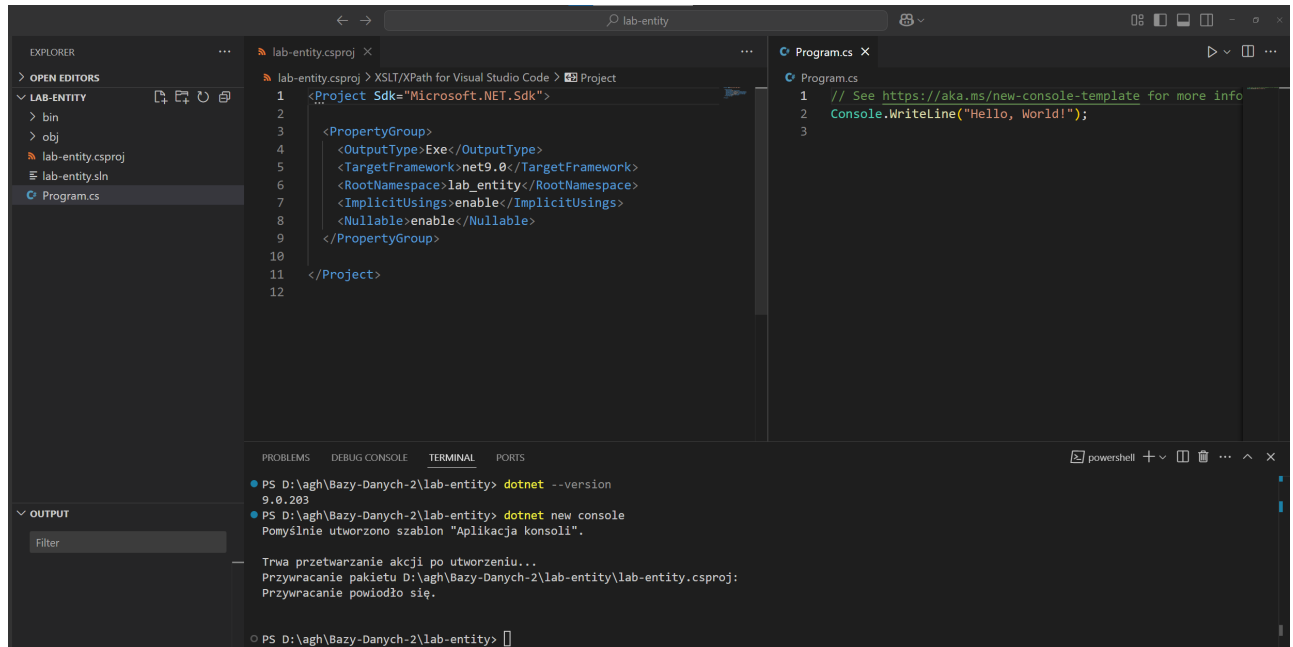


# Sprawozdanie z Entity Framework

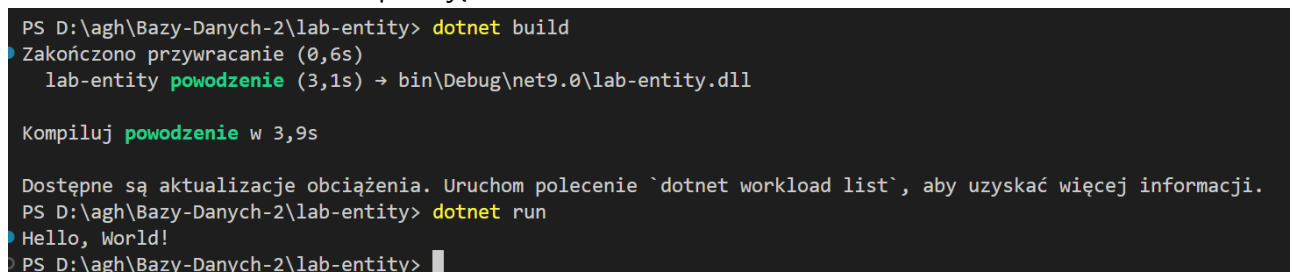
Szymon Żuk

## Część 1

1. Sprawdziłem wersję dotnet i utworzyłem projekt:



2. Zbudowałem i uruchomiłem aplikację:



3. Dodałem klasę Product

```
public class Product
{
    public int ProductID { get; set; }
    public string? ProductName { get; set; }
    public int UnitsInStock { get; set; }
}
```

4. Dodałem klasę ProdContext

```
public class ProdContext : DbContext { }
```

## 5. Dostałem błąd podczas budowania aplikacji

```
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet build
Zakończono przywracanie (0,6s)
  lab-entity powodzenie (3,1s) → bin\Debug\net9.0\lab-entity.dll

Zakończono przywracanie (0,5s)
  lab-entity zakończono niepowodzeniem, z błędami w liczbie: 1 (0,4s)
  D:\agh\Bazy-Danych-2\lab-entity\ProdContext.cs(1,28): error CS0246: Nie można znaleźć nazwy typu lub przestrzeni nazw „DbContext” (brak dyrektywy using lub odwołania do zestawu?)

Kompiluj zakończono niepowodzeniem, z błędami w liczbie: 1 w 1,3s

Dostępne są aktualizacje obciążenia. Uruchom polecenie `dotnet workload list`, aby uzyskać więcej informacji.
PS D:\agh\Bazy-Danych-2\lab-entity>
```

## 6. Dodałem using do ProdContext.cs

```
using Microsoft.EntityFrameworkCore;

public class ProdContext : DbContext { }
```

## 7. Dostałem inny błąd podczas budowania aplikacji:

```
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet build
Zakończono przywracanie (0,5s)
  lab-entity zakończono niepowodzeniem, z błędami w liczbie: 2 (0,3s)
  D:\agh\Bazy-Danych-2\lab-entity\ProdContext.cs(1,17): error CS0234: Typ lub przestrzeń nazw „EntityFrameworkCore” nie występuje w przestrzeni nazw „Microsoft” (czy nie brakuje odwołania do zestawu?)
  D:\agh\Bazy-Danych-2\lab-entity\ProdContext.cs(3,28): error CS0246: Nie można znaleźć nazwy typu lub przestrzeni nazw „DbContext” (brak dyrektywy using lub odwołania do zestawu?)

Kompiluj zakończono niepowodzeniem, z błędami w liczbie: 2 w 1,2s

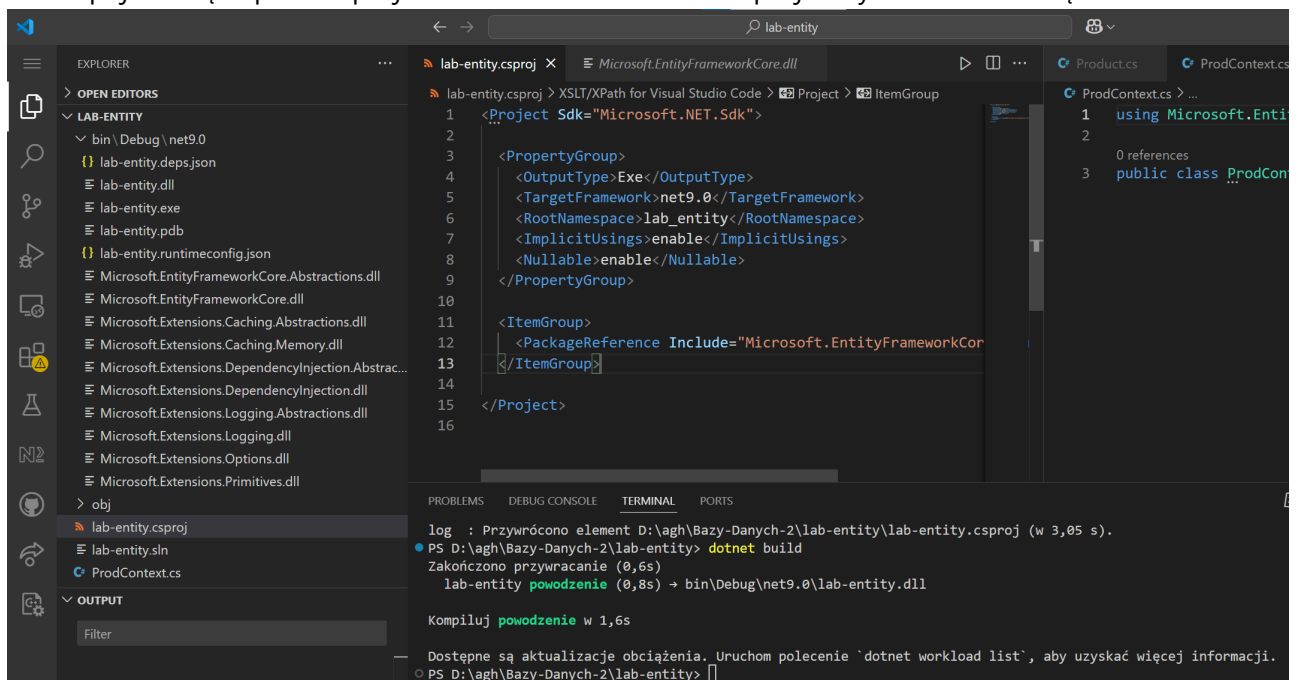
Dostępne są aktualizacje obciążenia. Uruchom polecenie `dotnet workload list`, aby uzyskać więcej informacji.
PS D:\agh\Bazy-Danych-2\lab-entity>
```

