Bölüm 2:

Proje Onion Arch. İle inşa edildi.

İlk başta boş bir solution oluşturuldu. Daha sonra 4 tane klasör oluşturuldu.

-Core

* Core katmanında en iç katman entity’ler ve aplikasyon işlemleri yapılmaktadır. Core katmanında 2 tane sınıf var birincisi Domain(modellerin) yer aldığı sınıf. İkincisi Application(interface,dtos,Usecasess(service))’lerin yer aldığı yapılar vardır.

-Application(Sınıfı) Oluşturuldu.

* + Dtos, Interfaces, Usecasess oluşturuldu.
  + Dto içerisinde her bir modelin dışarıya veya database nasıl alınması veya çıkarılması için ayrı ayrı sınıflar oluşturuldu.(Result,Create,Update,GetByID)
  + Interface içerisinde Repository pattern kullanılarak, Bir generic servis oluşturuldu.
  + Usecasess klasöründe, her bir modele ait servisler oluşturuldu(ICategoryService, CategoryService). Bu klasörde IRepository’nin implemente edilmiş sınıfı olan Repository’i CategoryService içerisinde Dependency Injection ile kullanıldı. Daha sonra bu ICategoryService CategoryApi’de kullanıldı.

-Domain(Sınıfı) Oluşturuldu.

* + Domain içerisinde modeller oluşturuldu.(Category)

-Infrastructure(Altyapı)

* Bu katmanda modellerin database’e aktarılması için context’ler, migration’lar, repository’ler oluşturuldu.
* Bu katmanda database’ile iletişim kurmak için tasarlanan ORM aracı EF Core kullanıldı. Bunun için 4 tane paket yüklendi. (Ef Core, EF Core.Design, EF Core.SQLServer, EF Core.Tools)

-Persistence(Kalıcı) Proje Sınıfı Oluşturuldu.

* + - Context klasörü oluşturuldu içerisinde SQL bağlantı adresi ve Dbset’ler oluşturuldu.
    - Repositories klasörü oluşturuldu. Bunun içerisinde ise sınıf oluşturularak generic IRepository interface’ten kalıtım aldı. IRepository’deki metotların işlemlerini Repository sınıfı oluşturarak dolduruldu. Repository sınıfında AppDbContext DI ile entegre edilip, EF Core kullanıldı.

-Presentation(Sunum Katmanı)

-WebApi Projesi oluşturuldu.

* Daha sonra ApiController oluşturuldu.(CategoriesController)
* CategoriesController içerisinde Api’ye gelen istek neticesinde veriler ya database’ten gönderileceği veya çekileceği için Repository ile iletişim kurmak gerekiyor. Bunun içinde DI ile ICategoryService çağrıldı. Çünkü ICategoryService’in implementasyonun içerisinde yani CategoryService içerisinde IRepository DI ile çağrılmış zaten onun da implementesi Repository olduğu için direkt database ile iletişimi kurmuş olduk.
* Her bir metot(post, get, put, delete) için ayrı ayrı Aksiyonlar tanımladık. (GetAllCategories, GetByIdCategory, CreateCategoryAsync, UpdateCategoryAsync, DeleteCategory)
* Verileri getirme işlemleri için bir değer döndürüldü. Crud işlemleri için döndürülmedi çünkü veriler database’e gidecek client tarafına gelmeyecek.
* API tarafını görebilmek için Swagger kurulması gerekiyor ilgili paket (Swashbuckle.AspNetCore)
* Daha sonra program.cs içerisinde DI’ların nereye gideceğini nasıl oluşturulacağının yöntemini oluşturmamız gerekmektedir. DbContext’i tanıtmamız gerekmektedir.
* Swagger ile Api çalıştıracağımızı ayarlamak için aynı şekilde.