28 Şubat- 7 Mart (1.Hafta)

İlk dönem projenin front-end kısmını bitirmiştim. Dönem arasında hiçbir bilgim olmayan ASP .Net Core dili hakkında videolar izleyerek, dokümanlar okuyarak, alıştırmalar yaparak web geliştirme konusunda bilgilenmeye başladım.

Öğrendiklerim:

ASP .Net Core ile web uygulamaları geliştirirken MVC yapısını kullanırız. MVC'deki M=Model

V=View

C=Controller anlamlarına geliyor.

MVC yapısını biraz kurcalayacak olursak:

Controller: Genellikle dosya yollarını belirtirken kullanırız. Projeye gelen ilk talebi controller'lar karşılar. Kullanıcının yaptığı istekleri alır. Ve onlara yanıt döndürür. Kullanıcıdan gelen HTTP isteklerini işler. (GET, POST, PUT, DELETE). Gelen verilerin hangilerinin View'e gideceğini de yine controlller belirler.

View: Projenin dosyaları %90 oranında burada yazılır. Statik dosyalar (wwwroot gibi) burada bulunur. Kullanıcının gördüğü arayüzdür diyebiliriz. HTML, CSS, JavaScript dosyaları da burada bulunur. Controller'dan gelen veriler burada kullanıcıya gösterilir. Genellikle *Razor View Engine* kullanarak dinamik içerikler oluşturur.

Model: Verilerin temsil edildiği yerdir. Genellikle veritabanı işlemleriyle ilgilenir.

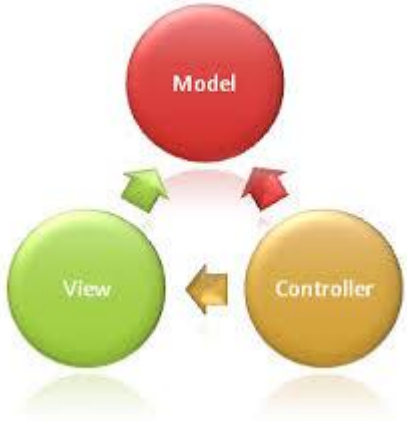
Entity Framework Core gibi ORM araçları ile birlikte kullanılır.

Özetlemek gerekirse kullanıcı tarafından bir istek geldiği anda:

Controller bu isteği alır ve Model'den de verileri alır.

View, gelen verileri işler ve HTML şeklinde kullanıcıya döndürür.

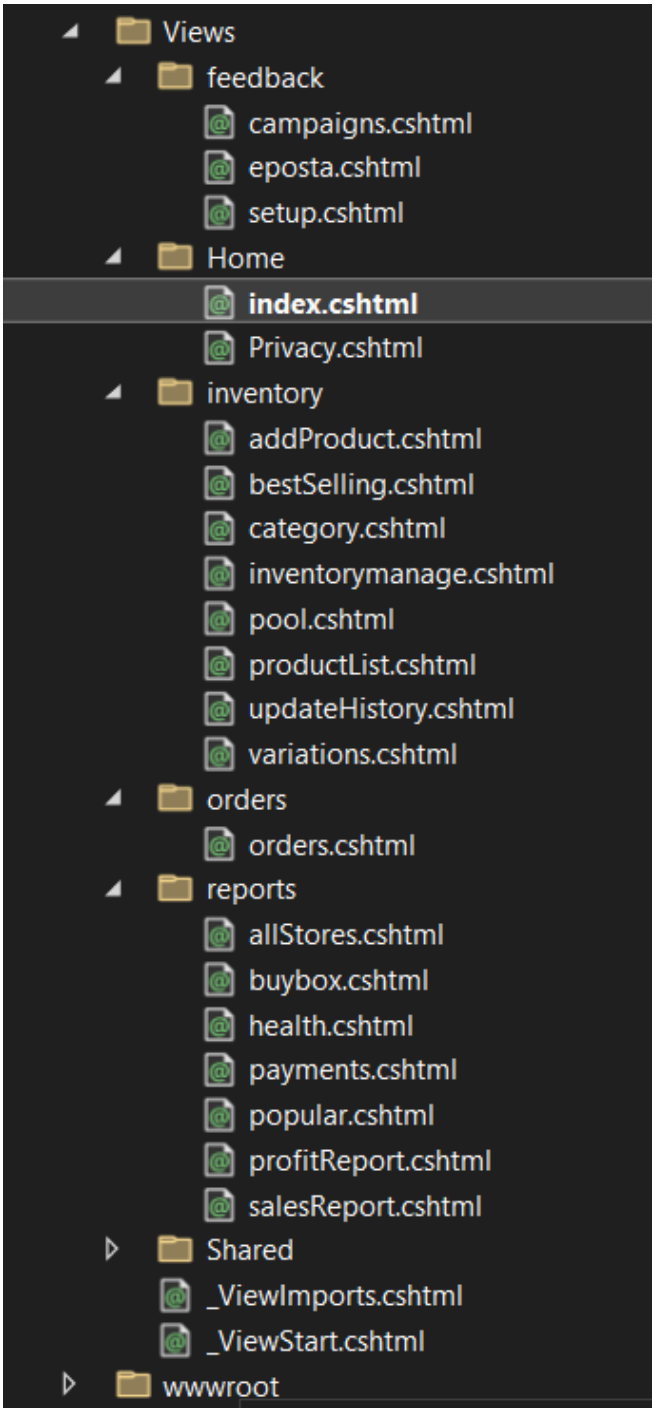
Kullanıcı sayfayı görüntüler.