## 8. Dodałem pakiet EntityFrameworkCore (używam najnowszej wersji dotneta więc nie musiałem podawać wersji pakietu)

```
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet add package Microsoft.EntityFrameworkCore

Kompiluj powodzenie w 0,6s
info : Walidacja łańcucha certyfikatów X.509 użyje domyślnego magazynu zaufania wybranego przez .NET do podpisywania kodu.
info : Walidacja łańcucha certyfikatów X.509 będzie korzystać z domyślnego magazynu zaufania wybranego przez .NET do oznaczania czasu.
info : Dodawanie elementu PackageReference dla pakietu „Microsoft.EntityFrameworkCore” do projektu „D:\agh\Bazy-Danych-2\lab-entity\lab-entity.csproj”.
info : GET https://api.nuget.org/v3/registrations5-gz-sewer2/microsoft.entityframeworkcore/index.json
```

## 9. Pakiet pojawił się w pliku .csproj i folderze bin. Zbudowałem projekt tym razem bez błędów



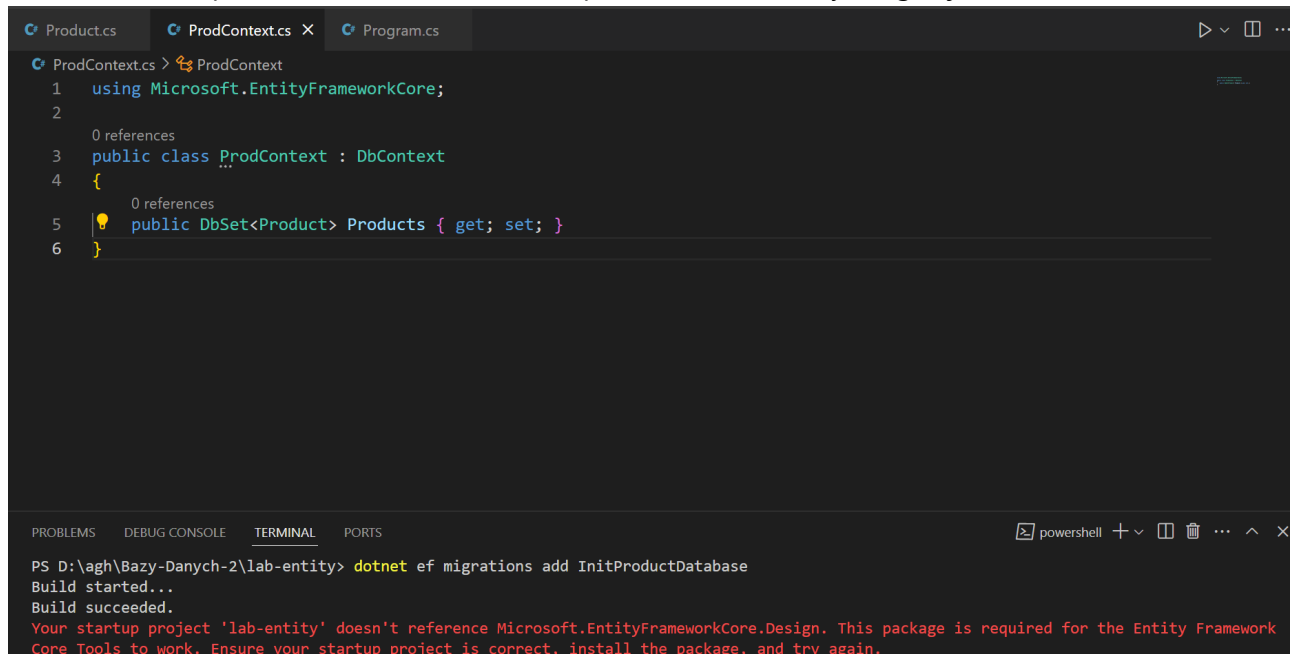
```
lab-entity.csproj X Microsoft.EntityFrameworkCore.dll
lab-entity.csproj > XSLT/XPath for Visual Studio Code > Project > ItemGroup
1 <Project Sdk="Microsoft.NET.Sdk">
2
3
4 <PropertyGroup>
5   <OutputType>Exe</OutputType>
6   <TargetFramework>net9.0</TargetFramework>
7   <RootNamespace>lab_entity</RootNamespace>
8   <ImplicitUsings>enable</ImplicitUsings>
9   <Nullable>enable</Nullable>
10 </PropertyGroup>
11
12 <ItemGroup>
13   <PackageReference Include="Microsoft.EntityFrameworkCore" />
14 </ItemGroup>
15 </Project>
16

PROBLEMS DEBUG CONSOLE TERMINAL PORTS
log : Przywrócono element D:\agh\Bazy-Danych-2\lab-entity\lab-entity.csproj (w 3,05 s).
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet build
Zakończono przywracanie (0,6s)
  lab-entity powodzenie (0,8s) → bin\Debug\net9.0\lab-entity.dll

Kompiluj powodzenie w 1,6s

Dostępne są aktualizacje obciążenia. Uruchom polecenie `dotnet workload list`, aby uzyskać więcej informacji.
PS D:\agh\Bazy-Danych-2\lab-entity>
```

## 10. Dodałem DbSet produktów do ProdContext i spróbowałem utworzyć migrację, ale dostałem błąd



```

ProdContext.cs > ProdContext
1  using Microsoft.EntityFrameworkCore;
2
3  0 references
4  public class ProdContext : DbContext
5  {
6      0 references
7      public DbSet<Product> Products { get; set; }
8  }

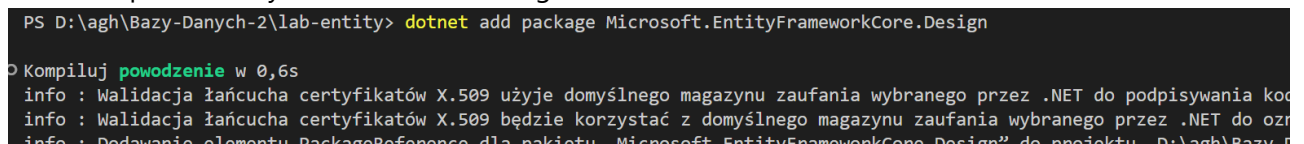
```

```

PS D:\agh\Bazy-Danych-2\lab-entity> dotnet ef migrations add InitProductDatabase
Build started...
Build succeeded.
Your startup project 'lab-entity' doesn't reference Microsoft.EntityFrameworkCore.Design. This package is required for the Entity Framework
Core Tools to work. Ensure your startup project is correct, install the package, and try again.

