**MiniProfiler\_API\_Project**

**Proje Raporu**

**İÇİNDEKİLER:**

1. **Projenin Amacı**
2. **Giriş**
3. **Yöntem**
   1. **Proje Planlaması ve Gereksinimlerin Belirlenmesi**
   2. **Proje Yapısının ve Mimarinin Oluşturulması**
   3. **Veritabanı Tasarımı ve Oluşturulması**
   4. **Entity Framework Core ile Veritabanı Bağlantısının Sağlanması**
   5. **MiniProfiler Kurulumu ve Yapılandırılması**
   6. **API Controller’larının Oluşturulması**
   7. **Service Katmanının Oluşturulması**
4. **MiniProfiler’ı Projeye Ekleme Adımları**
   1. **Nuget Paketlerini Yükleme**
   2. **MiniProfiler Middleware ve Servislerini Yapılandırma**
   3. **Controller Metotlarına Profil Adımları Eklemek**
   4. **MiniProfiler UI'sını Kontrol Etme**
5. **Gösterim ve Sonuç**
   1. **Profillerin gösterimi**
   2. **Profillerin yorumlanması ve sonuç**

**1.Projenin Amacı:**

Miniprofiler’ın ne işe yaradığını bu eklenti ile neler yapabileceğimizi ve projelerimizde kullanırken bize ne gibi faydaları olabileceğini gözlemlemek. Miniprofiler yapısının bir API projesi içinde nasıl oluşturulabileceğini kavramak.

**2.Giriş:**

Projem bir Web API içinde olup içine Miniprofiler eklentisi eklenmiştir. API projem “CRUD” işlemleri yapabilmekte ve veri tabanı olarak MSSQL kullanmaktadır. Miniprofiler eklentisi web istekleri ve veri tabanı sorgularını profillemek için kullanılmaktadır.  
  
  
**3.Yöntem:**

3.1 **Proje Planlaması ve Gereksinimlerin Belirlenmesi**

* Projenin kapsamı ve hedefleri belirlendi.
* Gereksinimler toplandı ve analiz edildi.

3.2 **Proje Yapısının ve Mimarinin Oluşturulması**

* N-tier mimari kullanılarak proje yapısı oluşturuldu.
* Katmanlar belirlendi: Rest, Business ve Database.

3.3 **Veritabanı Tasarımı ve Oluşturulması**

* MSSQL veritabanı olan Logo.API.Project oluşturuldu.
* CRUD işlemleri için gerekli tablo ve ilişkiler tasarlandı ve oluşturuldu.

3.4 **Entity Framework Core ile Veritabanı Bağlantısının Sağlanması**

* Entity Framework Core kullanılarak veritabanı bağlantısı kuruldu.
* LogoApiDbContext sınıfı oluşturularak DbSet tanımları yapıldı.

3.5 **MiniProfiler Kurulumu ve Yapılandırılması**

* MiniProfiler NuGet paketleri projeye eklendi.
* MiniProfiler middleware, Program.cs dosyasında yapılandırıldı.
* Profiling adımları controller metodlarına eklendi.

3.6 **API Controller’larının Oluşturulması**

* ProductsController sınıfı oluşturuldu.
* CRUD işlemlerini gerçekleştiren GET, POST, PUT, DELETE metotları yazıldı.