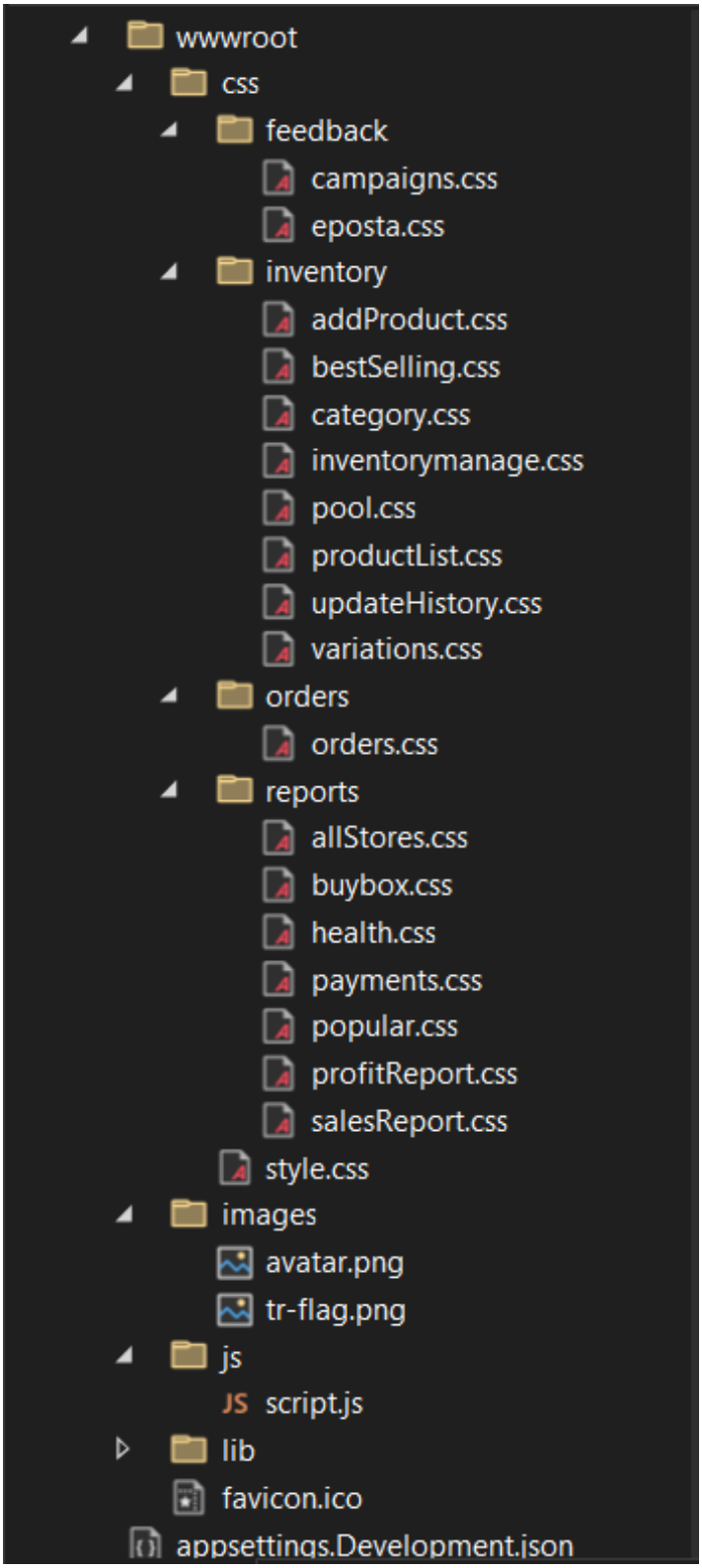
Bazı temel durumları öğrendim. Mesela ASP .Net Core ile yazılan html dosya uzantıları '.cshtml' şeklindedir. Ayrıca Layout dediğimiz bir yapı var. Bu dosya sürekli olarak kullanılan kodları yani mesela tüm sayfaların header kısmı ortak ise bu kodları tüm sayfalara teker teker yazmak yerine Layout içerisine yazıyoruz. Daha sonra sayfaların ilgili kısımlarına Layout dosyamızı bağlayarak kodları çağırıyoruz.