```

## 11. Dodałem pakiet EntityFrameworkCore.Design

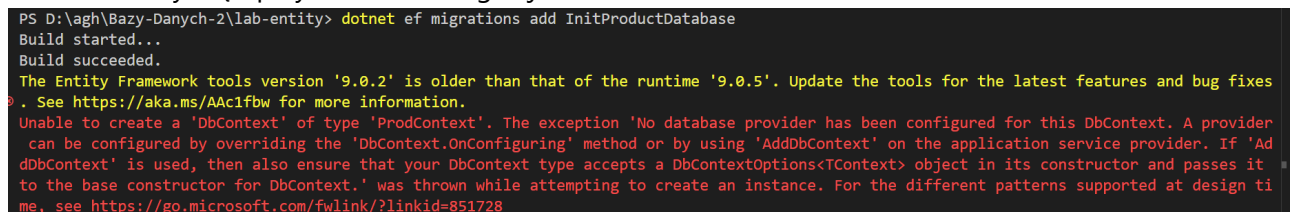


```

PS D:\agh\Bazy-Danych-2\lab-entity> dotnet add package Microsoft.EntityFrameworkCore.Design
Kompiluj powodzenie w 0,6s
info : Walidacja łańcucha certyfikatów X.509 użyje domyślnego magazynu zaufania wybranego przez .NET do podpisywania kodu
info : Walidacja łańcucha certyfikatów X.509 będzie korzystać z domyślnego magazynu zaufania wybranego przez .NET do odczytu
info : Dodawanie elementu PackageReference dla pakietu "Microsoft.EntityFrameworkCore.Design" do projektu "D:\agh\Bazy-Danych-2\lab-entity\lab-entity.csproj".

```

## 12. Dostałem inny błąd przy tworzeniu migracji



```

PS D:\agh\Bazy-Danych-2\lab-entity> dotnet ef migrations add InitProductDatabase
Build started...
Build succeeded.
The Entity Framework tools version '9.0.2' is older than that of the runtime '9.0.5'. Update the tools for the latest features and bug fixes
. See https://aka.ms/Aac1fbw for more information.
Unable to create a 'DbContext' of type 'ProdContext'. The exception 'No database provider has been configured for this DbContext. A provider
can be configured by overriding the 'DbContext.OnConfiguring' method or by using 'AddDbContext' on the application service provider. If 'Add
DbContext' is used, then also ensure that your DbContext type accepts a DbContextOptions<TContext> object in its constructor and passes it
to the base constructor for DbContext.' was thrown while attempting to create an instance. For the different patterns supported at design ti
me, see https://go.microsoft.com/fwlink/?linkid=851728

```

## 13. Dodałem przeładowanie metody onConfiguring z bazą danych Sqlite

```

using Microsoft.EntityFrameworkCore;

public class ProdContext : DbContext
{
    public DbSet<Product> Products { get; set; }

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

```

```
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet build
Zakończono przywracanie (0,6s)
  lab-entity zakończono niepowodzeniem, z błędami w liczbie: 1 (0,4s)
    D:\agh\Bazy-Danych-2\lab-entity\ProdContext.cs(10,24): error CS1061: Element „DbContextOptionsBuilder” nie zawiera definicji „UseSqlite” i nie odnaleziono dostępnej metody rozszerzenia „UseSqlite”, która przyjmuje pierwszy argument typu „DbContextOptionsBuilder” (czy nie brakuje dyrektywy using lub odwołania do zestawu?).

Kompiluj zakończono niepowodzeniem, z błędami w liczbie: 1 w 1,3s

Dostępne są aktualizacje obciążenia. Uruchom polecenie `dotnet workload list`, aby uzyskać więcej informacji.
```

```
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet add package Microsoft.EntityFrameworkCore.Sqlite
```

Kompiluj **powodzenie** w 0,6s

info : Walidacja łańcucha certyfikatów X.509 użyje domyślnego magazynu zaufania wybranego przez .NET do podpisywania kodu.  
info : Walidacja łańcucha certyfikatów X.509 będzie korzystać z domyślnego magazynu zaufania wybranego przez .NET do oznaczania czasu.

The screenshot shows the Visual Studio IDE with the following components:

- Explorer:** The 'Migrations' folder is expanded, showing files like '20250531192119\_InitProductDatabase.cs', '20250531192119\_InitProductDatabase.Designer.cs', and 'ProdContextModelSnapshot.cs'.
- Editor:** The 'ProdContextModelSnapshot.cs' file is open, showing the following code:
 

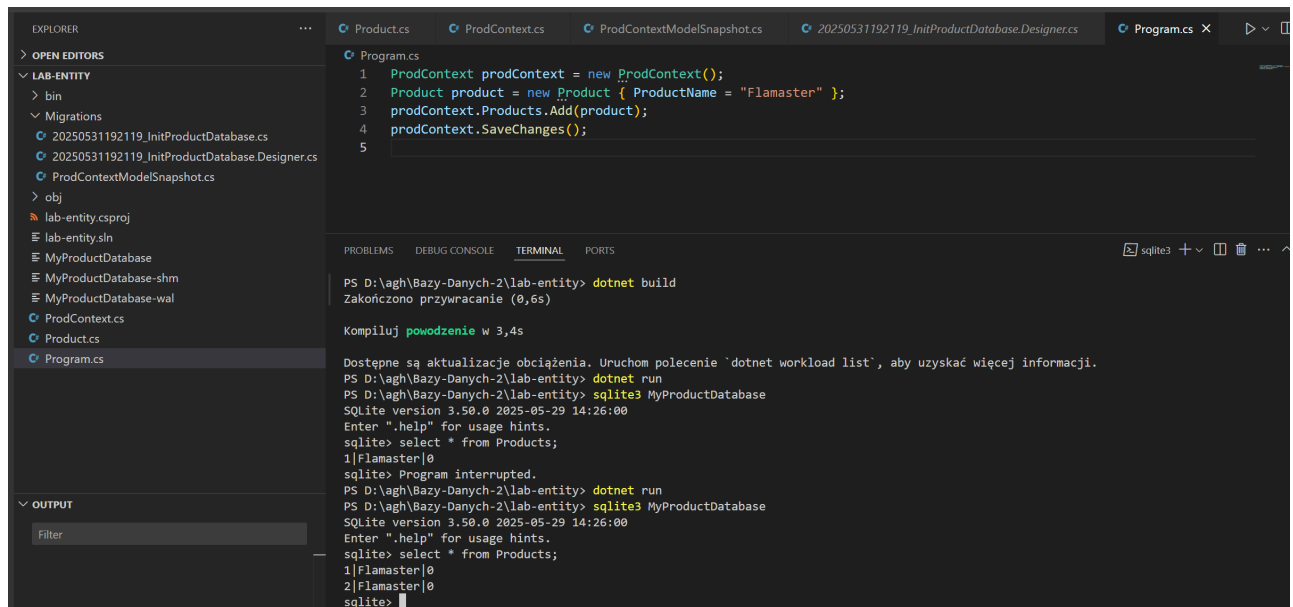
```

1 // <auto-generated />
2 using Microsoft.EntityFrameworkCore;
3 using Microsoft.EntityFrameworkCore.Infrastructure;
4 using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
5
6 #nullable disable
7
8 namespace lab_entity.Migrations
9 {
10     [DbContext(typeof(ProdContext))]
11     partial class ProdContextModelSnapshot : ModelSnapshot

```
- Terminal:** The terminal shows the command 'dotnet ef migrations add InitProductDatabase' being executed. The output indicates that the Entity Framework tools version '9.0.2' is older than the runtime '9.0.5', suggesting an update. The command successfully adds the 'InitProductDatabase' migration.

```
PS D:\agh\Bazy-Danych-2\lab-entity> sqlite3 MyProductDatabase
SQLite version 3.50.0 2025-05-29 14:26:00
Enter ".help" for usage hints.
sqlite> .tables
Products                __EFMigrationsHistory  __EFMigrationsLock
sqlite> .schema Products
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "ProductName" TEXT NULL,
    "UnitsInStock" INTEGER NOT NULL
);
sqlite>
```

4 / 23



```
Product.cs ProdContext.cs ProdContextModelSnapshot.cs 20250531192119_InitProductDatabase.Designer.cs Program.cs
```

