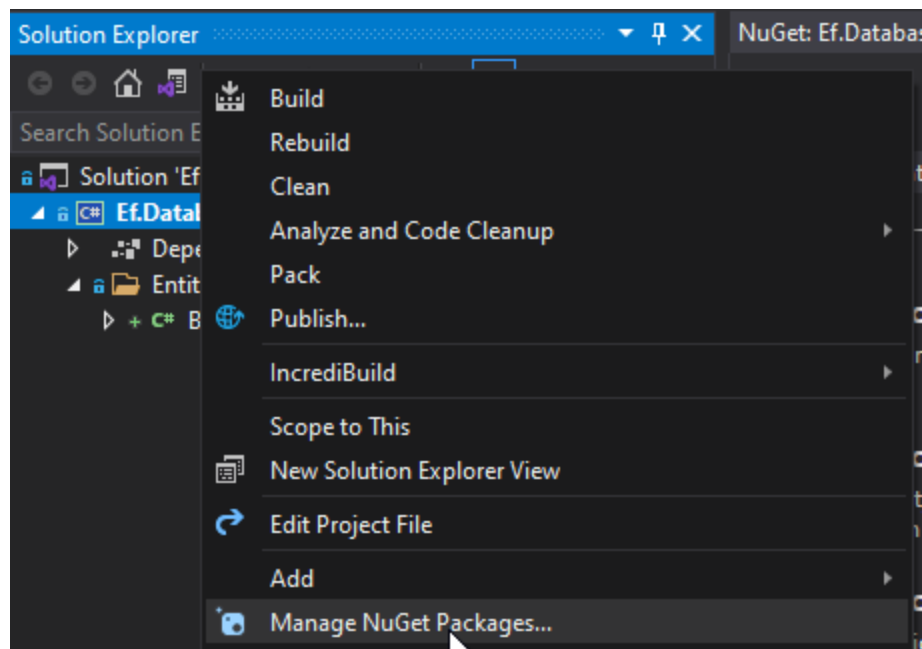


Student Development Program

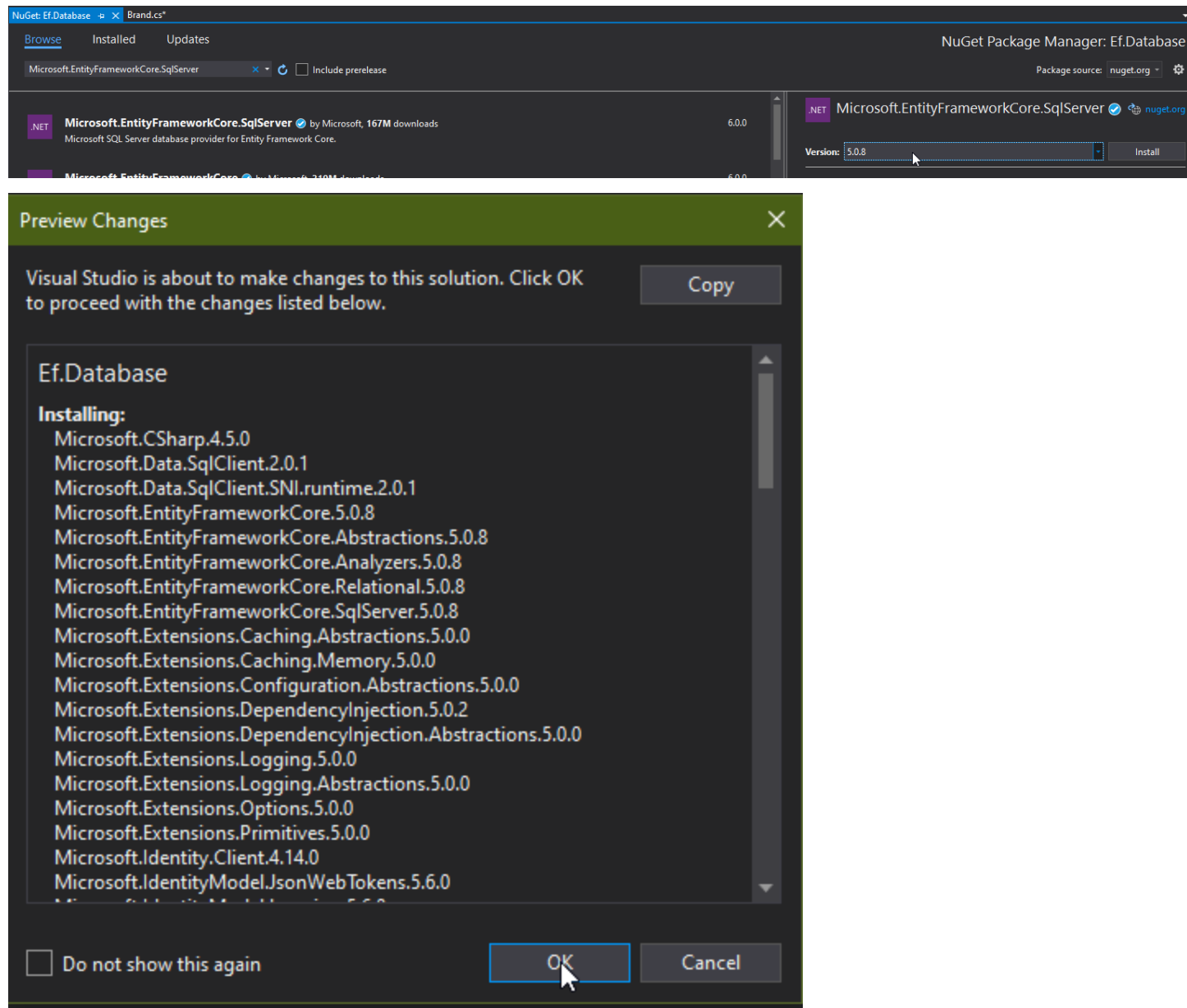
C# Entity Framework

Paweł Leśniak

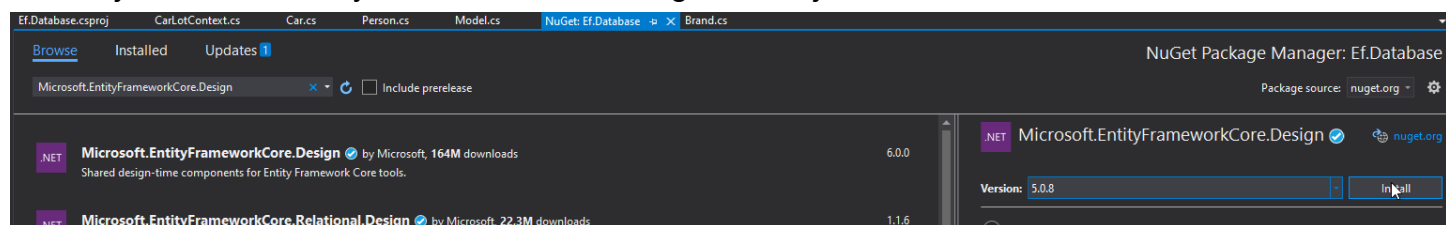
1. Instalacja dostawcy bazy danych.



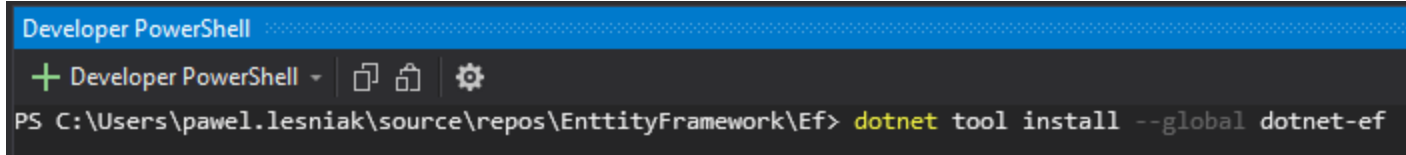
Wyszukanie Microsoft.EntityFrameworkCore.SqlServer i zainstalowanie wersji 5.0.8