Şimdi yaptıklarım ile bunların hepsini örneklendireceğim.

Öncelikle benim tüm kodlarım HTML uzantılıydı. Bunları ASP .Net Core'a uyarlamak için .cshtml uzantılı hâle getirmem gerekiyordu.



Ve ayrıca ASP .Net Core'da Scoped CSS dediğimiz olay yüzünden tüm dosyalar için özel CSS dosyaları oluşturdum.



Benim tüm projelerimde üst kısım aynı şekilde bulunuyor:

Bu kısmın kodlarını tüm sayfalarda teker teker yazmıştım. Ve bu durum beni zorlamıştı. ASP .Net Core dilinde layout yapısı bunun önüne geçmek için yapılmış. Ben de bu kısmın kodlarını layout.cshtml dosyasının içine taşıdım.

_Layout.cshtml

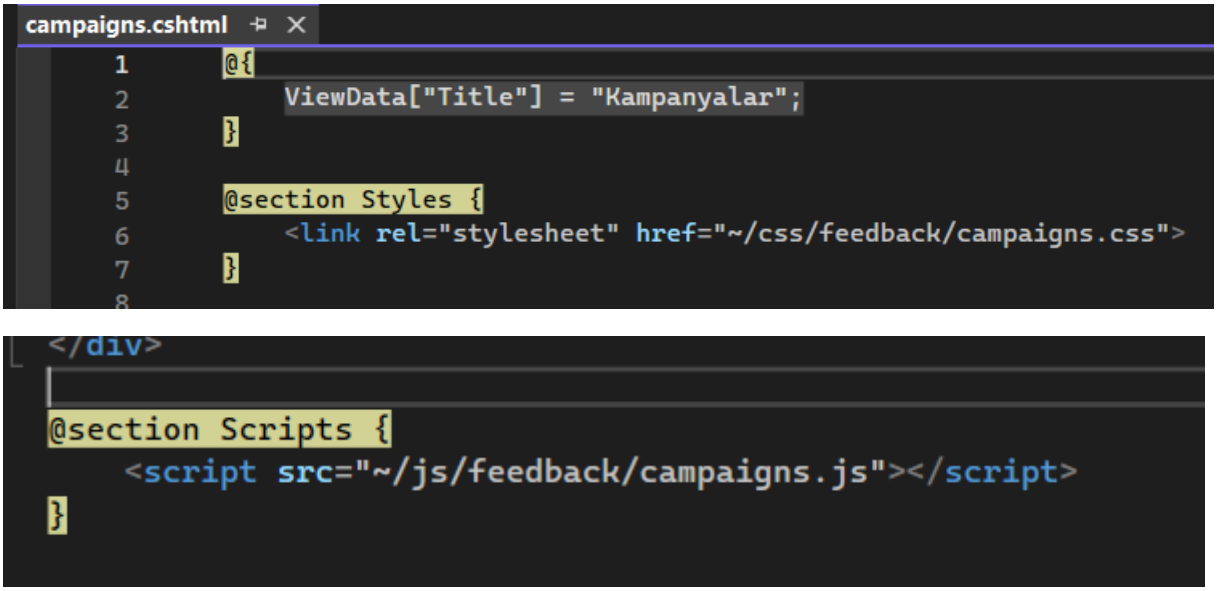
```
<!DOCTYPE html>
<html lang="tr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewData["Title"] - SellerRunning</title>
    <link rel="stylesheet" href="~/css/style.css">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

    @RenderSection("Styles", required: false)  @* Sayfalara özel stiller için *@
</head>
<body>
    <nav class="main-nav">
        <div class="nav-left">
            <div class="logo">
                <a href="/Home/index">
                    <img src="seller-running-logo.png" alt="SELLER RUNNING">
                </a>
            </div>
            <ul class="nav-menu">
                <li>
                    <a href="#">Siparişler</a>
                    <ul class="dropdown-menu">
                        <li><a href="/orders/orders">Siparişler</a></li>
                    </ul>
                </li>
                <li>
                    <a href="#">Envanter</a>
                    <ul class="dropdown-menu">
                        <li><a href="/inventory/inventorymanage">Envanter Yönetimi</a></li>
                        <li><a href="/inventory/addProduct">Yeni Ürün Ekle</a></li>
                        <li><a href="/inventory/variations">Ürün Varyasyonları</a></li>
                        <li><a href="/inventory/bestSelling">Çok Satan Ürünler</a></li>
                        <li><a href="/inventory/updateHistory">Güncelleme Geçmişi</a></li>
                        <li><a href="/inventory/category">Kategori Listeleri</a></li>
                        <li><a href="/inventory/productList">Ürün Listeleri</a></li>
                        <li><a href="/inventory/pool">Ortak Havuz</a></li>
```

