



**ISPARTA UYGULAMALI BİLİMLER ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**BLG-305 WEB TEKNOLOJİLERİ ve PROGRAMLAMA DERSİ**

**YIL SONU PROJE RAPORU**

---

**Dersin Öğretim Üyesi**  
Dr. Öğr. Üyesi Serdar PAÇACI

**Dersin Öğretim Elemanı**

---

**Teslim Tarihi**

XX.01.2025

**Proje Adı : Sinema Dünyası**

## **1. Öğrenci(ler)**

- **Öğrenci 1**

**Öğrenci Adı :** Amir ELAHMED

**Öğrenci Numarası :** 2112721307

- **Öğrenci 2**

**Öğrenci Adı :** Mohamad ALKASSEM

**Öğrenci Numarası :** 2212721320

- **Öğrenci 3**

**Öğrenci Adı :** Abdurrahman ALİ

**Öğrenci Numarası :** 2112721309

## **2. İçindekiler**

- Kapak Sayfası
- Giriş
- Proje Hakkında Genel Bilgi
- Projenin ui/ux kısmı
- Proje Gereksinimleri ve Karşılanması

### **3. Giriş :**

Bu proje, web teknolojileri dersinde öğrendiğimiz temel bilgileri uygulamak amacıyla geliştirilmiştir. ASP.NET Core MVC kullanarak öğrendiğimiz kavramları pratiğe dökmeyi hedefledik.

Projenin temel fikri şu şekildedir: Admin kullanıcısı, giriş yaparak film bilgilerini ekleyebilir. Adminin, film adı, yayınladığı yıl, fiyat, oyuncu kadrosu gibi bilgileri ekleme yetkisi vardır. Admin, bu bilgileri ekledikten sonra, kullanıcılar bu filmleri ana sayfada görüntüleyebilirler. Ayrıca kullanıcıların iletişim kurmak veya not göndermek istedikleri bir mesaj gönderme özelliği de bulunmaktadır. Gönderilen bu mesajlar, admin tarafından görüntülenebilir.

Bunun yanı sıra, kullanıcılar film satın alma işlemi de gerçekleştirebilirler. Genel hatlarıyla, projenin amacı ve işleyışı bu şekildedir.

### **4. Proje Hakkında Genel Bilgi :**

Bu projenin amacı, ASP.NET Core MVC kullanarak web teknolojileri dersinde öğrendiğimiz temel bilgileri pratiğe dökmektir. Proje, kullanıcılara film bilgilerini görüntüleme, film satın alma ve admin kullanıcılarına ise film bilgilerini ekleme ve yönetme yetkisi verme işlevlerini sunar.

Proje kapsamında gerçekleştirilen başlıca özellikler şunlardır :

- **Admin Paneli** : Admin kullanıcıları, giriş yaptıktan sonra film adı, yayınladığı yıl, fiyat, oyuncu kadrosu gibi bilgileri ekleyebilir ve düzenleyebilirler.
- **Kullanıcı Arayüzü** : Kullanıcılar, ana sayfada mevcut filmleri görüntüleyebilir ve film satın alma işlemi gerçekleştirebilirler.
- **İletişim Formu** : Kullanıcılar, admin ile iletişim kurmak veya not göndermek için mesaj gönderme özelliğini kullanabilirler. Gönderilen bu mesajlar admin tarafından görüntülenebilir.
- **Veritabanı yapısı** : Projemizde, kullanıcılara ve adminlere ait verilerin güvenli bir şekilde saklanması için bir veritabanı yapılandırılmıştır. Veritabanında aşağıdaki tablolar ve veriler bulunmaktadır:

**1 . Kullanıcılar ve Adminler** : Kullanıcıların ve adminlerin giriş yapabilmesi için gerekli olan bilgiler (kullanıcı adı, şifre, e-posta adresi vb.) bu tabloda saklanır.

**2 . Filmler** : Filmlere ait bilgiler (film adı, yayınladığı yıl, fiyat, oyuncu kadrosu vb.) bu tabloda saklanır.

**3 . Mesajlar** : Kullanıcıların admin ile iletişim kurmak veya not göndermek için gönderdikleri mesajlar bu tabloda saklanır.

veritabanı, projede kullanılan tüm verilerin organize bir şekilde saklanması ve yönetilmesini sağlar. Bu sayede, kullanıcılar ve adminler arasındaki etkileşimler ve film bilgileri güvenli bir şekilde işlenir.

