Минобрнауки России

ФГБОУ ВО «Удмуртский государственный университет»

Институт математики, информационных технологий и физики

Кафедра вычислительных систем и информационных технологий

Направление 09.03.01 «Информатика и вычислительная техника»

Зачетная работа по теме

"Создание базы данных для интернет-магазина мотоциклов в

Entity Framework Core"

Студента ОБ-09.03.01-21

Худайбердиева Эльеса Бахадировича

Научный руководитель:

к. т. н., доцент

Поярков Андрей Валерьевич

Ижевск 2021

Содержание

[Введение 3](#_Toc74968319)

[Глава 1. Выбор технологий для реализации базы данных 4](#_Toc74968320)

[Обзор технологий 4](#_Toc74968321)

[Описание Entity Framework 5](#_Toc74968322)

[Архитектура Entity Framework 6](#_Toc74968323)

[Описание платформы .NET 7](#_Toc74968324)

[Описание LINQ 10](#_Toc74968325)

[Среда разработки 11](#_Toc74968326)

# Введение

Цель работы: создание базы данных для интернет-магазина мотоциклов. База данных должна осуществлять хранение и обработку информации для динамического сайта данного интернет-магазина.

Для выполнения поставленной цели необходимо выполнить следующие задачи:

Разработать набор технологий, реализующий требования, которые относятся к базе данных для интернет-магазина автомобилей

Осуществить проектирование системы.

# Глава 1. Выбор технологий для реализации базы данных

## Обзор технологий

Entity Framework — это набор технологий в ADO.NET, которые поддерживают разработку программных приложений, ориентированных на данные.

ASP.NET — интернет-платформа для создания веб-сайтов и веб-приложений с помощью HTML, CSS и JavaScript. Также c помощью ASP.NET можно создавать веб-API и использовать технологии реального времени, такие как веб-сокеты.

LINQ (Language Integrated Query) — проект Microsoft по добавлению синтаксиса языка запросов, напоминающего SQL, в языки программирования платформы .NET. LINQ предоставляет возможности выполнения запросов к базе данных на уровне языка в C# и Visual Basic.

C# — объектно-ориентированный и типобезопасный язык программирования. C# позволяет создавать различные приложения, работающие в экосистеме .NET. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Entity был выбран для разработки базы данных по следующим причинам:

1. Так как Entity framework является частью программной платформы .NET framework, которая позволяет проектировать различное программное обеспечение, он является более предпочтительным в сравнении с другими модулями работы с базами данных, поскольку он не требует дополнительной настройки или интеграции в платформу .NET.
2. Entity Framework позволяет работать с данными в форме объектов и свойств (например, с клиентами и их адресами), при этом нет необходимости учитывать формат базовых таблиц и столбцов базы данных, где хранятся эти данные.
3. Entity Framework позволяет работать с данными на более высоком уровне абстракции, создавать и сопровождать приложения, ориентированные на работу с данными, а также сокращать объем кода для этих программ по сравнению с традиционными приложениями.
4. Начиная с версии entity 5.0, наиболее предпочтительным является подход code first, благодаря которому при проектировании базы данных нет необходимости придумывать отдельные сущности или разрабатывать свою систему управления базами данных, как это обычно делают в других приложениях. Этот подход заключается в том, что при разработке программы вы можете написать код модели, а затем она автоматически сгенерируется в базе данных.
5. Есть поддержка множества СУБД, благодаря чему нет необходимости учитывать особенности той или иной системы управления и производить запросы в одинаково для любой из них. Это упрощает разработку базы данных и дальнейшей работы с ней.

## Описание Entity Framework

Entity framework позволяет взаимодействовать с СУБД (система управления базами данных) при помощи сущностей, то есть классов и объектов .NET, а не таблиц баз данных. Данный фреймворк позволяет работать со следующими СУБД: SQL Server, SQLite, PostgreSQL, MySQL, MariaDB, MySQL, Oracle DB и другие.