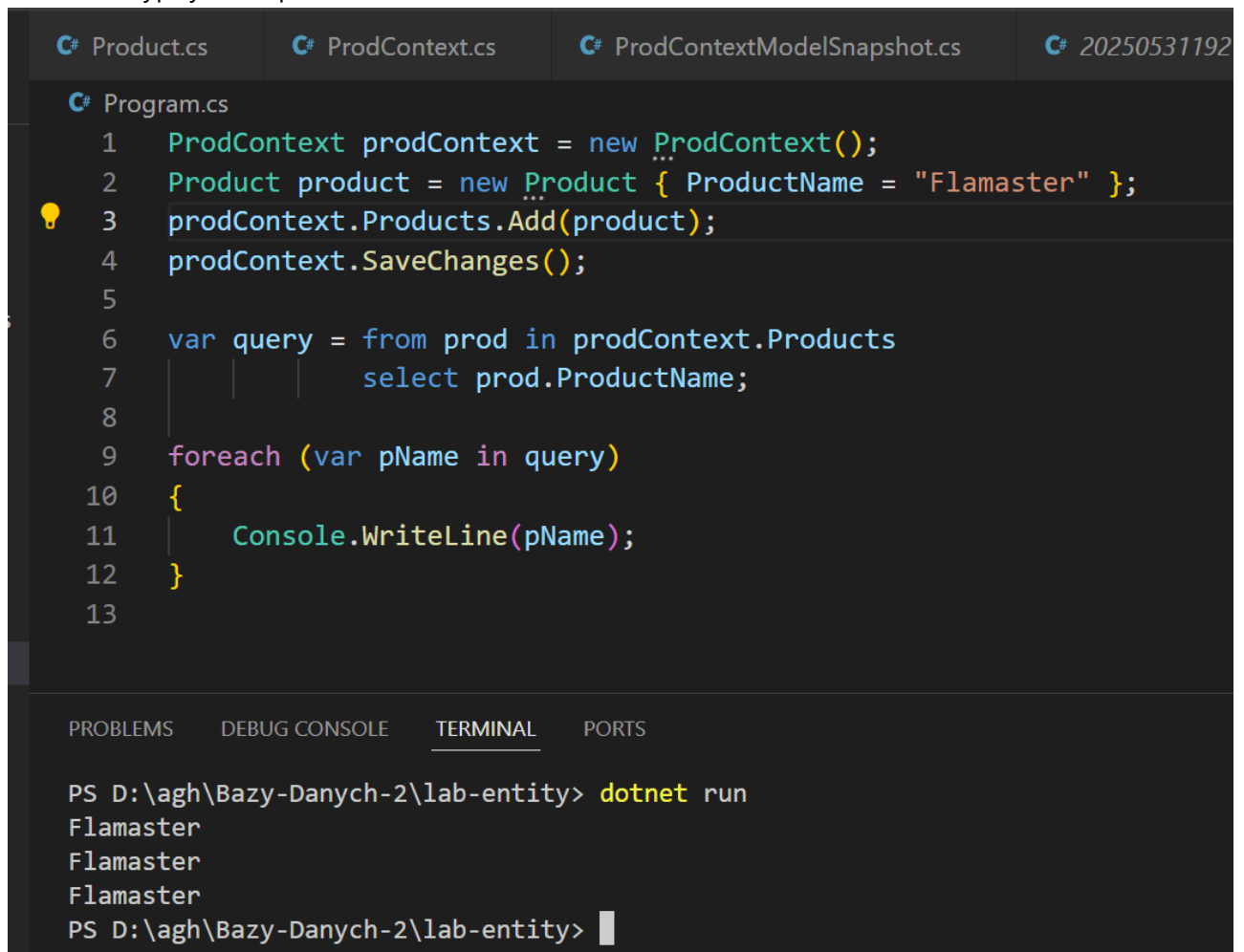
```
1 ProdContext prodContext = new ProdContext();
2 Product product = new Product { ProductName = "Flamaster" };
3 prodContext.Products.Add(product);
4 prodContext.SaveChanges();
5
```

```
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet build
Zakończono przywracanie (0,6s)

Kompiluj powodzenie w 3,4s

Dostępne są aktualizacje obciążenia. Uruchom polecenie 'dotnet workload list', aby uzyskać więcej informacji.
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet run
PS D:\agh\Bazy-Danych-2\lab-entity> sqlite3 MyProductDatabase
SQLite version 3.50.0 2025-05-29 14:26:00
Enter ".help" for usage hints.
sqlite> select * from Products;
1|Flamaster|0
sqlite> Program interrupted.
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet run
PS D:\agh\Bazy-Danych-2\lab-entity> sqlite3 MyProductDatabase
SQLite version 3.50.0 2025-05-29 14:26:00
Enter ".help" for usage hints.
sqlite> select * from Products;
1|Flamaster|0
2|Flamaster|0
sqlite>
```

19. Dodałem wypisywanie produktów z tabeli



```
Product.cs ProdContext.cs ProdContextModelSnapshot.cs 20250531192
```

```
1 ProdContext prodContext = new ProdContext();
2 Product product = new Product { ProductName = "Flamaster" };
3 prodContext.Products.Add(product);
4 prodContext.SaveChanges();
5
6 var query = from prod in prodContext.Products
7             select prod.ProductName;
8
9 foreach (var pName in query)
10 {
11     Console.WriteLine(pName);
12 }
13
```

```
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet run
Flamaster
Flamaster
Flamaster
PS D:\agh\Bazy-Danych-2\lab-entity>
```

20. Dodałem zapytanie o nazwę produktu i produkty Kredki i Krzesło

```

Program.cs
1  ProductContext prodContext = new ProductContext();
2
3  Console.WriteLine("Podaj nazwę produktu: ");
4  String? prodName = Console.ReadLine();
5
6  Product product = new Product { ProductName = prodName };
7  prodContext.Products.Add(product);
8  prodContext.SaveChanges();
9
10 var query = from prod in prodContext.Products
11              select prod.ProductName;
12
13 foreach (var pName in query)
14 {
15     Console.WriteLine(pName);
16 }

```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS

```

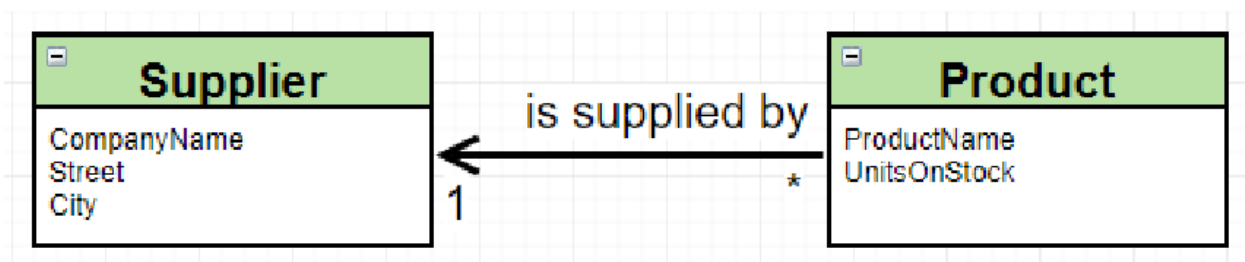
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet run
Podaj nazwę produktu:
Krzeseł
Flamaster
Flamaster
Flamaster
Krzeseł
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet run
Podaj nazwę produktu:
Kredki
Flamaster
Flamaster
Flamaster
Krzeseł
Kredki
PS D:\agh\Bazy-Danych-2\lab-entity>

```

## Część 2

Podpunkt a)

a. Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



- Stwórz nowego dostawcę.
- Znajdź poprzednio wprowadzony produkt i ustaw jego dostawcę na właśnie dodanego.
- Udokumentuj wykonane kroki oraz uzyskany rezultat (.schema table/diagram, select \* from....)