## 5. Projenin ui/ux kısmı

### 1- Kullanıcı sayfaları.

#### Üst Kısım

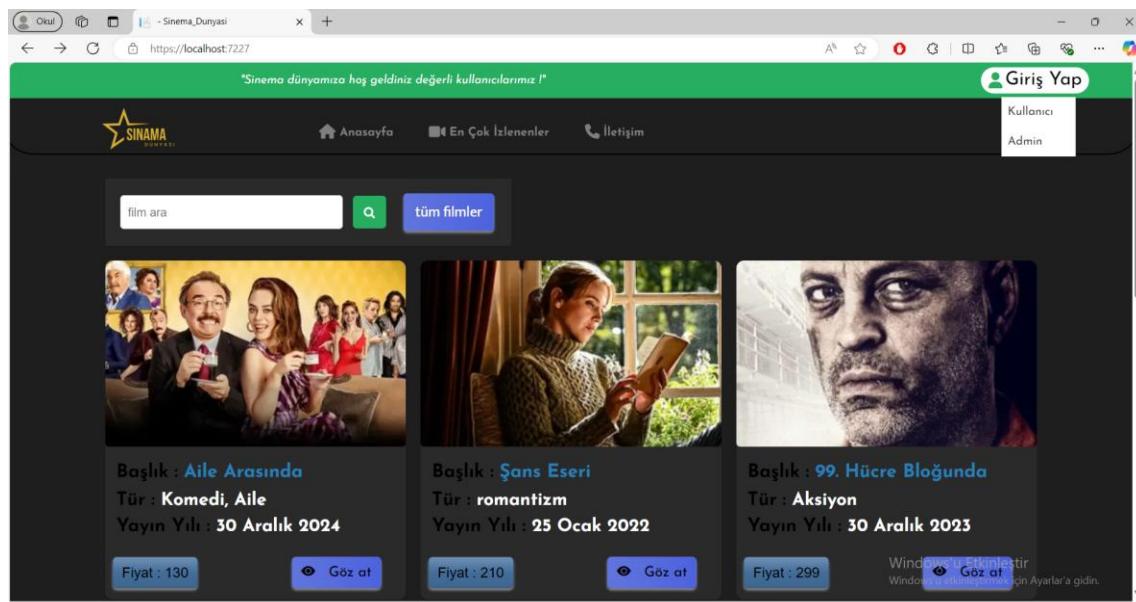
- Hoş Geldiniz Mesajı:** Sayfanın üst kısmında “Sinema dünyamıza hoş geldiniz, değerli kullanıcılarımız!” yazısı yer alır.
- Giriş Yap Butonu:** Kullanıcıların hesaplarına giriş yapmaları için.
- Kullanıcı Türü Seçimi:** “Kullanıcı” ve “Admin” seçenekleri ile kullanıcı türü seçimi yapılabilir.
- Menü Seçenekleri:** “Ana Sayfa”, “En Çok İzlenenler” ve “İletişim” gibi menü seçenekleri bulunur.

#### Orta Kısım

- Arama Çubuğu:** Kullanıcıların film araması yapabileceği bir alan.
- Tüm Filmler Butonu:** Tüm filmleri gösteren buton.

#### Fiyat Butonu

- Kullanıcı, fiyatlara tıkladığında eğer giriş yapmamışsa “Lütfen giriş yapmalısınız” şeklinde bir uyarı mesajı alır.