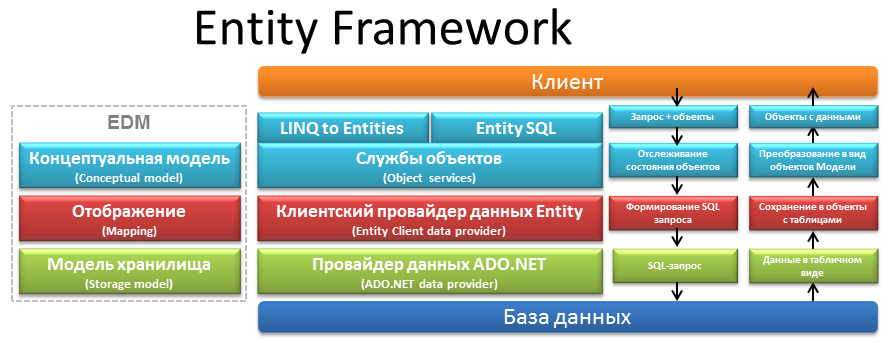
Entity framework является самым известным и функциональным ORM – инструментом C#.

ORM (Object-Relational Mapping — объектно-реляционное отображение, или преобразование) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

Есть 3 подхода ORM:

1. Database first.При таком подходе сначала создается база данных. Сложность этого подхода заключается в том, что сразу создать правильную базу данных довольно сложно, поэтому ее необходимо изменять в течение разработки основного приложения.
2. Code first.При подходе code first сначала пишется код на C#, а уже затем на основе созданных классов генерируется база данных. Данный подход чаще всего применяется при разработке приложений на C#.
3. Model first.При этом подходе сначала создается EDM (графическая) модель, а затем на ее основе генерируется сама база данных.

При разработке базы данных её приходится изменять. Для того, чтобы изменения происходили без потери уже имеющейся информации в базе данных, используют функцию миграции, которая позволяет изменять схему в базе данных для синхронизации ее с моделью данных, при этом уже существующая информация никак не изменяется.

Архитектура Entity FrameworkРисунок 1. Архитектура Entity Framework

Архитектура Entity Framework представлена на рисунке 1. Основными частями технологии являются: службы объектов (object services), клиентский провайдер данных Entity (Entity Client data provider) и провайдер данных ADO.NET (ADO.NET data provider). Эти три части соответствуют концептуальной модели, отображению и модели хранилища.

1. Слой службы объектов.Данный слой предназначен для создания запросов к базе данных и получения необходимой информации. Создание запросов производится с помощью LINQ. Также этот слой преобразовывает объекты, полученные из слоя клиентского провайдера данных в объекты моделей для того, чтобы с ними было удобно работать в C#.
2. Слой клиентского провайдера данных.Этот слой используется для взаимодействия с базой данных. Для упрощения архитектуры он не обращается к ней напрямую, а использует провайдер данных ADO.NET. На уровне этого слоя создаются сами SQL запросы, соответствующие используемой СУБД. Слой клиентского провайдера данных преобразует результаты запросов их простой табличной формы в специальные объекты и передает их службам объектов для дальнейшей обработки.
3. Слой провайдера данных ADO.NET.Провайдер данных ADO.NET предназначен для непосредственного обращения к СУБД.

## Описание платформы .NET

Платформа .NET Framework — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows.

При разработке платформы .NET Framework учитывались следующие цели:

1. Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.
2. Предоставление среды выполнения кода, в которой:
   1. сведена к минимуму вероятность конфликтов в процессе развертывания программного обеспечения и управления его версиями;
   2. Гарантируется безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем;
   3. исключаются проблемы с производительностью сред выполнения скриптов или интерпретируемого кода;
3. обеспечиваются единые принципы разработки для разных типов приложений, таких как приложения Windows и веб-приложения;
4. обеспечивается взаимодействие на основе промышленных стандартов, которое гарантирует интеграцию кода платформы .NET Framework с любым другим кодом.

Основными языками .NET являются C#, Visual C++, Visual Basic, F#.

CLI (Common Language Infrastructure — интерфейс командной строки) .NET — это кроссплатформенная цепочка инструментов для разработки, сборки, запуска и публикации приложений .NET.

CLR (Common Language Runtime — общеязыковая исполняющая среда) — исполняющая среда для байт-кода CIL, в который компилируются программы, написанные на .NET-совместимых языках программирования.

В CLI у есть компилятор для каждого языка, использующегося в .NET, благодаря чему любая программа, написанная на любом из этих языков, компилируется в промежуточный байт-код CIL (**Common Intermediate Language**), который в среде CLR компилируется в нативный код и начинает выполняться. Работа среды .NET также описана на рисунке 2.