2. Instalacja Microsoft.EntityFrameworkCore.Design w wersji 5.0.8

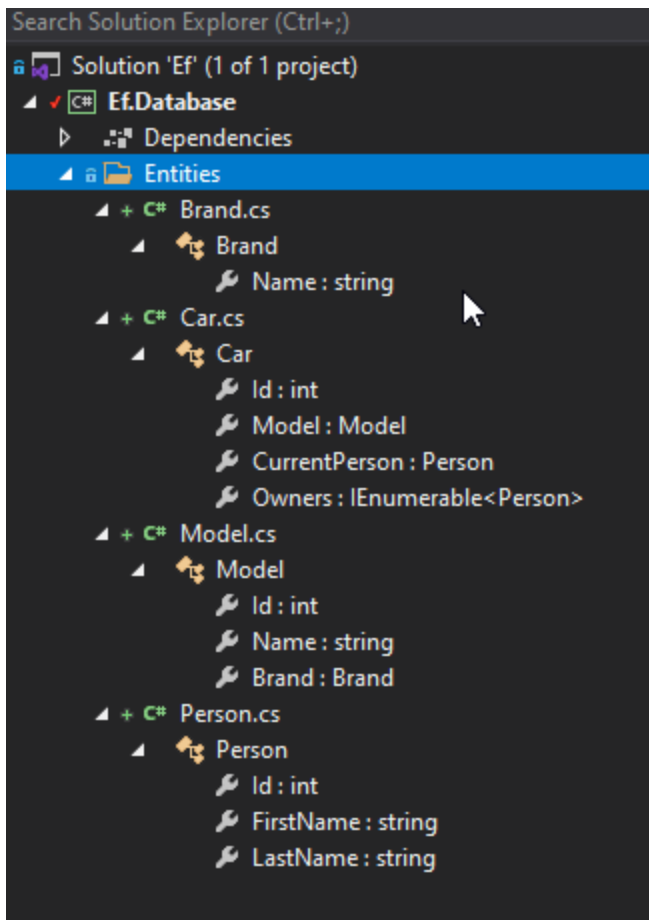


3. Instalacja narzędzi dotnet-ef

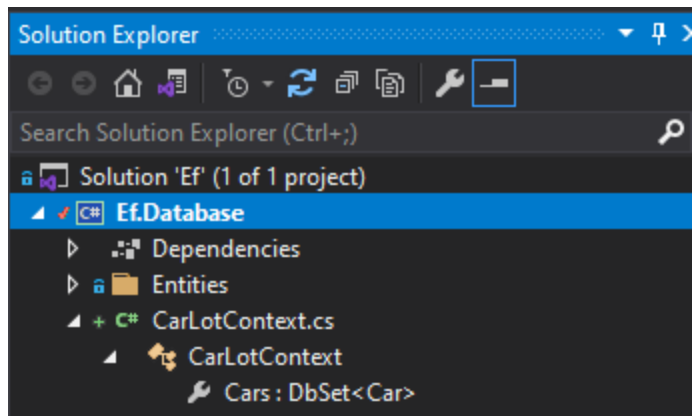


```
Developer PowerShell
+ Developer PowerShell
PS C:\Users\pawel.lesniak\source\repos\EntityFramework\Ef> dotnet tool install --global dotnet-ef
```

4. Dodanie klas domenowych.



5. Stworzenie kontekstu.



6. Konfiguracja kontekstu.

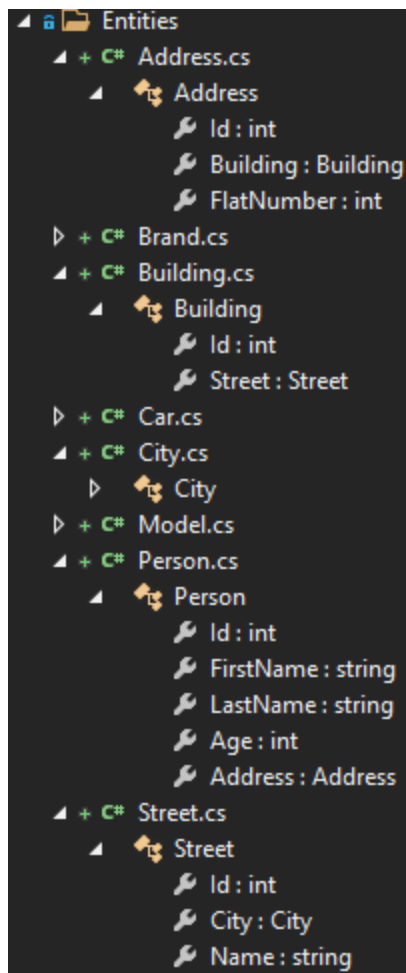
```
0 references | 0 changes | 0 authors, 0 changes
public class CarLotContext : DbContext
{
    0 references | 0 changes | 0 authors, 0 changes
    public DbSet<Car> Cars { get; set; }

    0 references | 0 changes | 0 authors, 0 changes
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        var connectionString = @"Data Source=localhost\sqlexpress;Initial Catalog=SDPEF;Integrated Security=True";
        optionsBuilder.UseSqlServer(connectionString);
    }
}
```

