Entity framework

Lab 2

1. Klasa Product

Klasa przedstawiająca nasz produkt

```
using System;
namespace BartlomiejKozeraProducts
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsOnStock { get; set; }
    }
}
```

2. Klasa ProductContext

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Sqlite;

namespace BartlomiejKozeraProducts
{
    public class ProcuctContext : DbContext
    {
        public DbSet<Product> Products { get; set;}

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("Datasource=ProductsDatabase");
        }
    }
}
```

Konfiguracja tabeli Products

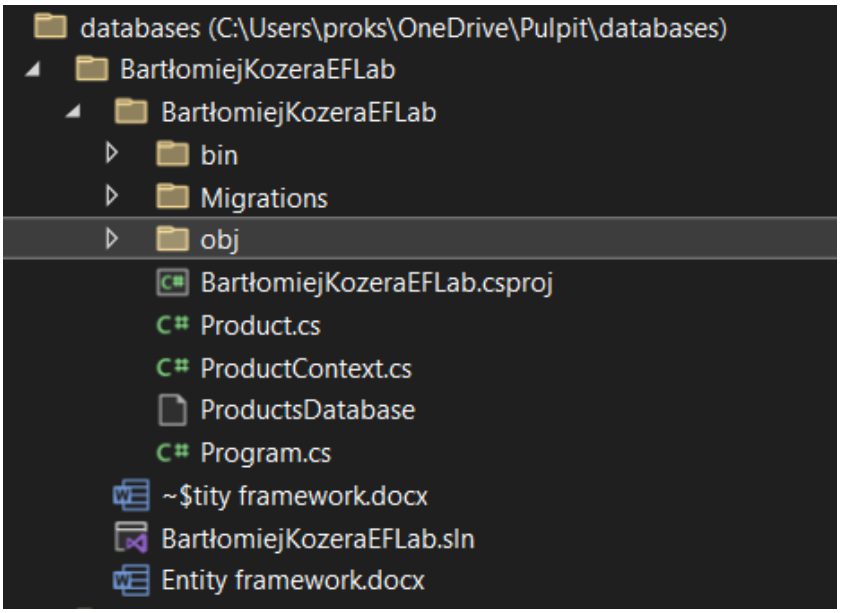
3. Migracje

```
PS C:\Users\proks\OneDrive\Pulpit\databases\BartłomiejKozeraEFLab\BartłomiejKozeraEFLab> dotnet ef migrations add InitProductDatabase
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
```

```
Migrations
    20230329151658_InitProductDatabase.cs
    20230329151658_InitProductDatabase.Designer.cs
    ProcuctContextModelSnapshot.cs
```

```
PS C:\Users\proks\OneDrive\Pulpit\databases\BartłomiejKozeraEFLab\BartłomiejKozeraEFLab> dotnet ef database update
Build started...
Build succeeded.
Applying migration '20230329151658_InitProductDatabase'.
Done.
```

Tworzenie tabeli naszej bazy



4. Kod odpowiadający za pobranie danych o produktach

```
using System;
namespace BartlomiejKozeraProducts
{
    class Program
    {
        static void Main(String[] args)
        {
            ProcuctContext productContext = new ProcuctContext();
            Product product = new Product { ProductName = "Flamaster" };
            productContext.Products.Add(product);
            productContext.SaveChanges();

            var query = from prod in productContext.Products
                        select prod.ProductName;
        }

    }
}
```

Kod służący do dodania produktu, w tym przypadku na sztywno flamaster

5. Wypisywanie kodu na standardowe wyjście

```
using System;

namespace BartlomiejKozeraProducts
{
    class Program
    {
        static void Main(String[] args)
        {
            ProductContext productContext = new ProductContext();
            Product product = new Product { ProductName = "Flamaster" };
            productContext.Products.Add(product);
            productContext.SaveChanges();

            var query = from prod in productContext.Products
                        select prod.ProductName;
            foreach (var pName in query)
            {
                Console.WriteLine(pName);
            }
        }

    }
}
```

```
Flamaster

C:\Users\proks\OneDrive\Pulpit\databases\BartłomiejKozeraEFLab\BartłomiejKozeraEFLab\bin\Debug\net6.0\BartłomiejKozeraEF
Lab.exe (proces 19932) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Wypisywanie produktów na konsole