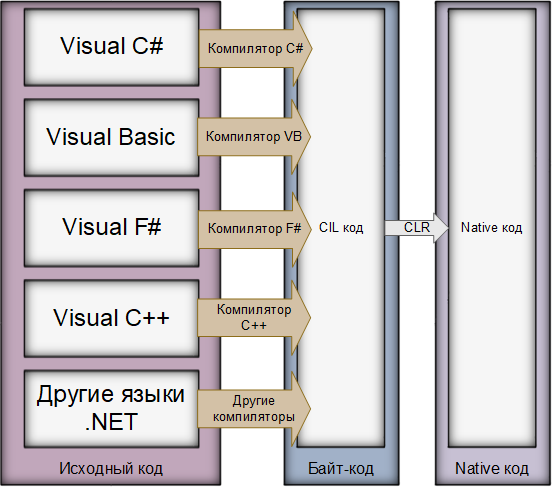


Рисунок 2. Работа среды .NET

## Описание LINQ

Для получения информации из базы данных широко используется технология LINQ, в Entity Framework Core она представлена в виде LINQ to Entities. Выражения LINQ создаются в виде методов в коде программы, а затем они транслируются в SQL запросы, которые понятны используемой базе данных. Это позволяет обеспечить удобную работу с СУБД. Используя некоторые новые особенности языка, LINQ позволяет использовать SQL-подобный синтаксис непосредственно в коде программы, написанной, например, на языке C#. Это язык предоставляет два вида запросов: локальные (для локальных коллекций) и интерпретируемые (для удаленных источников данных). Базовые единицы данных в LINQ - последовательности. Последовательность, - любая коллекция, реализующая интерфейс IEnumerable или IQueryable. Оператор запроса - метод, преобразующий последовательность. Запрос - выражение, которое преобразует последовательность с помощью операторов запроса. Оператор запроса никогда не изменяет входную последовательность, вместо этого он возвращает новую. Для использования LINQ необходимо подключить пространство имен System.Linq и наличие Framework 3.5 или выше.

## Среда разработки

В качестве среды разработки была использована Visual Studio Community.

Microsoft Visual Studio  — интегрированная среда разработки программного обеспечения, предназначенная для разработки и публикации различных приложений, а также отладки и сборки кода. Помимо этого, данная среда включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции, упрощающие процесс разработки. Также в данной среде можно легко делать ранее описанные миграции. Среда Visual Studio доступна для Windows и Mac. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения.

# Глава 2. Разработка базы данных

## Создание ER-модели

Для создания базы данных были выделены следующие сущности:

1. Компания
2. Мотоцикл

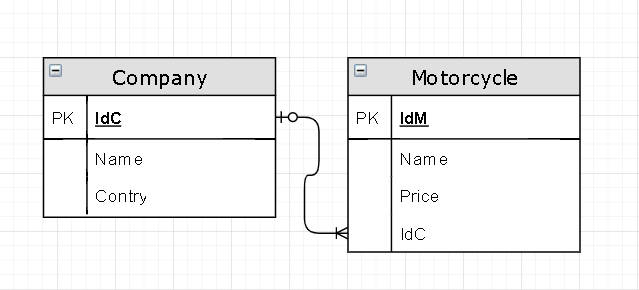


Рисунок 3.ER-модель

Модель была создана с помощью программы Draw.io

Между таблицами Company и Motorcycle присутствует связь 1 к Многим обязательно, поэтому передается внешний ключ таблице Motorcycle.

## Разработка проекта

Создание проекта начинается с установки пакетов.

1. Microsoft.EntityFrameworkCore.SqlServer
2. Microsoft.EntityFrameworkCore.Tools
3. Microsoft.EntityFrameworkCore.Design.

Для создания сущностей было необходимо создать классы, и выполнить команду Add – Migration в командной строке, чтобы на основе классов были созданы соответствующие таблицы. При необходимости нужно выполнить команду Update-Database при изменении таблицы.

После команды Add – Migration, cоздается папка Migrations в которой находится файл с нашей миграцией. В нем мы видим два метода – это метод Up, который создает нашу таблицу Motorcycle и Company со всеми её столбцами, а также присваивает первичный ключ столбцу со свойством Id и метод Down, который позволяет откатить наш проект в любую из доступных миграций.