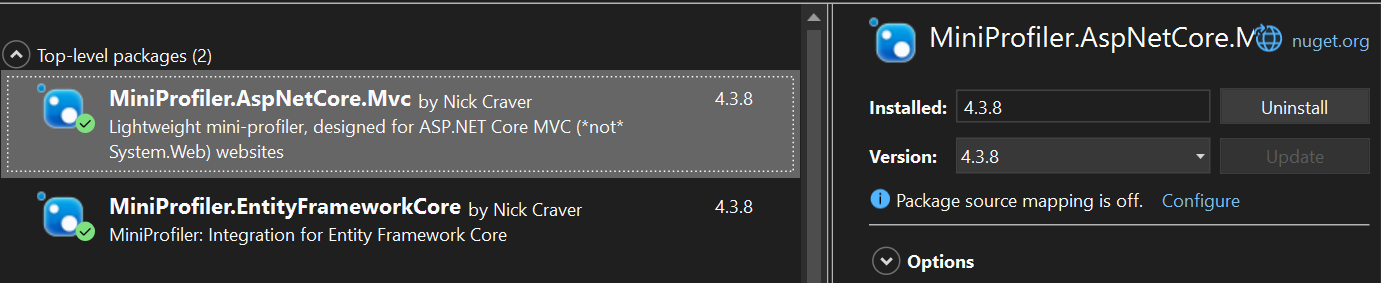
3.7 **Service Katmanının Oluşturulması**

* ProductService sınıfı oluşturuldu.
* Ürünlerle ilgili iş mantıkları (business logic) yazıldı.
* AddProductAsync, GetAllProductsAsync, GetProductByIdAsync, UpdateProductAsync, DeleteProductAsync metodları tanımlandı.

**4.MiniProfiler’ı Projeye Ekleme Adımları:**

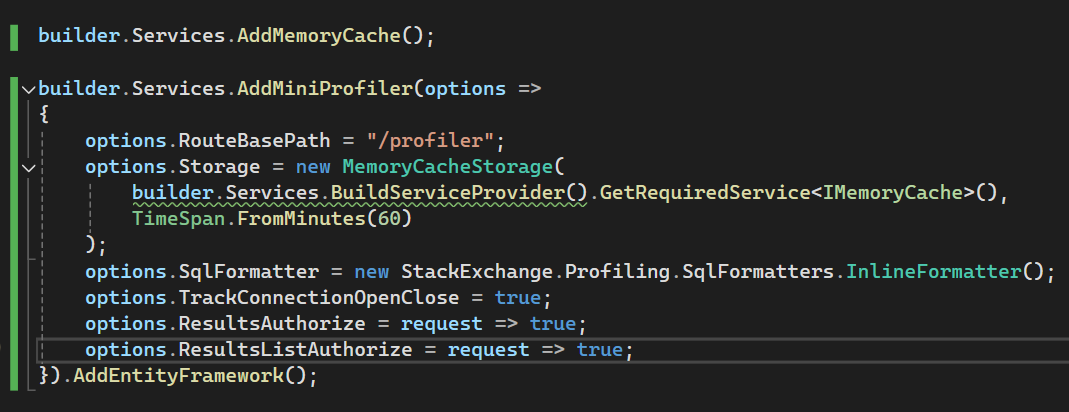
4.1 **Nuget Paketlerini Yükleme**

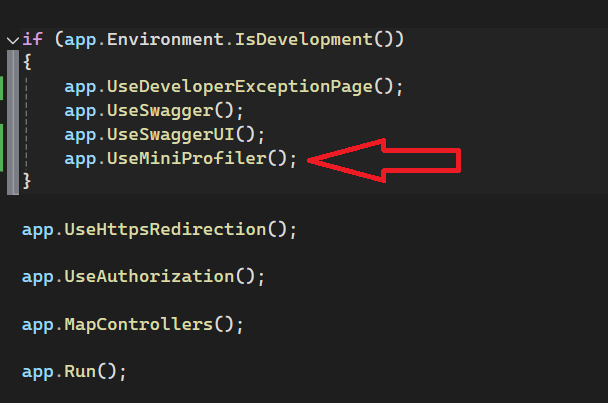
* MiniProfiler.AspNetCore.MVC ve MiniProfiler.EntityFrameworkCore paketleri eklendi.