## Giriş Sayfası

Kullanıcılar, "Kullanıcı" butonuna tıkladığında aşağıdaki giriş yap sayfası açılacaktır:

Giriş Yap Sayfasında Kullanıcının Yapabileceği İşlemler

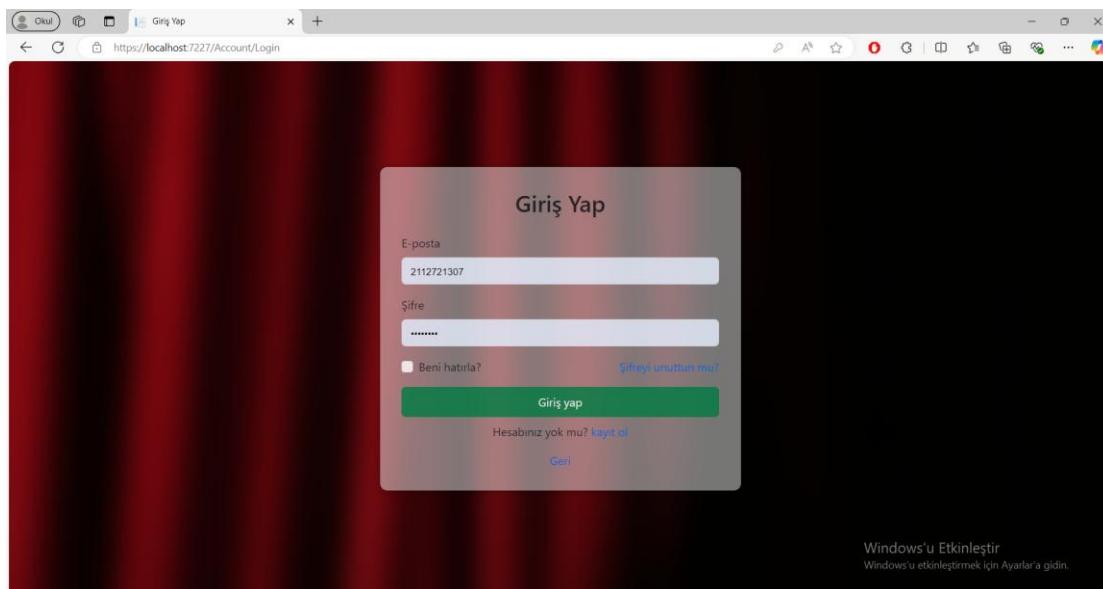
- **E-posta ve Şifre Giriş:** Kullanıcılar, e-posta adreslerini ve şifrelerini girer.
- **Beni Hatırla:** Kullanıcılar, hesap bilgilerinin hatırlanmasını isteyebilir.
- **Şifreni Unuttun mu:** Kullanıcılar şifrelerini unuttuklarında bu bağlantıya tıklayarak şifre yenileme sürecini başlatabilir.
- **Giriş Yap Butonu:** Kullanıcılar bilgilerini girip bu butona basarak sisteme giriş yapabilirler.

## Uyarı Mesajları

- **Geçersiz Hesap:** Kullanıcı geçerli olmayan bir e-posta adresi girdiğinde "Geçersiz hesap" uyarısı alır.
- **E-posta Formatı:** Kullanıcı e-posta formatında olmayan bir adres girdiğinde "Lütfen geçerli bir e-posta adresi giriniz" uyarısı alır.
- **Yanlış Şifre:** Kullanıcı doğru e-posta adresini girdiği halde yanlış şifre yazdığında "Şifreniz yanlış" uyarısı alır.

## Geri Butonu

- **Geri:** Kullanıcı "Geri" butonuna tıkladığında ana sayfaya geri döner.



## Kayıt Sayfası

Kullanıcı "Kayıt ol" kelimesine tıkladığında aşağıdaki kayıt sayfası açılacaktır:

Kayıt Sayfasında Kullanıcının Yapabileceği İşlemler

- **Ad:** Kullanıcı adını girer.
- **E-posta:** Kullanıcı e-posta adresini girer.
- **Şifre:** Kullanıcı şifresini belirler.
- **Şifre Tekrar:** Kullanıcı şifresini yeniden girer.