Modyfikacje modelu danych

```

// Dodałem klasę dostawcy
public class Supplier
{

```

```

    public int SupplierID { get; set; }
    public string? CompanyName { get; set; }
    public string? Street { get; set; }
    public string? City { get; set; }
}

```

```

public class ProdContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    // Dodałem DbSet dostawców
    public DbSet<Supplier> Suppliers { get; set; }

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

```

```

public class Product
{
    public int ProductID { get; set; }
    public string? ProductName { get; set; }
    public int UnitsInStock { get; set; }
    // Dodałem pole tworzące relację
    public Supplier? IsSuppliedBy { get; set; }
}

```

Wynik zapytania .schema po dodaniu migracji i aktualizacji bazy danych

```

CREATE TABLE IF NOT EXISTS "Suppliers" (
  "SupplierID" INTEGER NOT NULL CONSTRAINT "PK_Suppliers" PRIMARY KEY AUTOINCREMENT,
  "CompanyName" TEXT NULL,
  "Street" TEXT NULL,
  "City" TEXT NULL
);
CREATE TABLE IF NOT EXISTS "Products" (
  "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
  "IsSuppliedBySupplierID" INTEGER NULL,
  "ProductName" TEXT NULL,
  "UnitsInStock" INTEGER NOT NULL,
  CONSTRAINT "FK_Products_Suppliers_IsSuppliedBySupplierID" FOREIGN KEY ("IsSuppliedBySupplierID") REFERENCES "Suppliers" ("SupplierID")
);
CREATE INDEX "IX_Products_IsSuppliedBySupplierID" ON "Products" ("IsSuppliedBySupplierID");

```

Kod dodający dostawcę, znajdujący wcześniej dodany produkt i ustawiający jego dostawcę na nowo dodanego

```

var prodContext = new ProdContext();
// Dodanie nowego dostawcy
var supplier = new Supplier
{
    CompanyName = "Inpost",

```

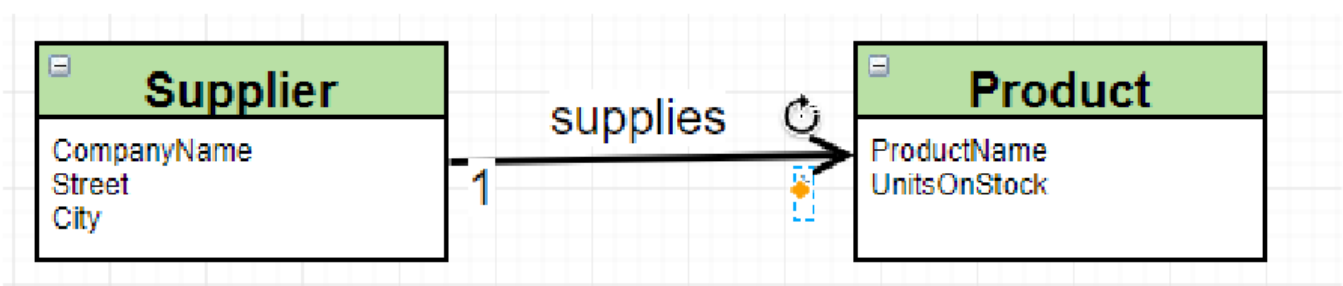
```
Street = "Kawiorzy",  
City = "Kraków"  
};  
prodContext.Suppliers.Add(supplier);  
// Znalezienie wcześniej dodanego produktu w bazie danych  
var query = from prod in prodContext.Products  
             where prod.ProductName == "Kredki"  
             select prod;  
var product = query.First();  
// Ustawienie dostawcy  
product.IsSuppliedBy = supplier;  
prodContext.SaveChanges();
```

Po wykonaniu kodu zmiany są widoczne w bazie danych (produkt Kredki ma ustawioną wartość kolumny "1", czyli ID dostawcy)

```
sqlite> select * from Suppliers;  
1|Inpost|Kawiorzy|Kraków  
sqlite> select * from Products;  
6||Flamaster|0  
7||Flamaster|0  
8||Flamaster|0  
9||Krzeseł|0  
10|1|Kredki|0  
sqlite> █
```

Podpunkt b)

b. Odwróć relację zgodnie z poniższym schematem



- i. Stwórz kilka produktów
- ii. Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę
- iii. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (.schema table/diagram z datagrip, select \* from....)**



## Modyfikacje modelu danych

```
public class Product
{
    public int ProductID { get; set; }
    public string? ProductName { get; set; }
    public int UnitsInStock { get; set; }
    // Usunąłem pole IsSuppliedBy
    // public Supplier? IsSuppliedBy { get; set; }
}
```

```
public class Supplier
{
    public int SupplierID { get; set; }
    public string? CompanyName { get; set; }
    public string? Street { get; set; }
    public string? City { get; set; }
    // Dodałem pole tworzące relację
    public ICollection<Product> Supplies { get; set; } = [];
}
```

## Wynik zapytania .schema po dodaniu migracji i aktualizacji bazy danych

```
CREATE TABLE IF NOT EXISTS "Suppliers" (
    "SupplierID" INTEGER NOT NULL CONSTRAINT "PK_Suppliers" PRIMARY KEY AUTOINCREMENT,
    "CompanyName" TEXT NULL,
    "Street" TEXT NULL,
    "City" TEXT NULL
);
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "ProductName" TEXT NULL,
    "SupplierID" INTEGER NULL,
    "UnitsInStock" INTEGER NOT NULL,
    CONSTRAINT "FK_Products_Suppliers_SupplierID" FOREIGN KEY ("SupplierID") REFERENCES "Suppliers" ("SupplierID")
);
CREATE INDEX "IX_Products_SupplierID" ON "Products" ("SupplierID");
```

Kod dodający kilka produktów, znajdujący wcześniej dodanego dostawcę i dodający produkty jako dostarczane przez tego dostawcę

```
var prodContext = new ProdContext();
// Dodanie kilku produktów
List<Product> products = [
    new Product
    {
        ProductName = "Ołówek",
        UnitsInStock = 4000
    },
    new Product
    {
        ProductName = "Laptop",
```

```
        UnitsInStock = 4
    },
    new Product
    {
        ProductName = "Zeszyt",
        UnitsInStock = 17
    }
    ];
foreach (var p in products)
{
    prodContext.Products.Add(p);
}

// Znalezienie wcześniej dodanego dostawcy
var query = from supp in prodContext.Suppliers
             where supp.CompanyName == "Inpost"
             select supp;
var supplier = query.First();

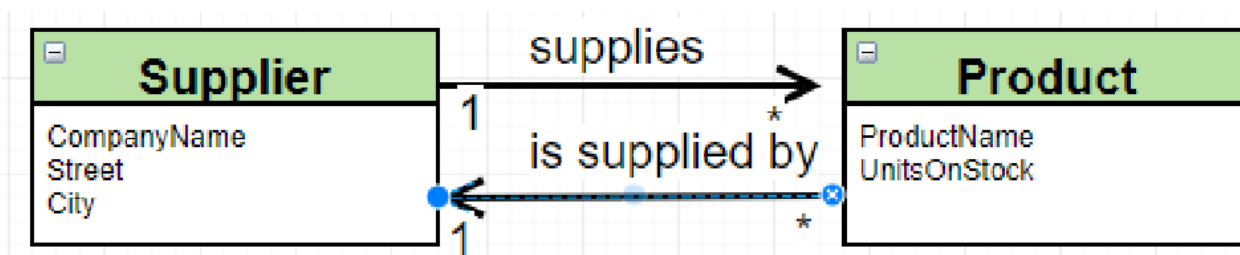
// Dodanie produktów jako dostarczanych przez tego dostawcę
foreach (var p in products)
{
    supplier.Supplies.Add(p);
}
prodContext.SaveChanges();
```

Po wykonaniu kodu zmiany są widoczne w bazie danych (nowe produkty mają ustawioną wartość kolumny "1", czyli ID dostawcy)

```
sqlite> select * from Suppliers;
1|Inpost|Kawiory|Kraków
sqlite> select * from Products;
6|Flamaster||0
7|Flamaster||0
8|Flamaster||0
9|Krzes||0
10|Kredki|1|0
11|Ołówek|1|4000
12|Laptop|1|4
13|Zeszyt|1|17
sqlite> 
```