4.2 **MiniProfiler Middleware ve Servislerini Yapılandırma**

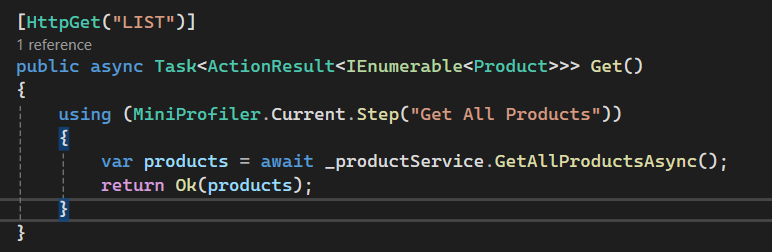
* Program.cs dosyası içine aşağıdaki kodları ekleyerek MiniProfiler için gerekli servisler ve middleware yapılandırıldı.

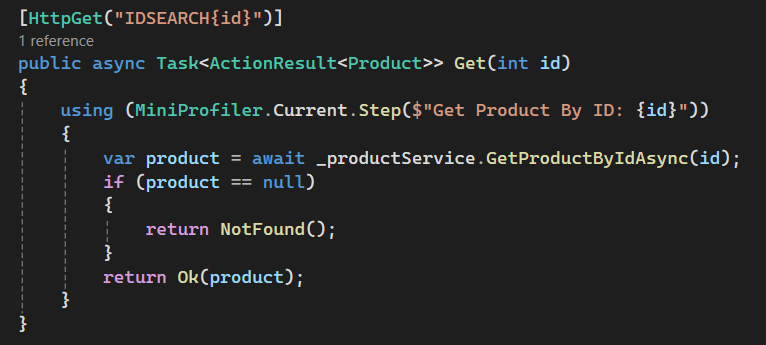


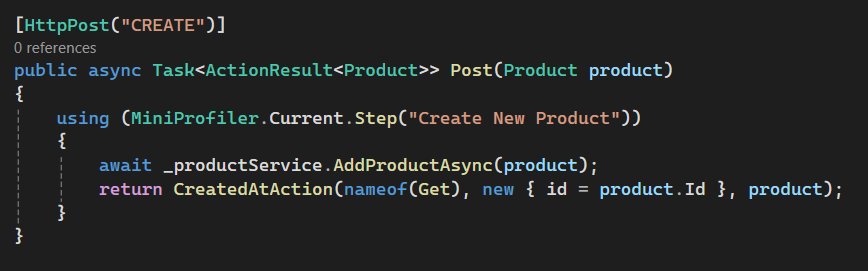


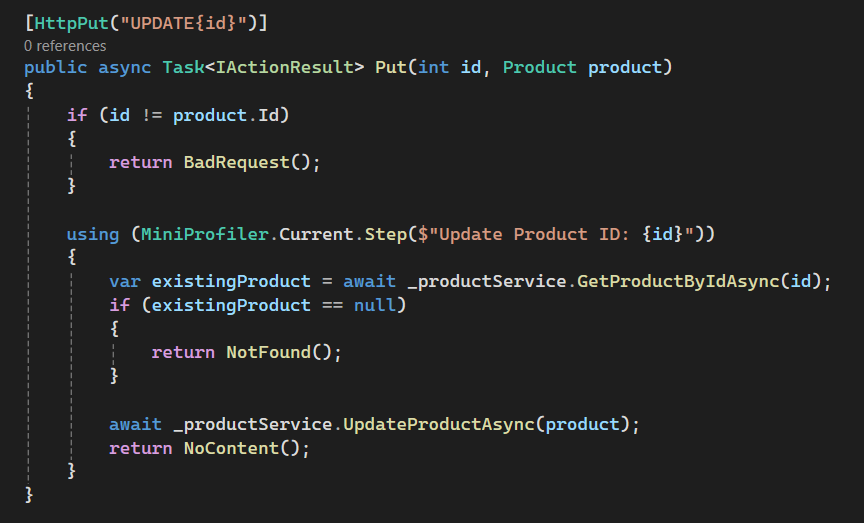
4.3 **Controller Metotlarına Profil Adımları Eklemek**

