ЛАБОРАТОРНАЯ РАБОТА №8

Знакомство с технологией ADO.NET Entity Framework

1 Цель работы

Изучить возможности использования автономной модели данных на основе классов Entity Framework.

2 Постановка задачи

Для заданной в индивидуальном задании предметной области, используя подход «Code First» создать базу данных, содержащую одну таблицу.

Реализовать для спроектированной БД возможность просмотра и редактирования данных с соблюдением следующих условий:

- при выводе списка всех записей в DataGrid свойство «Id» (ключевое поле) не должно отображаться;
- в элементе DataGrid заголовки столбцов должны быть на русском языке;
- текст в первой колонке выровнять по правому краю;
- строки списка должны быть пронумерованы;
- редактирование и добавление данных должно осуществляться в отдельном окне.

3 Индивидуальные задания

Варианты предметной области для проектирования БД:

1. Автотранспорт
2. Жилищно-коммунальная сфера
3. Здравоохранение
4. Бытовое обслуживание населения
5. Образование
6. Муниципальное управление

7. Железнодорожный транспорт
8. Авиаперевозки
9. Компьютерная техника
10. Энергетика

4 Пример работы приложения

Реакция программы на нажатие кнопки «Delete student» - см. Рисунок 1

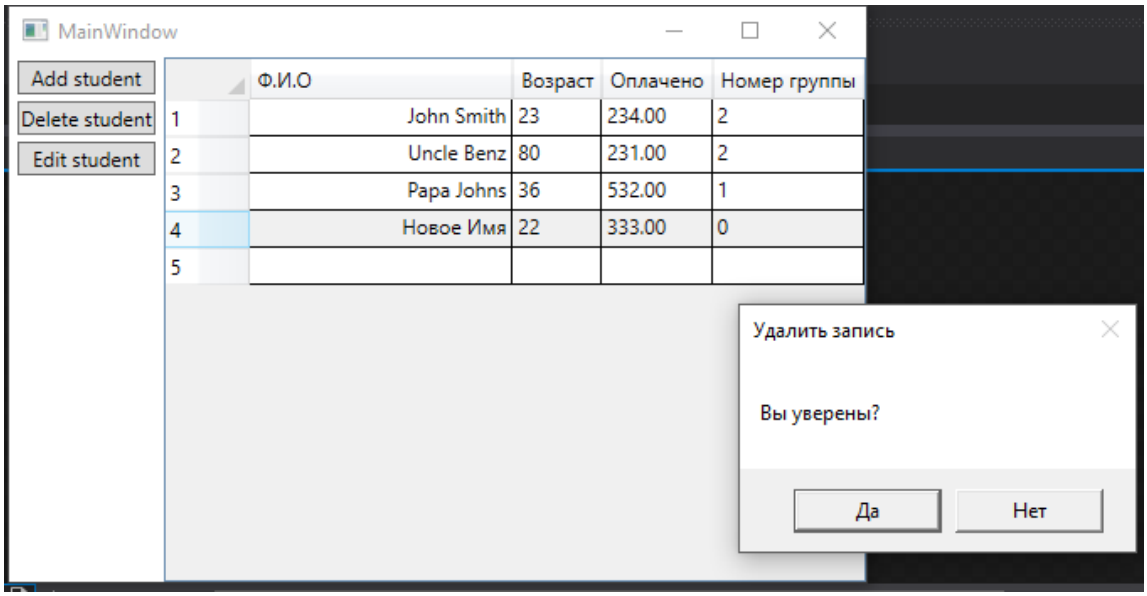

Рисунок 1 Реакция программы на нажатие кнопки «Delete student»

