МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Разработка приложений баз данных для информационных систем»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Разработка серверной части информационной системы в СУБД MS SQL Server»

Вариант № 4

Выполнил:

студент группы ИТП-31, Бурлаков А.О.

Принял:

Гуменников Е.Д.

Гомель 2024

**ЗАДАНИЕ**

3.1. 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.

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

1. Выборку всех данных из таблицы, стоящей в схеме базы данных нас стороне отношения «один» – 1 шт.
2. Выборку данных из таблицы, стоящей в схеме базы данных нас стороне отношения «один», отфильтрованные по определенному условию, налагающему ограничения на одно или несколько полей – 1 шт.
3. 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 нас стороне отношения «многие» – 1 шт.
4. Выборку данных из двух полей двух таблиц, связанных между собой отношением «один-ко-многим» – 1 шт.
5. 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.
6. Вставку данных в таблицы, стоящей на стороне отношения «Один» – 1 шт.
7. Вставку данных в таблицы, стоящей на стороне отношения «Многие» – 1 шт.:
8. Удаление данных из таблицы, стоящей на стороне отношения «Один» – 1 шт.
9. Удаление данных из таблицы, стоящей на стороне отношения «Многие» – 1 шт.
10. Обновление удовлетворяющих определенному условию записей в любой из таблиц базы данных – 1 шт.

**Ссылка на репозиторий GitHub:** https://github.com/Andrew3574/3Course.git

На рисунках 1,2,3,4,5,6 представлен результат выполнения задания согласно варианту. Текст программы находится в приложении А

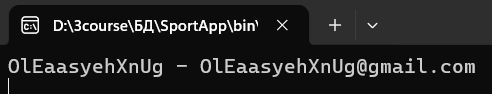


Рисунок 1 – Выборка на стороне «один»

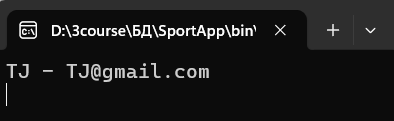


Рисунок 2 – Выборка на стороне «один» с фильтром на длину логина пользователя

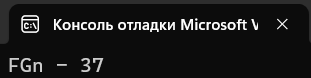


Рисунок 3 – Группировка пользователей по количеству созданных комплексов упражнений



Рисунок 4 – Выборка на стороне «многие»



Рисунок 5 – Добавление нового пользователя



Рисунок 6 – Изменение логина пользователя

**Вывод:** в ходе лабораторной работы было создано консольное приложение с использованием *Entity Framework* содержащее набор классов, моделирующих соответствующей своему варианту предметную область, на основе созданной в лабораторной работе № 1 базой данных MS SQL Server

**ПРИЛОЖЕНИЕ А**

Текст программы

**Program.cs**

using Microsoft.EntityFrameworkCore;

using SportApp.Models;

namespace SportApp

{

class Program

{

static void Main()

{

SportDbContext db = new SportDbContext();

using (db)

{

var users = db.Users.ToList();

var u = users.FirstOrDefault();

Console.WriteLine($"{u.Login} - {u.Email}");

Console.ReadLine();

Console.Clear();

var sortedUser = users.Where(u => u.Login!.Length < 5).FirstOrDefault();

Console.WriteLine($"{sortedUser!.Login} - {sortedUser.Email}");

Console.ReadLine();

Console.Clear();

var sets = db.Sets.Include(u=>u.User).ToList();

var countSet = sets

.GroupBy(u=>u.User!.Login)

.Select(g => new

{

g.Key,

Count = g.Count()

})

.FirstOrDefault();

Console.WriteLine($"{countSet.Key} - {countSet.Count}");

var vset = sets.FirstOrDefault();

Console.WriteLine($"{vset.Name} - {vset.User!.Login}");

var sortedSet = sets.Where(u => u.User!.Login!.Length < 5).FirstOrDefault();

Console.WriteLine($"{sortedSet!.Name} - {sortedSet.User!.Login}");

User user1 = new User()

{

Login="qwerty",

Password="qwerty",

Email="qwrew@gmai.com"

};

db.Users.Add(user1);

db.SaveChanges();

Console.WriteLine($"Added user: {user1.Login} {user1.Email}");

Set set1 = new Set()

{

Name="qwertySet",

User=user1

};

db.Sets.Add(set1);

db.SaveChanges();

user1.Login = "HumanName";

db.Users.Update(user1);

Console.WriteLine($"Updated user: {user1.Login} {user1.Email}");

db.Users.Remove(user1);

db.SaveChanges();

}

}

}

}

**Dotnet-desktop.yml**

name: .NET 8.0 Build

on:

push:

branches:

- main

pull\_request:

branches:

- main

workflow\_dispatch:

jobs:

build:

runs-on: ${{ matrix.os }}

strategy:

matrix:

os: [windows-latest,unubtu-latest]

dotnet-version: ['8.0.x']

steps:

- name: Checkout the repository

uses: actions/checkout@v4

- name: Setup .NET

uses: actions/setup-dotnet@v4

with:

dotnet-version: ${{ matrix.dotnet-version }}

continue-on-error: true

- name: Restore dependencies

run: dotnet restore ./SportApp/SportApp.sln

- name: Build the solution

run: dotnet build ./SportApp/SportApp.sln --configuration Release --no-restore

- name: Run tests

run: dotnet test ./SportApp/SportApp.sln --configuration Release --no-build --verbosity normal