## Kayıt Sayfası Uyarı Mesajları

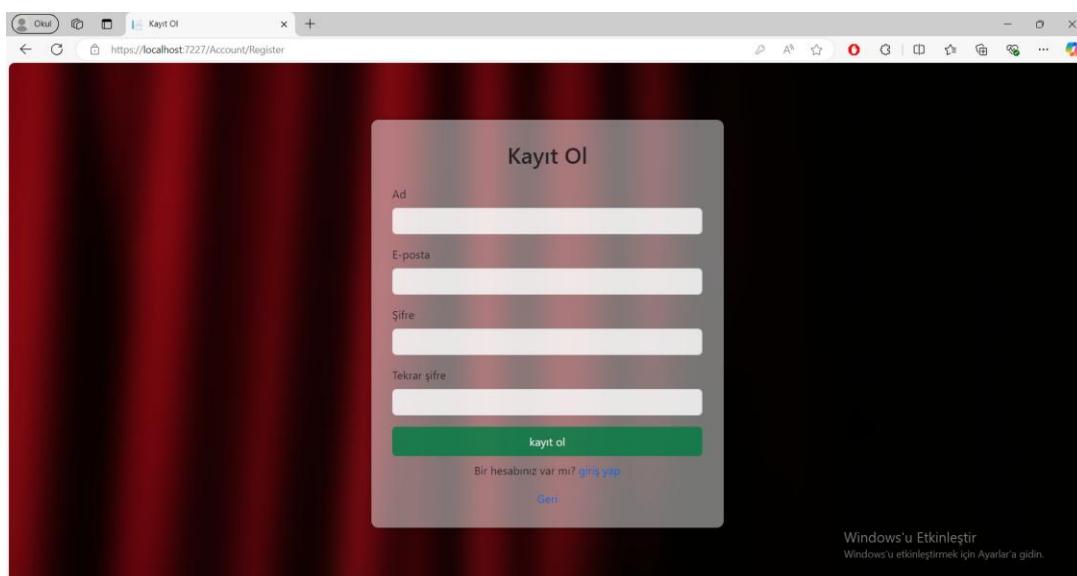
- **Boş Alan Uyarısı:** Kullanıcı herhangi bir alanı boş bıraktığında "Lütfen tüm alanları doldurunuz" uyarısı alır.
- **Geçersiz E-posta Uyarısı:** Kullanıcı geçerli formatta olmayan bir e-posta adresi girdiğinde "Lütfen geçerli bir e-posta adresi giriniz" uyarısı alır.
- **Şifre Uyumsuzluğu Uyarısı:** Kullanıcının girdiği şifreler birbirile uyusmadığında "Şifreler eşleşmiyor" uyarısı alır.

## Kayıt Ol Butonu

- Kullanıcı bilgilerini doğru bir şekilde girdikten sonra "Kayıt Ol" butonuna basar.
- Hata yoksa hesap veritabanına eklenir ve kullanıcı giriş sayfasına yönlendirilir.

## Geri Butonu

- **Geri:** Kullanıcı "Geri" butonuna tıkladığında giriş sayfasına geri döner.



## Şifre Unuttum , Hesap Doğrulama Sayfası

Kullanıcı "Şifre unuttum" kelimesine tıkladığında aşağıdaki şifre unuttum sayfası açılacaktır:

Şifre Unuttum Sayfasında Kullanıcının Yapabileceği İşlemler

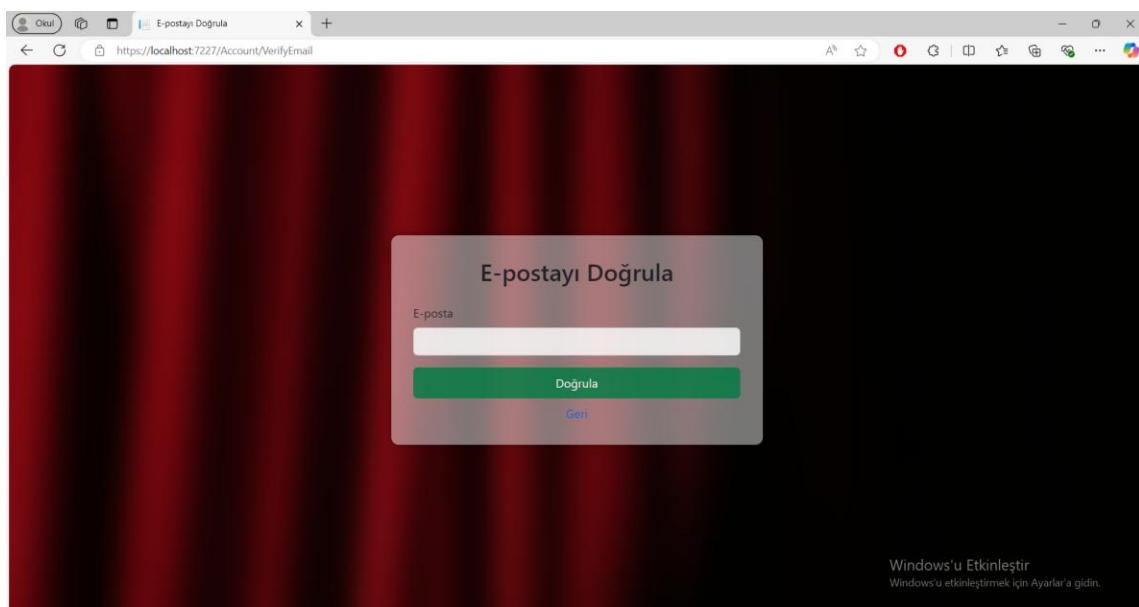
- **E-posta Girişи:** Kullanıcı geçerli bir e-posta adresi girmelidir.
- **Doğrulama Butonu:** Kullanıcı "Doğrula" butonuna basarak işlemi başlatır.