1. public partial class v1 : Migration
2. {
3. protected override void Up(MigrationBuilder migrationBuilder)
4. {
5. migrationBuilder.CreateTable(
6. name: "Companies",
7. columns: table => new
8. {
9. Id = table.Column<int>(type: "int", nullable: false)
10. .Annotation("SqlServer:Identity", "1, 1"),
11. Name = table.Column<string>(type: "nvarchar(50)", maxLength: 50, nullable: true),
12. Country = table.Column<string>(type: "nvarchar(max)", nullable: true)
13. },
14. constraints: table =>
15. {
16. table.PrimaryKey("PK\_Companies", x => x.Id);
17. });
18. migrationBuilder.CreateTable(
19. name: "Motorcycles",
20. columns: table => new
21. {
22. Id = table.Column<int>(type: "int", nullable: false)
23. .Annotation("SqlServer:Identity", "1, 1"),
24. Name = table.Column<string>(type: "nvarchar(50)", maxLength: 50, nullable: true),
25. Price = table.Column<int>(type: "int", nullable: false),
26. CompanyId = table.Column<int>(type: "int", nullable: false)
27. },
28. constraints: table =>
29. {
30. table.PrimaryKey("PK\_Motorcycles", x => x.Id);
31. table.ForeignKey(
32. name: "FK\_Motorcycles\_Companies\_CompanyId",
33. column: x => x.CompanyId,
34. principalTable: "Companies",
35. principalColumn: "Id",
36. onDelete: ReferentialAction.Cascade);
37. });
38. migrationBuilder.CreateIndex(
39. name: "IX\_Motorcycles\_CompanyId",
40. table: "Motorcycles",
41. column: "CompanyId");
42. }
43. protected override void Down(MigrationBuilder migrationBuilder)
44. {
45. migrationBuilder.DropTable(
46. name: "Motorcycles");
47. migrationBuilder.DropTable(
48. name: "Companies");
49. }
50. }

Далее приведен код каждого класса:

Сущность Company. Имеет первичный ключ Id, поле названия компании и название страны. Для сохранения эстетики название компании может иметь не больше 50 символов. Для связи с таблицей Motorcycle, класс Company содержит поле Motorcycles.

1. public class Company
2. {
3. public int Id { get; set; }
4. [MaxLength(50)]
5. public string Name { get; set; }
6. public string Country { get; set; }
7. public List<Motorcycle> Motorcycles { get; set; }
8. }

Сущность Motorcycle. В классе содержится первичный ключ Id, название мотоцикла, цена, внешний ключ CompanyId и навигационное свойство Company. [Key] означает, что поле Id является первичным ключом, в данном примере это необязательно т.к поле содержит слово “Id”. MaxLength(50) и MinLength(5) означают, что название может содержать от 5 до 50 символов.

1. public class Motorcycle
2. {
3. [Key]
4. public int Id { get; set; }
5. [MaxLength(50), MinLength(5)]
6. public string Name { get; set; }
7. public int Price { get; set; }
8. public int CompanyId { get; set; }
9. public Company Company { get; set; }
10. }

Для создание базы данных и ее таблиц необходим прописать класс AppContext наследованный от DbContext.

1. public class AppContext : DbContext.
2. {
3. public DbSet<Motorcycle> Motorcycles { get; set; }
4. public DbSet<Company> Companies { get; set; }
5. protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
6. {
7. optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=MotorcycleShop;Trusted\_Connection=True;");
8. }
9. }

Метод - OnConfiguring определяет параметры базы данных. Входным качестве входного параметра находится объект класса DbContextOptionsBuilder, предназначенный для настройки контекста. С помощью его метода UseSqlServer задаются параметры для инициализации базы данных, в него передается строка с нужными параметрами. В данном случае параметр « Database=MotorcycleShop » означает, что база данных будет называться MotorcycleShop.

Нам необходим создавать базу данных при запуске, для этого у нас есть стандартный класс Program, в котором записываем данные и выводим данные на консоль. В классе определен основной метод Main, вызывающийся при запуске программы.