Пример окна редактирования данных приведен на Рисунок 2

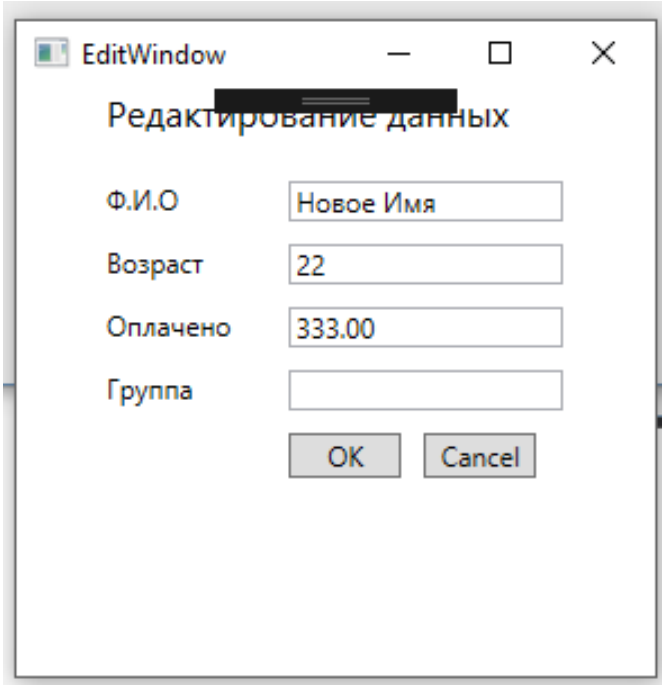

Рисунок 2 Окно редактирования данных

5 Рекомендации к выполнению задания

5.1 Подключение библиотеки EntityFramework к проекту

Откройте окно диспетчера пакетов NuGet (см. Рисунок 3)

В открывшемся окне в строке поиска введите EntityFramework (см. Рисунок 4). В появившемся списке выберите пакет EntityFramework. Поставьте галочку в CheckBox с названием вашего проекта и нажмите кнопку «Установить».

В узле «Ссылки» (References) вашего проекта появятся новые библиотеки, необходимые для работы Entity Framework.

5.2 Добавление строки подключения

В файле app.config **после** раздела <startup> добавьте раздел, описывающий строки подключения, например:

```
<startup>
    . . .
 </startup>
 <connectionStrings>
    <add name="TestDbConnection"
         connectionString="Data Source=(localdb)\mssqllocaldb;Initial
Catalog=Lab8Db;Integrated Security=True"
```

```
        providerName="System.Data.SqlClient"/>
</connectionStrings>
```