## Uyarı Mesajları

- Geçersiz E-posta Uyarısı: Kullanıcı geçerli formatta olmayan bir e-posta adresi girdiğinde "Lütfen geçerli bir e-posta adresi giriniz" uyarısı alır.
- Veritabanında Bulunmayan E-posta Uyarısı: Kullanıcının girdiği e-posta adresi veritabanında bulunmuyorsa "E-posta adresi sistemimizde kayıtlı değil" uyarısı alır.

## Geri Butonu

- **Geri:** Kullanıcı "Geri" butonuna tıkladığında giriş sayfasına geri döner.



## Hesap Doğrulama Sonrası Şifre Değiştirme Sayfası

Eğer kullanıcı doğru bir e-posta adresi girerse ve hesap doğrulanırsa aşağıdaki sayfa görünecektir:

### Şifre Değiştirme Sayfası

- E-posta:** Kullanıcı adı ve e-posta adresi otomatik olarak doldurulmuş (sadece okunabilir, readonly).
- Yeni Şifre:** Kullanıcının yeni şifreyi girmesi için boş alan.
- Yeni Şifre Tekrar:** Kullanıcının yeni şifreyi tekrar girmesi için boş alan.

### Uyarı Mesajları

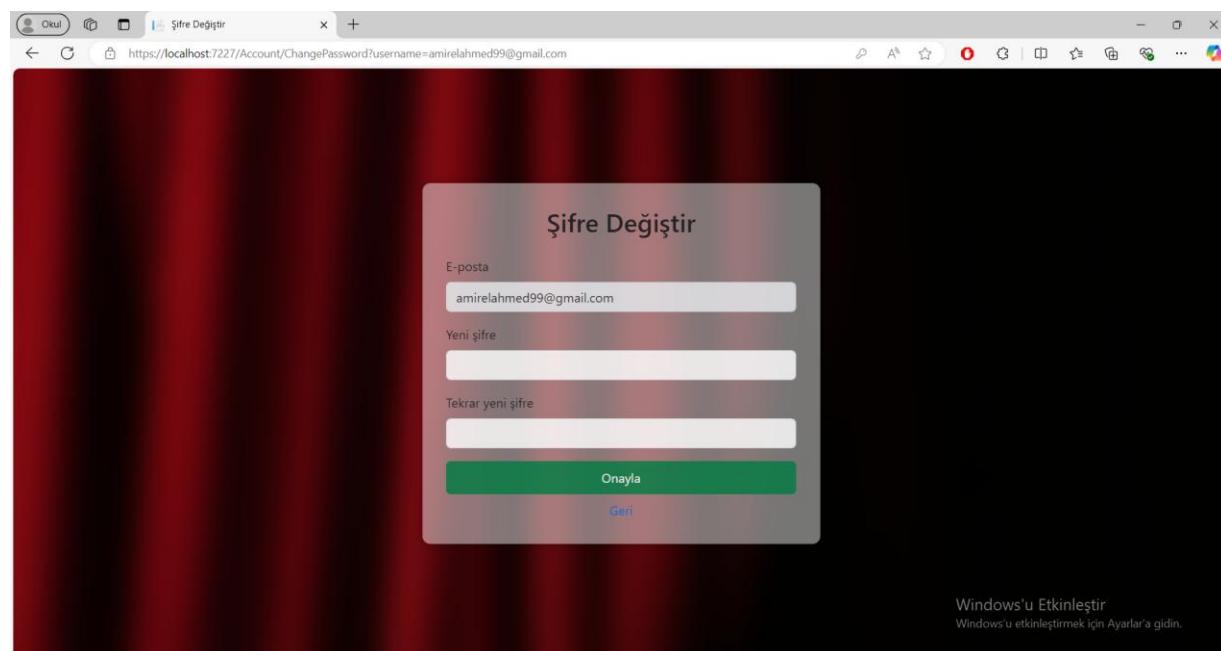
- Boş Alan Uyarısı:** Kullanıcı herhangi bir alanı boş bıraktığında "Lütfen tüm alanları doldurunuz" uyarısı alır.
- Şifre Uyumsuzluğu Uyarısı:** Kullanıcının girdiği şifreler birbirile uyışmadığında "Şifreler eşleşmiyor" uyarısı alır.

