1. Northwind veri tabanı SQL Server içerisinde hazır hale getirilir.

2. CustomerCustomerDemo ve CustomerDemographics isimli tablolar sırasıyla silinsin. Bunlar hiçbir işimize yaramayacağı için siliyoruz.

3. Order Details tablosunun ismindeki boşluk silinsin. Region tablosunun ismi Regions olarak değiştirilsin. (Bunu yapmamızın amacı Scaffolding esnasında singularizing (tekil ada dönüştürme) işleminin güzelce uygulanmasını sağlamak.)

4. .NET MVC CORE projesi açalım.

5. Microsoft.EntityFrameworkCore.Tools ve Microsoft.EntityFrameworkCore.SqlServer paketleri nuget aracılığıyla projeye eklensin.

6. Proje içerisine bu veri tabanının ilişkisel hali olan DBContext eklensin.

Bunun için package manager console'da default project olarak projemiz seçildikten sonra solution içindeki projeye change directory yaptıktan sonra aşağıdaki komut çalıştırılmalıdır.

```
Scaffold-DbContext "Server=.\SQLEXPRESS;Database=NorthwindDB;Trusted_Connection=True;TrustServerCertificate=True;" Microsoft.EntityFrameworkCore.SqlServer -t Categories,Customers,Employees,Orders,OrderDetails,Products,Regions,Shippers,Suppliers,Territories,EmployeeTerritories -OutputDir Context –force
```

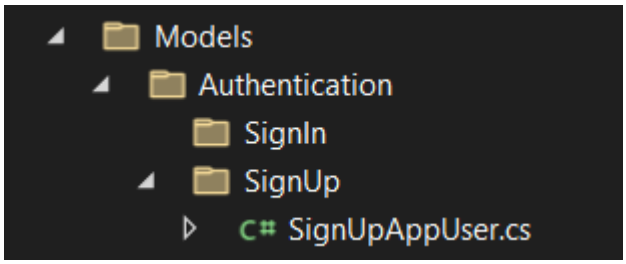
7. Oluşturulan DBContext'imizi servislere dahil edelim.

8. HomeController constructor'ında Northwind için bir context ataması yapalım. Bu nesne her zaman yaptığımız _context nesnesidir.

9. Veri tabanı çalışıyor mu diye Home/Index'te OkResult ile herhangi bir product nesnesini cevap içerisinde gönderiniz.

1. Microsoft.AspNetCore.Identity.EntityFrameworkCore (6.0.15 sürümü) ve Microsoft.AspNetCore.Authentication.JwtBearer (6.0.15 sürümü) paketlerini projemize dahil edelim.

2. Aşağıdaki gibi klasör sistemi oluşturunuz ve SignUpAppUser.cs adında sınıfı ekleyiniz. Bu sınıf kayıt olan kullanıcıların verilerini bind etmek için kullanılacaktır.



SignUpAppUser.cs dosyası aşağıdaki gibidir.

```
public class SignUpAppUser
{
	[Required(ErrorMessage = "Bu bilginin girilmesi zorunludur.")]
	public string? Username { get; set; }
	[EmailAddress]
	[Required(ErrorMessage = "Bu bilginin girilmesi zorunludur.")]
	public string? Email { get; set; }
	[MinLength(8)]
	[Required(ErrorMessage = "Bu bilginin girilmesi zorunludur.")]
	[DataType(DataType.Password)]
	public string? Password { get; set; }
}
```

3. Northwind DBContext sınıfının miras aldığı sınıfı DBContext yerine aşağıdaki gibi ayarlıyoruz.

```
NorthwindContext : IdentityDbContext<IdentityUser>
```

4. OnModelCreating metodu içerisine Identity tablolarının da özelliklerinin belirlenmesi için aşağıdaki satırı ekliyoruz.

```
base.OnModelCreating(modelBuilder);
```

5. Identity ve Authentication servislerini ekliyoruz.

```
builder.Services.AddIdentity<IdentityUser,
IdentityRole>().AddEntityFrameworkStores<NorthwindContext>().AddDefaultTokenProvi
ders();

builder.Services.AddAuthentication(
		opt =>
		{
			opt.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
			opt.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
			opt.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;

		}
);
```

6. Yaptığımız ayarlamaları veri tabanına da enjekte etmeye çalışacağız. Buna migration yani göç ettirme deniyor. Önce bir göç etme kuralı oluşturalım bunun için aşağıdaki komutu package manager console'da çalıştırıyoruz.

add-migration v1.0.0

Sonrasında Migrations klasörü oluşuyor.

Bizim veri tabanımız daha önceden hazır olduğu için bu tablolar ve Identity tabloları göç ettirme kuralına işlendi. Veri tabanımızda var olan tabloları bu göç ettirme kuralı dosyasından silelim.

Nasıl olması gerektiğine size göndermiş olduğum örnek projedeki Migrations klasörüm içerisindeki 20230317210416_v1.0.0.cs isimli dosyadan bakabilirsiniz.

Bu göç ettirme kuralının veri tabanına da uygulanması için aşağıdaki komutu package manager console'da çalıştırıyoruz.

update-database

7. 3 tane rol ekleyelim. Bunun için NorthwinContext içerisinden ekleme yapacağız. AddDummyRoles adından metot oluşturup 3 dataya sahip olduğu bilgisini vereceğiz ve bu metotu OnModelCreating metotu içerisinden çağıracağız. Ve sonra migration ekleyip update-database diyerek bu 3 rolün veri tabanına eklenmesini sağlayacağız. OnModelCreating metotu model oluşturulurken devreye girer. migration eklediğimizde bu model oluşturulur böylelikle migration dosyamızda hangi verileri eklediğimiz bilgisi de yer alır. update-database diyerek bu migration dosyası içerisindeki bilgiler veri tabanına uygulanmış olur.

8. UserController adında bir Controller oluşturalım. Bu controller örnek projedeki gibidir. Doğrudan kodları kullanabilirsiniz ama muhakkak anlamanız gerekmektedir. Yoksa değişiklik yapabilme imkanınız olmaz. Milyonlarca ihtimale ulaşabilmek için değişiklik yapabilmemiz şarttır!

Eklediğimiz dosyalar aşağıdaki gibidir. Proje içerisinden bulup inceleyiniz.

AppResponse.cs

SignUp.cshtml