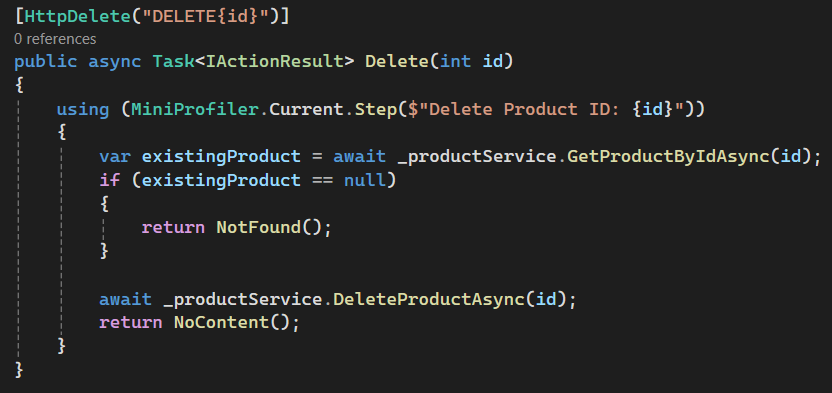
* Profil adımlarını belirlemek için ProductsController sınıfında CRUD işlemlerine MiniProfiler.Current.Step metodunu kullanılarak adımlar eklendi.