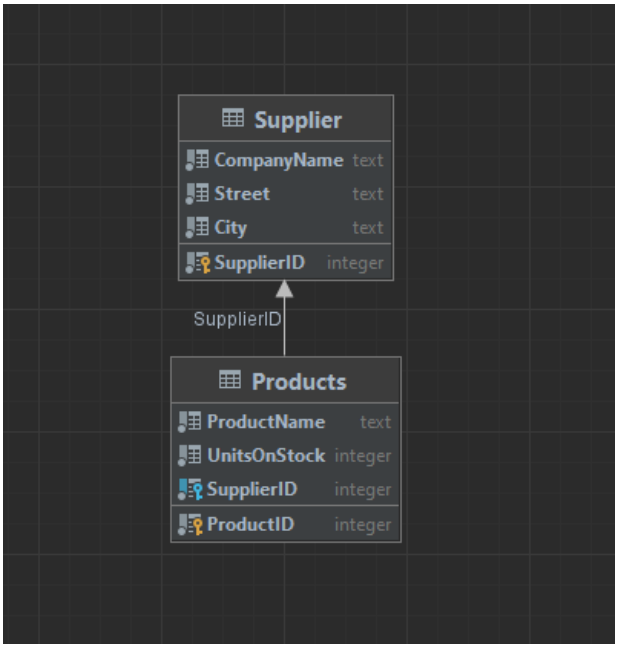
6. Dodawanie produktów do bazy

```
namespace BartlomiejKozeraProducts
{
    class Program
    {
        static void Main(String[] args)
        {

            Console.WriteLine("Podaj nazwę produktu: ");
            string ProductName = Console.ReadLine();

            Console.WriteLine("Lista produktów: ");
            ProductContext productContext = new ProductContext();
            Product product = new Product { ProductName = ProductName };
            productContext.Products.Add(product);
            productContext.SaveChanges();

            var query = from prod in productContext.Products
                        select prod.ProductName;
            foreach (var pName in query)
            {
                Console.WriteLine(pName);
            }
        }

    }
}
```

Dodawanie produktów podanych na wejściu przez użytkownika, oraz wypisywanie ich.

```
Podaj nazwę produktu:
Zeszyt
Lista produktów:
Flamaster
Kredki
Zeszyt

C:\Users\proks\OneDrive\Pulpit\databases\BartłomiejKozeraEFLab\BartłomiejKozeraEFLab\bin\Debug\net6.0\BartłomiejKozeraEF
Lab.exe (proces 5400) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

7. Relacja między produktami a dostawcami




```
select * from Supplier;
select * from Products;
```

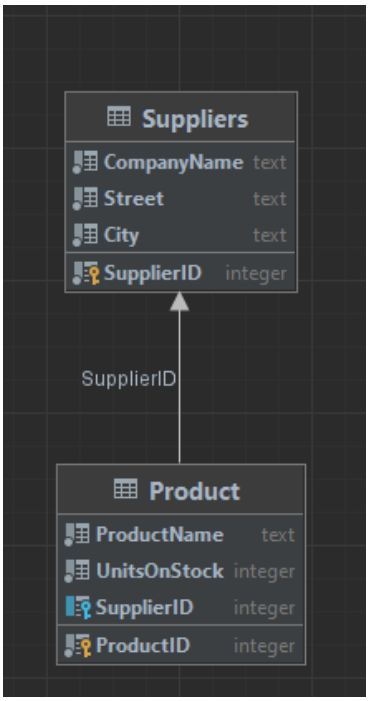
	ProductID	ProductName	UnitsOnStock	SupplierID
1	1	Flamaster	0	1
2	2	Kredki	0	1

	SupplierID	CompanyName	Street	City
1	1	DPD	Witolda Budryka 2	Krakow

8.Odwracanie relacji

```
using System;
namespace BartlomiejKozeraProducts
{
    public class Supplier
    {
        public int SupplierID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }

