Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5

Управление сущностями

тема

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | А.К. Погребников |
|  | подпись, дата | инициалы, фамилия |
| Студент КИ18-16б 031831229 |  | В.А. Прекель |
| номер группы, зачетной книжки | подпись, дата | инициалы, фамилия |

Красноярск 2020

# 1 Цель работы

Настроить ORM.

# 2 Общая постановка задачи

В рамках данной практической работы необходимо реализовать модель данных на программной стороне для мапинга с таблицами БД.

# 3 Ход работы

Используется Entity Framework Core. Модели были реализованы в практичекой работе 2.

Листинг 1 – MyStore/MyStore.EfCore/Program.cs

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
using MyStore.Data;  
using MyStore.Data.Entity;  
  
Console.OutputEncoding = Encoding.UTF8;  
  
using var context = new Context();  
  
var name = "Юлия";  
var g = context.SupportQuestions  
 .Join(context.SupportTickets, question => question.SupportTicketId, ticket => ticket.SupportTicketId,  
 (question, ticket) => new {question, ticket})  
 .Join(context.Customers, arg => arg.ticket.CustomerId, customer => customer.CustomerId,  
 (arg, customer) => new {arg.question, arg.ticket, customer})  
 .Where(arg => arg.customer.FirstName == name)  
 .Select(arg => new  
 {  
 arg.question.SupportTicketId,  
 IsQuestion = true,  
 arg.question.SendTimestamp,  
 ReadTimestamp = arg.question.ReadTimestamp ?? DateTimeOffset.MinValue,  
 arg.customer.FirstName,  
 arg.customer.LastName,  
 arg.question.Text  
 }).Union(context.SupportAnswers  
 .Join(context.SupportOperators, answer => answer.SupportOperatorId,  
 operator\_ => operator\_.SupportOperatorId,  
 (answer, operator\_) => new {answer, operator\_})  
 .Where(arg => arg.operator\_.FirstName == name)  
 .Select(arg => new  
 {  
 arg.answer.SupportTicketId,  
 IsQuestion = false,  
 arg.answer.SendTimestamp,  
 ReadTimestamp = DateTimeOffset.MinValue,  
 arg.operator\_.FirstName,  
 arg.operator\_.LastName,  
 arg.answer.Text  
 }))  
 .OrderBy(arg => arg.SupportTicketId)  
 .ThenBy(arg => arg.SendTimestamp);  
foreach (var i in g)  
{  
 Console.WriteLine(i);  
}  
Console.WriteLine();  
  
string GetCustomerFirstName(int id)  
{  
 return context.Customers.First(customer => customer.CustomerId == id).FirstName;  
}  
  
Console.WriteLine("Имя покупателя с id=2");  
Console.WriteLine(GetCustomerFirstName(2));  
Console.WriteLine();  
  
IEnumerable<string> GetCustomerFirstNames(int startId, int endId)  
{  
 return context.Customers  
 .Where(customer => startId <= customer.CustomerId && customer.CustomerId <= endId)  
 .Select(customer => customer.FirstName);  
}  
  
Console.WriteLine("Имена покупателей с id от 4 по 9");  
foreach (var i in GetCustomerFirstNames(4, 9))  
{  
 Console.WriteLine(i);  
}  
Console.WriteLine();  
  
int AddCustomer(Customer customer)  
{  
 var c1 = context.Customers.Add(customer);  
 context.SaveChanges();  
 return c1.Entity.CustomerId;  
}  
  
var salt = Crypto.GenerateSaltForPassword();  
var newId = AddCustomer(new Customer  
{  
 FirstName = "Тимофей",  
 LastName = "Тимофеев",  
 Honorific = "Даыо.",  
 Email = "dawfdfwa@fsaf.ru",  
 PasswordHash = Crypto.ComputePasswordHash("123456", salt),  
 PasswordSalt = salt  
});  
Console.WriteLine($"Создан новый покупатель с id={newId} и именем {GetCustomerFirstName(newId)}");