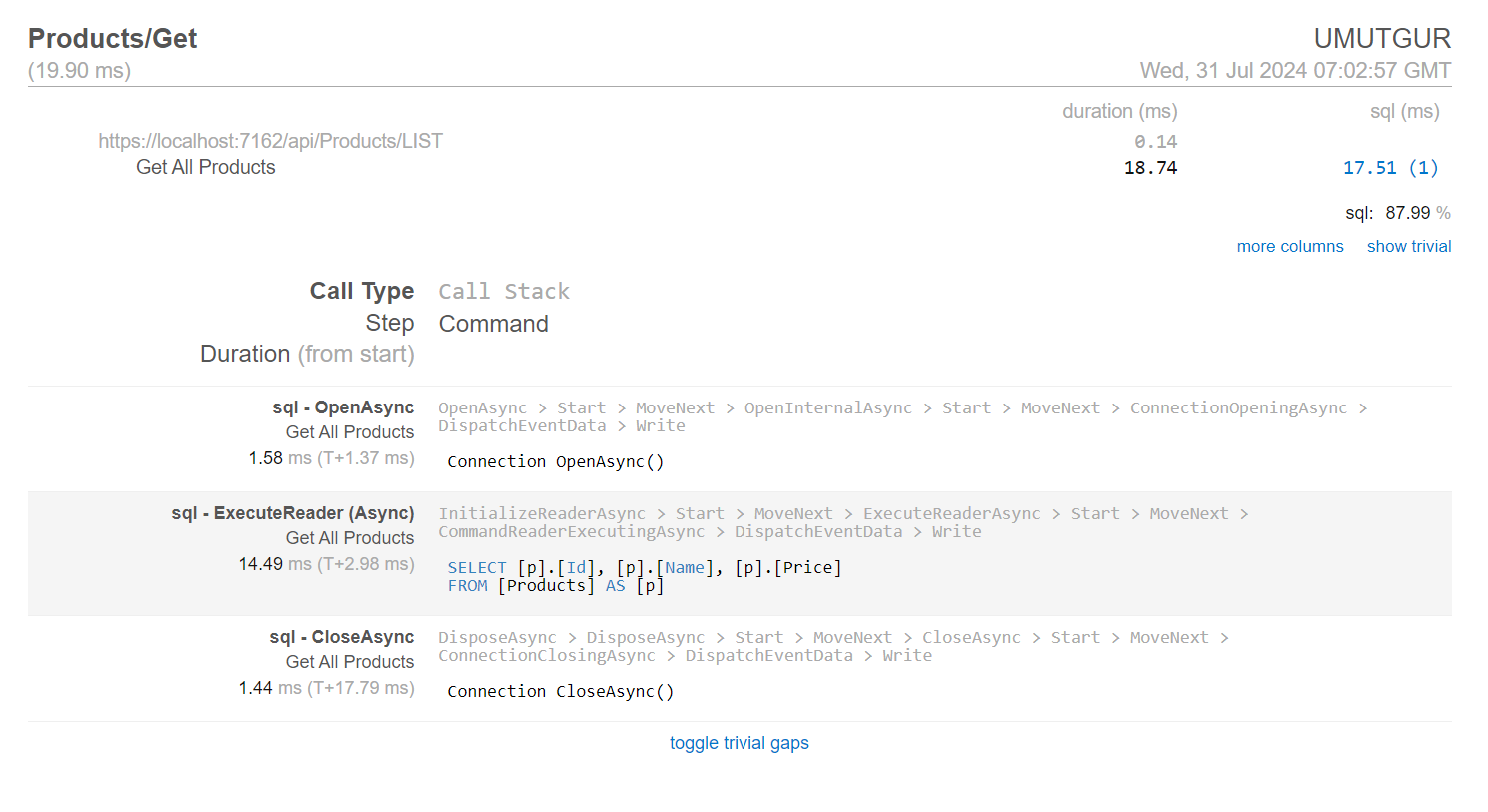


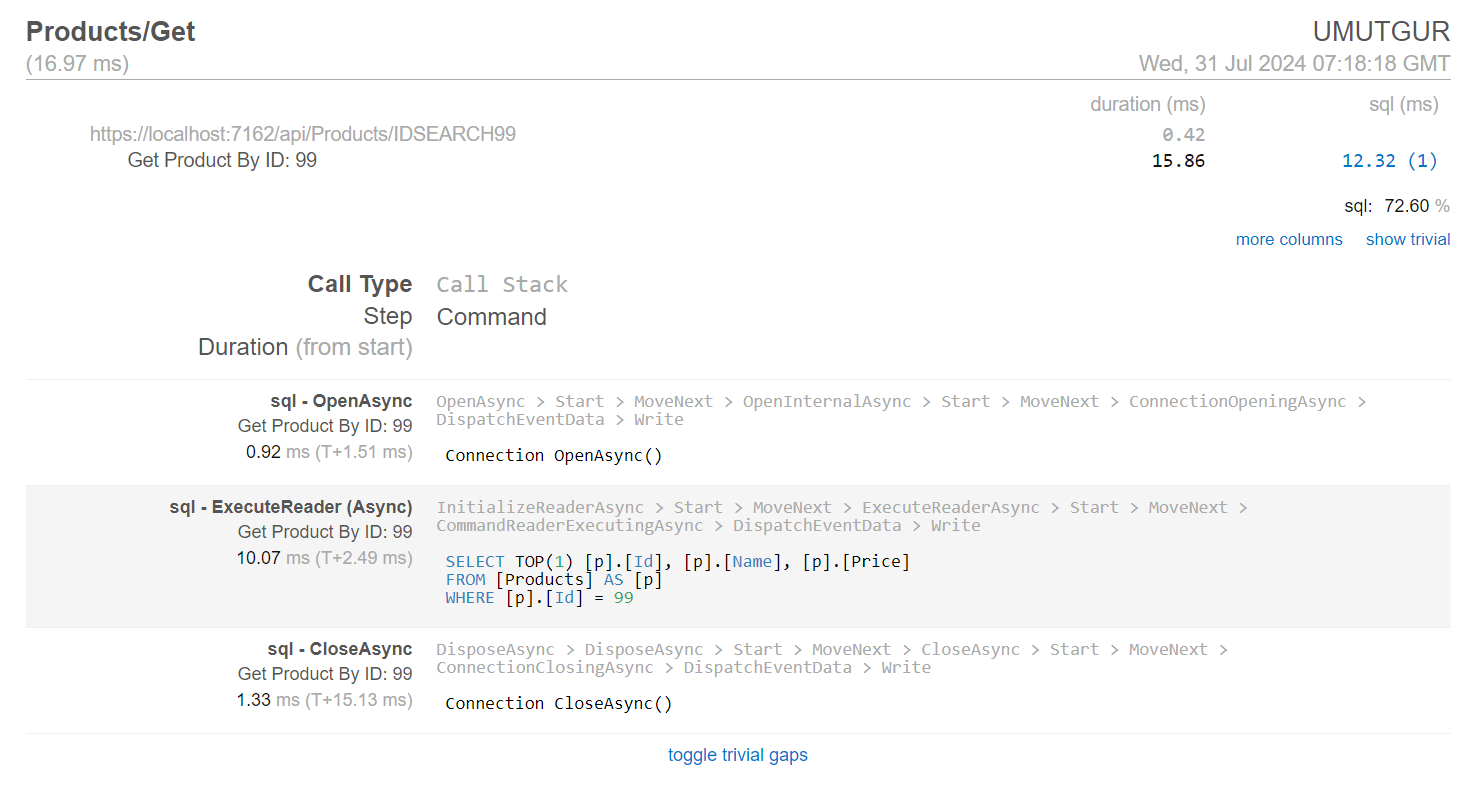
4.4 **MiniProfiler UI'sını Kontrol Etme**

