контрактов по тарифным планам:");

foreach (var group in groupedSubscribers)

{

Console.WriteLine($"Тарифный план ID: {group.TariffPlanId}, Количество контрактов: {group.Count}");

}

var subscribersWithContracts = context.Subscribers

.Include(s => s.Contracts)

.Select(s => new

{

SubscriberName = s.FullName,

Contracts = s.Contracts.Select(c => c.PhoneNumber)

}).ToList();

Console.WriteLine("\nАбоненты и их контракты:");

foreach (var subscriber in subscribersWithContracts)

{

Console.WriteLine($"Абонент: {subscriber.SubscriberName}");

foreach (var contract in subscriber.Contracts)

{

Console.WriteLine($" Телефон: {contract}");

}

}

var filteredContracts = context.Contracts

.Where(c => c.SubscriberId == 1)

.Include(c => c.Subscriber)

.ToList();

Console.WriteLine("\nКонтракты абонента с ID 1:");

foreach (var contract in filteredContracts)

{

Console.WriteLine($"ID контракта: {contract.ContractId}, Телефон: {contract.PhoneNumber}");

}

var newSubscriber = new Subscriber

{

FullName = "Петров Петр Петрович",

HomeAddress = "Улица Пушкина, дом 10",

PassportData = "1234 567890"

};

context.Subscribers.Add(newSubscriber);

context.SaveChanges();

Console.WriteLine($"\nДобавлен новый абонент: ID = {newSubscriber.SubscriberId}, ФИО = {newSubscriber.FullName}");

var existingStaff = context.Staff.FirstOrDefault();

if (existingStaff == null)

{

Console.WriteLine("Нет доступных сотрудников для связывания с контрактом.");

return;

}

var newContract = new Contract

{

SubscriberId = newSubscriber.SubscriberId,

TariffPlanId = 1,

ContractDate = DateTime.Now,

PhoneNumber = "89001234567",

StaffId = existingStaff.StaffId

};

context.Contracts.Add(newContract);

context.SaveChanges();

Console.WriteLine($"Добавлен новый контракт: ID = {newContract.ContractId}, Телефон = {newContract.PhoneNumber}");

var subscriberToRemove = context.Subscribers.Find(newSubscriber.SubscriberId);

if (subscriberToRemove != null)

{

context.Subscribers.Remove(subscriberToRemove);

context.SaveChanges();

Console.WriteLine($"Удален абонент: ID = {subscriberToRemove.SubscriberId}, ФИО = {subscriberToRemove.FullName}");

}

var contractToRemove = context.Contracts.Find(newContract.ContractId);

if (contractToRemove != null)

{

context.Contracts.Remove(contractToRemove);

context.SaveChanges();

Console.WriteLine($"Удален контракт: ID = {contractToRemove.ContractId}, Телефон = {contractToRemove.PhoneNumber}");

}

var subscriberToUpdate = context.Subscribers

.FirstOrDefault(s => s.SubscriberId == 1);

if (subscriberToUpdate != null)

{

subscriberToUpdate.FullName = "Обновленное Имя";

context.SaveChanges();

Console.WriteLine($"Обновлено имя абонента: ID = {subscriberToUpdate.SubscriberId}, Новое ФИО = {subscriberToUpdate.FullName}");

}

}

}

}

**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);

}