7. Stworzenie migracji

```
Developer PowerShell
+ Developer PowerShell | [ ] [ ] [ ]
PS C:\Users\pawel.lesniak\source\repos\EntityFramework\Ef\Ef.Database> dotnet ef migrations add InitialMigration
```

8. Modyfikacja klas domenowych poprzez dodanie dowolnych właściwości i dodanie nowych klas domenowych.



9. Stworzenie kolejnej migracji (migracji należy nadać inną nazwę, niż ta, która została określona przy poprzednich migracjach)

```
PS C:\Users\pawel.lesniak\source\repos\EntityFramework\Ef\Ef.Database> dotnet ef migrations add UpdateMigration1
```

10. Stworzenie bazy danych

```
PS C:\Users\pawel.lesniak\source\repos\EntityFramework\Ef\Ef.Database> dotnet ef database update
```

11. Dodanie brakującej właściwości klasy Building (int Number), dodanie go i utworzenie nowej migracji.
12. Aktualizacja bazy danych.

13. Dodanie klasy Repository.

```
public static class Repository
{
    0 references | 0 changes | 0 authors, 0 changes
    public static void Add<T>(T item) where T : class
    {
        using var context = new CarLotContext();
        context.Set<T>().Add(item);
        context.SaveChanges();
    }

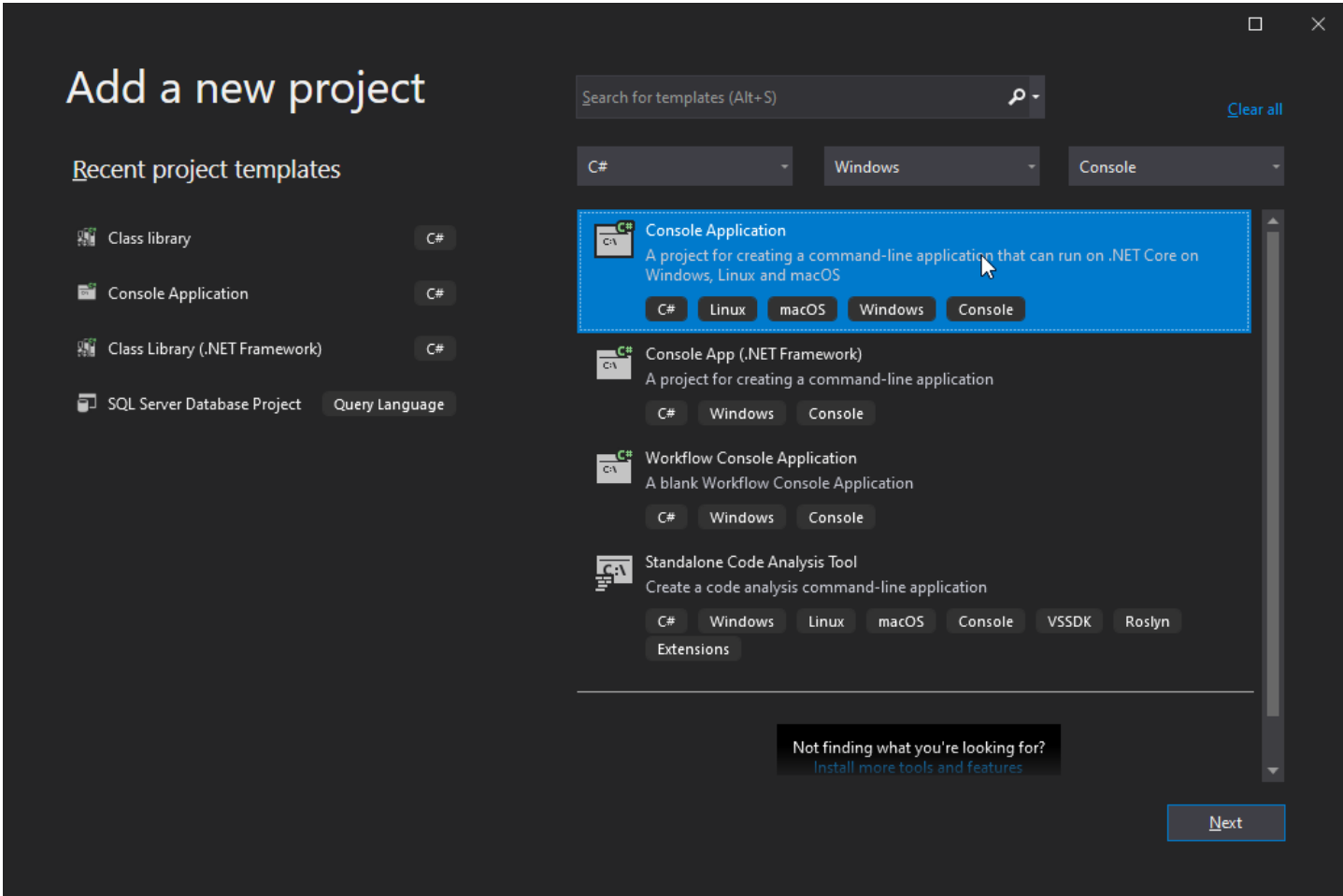
    0 references | 0 changes | 0 authors, 0 changes
    public static void Update<T>(T item) where T : class
    {
        using var context = new CarLotContext();
        context.Set<T>().Update(item);
        context.SaveChanges();
    }

    0 references | 0 changes | 0 authors, 0 changes
    public static void Remove<T>(T item) where T : class
    {
        using var context = new CarLotContext();
        context.Set<T>().Remove(item);
        context.SaveChanges();
    }

    0 references | 0 changes | 0 authors, 0 changes
    public static T Get<T>(int id) where T : class
    {
        using var context = new CarLotContext();
        return context.Set<T>().Find(id);
    }

    0 references | 0 changes | 0 authors, 0 changes
    public static List<T> Find<T>() where T : class
    {
        using var context = new CarLotContext();
        return context.Set<T>().ToList();
    }
}
```

