2. SQL Profiler – nasłuch endpointu

📌 Zadanie:

- Uruchom SQL Server Profiler (lub EF Core Logging).
- Wybierz konkretny endpoint API.
- Uruchom aplikację → wywołaj endpoint → zrób screenshot z Profilerem pokazującym zapytanie.
- Dodaj screenshoty + opis działania zapytania + krótki komentarz.

1. Nazwa Endpointu / Akcji: ServiceOrdersController.Details/{id}
   Opis: Ta akcja wyświetla pełne szczegóły zlecenia serwisowego, wraz z danymi pojazdu, klienta, mechanika, czynnościami, częściami i komentarzami. Jest to jedno z najbardziej złożonych zapytań w systemie. Poniżej przedstawiono zapytania SQL przechwycone podczas wywołania endpointu ServiceOrdersController.Details/{id}. Widoczny jest podział na wiele zapytań SELECT, co jest efektem zastosowania metody AsSplitQuery() w logice pobierania danych w serwisie.

```
Zapytanie 1 (Główne dane zlecenia, pojazdu, klienta, mechanika)


SELECT TOP(1) [s].[Id], [s].[AssignedMechanicId], [s].[CompletionDate],
[s].[CreationDate], [s].[Description], [s].[RowVersion], [s].[Status],
[s].[VehicleId], [v].[Id], [v].[Brand], [v].[CustomerId],
[v].[ImageUrl], [v].[Model], [v].[RegistrationNumber], [v].[VIN],
[v].[Year], [c].[Id], [c].[FullName], [c].[PhoneNumber], [a].[Id],
[a].[AccessFailedCount], [a].[ConcurrencyStamp], [a].[Email],
[a].[EmailConfirmed], [a].[FirstName], [a].[LastName],
[a].[LockoutEnabled], [a].[LockoutEnd], [a].[NormalizedEmail],
[a].[NormalizedUserName], [a].[PasswordHash], [a].[PhoneNumber],
[a].[PhoneNumberConfirmed], [a].[SecurityStamp], [a].[TwoFactorEnabled],
[a].[UserName]
FROM [ServiceOrders] AS [s]
INNER JOIN [Vehicles] AS [v] ON [s].[VehicleId] = [v].[Id]
INNER JOIN [Customers] AS [c] ON [v].[CustomerId] = [c].[Id]
LEFT JOIN [AspNetUsers] AS [a] ON [s].[AssignedMechanicId] = [a].[Id]
WHERE [s].[Id] = @__id_0
ORDER BY [s].[Id], [v].[Id], [c].[Id], [a].[Id];

Zapytanie 3 (Użyte części)

SELECT [s0].[Id], [s0].[Description], [s0].[LaborCost],
[s0].[ServiceOrderId]
FROM [ServiceTasks] AS [s0]
WHERE [s0].[ServiceOrderId] = @__id_0 ORDER BY [s0].[Id];
```

Zapytanie 3 (Użyte części)

```
SELECT [t0].[Id], [t0].[PartId], [t0].[Quantity], [t0].[ServiceTaskId],
[p].[Id] AS [Id0], [p].[Name], [p].[Type], [p].[UnitPrice], [t].[Id],
[t].[Id0], [t].[Id1], [t].[Id2], [s0].[Id]
FROM (
    SELECT TOP(1) [s].[Id], [v].[Id] AS [Id0], [c].[Id] AS [Id1],
[a].[Id] AS [Id2]
    FROM [ServiceOrders] AS [s]
    INNER JOIN [Vehicles] AS [v] ON [s].[VehicleId] = [v].[Id]
    INNER JOIN [Customers] AS [c] ON [v].[CustomerId] = [c].[Id]
    LEFT JOIN [AspNetUsers] AS [a] ON [s].[AssignedMechanicId] =
[a].[Id]
    WHERE [s].[Id] = @__id_0
) AS [t]
INNER JOIN [ServiceTasks] AS [s0] ON [t].[Id] = [s0].[ServiceOrderId]
INNER JOIN [UsedParts] AS [t0] ON [s0].[Id] = [t0].[ServiceTaskId]
INNER JOIN [Parts] AS [p] ON [t0].[PartId] = [p].[Id] -- Tutaj p.Id
powinno być OK
ORDER BY [t].[Id], [t].[Id0], [t].[Id1], [t].[Id2], [s0].[Id],
[t0].[Id];
```

Zapytanie 4 (Komentarze)

```
SELECT [t].[Id], [t].[AuthorId], [t].[Content], [t].[CreatedAt],
[t].[OrderId], [t0].[Id], [t0].[AccessFailedCount],
[t0].[ConcurrencyStamp], [t0].[Email], [t0].[EmailConfirmed],
[t0].[FirstName], [t0].[LastName], [t0].[LockoutEnabled],
[t0].[LockoutEnd], [t0].[NormalizedEmail], [t0].[NormalizedUserName],
[t0].[PasswordHash], [t0].[PhoneNumber], [t0].[PhoneNumberConfirmed],
[t0].[SecurityStamp], [t0].[TwoFactorEnabled], [t0].[UserName]
FROM [Comments] AS [t]
INNER JOIN [AspNetUsers] AS [t0] ON [t].[AuthorId] = [t0].[Id]
WHERE [t].[OrderId] = @__id_0
ORDER BY [t].[Id];
```

Zapytanie 5 (Lista wszystkich części)

```
SELECT [p].[Id], [p].[Name], [p].[Type], [p].[UnitPrice]
FROM [Parts] AS [p];
```

2. Wnioski :
   Logowanie zapytań EF Core to super sprawa – widać czarno na białym, jak nasze LINQ-owe zapytania są tłumaczone na SQL. Pomaga to wychwycić, co dzieje się "pod maską" i sprawdzić, czy optymalizacje (jak AsSplitQuery()) działają tak, jak powinny, zapewniając szybkie ładowanie danych.