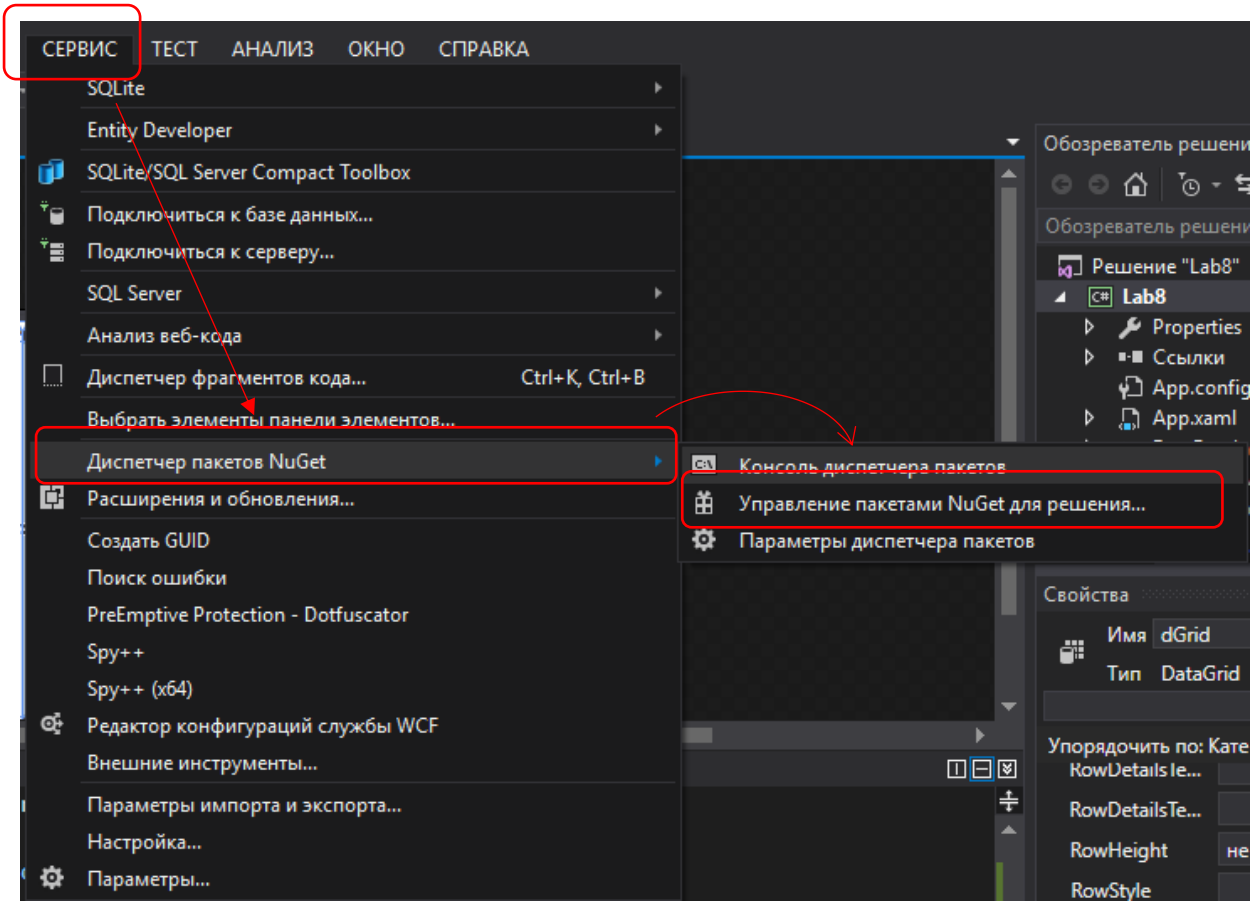

Рисунок 3 Управление пакетами NuGet

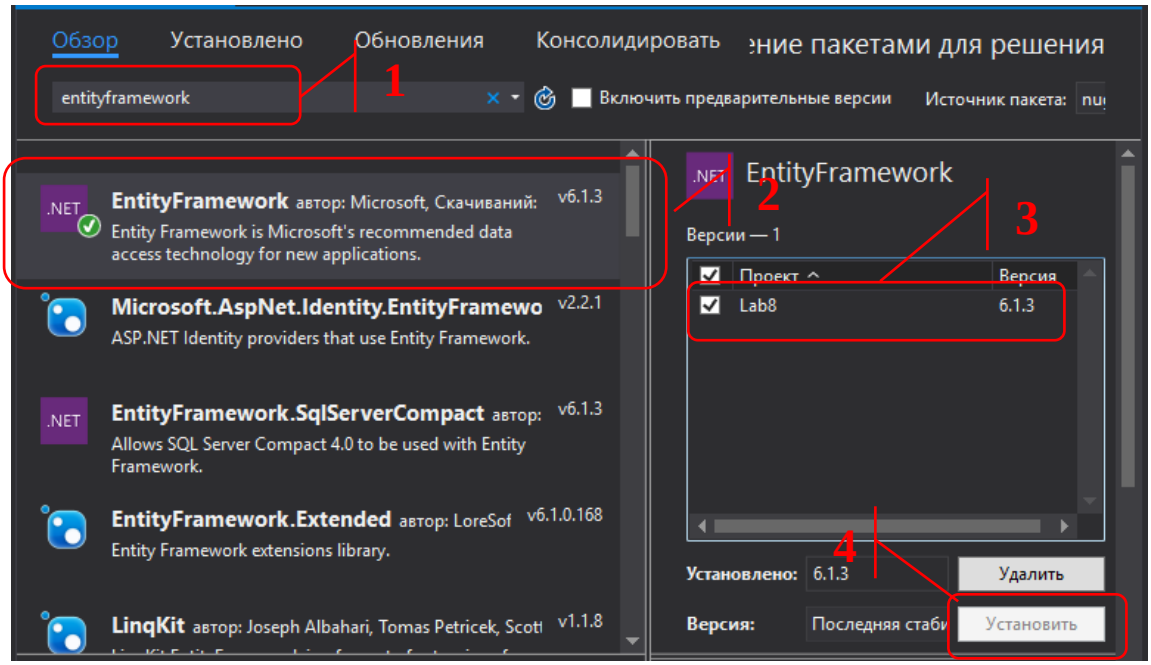

Рисунок 4 Окно управления пакетами NuGet

5.3 Создание классов сущностей и контекста

Создайте класс, описывающий сущность предметной области. Это класс впоследствии будет отображаться на таблицу базы данных.

```
public class Student
{
        public int StudentId { get; set; }
        public String FullName { get; set; }
        public int Age { get; set; }
        public decimal Payment { get; set; }
        public int GroupId { get; set; }
}
```

Примечание: по соглашению, принятому в Entity Framework свойство, название которого включает имя класса с добавлением id (до или после) или просто называется id, будет преобразовано в ключевое поле таблицы БД

Создайте класс контекста данных:

```
using System.Data.Entity;

namespace Lab8
{
    class EntityContext:DbContext
    {
        public EntityContext(string name) : base(name)
        {

        }
       public DbSet<Student> Students { get; set; }
    }
}
```

В приведенном тексте параметр name – это имя строки подключения в файле конфигурации.

Для того, чтобы новая база данных при создании заполнялась исходными данными, создайте класс инициализации базы данных.

Класс должен быть унаследован от класса `DropCreateDatabaseIfModelChanges<T>` или от клвсса `DropCreateDatabaseAlways<T>`. В созданном классе переопределите метод Seed для инициализации начальных значений :

```
using System.Data.Entity;

namespace Lab8
{
    class DataBaseInitializer:
DropCreateDatabaseIfModelChanges<EntityContext>
    {
        protected override void Seed(EntityContext context)
        {
            context.Students.AddRange(new Student[] {
                new Student { FullName="John Smith", Age=23,
                              Payment=234, GroupId=2 },
                new Student { FullName="Uncle Benz", Age=80,
                              Payment=231, GroupId=2 },
                new Student { FullName="Papa Johns", Age=36,
                              Payment=532, GroupId=1 },
            });
        }
    }
}
```

