**Лабораторная работа №4**

**Оглавление**

[Лабораторная работа № 4. Работа с базой данных, Entity Framework 3](#_Toc163821059)

[Содержание пояснительной записки 9](#_Toc163821060)

[Используемое программное обеспечение 10](#_Toc163821061)

[Список литературы 11](#_Toc163821062)

Лабораторная работа № 4.  
Работа с базой данных, Entity Framework

**Цель лабораторной работы**

Закрепление теоретических знаний по основам работы с баззой данных при помощи технологии Entity Framework.

**Постановка задачи**

**Ба́за да́нных** — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

**Систе́ма управле́ния ба́зами да́нных, сокр. СУБД** — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных. СУБД — комплекс программ, позволяющих создать базу данных и манипулировать данными.

**ADO.NET Entity Framework** — объектно-ориентированная технология доступа к данным, является object-relational mapping (отображение объектов и связей) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL.

Для работы с БД будем использовать СУБД SQLite и EntityFramework. Необходимо перейти в диспетчер пакетов Nuget и установить следующие пакеты (рисунок 1):

* Microsoft.EntityFrameworkCore,
* Microsoft.EntityFrameworkCore.Sqlite.

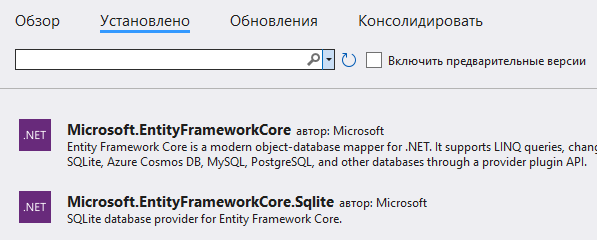


Рисунок 1 – Установка пакетов для работы с базой данных

**Таблица** — совокупность связанных данных, хранящихся в структурированном виде в базе данных. Она состоит из столбцов и строк. Таблица содержит определенное число столбцов, но может иметь любое количество строк. Каждая строка однозначно определяется одним или несколькими уникальными значениями, которые принимают её ячейки из определенного подмножества столбцов. Подмножество столбцов, которое уникально идентифицирует строку, называется первичным ключом.

**Ячейка** — место, где строка и столбец пересекаются.

**Первичный ключ** – поле, которое уникально идентифицирует запись в таблице. Обычно в качестве ключа используется тип int и имя Id.

Чтобы задать имя таблицы используется атрибут [Table(“Имя”)].

Чтобы явным образом указать первичный ключ нужно использовать атрибут [Key].

Чтобы задать минимальную или максимальную длину строки используются атрибуты [MinLength(N)] и [MaxLength(N)].

[Table("Покупатель")]

public class Customer

{

[Key] public int Id { get; set; }

[MaxLength(200)] public string? Name { get; set; }

public int Age { get; set; }

}

Добавим класс банковского аккаунта пользователя:

public class BankAccount

{

public int Id { get; set; }

public int CustomerId { get; set; } //Внешний ключ

public double Money { get; set; }

public Customer? Customer { get; set; } //Навигационное свойство

}

Добавим класс, описывающий товар:

public class Product

{

public int Id { get; set; }

public string? Name { get; set; }

public double Price { get; set; }

}

Взаимодействие с базой данных в **Entity Framework Core** происходит посредством специального класса - **контекста данных**. Поэтому добавим в наш проект новый класс, который назовем ApplicationContext и который будет иметь следующий код:

public class ApplicationContext : DbContext

{

public DbSet<Customer> Customers => Set<Customer>();

public DbSet<Product> Products => Set<Product>();

public DbSet<BankAccount> BankAccounts => Set<BankAccount>();

public ApplicationContext() => Database.EnsureCreated();

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlite("Data Source=MyDataBase.db");

}

}

Посмотрим на примерах как можно работать с таблицами, заполним БД некоторыми данными:

using ApplicationContext db = new();

db.Customers.AddRange(new List<Customer> {

new Customer{Name = "Иван", Age = 25},

new Customer{Name = "Петр", Age = 35},

new Customer{Name = "Елена", Age = 23},

new Customer{Name = "Анна", Age = 40},

});

db.Products.AddRange(new List<Product> {

new Product{Name = "Стол", Price = 12000},

new Product{Name = "Стул", Price = 2500},

new Product{Name = "ВАЗ 2107", Price = 75000},

new Product{Name = "Яблоки 1кг", Price = 120},

});

db.SaveChanges();