Podpunkt c)

c. Zamodeluj relację dwustronną jak poniżej:



- i. Tradycyjnie: Stwórz kilka produktów
- ii. Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę
- iii. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (.schema table/diagram z datagrip, select \* from....)**

Modyfikacje modelu danych

```

public class Product
{
    public int ProductID { get; set; }
    public string? ProductName { get; set; }
    public int UnitsInStock { get; set; }
    // Dodałem wcześniej usunięte pole tworzące relację
    public Supplier? IsSuppliedBy { get; set; }
}
  
```

Wynik zapytania .schema po dodaniu migracji i aktualizacji bazy danych

```

CREATE TABLE IF NOT EXISTS "Suppliers" (
    "SupplierID" INTEGER NOT NULL CONSTRAINT "PK_Suppliers" PRIMARY KEY AUTOINCREMENT,
    "CompanyName" TEXT NULL,
    "Street" TEXT NULL,
    "City" TEXT NULL
);
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "IsSuppliedBySupplierID" INTEGER NULL,
    "ProductName" TEXT NULL,
    "UnitsInStock" INTEGER NOT NULL,
    CONSTRAINT "FK_Products_Suppliers_IsSuppliedBySupplierID" FOREIGN KEY ("IsSuppliedBySupplierID") REFERENCES "Suppliers" ("SupplierID")
);
  
```

Kod dodający kilka produktów, znajdujący wcześniej dodanego dostawcę, dodający produkty jako dostarczane przez tego dostawcę i ustawiający dostawcę dla każdego produktu

```

var prodContext = new ProdContext();
// Dodanie kilku produktów
List<Product> products = [
    new Product
    {
        ProductName = "Klawiatura",
        UnitsInStock = 35
    },
    new Product
  
```

```
{
    ProductName = "Myszka",
    UnitsInStock = 22
},
new Product
{
    ProductName = "Zasilacz",
    UnitsInStock = 0
}];
foreach (var p in products)
{
    prodContext.Products.Add(p);
}

// Znalezienie wcześniej dodanego dostawcy
var query = from supp in prodContext.Suppliers
             where supp.CompanyName == "Inpost"
             select supp;
var supplier = query.First();

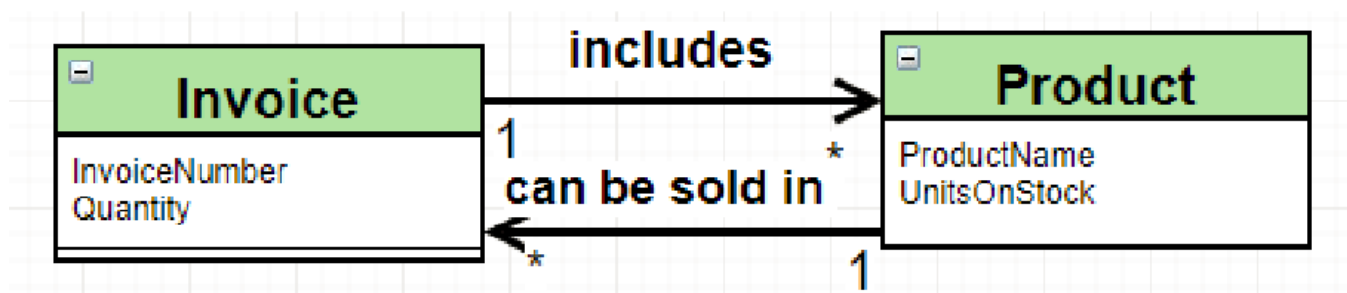
// Dodanie produktów jako dostarczanych przez tego dostawcę i ustawienie dostawcy
dla każdego produktu
foreach (var p in products)
{
    supplier.Supplies.Add(p);
    p.IsSuppliedBy = supplier;
}
prodContext.SaveChanges();
```

Po wykonaniu kodu zmiany są widoczne w bazie danych (nowe produkty mają ustawioną wartość kolumny "1", czyli ID dostawcy)

```
sqlite> select * from Products;
6||Flamaster|0
7||Flamaster|0
8||Flamaster|0
9||Krzes|0
10|1|Kredki|0
11|1|Ołówek|4000
12|1|Laptop|4
13|1|Zeszyt|17
17|1|Klawiatura|35
18|1|Myszka|22
19|1|Zasilacz|0
sqlite> 
```

## Podpunkt d)

d. Zamodeluj relację wiele-do-wielu, jak poniżej:



- i. Stórz kilka produktów i “sprzedaj” je na kilku transakcjach.
- ii. Pokaż produkty sprzedane w ramach wybranej faktury/transakcji
- iii. Pokaż faktury, w ramach których sprzedany został wybrany produkt
- iv. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (.schema table/diagram z datagrip, select \* from....)**

## Modyfikacje modelu danych

```

// Dodałem klasę faktury
public class Invoice
{
    public int InvoiceID { get; set; }
    public int InvoiceNumber { get; set; }
    public int Quantity { get; set; }
    public ICollection<Product> Includes { get; set; } = [];
}
  
```

```

public class ProdContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<Supplier> Suppliers { get; set; }
    // Dodałem DbSet faktur
    public DbSet<Invoice> Invoices { get; set; }

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

```
public class Product
{
    public int ProductID { get; set; }
    public string? ProductName { get; set; }
    public int UnitsInStock { get; set; }
    public Supplier? IsSuppliedBy { get; set; }
    // Dodałem pole tworzące relację
    public ICollection<Invoice> CanBeSoldIn { get; set; } = [];
}
```

Wynik zapytania .schema po dodaniu migracji i aktualizacji bazy danych. Oprócz nowej tabeli Invoice pojawiła się tabela InvoiceProduct modelująca relację wiele do wielu

```
CREATE TABLE IF NOT EXISTS "Suppliers" (
    "SupplierID" INTEGER NOT NULL CONSTRAINT "PK_Suppliers" PRIMARY KEY AUTOINCREMENT,
    "CompanyName" TEXT NULL,
    "Street" TEXT NULL,
    "City" TEXT NULL
);
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "IsSuppliedBySupplierID" INTEGER NULL,
    "ProductName" TEXT NULL,
    "UnitsInStock" INTEGER NOT NULL,
    CONSTRAINT "FK_Products_Suppliers_IsSuppliedBySupplierID" FOREIGN KEY ("IsSuppliedBySupplierID") REFERENCES "Suppliers" ("SupplierID")
);
CREATE INDEX "IX_Products_IsSuppliedBySupplierID" ON "Products" ("IsSuppliedBySupplierID");
CREATE TABLE IF NOT EXISTS "Invoices" (
    "InvoiceID" INTEGER NOT NULL CONSTRAINT "PK_Invoices" PRIMARY KEY AUTOINCREMENT,
    "InvoiceNumber" INTEGER NOT NULL,
    "Quantity" INTEGER NOT NULL
);
CREATE TABLE IF NOT EXISTS "InvoiceProduct" (
    "CanBeSoldInInvoiceID" INTEGER NOT NULL,
    "IncludesProductID" INTEGER NOT NULL,
    CONSTRAINT "PK_InvoiceProduct" PRIMARY KEY ("CanBeSoldInInvoiceID", "IncludesProductID"),
    CONSTRAINT "FK_InvoiceProduct_Invoices_CanBeSoldInInvoiceID" FOREIGN KEY ("CanBeSoldInInvoiceID") REFERENCES "Invoices" ("InvoiceID") ON DELETE CASCADE,
    CONSTRAINT "FK_InvoiceProduct_Products_IncludesProductID" FOREIGN KEY ("IncludesProductID") REFERENCES "Products" ("ProductID") ON DELETE CASCADE
);
CREATE INDEX "IX_InvoiceProduct_IncludesProductID" ON "InvoiceProduct" ("IncludesProductID");
```

Kod dodający produkty i faktury, znajdujący produkty sprzedane w ramach danej faktury, i faktury, w których sprzedano dany produkt