_Layout.cshtml

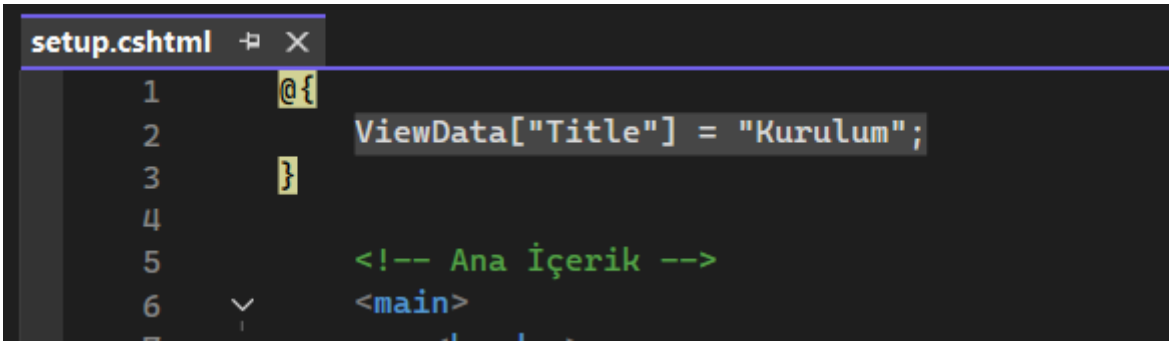
```
                        <li><a href="/reports/popular">Popüler Zamanlar</a></li>
                    </ul>
                </li>
                <li>
                    <a href="#">Feedback</a>
                    <ul class="dropdown-menu">
                        <li><a href="/feedback/setup">Kurulum</a></li>
                        <li><a href="/feedback/campaigns">Kampanyalar</a></li>
                        <li><a href="/feedback/eposta">Gönderilen E-Postalar</a></li>
                    </ul>
                </li>
            </ul>
        </div>
        <div class="nav-right">
            <div class="search-bar">
                <i class="material-icons">search</i>
                <input type="text" placeholder="Ara: ASIN, SKU, Sipariş no">
            </div>
            <div class="user-controls">
                <div class="language-select">
                    <img src="~/images/tr-flag.png" alt="TR" class="flag-icon">
                </div>
                <div class="user-avatar">
                    <img src="~/images/avatar.png" alt="Kullanıcı">
                </div>
            </div>
        </div>
    </nav>

    <main>
        @RenderBody()  @* Sayfaların içeriği burada yüklenir *@
    </main>

    @RenderSection("Scripts", required: false)  @* Sayfalara özel scriptler için *@
</body>
</html>

```

Şeklinde layout dosyamızın içi…

Biz neredeyse bütün sayfalar için özel CSS dosyası oluşturmuştuk. Bunu da göze alalım. Ve layout içerisindeki kodları çağıralım. Örnek olarak campaigns.cshtml dosyasının içini göstereceğim:



Burada layout içerisinde HTML, BODY etiketleri kullanıldığı bu tagları da kaldırdım. Layout içerisinde bulunan tüm kodları sayfalardan temizledim ve hepsini yukarıdaki gibi uyarladım.



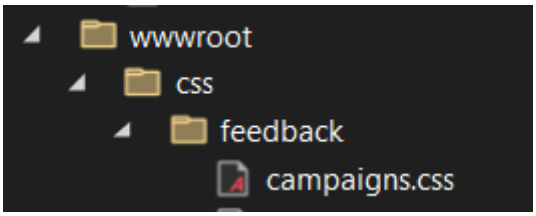
Burada da setup.cshtml dosyasının içeriği var. Bunun için özel bir CSS dosyasına henüz gerek duymadığım için eklemedim.

Biz bunları yaparak:

- Kod tekrarını önledik.
- Kodlar daha temiz ve okunaklı bir hâle geldi.

