**TechCareerShoppingList Proje Detayları**

**Kullanılan Mimari :** CQRS

**Entity FrameWork Yaklaşımı :** CodeFirst

**BackEnd :** Asp.NetCoreWepApi

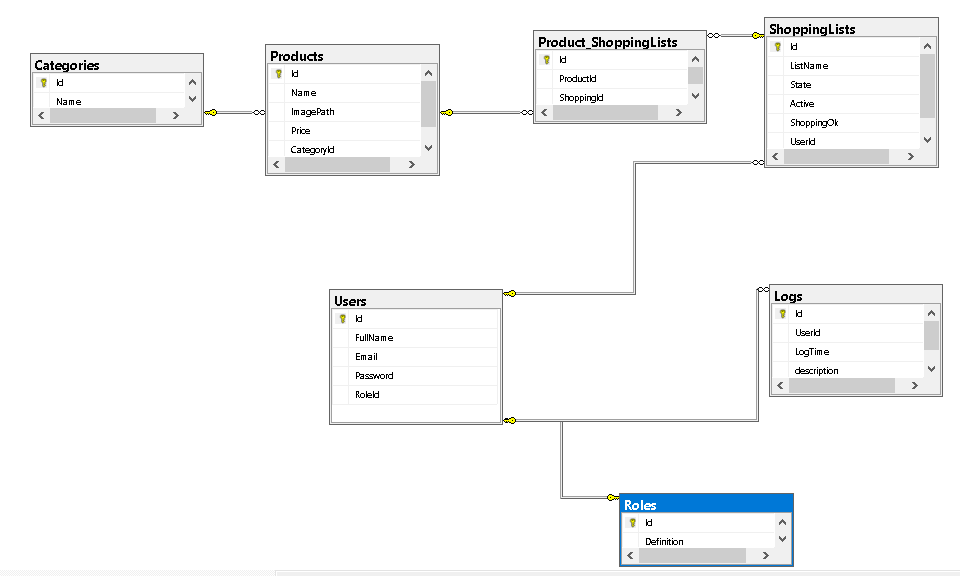
**FrontEnd :** Asp.NetCoreWepApp

**Yazılım Geliştirme Modelleri :** Code & Fix

**Kimlik Doğrulama :** Jwt

**Not: Proje teslimine son bir hafta kalasıya bilgisayarım kırıldı ve çok yavaş çalıştı. Bilgisayarımın yavaşlığı videoya çekildi ve Ek klasörü altında paylaşıldı. Bu sebep dolayısıyla planlayıpta yapamadığım yerler aşağıdaki anlatımda teknik aksaklı adı altı da açıklandı.**

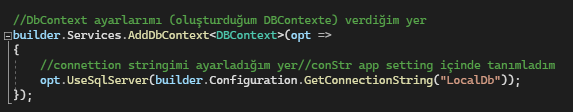
DataBase



Tablolarım ve tablolarımım arasındaki ilişki yukarıdaki gibidir.

Categori&Product, User&Log, User&Roles, User & ShoppinList arasında one to many ilişki varken Product ve ShoppingList arasında(bir ürün birden fazla listede olacağı gibi bir listenin birden fazla ürünü olabilir) many to many ilişki vardır.

Database, CodeFirst yaklaşımıyla oluşturuldu. Projede Data klasörü altında gerekli Entity’ler oluşturuldu. Tablolardaki field’ların özellikleri ilk olarak (kaç karekterli olacağı, boş geçilemez gibi..) data annotion yöntemiyle attribute eklenerek yapıldı. Daha sonra data annotion işlemleri ile yaptığım işlemlerimi iptal ederek, fluent api yöntemiyle DBContext sınıfımda bir araya getirildi. Solid Prensiplerinden Single Responsibility aykırı olduğu için ve kod kalabağı olduğu için IEntityTypeConfiguration arayüzünden implement alınarak her entity için ayrı ayrı cofigurasyon sınıfları oluşturuldu ve DBContext’e bu sınıflar çağrıldı. DBContext sınıfım için ayarlar(connection string gibi) program.cs de verildi.

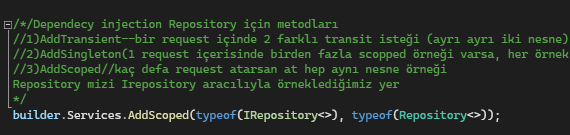


Program ilk defa ayağı kalktığında verilecek Fake datalar oluşturuldu(Role,Category,User,Product). Program ilk defa ayağa kalktığında Fake datalarım database eklenecek ve appseting içerinde tanımlayacağım global settings ayarımla, sadece program ilk defa ayağı kalktığında kayıt yapılacağı kontrolü yapılacaktı. Daha farklı nasıl yapılabilir diye araştırıldı ve configurasyon işlemleri sırasında HasData fonksiyonuyla yapılabileceği bulundu.

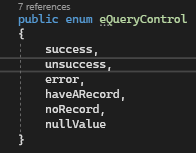


Tek başına Category veya Role eklemede sıkıntı yaşanılmadı ama User ve Product eklenirken hatayla karşılaşıldı(Foreingkeyden dolayı).Teknik sebeplerde dolayı yeterince test yapamadığım için fake data ekleme işlemi şuan kodumda yorum satırı halinde. Gerekli ayarlamalar yapıldıktan sonra Migration oluşturuldu ve Database gönderildi.