```
var prodContext = new ProdContext();

var tulipan = new Product
{
    ProductName = "Tulipan",
    UnitsInStock = 20,
};
var fiolek = new Product
{
    ProductName = "Fiołek",
    UnitsInStock = 5
};
var storczyk = new Product
{
    ProductName = "Storczyk",
    UnitsInStock = 3
};
var roza = new Product
{
```

```
        ProductName = "Róża",
        UnitsInStock = 2
    };
    prodContext.Products.AddRange(tulipan, fiolek, storczyk, roza);

    var invoice1 = new Invoice
    {
        InvoiceNumber = 1,
        Includes = [tulipan, fiolek, storczyk, roza]
    };
    tulipan.CanBeSoldIn.Add(invoice1);
    fiolek.CanBeSoldIn.Add(invoice1);
    storczyk.CanBeSoldIn.Add(invoice1);
    roza.CanBeSoldIn.Add(invoice1);

    var invoice2 = new Invoice
    {
        InvoiceNumber = 2,
        Includes = [tulipan, fiolek, storczyk]
    };
    tulipan.CanBeSoldIn.Add(invoice2);
    fiolek.CanBeSoldIn.Add(invoice2);
    storczyk.CanBeSoldIn.Add(invoice2);

    var invoice3 = new Invoice
    {
        InvoiceNumber = 3,
        Includes = [tulipan]
    };
    tulipan.CanBeSoldIn.Add(invoice3);

    var invoice4 = new Invoice
    {
        InvoiceNumber = 3,
        Includes = [fiolek, roza]
    };
    fiolek.CanBeSoldIn.Add(invoice4);
    roza.CanBeSoldIn.Add(invoice4);

    prodContext.Invoices.AddRange(invoice1, invoice2, invoice3, invoice4);
    prodContext.SaveChanges();

    // Pokazanie produktów sprzedanych w ramach faktury nr. 1
    var query1 = from inv in prodContext.Invoices
                  where inv.InvoiceNumber == 1
                  select inv.Includes;
    var products = query1.First();
    Console.WriteLine("Produkty sprzedane w ramach faktury nr. 1: ");
    foreach (var p in products)
    {
        Console.WriteLine(p.ProductName + " ");
    }
    Console.WriteLine("\n");
```

```
// Pokazanie faktur w których sprzedano tulipany
var query2 = from prod in prodContext.Products
              where prod.ProductName == "Tulipan"
              select prod.CanBeSoldIn;
var invoices = query2.First();
Console.WriteLine("Numery faktur, w których sprzedano tulipany: ");
foreach (var i in invoices)
{
    Console.WriteLine(i.InvoiceNumber + " ");
}
```

Wynik działania kodu jest zgodny z oczekiwaniami

```
PS D:\agh\Bazy-Danych-2\lab-entity> dotnet run
● Produkty sprzedane w ramach faktury nr. 1: Tulipan Fiołek Storczyk Róża
  Numery faktur, w których sprzedano tulipany: 1 2 3
PS D:\agh\Bazy-Danych-2\lab-entity> █
```

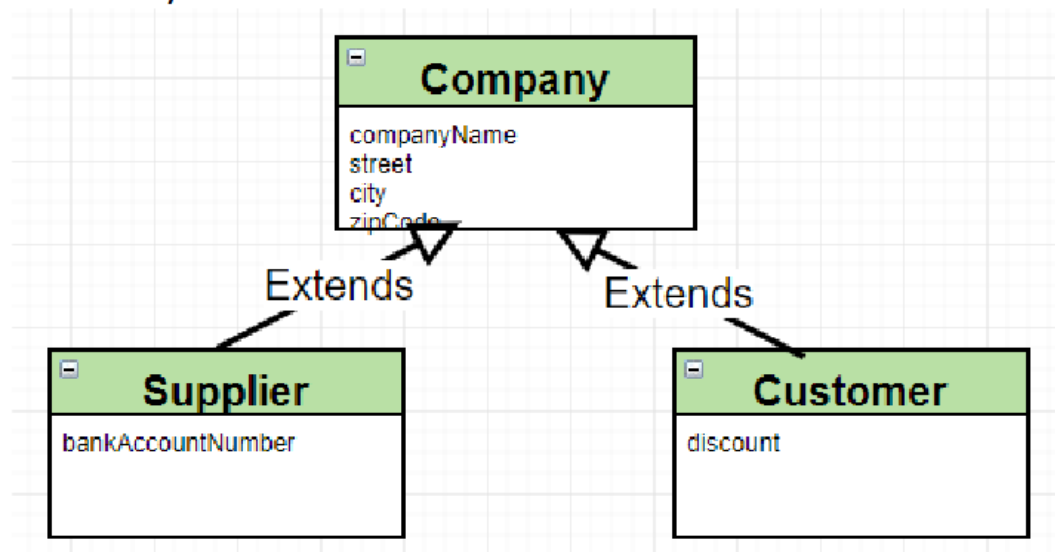
Zmiany są widoczne w bazie danych, w tabeli InvoiceProduct są powiązania nowych produktów z fakturami



```
sqlite> select * from Invoices;
1|1|0
2|2|0
3|3|0
4|3|0
sqlite> select * from Products;
6||Flamaster|0
7||Flamaster|0
8||Flamaster|0
9||Krzes|0
10|1|Kredki|0
11|1|Ołówek|4000
12|1|Laptop|4
13|1|Zeszyt|17
17|1|Klawiatura|35
18|1|Myszka|22
19|1|Zasilacz|0
23||Tulipan|20
24||Fiołek|5
25||Storczyk|3
26||Róża|2
sqlite> select * from InvoiceProduct;
1|23
1|24
1|25
1|26
2|23
2|24
2|25
3|23
4|24
4|26
sqlite> █
```

## Podpunkt e)

- e. Wprowadź do modelu poniższą hierarchie dziedziczenia używając strategii Table-Per-Hierarchy:



- i. Dodaj i pobierz z bazy danych kilka firm obu rodzajów
- ii. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (.schema table/diagram z datagrip, select \* from....)**

## Modyfikacje modelu danych

```

// Dodałem klasę firmy
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; }
}
  
```

```

// Dodałem klasę klienta
public class Customer : Company
{
    public double Discount { get; set; }
}
  
```

```

// Zmodyfikowałem klasę dostawcy
public class Supplier : Company
  
```

```
{
    public string? BankAccountNumber { get; set; }
}
```

```
public class ProdContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<Invoice> Invoices { get; set; }
    // Dodałem DbSet firm
    public DbSet<Company> Companies { get; set; }
    public DbSet<Supplier> Suppliers { get; set; }
    // Dodałem DbSet klientów
    public DbSet<Customer> Customers { get; set; }

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

Wynik zapytania .schema po dodaniu migracji i aktualizacji bazy danych. W tabeli Companies są kolumny BankAccountNumber i Discount. Znajduje się tam też dodatkowa kolumna Discriminator

```
CREATE TABLE IF NOT EXISTS "Invoices" (
    "InvoiceID" INTEGER NOT NULL CONSTRAINT "PK_Invoices" PRIMARY KEY AUTOINCREMENT,
    "InvoiceNumber" INTEGER NOT NULL,
    "Quantity" INTEGER NOT NULL
);
CREATE TABLE IF NOT EXISTS "InvoiceProduct" (
    "CanBeSoldInInvoiceID" INTEGER NOT NULL,
    "IncludesProductID" INTEGER NOT NULL,
    CONSTRAINT "PK_InvoiceProduct" PRIMARY KEY ("CanBeSoldInInvoiceID", "IncludesProductID"),
    CONSTRAINT "FK_InvoiceProduct_Invoices_CanBeSoldInInvoiceID" FOREIGN KEY ("CanBeSoldInInvoiceID") REFERENCES "Invoices" ("InvoiceID") ON DELETE CASCADE,
    CONSTRAINT "FK_InvoiceProduct_Products_IncludesProductID" FOREIGN KEY ("IncludesProductID") REFERENCES "Products" ("ProductID") ON DELETE CASCADE
);
CREATE INDEX "IX_InvoiceProduct_IncludesProductID" ON "InvoiceProduct" ("IncludesProductID");
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "IsSuppliedByCompanyID" INTEGER NULL,
    "ProductName" TEXT NULL,
    "UnitsInStock" INTEGER NOT NULL,
    CONSTRAINT "FK_Products_Companies_IsSuppliedByCompanyID" FOREIGN KEY ("IsSuppliedByCompanyID") REFERENCES "Companies" ("CompanyID")
);
CREATE TABLE IF NOT EXISTS "Companies" (
    "CompanyID" INTEGER NOT NULL CONSTRAINT "PK_Companies" PRIMARY KEY AUTOINCREMENT,
    "BankAccountNumber" TEXT NULL,
    "City" TEXT NULL,
    "CompanyName" TEXT NULL,
    "Discount" REAL NULL,
    "Discriminator" TEXT NOT NULL,
    "Street" TEXT NULL,
    "ZipCode" TEXT NULL
);
CREATE INDEX "IX_Products_IsSuppliedByCompanyID" ON "Products" ("IsSuppliedByCompanyID");
```

