Wymagania wstępne

1. Utworzyć projekt WebAPI
2. Zainstalować narzędzie CLI - **dotnet-ef**
   https://learn.microsoft.com/pl-pl/ef/core/cli/dotnet
3. Zainstalować pakiety Nuget:
   Microsoft.EntityFrameworkCore.Design
   Microsoft.EntityFrameworkCore.SqlServer

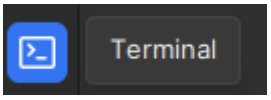
Odtworzenie bazy danych – Scaffold

Przed wykonaniem odtworzenia bazy danych trzeba utworzyć bazę danych i wykonać skrypt **cw7_create.sql**, który utworzy odpowiednie tabele i zaseeduje dane.

Na początku należy dodać ConnectionString do pliku **appsettings.json**. Przykładowa zawartość tego pliku:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "Default": "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=scaffold;Integrated Security=True;"
  }
}
```

Po zapisaniu konfiguracji aplikacji uruchamiamy terminal w Riderze. Poniżej znajduje się przykładowa ikona terminalu:

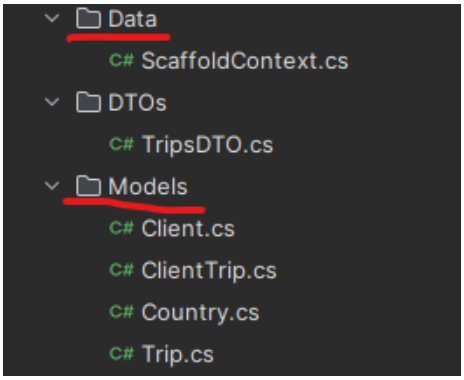


Używając interfejsu wiersza poleceń należy zmienić katalog na folder z projektem. Domyślnie terminal uruchomi się w katalogu solucji, dlatego przechodzimy do projektu wykonując taką komendę:

```
cd NazwaProjektu
```

Będąc w katalogu możemy wykonać odtworzenie bazy danych za pomocą komendy dotnet ef dbcontext scaffold. Wymagane są dwa parametry ConnectionString oraz sterownik do bazy danych. Dodatkowo można dodać parametr **`--context-dir`** w celu zdefiniowania gdzie zostanie umieszczony kontekst bazy danych. Warto też skorzystać z opcjonalnego parametru **`--output-dir`**, który pozwoli na umieszczenie modeli z bazy w jednym wskazanym folderze. Poniżej znajduje się cała komenda, którą należy wykonać:

```
dotnet ef dbcontext scaffold "Name=ConnectionStrings:Default" --context-dir
Data --output-dir Models Microsoft.EntityFrameworkCore.SqlServer
```



Kontekst bazy danych (powyżej ScaffoldContext.cs) zawiera konfigurację połączenia do bazy danych. Poniższą linię kodu należy usunąć, ponieważ konfiguracja będzie wykonana podczas wstrzykiwania zależności.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    => optionsBuilder.UseSqlServer("Name=ConnectionStrings:Default");
```

W klasie Program.cs wstrzykujemy zależność z kontekstem:

```
builder.Services.AddDbContext<ScaffoldContext>(
    options => options.UseSqlServer("Name=ConnectionStrings:Default"));
```

Wykonywanie zapytań

EntityFramework ma kilka możliwości na pobieranie danych. My wykorzystamy do tego technikę EagerLoading.

Najpierw należy przekazać kontekst jako argument konstruktora:

```
[Route("api/[controller]")]
[ApiController]
public class TripsController : ControllerBase
{
    private readonly ScaffoldContext _context;

    public TripsController(ScaffoldContext context)
    {
        _context = context;
    }
}
```

W celu stworzenia zapytania należy wykorzystać składnię LINQ, która była na poprzednich ćwiczeniach. Poniżej jest przykład pobrania wycieczek (dane są niepełne):

```
var trips = await _context.Trips.Select(e => new TripDTO()
{
    Name = e.Name,
    DateFrom = e.DateFrom,
    MaxPeople = e.MaxPeople,
    Clients = e.ClientTrips.Select(e => new ClientDTO()
    {
        FirstName = e.IdClientNavigation.FirstName,
        LastName = e.IdClientNavigation.LastName
    })
})
    .OrderBy(e => e.DateFrom)
    .ToListAsync();
```

Metody Skip i Take mogą się przydać do stronicowania:

```
.Skip((page-1) * pageSize)
.Take(pageSize)
```

W przypadku dodawania i usuwania danych proszę obejrzeć wykład :)