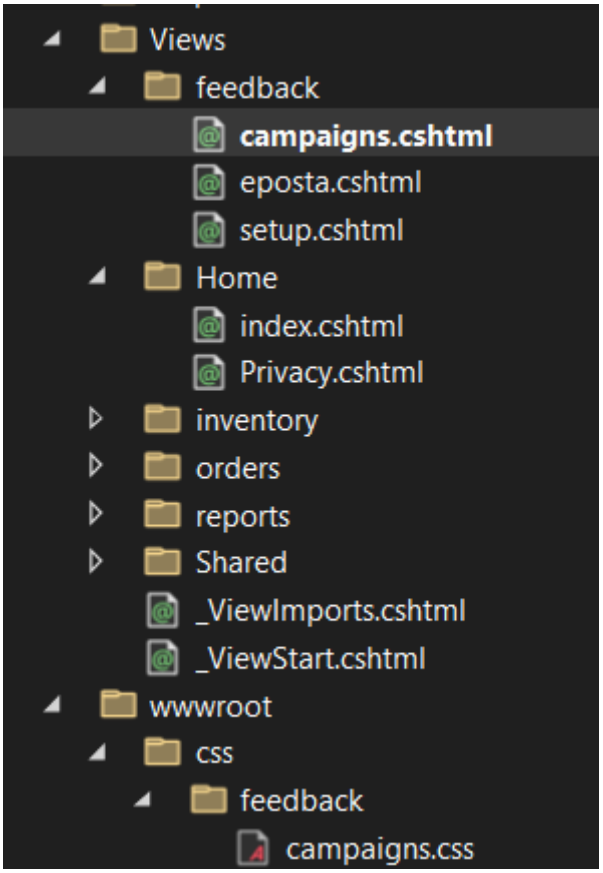
STATİK DOSYALARIN ÇAĞRILMASI:

Öğrendiğim şeylerden birisi de bu. Statik dosyalar güvenlik gerekçesi ile wwwroot klasörü altında bulunur. CSS, JavaScript, images dosyaları burada bulunuyor. Örnekle göstereceğim.



Burada wwwroot dizini altında css dizini içerisinde feedback klasörü içerisinde bulunan campaign.css dosyasını çağırırken:

Normalde HTML kodu yazarken: ../../wwwroot/css/feedback/campaign.css şeklinde çağırıyoruz.



Ama ASP .Net Core dilinde

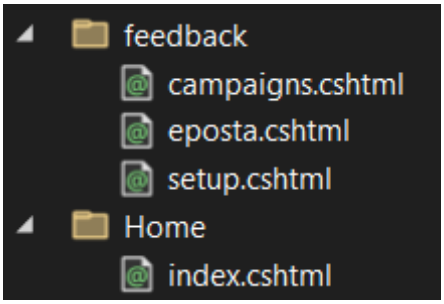
```
<link rel="stylesheet" href="~/css/feedback/campaigns.css">
```

Şeklinde çağırıyoruz. ~ işareti kök dizinini gösterir.

```
<img src="~/images/avatar.png" alt="Kullanıcı">
```

HREF İle Dosya Yolu Belirtme:

HTML kullanırken dosya yolunu:



index'den setup'a gitmek için

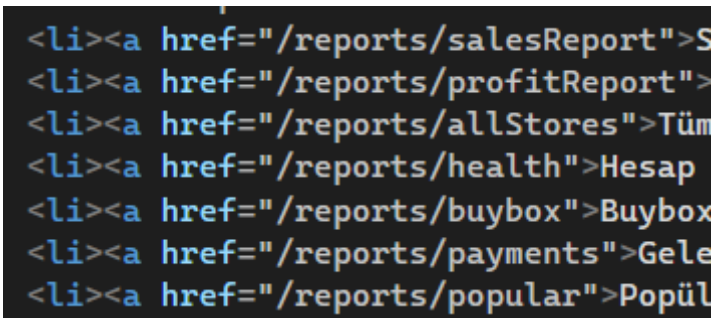
../feedback/setup.html şeklinde kod yazardık.

ASP .Net Core dilinde ise bu durum çok farklı.

```
<li><a href="/feedback/setup">Kurulum</a></li>
```

Şeklinde o sayfanın yolu belirtilir. Burada farklı olarak:

- "../" kullanılmıyor.
- ".cshtml" şeklinde dosya yolu yazılmıyor.



Bütün dosya yolları bütün sayfalarda düzeltildi.

Özet olarak HTML dosyalarını ASP .Net Core'a uyarladım. Layout ile kod temizlemesi yaptım. Dosya hakkında videoyu Google Drive'a yükledim. Linki burada.

GOOGLE DRİVE