CRUD işlemlerimini her tablo için sürekli kod tekrarı yapılması için ve sınıflar arasındaki bağımlılıkları azaltmak için IRepository Generic arayüzü ve bu arayüzden implement alan Repository tanımlandı. Program.cs içinde IRepository arayüzüm Repository sınıfıma injection edildi. Yani örneklemesi alındı.

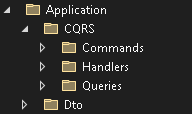


Daha önceki projelerimde kullandığım AsyncSemahore Tools eklenip Repository çağrıldı. Sorgunun durumu kontrol edecek enum değerler verildi.

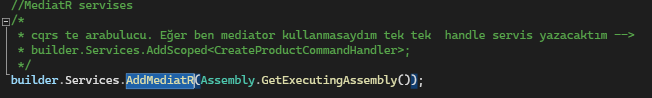


Ve yine Tools içinde QueryControl sınıfında kullanıldı. Ancak sınıfım sorgularda kullanılmadı. Bu kotrolleri en son yapmayı planlamıştım. Yetişmedi.

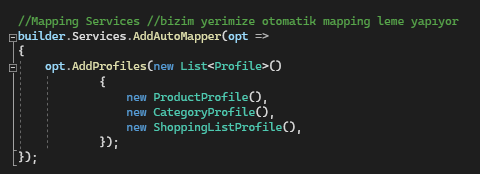
CQRS mimarisini alt yapısı oluşturuldu.



CQRS’te arabulucu sınıf olan MediatR paketi eklendi ve program.cs gerekli ayarlamalar yapılarak kullanıma hazır hale getirildi.

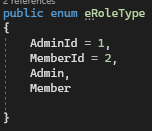


Yine otomatik Mappingleme için AutoMapper paketi eklendi.



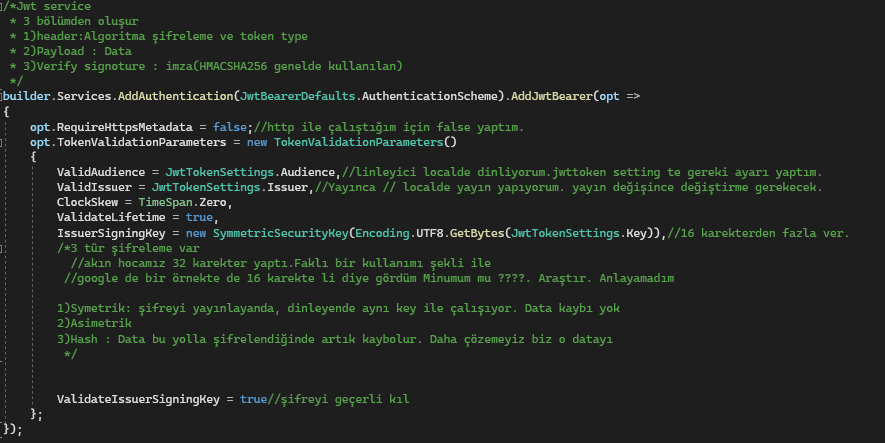
Category, Product,User, ShoppingList tablolarımla ilgili CRUD işlemleri gerçekleştirildi.

Role tablomdaki veri Fake data olarak verilmesi planlandığı için herhangi bir CRUD işlemi yapılmadı. Enum değer verildi.



Log Tablom ise hangi Crud işlemi hangi kullanıcı tarafından yapılmış bilgisini tutacaktı. Bu tablo sonradan eklenmiş bir tablo olduğu için Handlel’arı oluşturulmadı. QueryControl yaparken log tutma işlemi gerçekleşecekti. Bilgisayarımdaki teknik arıza nedeniyle yapılamadı. Log tablosuyla beraber düzenleme yaptığım ilişki product ve shoppingList arasında many to many ilişki getirmek. Kodlamada şuanki haliyle one to many şeklinde. Düzeltilmesi sonraya bırakılmıştı. Teknik arıza sebebiyle buda düzeltilemedi.

Kullanıcı kimlik doğrulama için Jwt paketi eklendi ve Tools klasör altında gerekli ayarları yapıldı. Aynı zamanda program.cs de de bu ayarlar verildi.

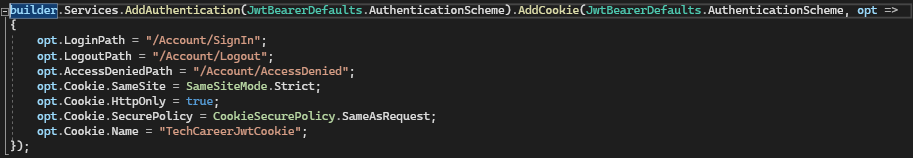


Kullanıcı girişi de sağlandıktan sonra Frontend kodlamasına geçildi.

Frontend kısmında ilk olarak MVC yapısı oluşturuldu. HttpClient ile iki proje arasında haberleşme sağlandı. HttpClient ayarı program cs.de verildi.



Kullanıcı kimlik doğrulama için cookie eklendi.



Kullanıcı girişi için arayüz oluşturuldu ve kodlaması gerçekleştirildi. Role göre farklı iki sayfaya yönlendirildi. Admin ve Member. AdminPage ve MemberPage sayfaları tasarlanmadı. AdminPagede Category, Product… ekle sil güncelleme gibi işlemler yapılacaktı. MemberPage de Productlerin listelenmesi, ShoppingList oluşturma gibi işlemler yapılacaktı. Kullanıcı arayüzünden oluşturulan token kullanılarak Category ekleme silme güncelleme, listeleme işlemleri gerçekleştirildi.