* Uygulamamızı çalıştırıldı ve tarayıcımızda MiniProfiler UI'yı açıldı profil adımlarını kontrol edildi. Profil sonuçlarını https://localhost:5001/profiler/results-index ve https://localhost:5001/profiler/results adresinde görüntülendi.

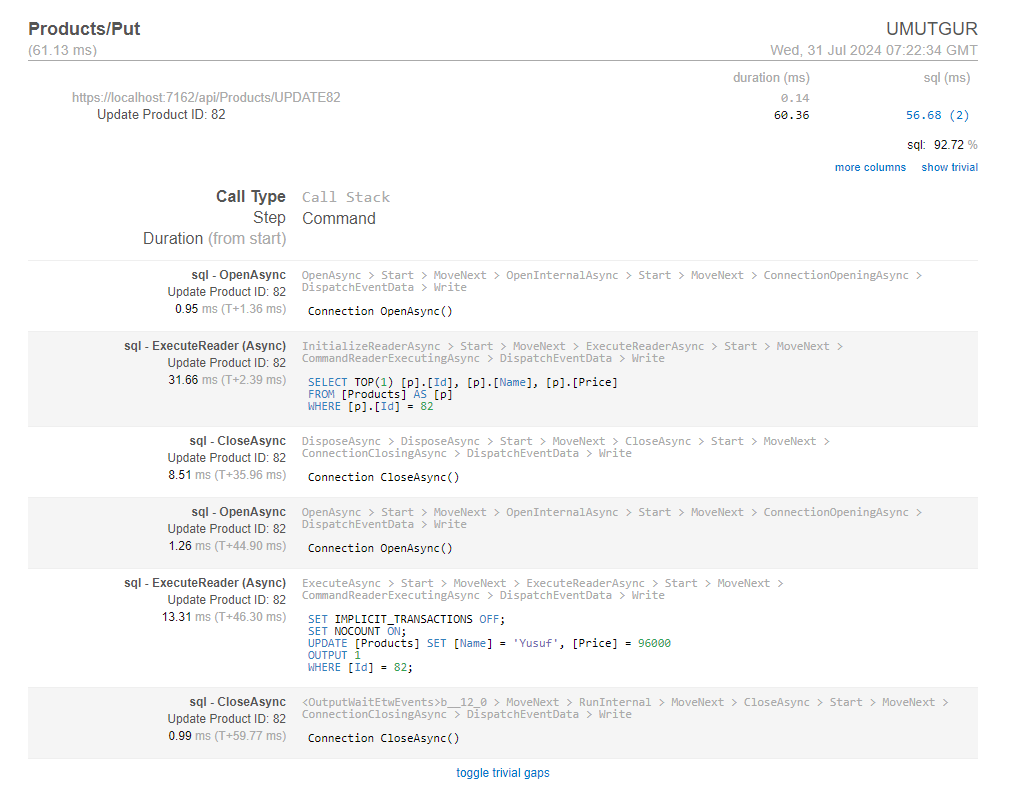
**5.Gösterim ve Sonuç:**

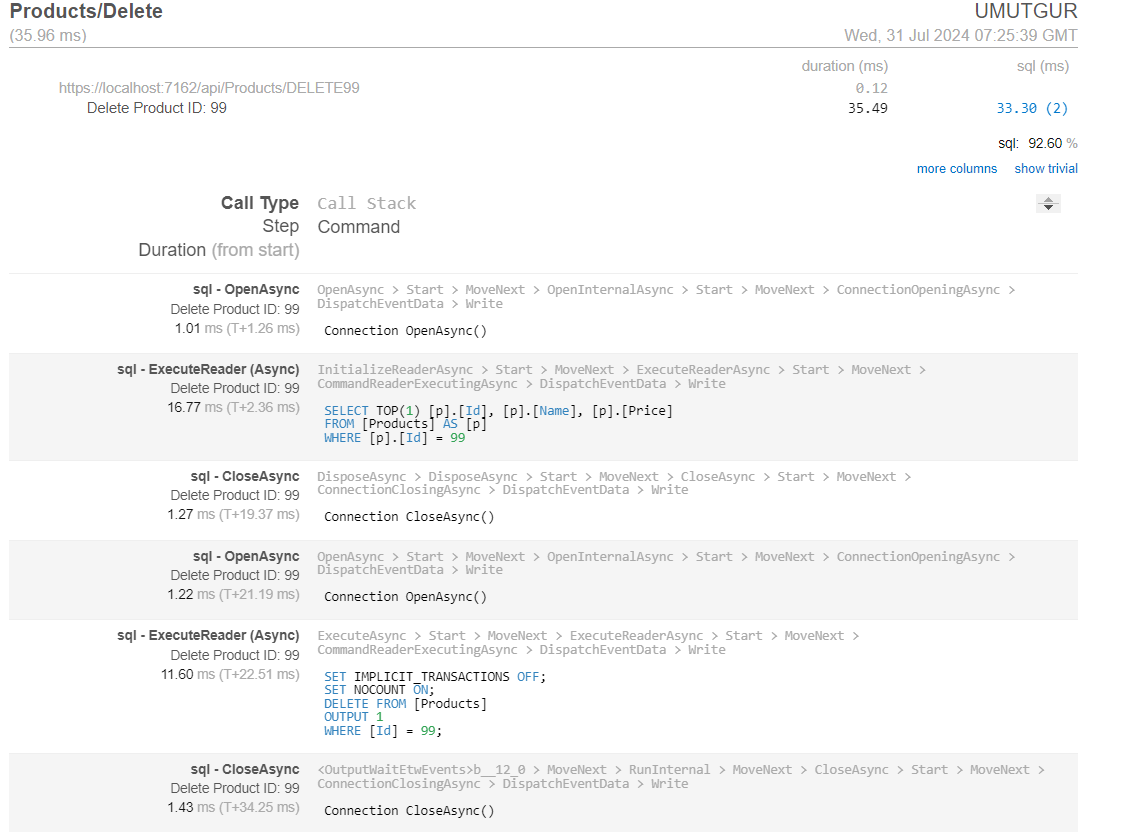
5.1 **Profillerin gösterimi**

****







****

5.2 **Profillerin yorumlanması ve sonuç**

Profil çıktılarımı incelediğimde yapılması en uzun süren işlemlerin sırasıyla Update Product ID, Delete Product ID, Create New Product, Get All Products ve Get Product By ID steplerinin olduğunu gözlemledim.  
  
İşlemlerin detaylarına bakıldığında tüm işlemlerde şüphesiz en çok zaman harcayan işlem sql-ExecuteReader işlemi gözükmekte. Profillere bakıldığında genellikle işlemlerin %80 inden çoğunu sql işlemlerinin harcadığını gözlemledim.  
  
Bu sonuçlara bakıldığında projenin hızlanırılabilmesi için veritabanı kısmda geliştirmeler yapılabileceği söylenebilir.