        public ICollection<Product> Products { get; set; }
    }
}
```

```
using System;
namespace BartlomiejKozeraProducts
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsOnStock { get; set; }
    }
}
```

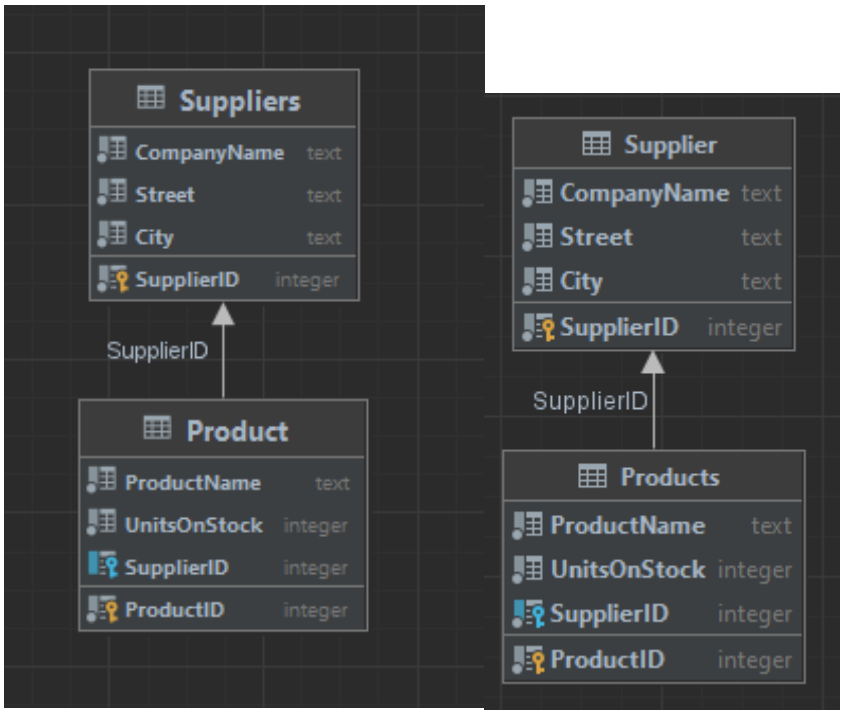



```
select * from Suppliers;
select * from Product;
```

	SupplierID	CompanyName	Street	City
1	1	DPD	Witolda Budryka 2	Kraków

	ProductID	ProductName	UnitsOnStock	SupplierID
1	1	Flamaster	0	1
2	2	Kredki	0	1

9.Obustronna relacja




```
select * from Suppliers;
select * from Product;
```

	SupplierID	CompanyName	Street	City
1	1	DPD	Witolda Budryka 2	Kraków

	ProductID	ProductName	UnitsOnStock	SupplierID
1	1	Flamaster	0	1
2	2	Kredki	0	1

```
select * from Supplier;
select * from Products;
```

	ProductID	ProductName	UnitsOnStock	SupplierID
1	1	Flamaster	0	1
2	2	Kredki	0	1

	SupplierID	CompanyName	Street	City
1	1	DPD	Witolda Budryka 2	Krakow

10. Relecja many to many

Tworze nową tabelę:

```
using System;
using System.ComponentModel.DataAnnotations;

namespace BartlomiejKozeraProducts
{
    public class Invoice
    {
        public Invoice()
        {
            this.Products = new HashSet<Product>();
        }
        public int InvoiceID { get; set; }
        [Required]
        public int invoiceNumber { get; set; }
        [Required]
        public int quantity { get; set; }
        public virtual ICollection<Product> Products { get; set; }
    }
}
```

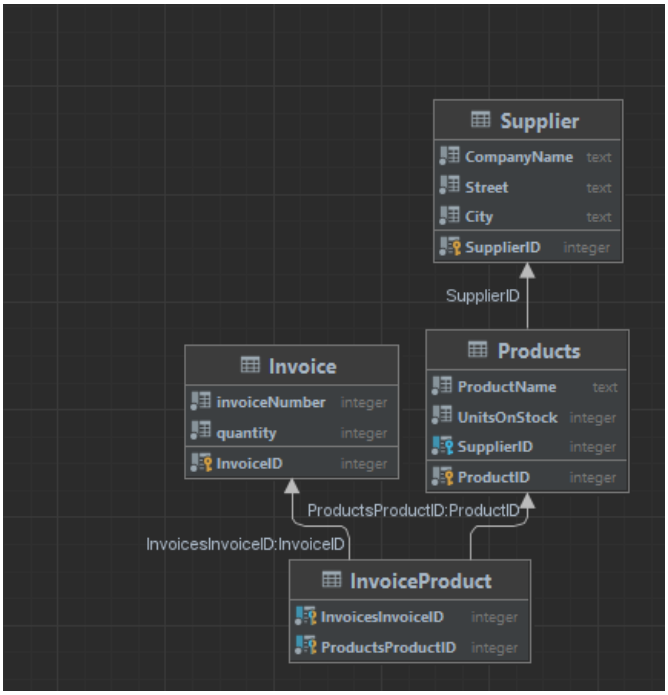
```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Sqlite;

namespace BartlomiejKozeraProducts
{
    public class InvoiceContext : DbContext
    {
        public DbSet<Invoice> Invoices { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("Datasource=InvoicesDatabase");
        }
    }
}
```

Oraz tabelę pomocniczą, przechowującą indeksy z tabel Products oraz Invoices