В конструкторе класса контекста укажите необходимость использования созданного инициализатора БД:

```
using System.Data.Entity;

namespace Lab8
{
    class EntityContext:DbContext
    {
        public EntityContext(string name) : base(name)
        {
            Database.SetInitializer(new DataBaseInitializer());
        }
       public DbSet<Student> Students { get; set; }

    }
}
```

5.4 Создание базовой разметки главного окна

Пример разметки:

```
<Window.Resources>
        <Style TargetType="{x:Type Button}">
            <Setter Property="Margin" Value="5,2"/>
        </Style>
</Window.Resources>
<Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto"></ColumnDefinition>
```

```
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <DataGrid x:Name="dGrid" Grid.Column="1"
                  ItemsSource="{Binding
        </DataGrid>
        <StackPanel x:Name="stackButtons">
            <Button x:Name="btnAdd" Click="btnAdd_Click">
                Add student</Button>
            <Button x:Name="btnDelete" Click="btnDelete_Click">
                Delete student</Button>
            <Button x:Name="btnEdit" Click="btnEdit_Click">
                Edit student</Button>
        </StackPanel>
</Grid>
```

В приведенной разметке в ресурсах окна определен стиль для всех кнопок. Этот стиль определяет зазор между кнопкой и соседним элементом (5 по горизонтали и 2 по вертикали).

В коде окна создайте контекст базы данных и поместите свойство Sudents в DataContext элемента DataGrid:

```
using System.Data.Entity;

namespace Lab8
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        EntityContext context;

        public MainWindow()
        {
            context = new EntityContext("TestDbConnection");
            InitializeComponent();
            InitStudentList();

        }

        private void InitStudentList()
        {
            context.Students.Load();
            dGrid.DataContext = context.Students.Local;
        }


    . . .
```

```
}
```

Можно запустить проект и убедиться, что таблица заполнена исходными данными. После этого в окне обозревателя серверов можно подключиться к созданной базе данных.