* 1. Company Company1 = new Company { Name = "BMW", Country = "Германия" };
  2. Company Company2 = new Company { Name = "Ducati", Country = "Италия" };
  3. Company Company3 = new Company { Name = "Honda", Country = "Япония" };
  4. Company Company4 = new Company { Name = "Kawasaki ", Country = "Япония" };
  5. Company Company5 = new Company { Name = "KTM", Country = "Австрия" };
  6. db.Companies.AddRange(Company1, Company2, Company3, Company4, Company5);
  7. db.SaveChanges();
  8. Motorcycle motorcycle1 = new Motorcycle { Name = "BMW R 1250 GS ADVENTURE", Price = 2051000, CompanyId = 1 };
  9. Motorcycle motorcycle2 = new Motorcycle { Name = "BMW R 1250 GS ADVENTURE", Price = 2051000, CompanyId = 1 };
  10. Motorcycle motorcycle3 = new Motorcycle { Name = "BMW R NINET SCRAMBLER", Price = 1318000, CompanyId = 1 };
  11. Motorcycle motorcycle4 = new Motorcycle { Name = "DUCATI XDIAVEL S - THRILLING BLACK", Price = 2542000, CompanyId = 2 };
  12. Motorcycle motorcycle5 = new Motorcycle { Name = "DUCATI XDIAVEL S - ICEBERG WHITE", Price = 2688000, CompanyId = 2 };
  13. Motorcycle motorcycle6 = new Motorcycle { Name = "DUCATI MULTISTRADA 1260 ENDURO", Price = 203200, CompanyId = 2 };
  14. Motorcycle motorcycle7 = new Motorcycle { Name = "HONDA CBR1000RR-R FIREBLADE", Price = 2215900, CompanyId = 3 };
  15. Motorcycle motorcycle8 = new Motorcycle { Name = "HONDA CB1000R BLACK EDITION", Price = 1429900, CompanyId = 3 };
  16. Motorcycle motorcycle9 = new Motorcycle { Name = "HONDA CRF450R 2021", Price = 852900, CompanyId = 3 };
  17. Motorcycle motorcycle10 = new Motorcycle { Name = "KAWASAKI Z900", Price = 915000, CompanyId = 4 };
  18. Motorcycle motorcycle11 = new Motorcycle { Name = "KAWASAKI KX450XC", Price = 915000, CompanyId = 4 };
  19. Motorcycle motorcycle12 = new Motorcycle { Name = "KAWASAKI VULCAN S ABS ЧЕРНЫЙ", Price = 765000, CompanyId = 4 };
  20. Motorcycle motorcycle13 = new Motorcycle { Name = "KTM 150 EXC TPI", Price = 869900, CompanyId = 5 };
  21. db.Motorcycles.AddRange(
  22. motorcycle1, motorcycle2, motorcycle3,
  23. motorcycle4, motorcycle5, motorcycle6,
  24. motorcycle7, motorcycle8, motorcycle9,
  25. motorcycle10, motorcycle11, motorcycle12, motorcycle13
  26. );
  27. db.SaveChanges();

В данном примере db.Companies.AddRange() и db.Motorcycles.AddRange() добавляют данные в базу данных, а db.SaveChanges() сохраняют изменения в базе данных.

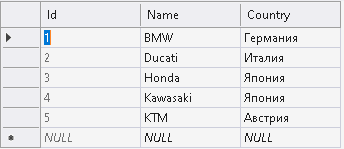


Рисунок 4. Таблица Company

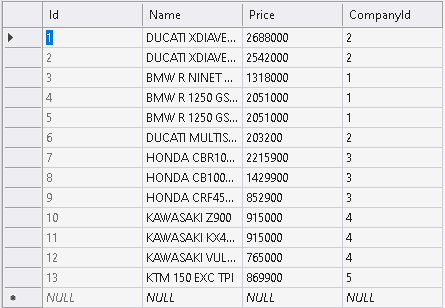


Рисунок 5. Таблица Motorcycle

Реализован вывод в консоль. Для загрузки наших данных из таблиц и их представления в объединенном виде будем использовать Eager Loading, для которого возьмем метод Include. Этот метод подгружает в свой контект данные сущности, передаваемые в него. А также метод ToList, который собирает все сущности в один общий лист.

1. var companies = db.Companies
2. .Include(p => p.Motorcycles)
3. .ToList();
4. Console.WriteLine("Список :");
5. foreach (var company in companies)
6. {
7. Console.WriteLine($"Компания: {company.Name} Страна: {company.Country}");
8. foreach (var motorcycle in company.Motorcycles)
9. Console.WriteLine($"Модель: {motorcycle.Name} Цена: {motorcycle.Price}");
10. Console.WriteLine("----------------------");
11. }

Вывод в консоль:

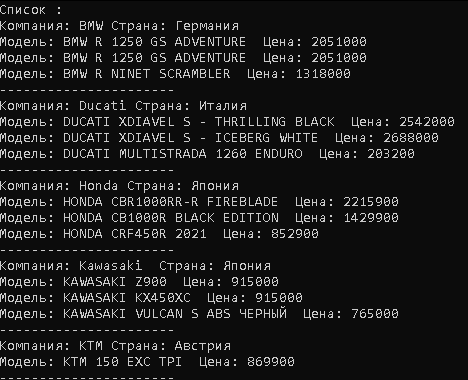


Рисунок 6. Вывод данных

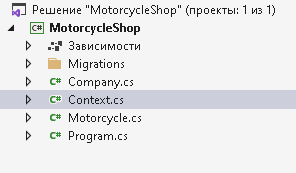


Рисунок 7.Архитектура проекта

Результат работы: при создании классов сущностей в фреймворке Entity Framework Core мы получаем готовую базу данных для реализации интернет-магазина мотоциклов по технологии MVC на платформе .NET.

# Заключение

В ходе выполнения курсовой работы была изучена информация о работе с Entity Framework и разработана база данных для интернет-магазина мотоциклов. Приобретены практические навыки работы с технологиями .NET, LINQ, языком программирования C#, разработки инфологической модели базы данных и средой разработки Visual Studio Community.

# Список литературы

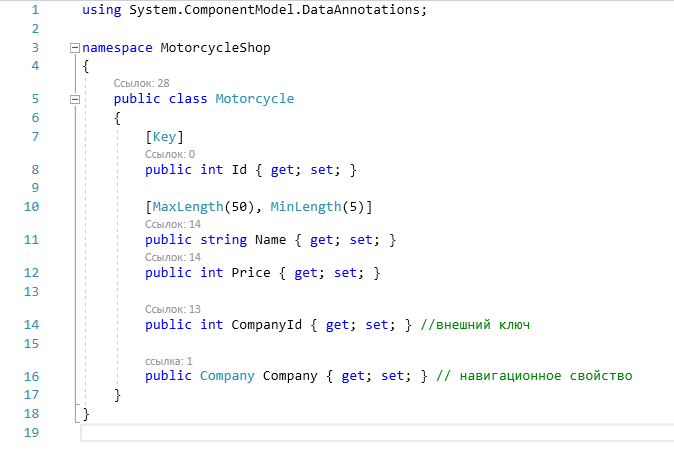
1. Раттц-мл., Джозеф С. – LINQ: язык интегрированных запросов в C# 2008 для профессионалов. : Пер. с англ. — М. : ООО «И.Д. Вильямс», 2008. — 560 с. : ил. — Парал. тит. англ.
2. Герберт Шилдт – C# 4.0: полное руководство.: Пер. с англ. — М.: ООО «И.Д. Вильямс», 2011. — 1056 с.: ил. — Парал. тит. англ.
3. Эндрю Троелсен – Язык программирования C# 5.0 и платфор
4. Прайс М. C# 8 и .NET Core. Разработка и оптимизация/ М.Прайс — СПб.: Питер, 2021. — 816 с.
5. Шустова Л.И. Базы данных. Учебник/ Л.И. Шустова, О.В. Тараканов—Москва: Инфра-М, 2018. — 304 с.
6. Документация и учебный ресурс Майкрософт для разработчиков и технических специалистов. [Электронный ресурс] // Режим доступа: <https://docs.microsoft.com/ru-ru/>, свободный (дата обращения 10.06.2021).
7. Entity Framework Core [Электронный ресурс]. URL: https://metanit.com/sharp/entityframeworkcore/1.1.php (Дата обращения 02.06.2021)

# Приложение

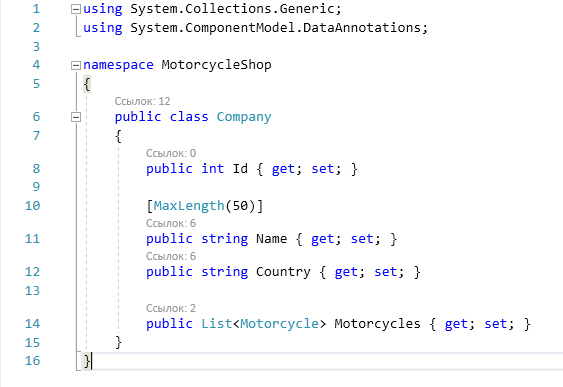
Program.cs



Motorcycle.cs



Company.cs



Context.cs