Kod dodający dostawców i klientów, pobierający dane najpierw z całego agregatu Companies, a potem tylko z Suppliers

```
using System.Text.Json;

var prodContext = new ProdContext();

var supplier1 = new Supplier
```

```

{
    CompanyName = "DHL",
    Street = "Wielicka",
    City = "Kraków",
    BankAccountNumber = "PL91 1378 1462 6908 8595 1014 0748"
};
var supplier2 = new Supplier
{
    CompanyName = "MAERSK",
    Street = "Długa",
    City = "Warszawa",
    BankAccountNumber = "PL37 5269 6062 3118 6527 8335 6401"
};
prodContext.Suppliers.AddRange(supplier1, supplier2);

var customer1 = new Customer
{
    CompanyName = "Wedel",
    Street = "Myślenicka",
    City = "Kraków",
    Discount = 0.05
};
var customer2 = new Customer
{
    CompanyName = "Wedel",
    Street = "Prosta",
    City = "Wrocław",
    Discount = 0.07
};
prodContext.Customers.AddRange(customer1, customer2);

prodContext.SaveChanges();

// Pobranie firmy o danej nazwie z całego agregatu Companies
var query1 = from comp in prodContext.Companies
              where comp.CompanyName == "Wedel"
              select comp;
var company = query1.First();
Console.WriteLine(JsonSerializer.Serialize(company));

// Pobranie dostawcy o danym numerze konta w banku z agregatu Suppliers
var query2 = from supp in prodContext.Suppliers
              where supp.BankAccountNumber == "PL91 1378 1462 6908 8595 1014 0748"
              select supp;
var supplier = query2.First();
Console.WriteLine(JsonSerializer.Serialize(supplier));

```

Wynik działania kodu, jak widać przy pobieraniu danych z prodContext.Companies pole Discount nie jest widoczne

```

PS D:\agh\Bazy-Danych-2\lab-entity> dotnet run
{"CompanyID":2,"CompanyName":"Wedel","Street":"My\u015Blenicka","City":"Krak\u00F3w","ZipCode":null}
{"BankAccountNumber":"PL91 1378 1462 6908 8595 1014 0748","CompanyID":4,"CompanyName":"DHL","Street":"Wielicka","City":"Krak\u00F3w","ZipCode":null}

```

Zmiany są widoczne w bazie danych

```
sqlite> select * from Companies;
1|Kraków|Inpost||Kawiory|
2|Kraków|Wedel|0.05|Customer|Myślenicka|
3|Wrocław|Wedel|0.07|Customer|Prosta|
4|PL91 1378 1462 6908 8595 1014 0748|Kraków|DHL||Supplier|Wielicka|
5|PL37 5269 6062 3118 6527 8335 6401|Warszawa|MAERSK||Supplier|Długa|
```

Podpunkt f)

- f. Zamodeluj tę samą hierarchię dziedziczenia, ale tym razem użyj strategii Table-Per-Type
  - i. Dodaj i pobierz z bazy danych kilka firm obu rodzajów
  - ii. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (.schema table/diagram z datagrip, select \* from....)**

Modyfikacje modelu danych

```
public class ProdContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<Invoice> Invoices { get; set; }
    public DbSet<Company> Companies { get; set; }
    public DbSet<Supplier> Suppliers { get; set; }
    public DbSet<Customer> Customers { get; set; }

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

    // Dodałem przeładowanie OnModelCreating
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
        modelBuilder.Entity<Company>().ToTable("Companies");
        modelBuilder.Entity<Supplier>().ToTable("Suppliers");
        modelBuilder.Entity<Customer>().ToTable("Customers");
    }
}
```

Ze względu na błędy wynikające z relacji musiałem to zadanie wykonać na nowej bazie danych. Wynik zapytania .schema po dodaniu migracji i aktualizacji bazy danych, są osobne tabele Companies, Suppliers i Customers

```

CREATE TABLE IF NOT EXISTS "Customers" (
  "CompanyID" INTEGER NOT NULL CONSTRAINT "PK_Customers" PRIMARY KEY AUTOINCREMENT,
  "Discount" REAL NOT NULL,
  CONSTRAINT "FK_Customers_Companies_CompanyID" FOREIGN KEY ("CompanyID") REFERENCES "Companies" ("CompanyID") ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS "Suppliers" (
  "CompanyID" INTEGER NOT NULL CONSTRAINT "PK_Suppliers" PRIMARY KEY AUTOINCREMENT,
  "BankAccountNumber" TEXT NULL,
  CONSTRAINT "FK_Suppliers_Companies_CompanyID" FOREIGN KEY ("CompanyID") REFERENCES "Companies" ("CompanyID") ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS "Products" (
  "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
  "IsSuppliedByCompanyID" INTEGER NULL,
  "ProductName" TEXT NULL,
  "UnitsInStock" INTEGER NOT NULL,
  CONSTRAINT "FK_Products_Suppliers_IsSuppliedByCompanyID" FOREIGN KEY ("IsSuppliedByCompanyID") REFERENCES "Suppliers" ("CompanyID")
);
CREATE TABLE IF NOT EXISTS "Companies" (
  "CompanyID" INTEGER NOT NULL CONSTRAINT "PK_Companies" PRIMARY KEY AUTOINCREMENT,
  "City" TEXT NULL,
  "CompanyName" TEXT NULL,
  "Street" TEXT NULL,
  "ZipCode" TEXT NULL
);
CREATE INDEX "IX_Products_IsSuppliedByCompanyID" ON "Products" ("IsSuppliedByCompanyID");

```

Aktywuj system Win

Kod dodający dostawców i klientów, a potem pobierający dane, jest taki sam jak w poprzednim podpunkcie. Wynik działania kodu też jest taki sam

```

PS D:\agh\Bazy-Danych-2\lab-entity> dotnet run
{"CompanyID":1,"CompanyName":"Wedel","Street":"My\u015Blenicka","City":"Krak\u00F3w","ZipCode":null}
{"BankAccountNumber":"PL91 1378 1462 6908 8595 1014 0748","CompanyID":3,"CompanyName":"DHL","Street":"Wielicka","City":"Krak\u00F3w","ZipCode":null}

```

Zmiany są widoczne w bazie danych

```

sqlite> select * from Companies;
1|Kraków|Wedel|Myślenicka|
2|Wrocław|Wedel|Prosta|
3|Kraków|DHL|Wielicka|
4|Warszawa|MAERSK|Długa|
sqlite> select * from Customers;
1|0.05
2|0.07
sqlite> select * from Suppliers;
3|PL91 1378 1462 6908 8595 1014 0748
4|PL37 5269 6062 3118 6527 8335 6401
sqlite> 

```

Podpunkt g)

g. Porównaj (i skomentuj/opisz w raporcie) obie strategie modelowania dziedziczenia

Z poziomu interakcji z bazą danych poprzez kod obie strategie są identyczne. W strategii TPH mamy tylko jedną tabelę, ale są w niej wartości ustawione na null, co może być problematyczne jeśli dziedziczące klasy

mają wiele własnych pól. W strategii TPT nie ma dodatkowych wartości null, ale za to jest wiele tabel, więc przy zapytaniach trzeba używać join-ów, co może je spowalniać.