**Примеры** работы с БД.

Извлечение данных:

var customers = db.Customers.ToList();

Извлекаем из БД молодых покупателей:

var youngCustomers = db.Customers

.Where(c => c.Age <= 30)

.ToList();

Извлекаем самый дорогой товар:

var mostExpensive = db.Products

.ToList()

.MaxBy(p => p.Price);

Удаляем аккаунты с количеством денег меньше 10000:

var customersToRemove = db.BankAccounts

.Where(b => b.Money <= 1e4)

.Select(b => b.Customer)

.ToList();

foreach (var customer in customersToRemove)

{

db.Customers.Remove(customer);

}

Сохраняем изменения:

db.SaveChanges();

Дополнительная информация по работе с Entity Framework: <https://metanit.com/sharp/efcore/1.2.php>

**Пример** использования полиморфизма для загрузки данных.

// Класс человека в сокращенном виде

public record Person(int Id, string Name, int Age, double Money);

// Интерфейс для загрузки объекта человека

public interface IPersonLoader

{

public Person? LoadByName(string dataBasePath, string name);

}

using Newtonsoft.Json.Linq;

// Класс для загрузки человека из JSON файла

public class JsonPersonLoader : IPersonLoader

{

public Person? LoadByName(string dataBasePath, string name)

{

var data = File.ReadAllText(dataBasePath);

var jsonArray = JArray.Parse(data);

foreach (var item in jsonArray)

{

var tokenName = item.Value<string>("Name");

if (tokenName == name)

{

var tokenAge = item.Value<int>("Age");

var tokenMoney = item.Value<double>("Money");

return new Person(default, tokenName, tokenAge, tokenMoney);

}

}

return null;

}

}

// Класс для загрузки человека из БД SQLite

public class SQLitePersonLoader : IPersonLoader

{

public Person? LoadByName(string dataBasePath, string name)

{

using ApplicationDBContext db = new(dataBasePath);

return db.Persons.SingleOrDefault(p => p.Name == name);

}

}

// Интерфейс для сохранения объекта человека

public interface IPersonSaver

{

public void SavePerson(string dataBasePath, Person person);

}

using Newtonsoft.Json.Linq;

// Класс для сохранения человека в JSON

public class JsonPersonSaver : IPersonSaver

{

public void SavePerson(string dataBasePath, Person person)

{

var json = File.ReadAllText(dataBasePath);

var jArr = JArray.Parse(json);

var jsonPerson = new JObject();

jsonPerson["Name"] = person.Name;

jsonPerson["Age"] = person.Age;

jsonPerson["Money"] = person.Money;

jArr.Add(jsonPerson);

File.WriteAllText(dataBasePath, jArr.ToString());

}

}

// Класс для сохранения человека в БД SQLite

public class SQLitePersonSaver : IPersonSaver

{

public void SavePerson(string dataBasePath, Person person)

{

using ApplicationDBContext db = new(dataBasePath);

db.Persons.Add(person);

db.SaveChanges();

}

}

**Задание на лабораторную работу**

1. Модифицировать программу из лабораторной работы 3. Добавить загрузку данных из базы данных SQLite.
2. Перенести / добавить данные, которые будут храниться в базе данных. Например, данные пользователей при регистрации, параметры противников или игрок, игровая статистика и др. по выбору.
3. Добавить уровни сложности в игру.
4. Добавить звуковые эффекты и музыкальное сопровождение.
5. Предусмотреть обучение нового пользователя при первом запуске программы.
6. Защита работы включает демонстрацию работы программы.
7. Для успешной защиты:

* программа должна быть задокументирована с помощью комментариев,
* код программы не должен содержать не используемые блоки, должен быть «чистым» и стилистически верно оформлен,
* интерфейс программы должен быть гибким и удобным.

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»