```
using BartlomiejKozeraProducts;
using Microsoft.EntityFrameworkCore;
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace BartlomiejKozeraProducts
{
    public class ProductInvoices
    {
        public int ProductInvoicesID { get; set; }
        [ForeignKey("Product")]
        public ICollection<Product> ProductsID { get; set; }
        [ForeignKey("Product")]
        public ICollection<Invoice> InvoicesID { get; set; }
    }
}
```

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Sqlite;

namespace BartlomiejKozeraProducts
{
    public class ProductInvoicesContext : DbContext
    {
        public DbSet<ProductInvoices> ProductInvoices { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("Datasource=ProductInvoicesDatabase");
        }
    }
}
```

Poniżej schemat:




```
select * from Invoice;
```

	InvoiceID	invoiceNumber	quantity	Product
1	1	78644432782110	5	2
2	2	78714865732761	10	1
3	3	41572135732157	3	3

```
select * from InvoiceProduct;
```

	InvoicesInvoiceID	ProductsProductID
1	1	2
2	2	1
3	3	3

11. Table-per-Hierarchy

Stworzyłem tabelę główną Company

```
using System;
using System.ComponentModel.DataAnnotations;

namespace BartlomiejKozeraProducts
{
    public class Company
    {
        public int CompanyID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public string ZipCode { get; set; }

        public ICollection<Product> Products { get; set; }
    }
}
```

Oraz jej widok kontekstowy:

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Sqlite;

namespace BartlomiejKozeraProducts
{
    public class CompanyContext : DbContext
    {
        public DbSet<Company> Companies { get; set; }
        public DbSet<Customer> Customers { get; set; }
        public DbSet<Supplier> Suppliers { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseSqlite("Datasource=CompaniesDatabase");
        }
    }
}
```

Wymagało to przerobienia klasy Suppliers

```
using System;
using System.ComponentModel.DataAnnotations;

namespace BartlomiejKozeraProducts
{
    public class Supplier : Company
    {
        public int BankAccountNumber { get; set; }
    }
}
```

Oraz dodania klasy Customer

```
using System;

namespace BartlomiejKozeraProducts
{
    public class Customer : Company
    {
        public double Discount { get; set; }

    }
}
```

Kod generujący dane z tabeli wygląda następująco:

```
IQueryable<Company> linqQuery2 = from b in companyContext.Companies select b;
List<Company> companiesDetails = linqQuery2.ToList();
foreach (Company name in companiesDetails)
{
    System.Console.WriteLine(name.CompanyName);
}
var linqQuery = from b in companyContext.Suppliers select b.CompanyName;

foreach(string name in linqQuery)
{
    System.Console.WriteLine(name);
}
IQueryable<Customer> query = from b in companyContext.Companies.OfType<Customer>()
                             select b;
foreach (Customer name in query)
{
    System.Console.WriteLine(name.CompanyName);
}
```

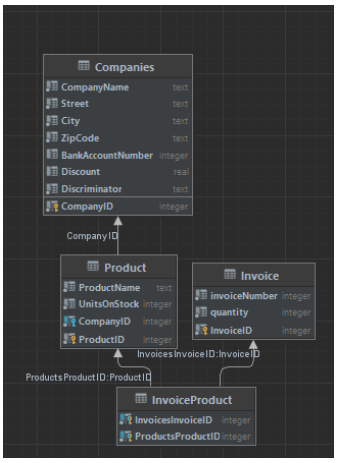
```
D:\databases\BartłomiejKozeraEFLab\BartłomiejKozeraEFLab\bin\Debug\net6.0\BartłomiejKozeraEFLab.exe
DPD
Audi HSO
Babyn
Gouda Gruchała
DPD
Audi HSO
Babyn
Gouda Gruchała
```

Do tabeli dodawałem dane za pomocą:

```
Customer company = new Customer{CompanyName = "Gouda Gruchałą", Street
companyContext.Customers.Add(company);
companyContext.SaveChanges();
```

Przy czym, przy dodawaniu obiektów typu Supplier typ zmiennej zmienia się na Supplier

Poniżej schemat bazy.



12. Table-per-Type

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace BartlomiejKozeraProducts
{
    [Table("Supplier")]
    public class Supplier : Company
    {
        public int BankAccountNumber { get; set; }
    }
}
```

```
namespace BartlomiejKozeraProducts
{
    [Table("Customer")]
    public class Customer : Company
    {
        public double Discount { get; set; }

    }
}
```

```
namespace BartlomiejKozeraProducts
{
    public abstract class Company
    {
        public int CompanyID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public string ZipCode { get; set; }

        public ICollection<Product> Products { get; set; }
    }
}
```