14. Dodanie aplikacji konsolowej Ef.Console.



Configure your new project

Console Application

C#LinuxmacOSWindowsConsole

Project name

Ef.Console

Location

C:\Users\pawel.lesniak\source\repos\EntityFramework\Ef

z''

Back

Next

15. Dodanie nowego elementu Car do repozytorium.

```
static void Main(string[] args)
{
    var newCar = new Car
    {
        CurrentPerson = new Person
        {
            Address = new Address
            {
                Building = new Building
                {
                    Number = 5,
                    Street = new Street
                    {
                        City = new City
                        {
                            Name = "Szczecin"
                        },
                        Name = "Wojska Polskiego"
                    }
                }
            },
            Model = new Model
            {
                Brand = new Brand
                {
                    Name = "Ford"
                },
                Name = "Focus"
            },
            Owners = null
        };

    Repository.Add<Car>(newCar);
}
```

16. Zaimplementowanie dodania poszczególnych Encji osobno.

17. Dodanie kluczy unikalnych w uzasadnionych miejscach (właściwościach poszczególnych encji).

```
[Index( params: nameof(Name), IsUnique = true)]
2 references | 0 changes | 0 authors, 0 changes
public class City
{
    0 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }
    2 references | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }
}
```

18. Należy pamiętać, aby po każdej aktualizacji modeli dodać kolejną migrację i zaktualizować bazę danych.

19. Zmienić nazwy tabel przez określenie atrybutu.

```
[Table( name: "tblPerson")]
3 references | 0 changes | 0 authors, 0 changes
public class Person
{
    0 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public string FirstName { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public string LastName { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public int Age { get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public Address Address { get; set; }
}
```

```
[Index( params: nameof(Name), IsUnique = true)]
[Table( name: "tblCity")]
2 references | 0 changes | 0 authors, 0 changes
public class City
{
    0 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }
    2 references | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }
}
```

20. Zadanie domowe:

- Zaimplementować klasę niestaticzną klasę repozytorium z metodą Save, której wywołanie będzie skutkowało zapisaniem danych do bazy danych.
- W aplikacji konsolowej zaimplementować sekwencję dodawania, aktualizacji i usuwania danych wraz z wyświetlaniem (Console WriteLine) identyfikatorów samochodów przy

czym aktualizacja ma zmieniać właściciela dodając poprzedniego do listy poprzednich właścicieli z wykorzystaniem klasy repozytorium z poprzedniego punktu.