5.5 Создание окна редактирования

Для редактирования или создания нового объекта необходимо этот объект передать окну. Для этого можно создать параметризированный конструктор, принимающий нужный объект в качестве параметра. Полученный объект затем нужно привязать к элементам ввода:

```
public partial class EditWindow : Window
    {
        Student student;
        public EditWindow()
        {
            InitializeComponent();
        }

        public EditWindow(Student student):this()
        {
            this.student = student;
            grid.DataContext = student;
        }

     . . .
}
```

Обработка событий нажатия кнопок может выглядеть так:

```
        private void btnOK_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = true;
        }

        private void btnCancel_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
```

5.6 Обработка событий нажатия кнопок главного окна

Кнопки добавления и редактирования объекта должны вызывать открытия диалогового окна редактирования. Если при закрытии окна редактирования была нажата кнопка ОК, то данные нужно сохранить:

```
private void btnAdd_Click(object sender, RoutedEventArgs e)
        {
            var stud = new Student();
            EditWindow ew = new EditWindow(stud);
            var result = ew.ShowDialog();
            if (result==true)
            {
                context.Students.Add(stud);
                context.SaveChanges();
                ew.Close();
            }
}
private void btnEdit_Click(object sender, RoutedEventArgs e)
        {
            Student stud = dGrid.SelectedItem as Student;

            EditWindow ew = new EditWindow(stud);
            var result = ew.ShowDialog();
            if (result == true)
            {
                context.SaveChanges();
                ew.Close();
            }
            else
            {
                // вернуть начальное значение
                context.Entry(stud).Reload();

                // перегрузить DataContext
                dGrid.DataContext = null;
                dGrid.DataContext = context.Students.Local;
            }
}
```

При удалении объекта нужно вывести предупреждение:

```
private void btnDelete_Click(object sender, RoutedEventArgs e)
        {
            var result=MessageBox.Show("Вы уверены?", "Удалить
запись", MessageBoxButton.YesNo);
            if(result==MessageBoxResult.Yes)
            {
                Student stud = dGrid.SelectedItem as Student;
                context.Students.Remove(stud);
```

```
                context.SaveChanges();
            }
}
```

Можно запустить проект и проверить правильность работы кнопок

5.7 Нумерация строк в таблице

Для вывода номера строки можно воспользоваться событием LoadingRow элемента управления DataGrid

```
<DataGrid x:Name="dGrid" Grid.Column="1"
                  ItemsSource="{Binding}"
                  LoadingRow="dGrid_LoadingRow"
                  RowHeaderWidth="50">
```

Обработка события:

```
private void dGrid_LoadingRow(object sender, DataGridRowEventArgs e)
        {
            e.Row.Header = e.Row.GetIndex() + 1;
        }
```

5.8 Окончательное оформление таблицы

Для описания столбцов нужно запретить автоматическое генерирование столбцов и описать столбцы вручную. Для первого столбца также нужно добавить стиль для выравнивания текста:

```
<DataGrid x:Name="dGrid" Grid.Column="1"
                  ItemsSource="{Binding}"
                  AutoGenerateColumns="False"
                  LoadingRow="dGrid_LoadingRow"
                  RowHeaderWidth="50">
      <DataGrid.Columns>
            <DataGridTextColumn Header="Ф.И.О"
                                Binding="{Binding FullName}"
                                    Width="*">
                  <DataGridTextColumn.CellStyle>
                        <Style>
                            <Setter Property="TextBlock.TextAlignment"
                                    Value="Right"/>
                        </Style>
                  </DataGridTextColumn.CellStyle>
            </DataGridTextColumn>
            <DataGridTextColumn Header="Возраст"
                                Binding="{Binding Age}"/>
            <DataGridTextColumn Header="Оплачено"
```

```
                                Binding="{Binding Payment}"/>
            <DataGridTextColumn Header="Номер группы"
                                Binding="{Binding GroupId}"/>
        </DataGrid.Columns>
</DataGrid>
```

Запустите проект и проверьте результат