### Onayla Butonu

- Kullanıcı bilgilerini doğru bir şekilde girdikten sonra "Onayla" butonuna basar.
- Hata yoksa yeni şifre veritabanına eklenir ve kullanıcı giriş sayfasına yönlendirilir.

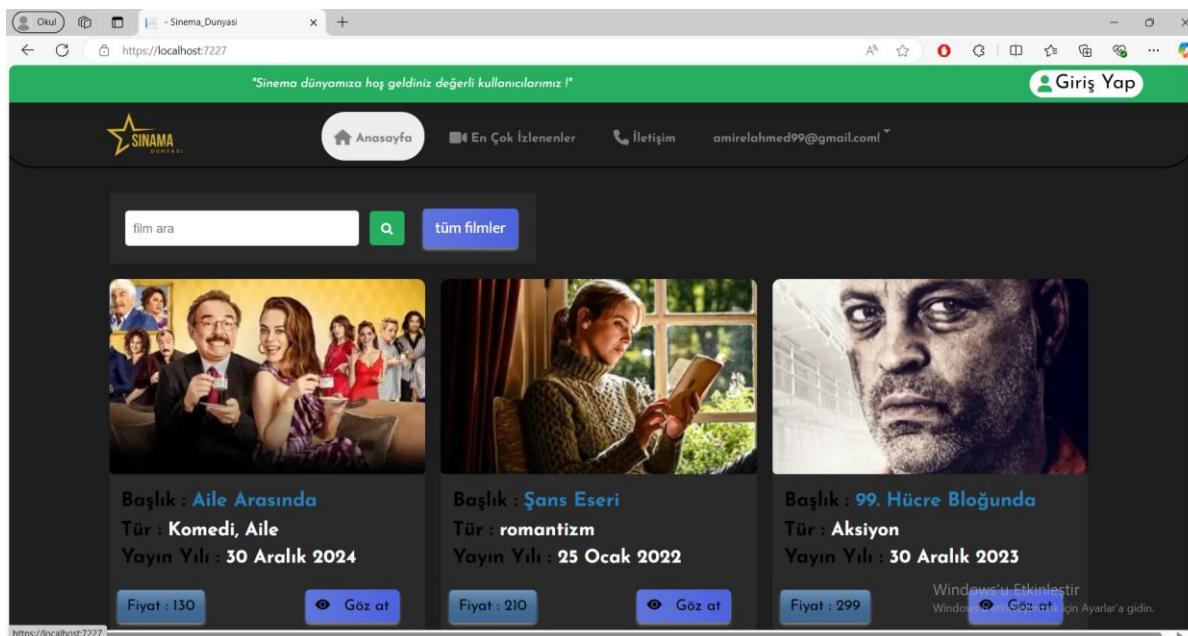
### Geri Butonu

- Geri:** Kullanıcı "Geri" butonuna tıkladığında şifre unuttum sayfasına geri döner.



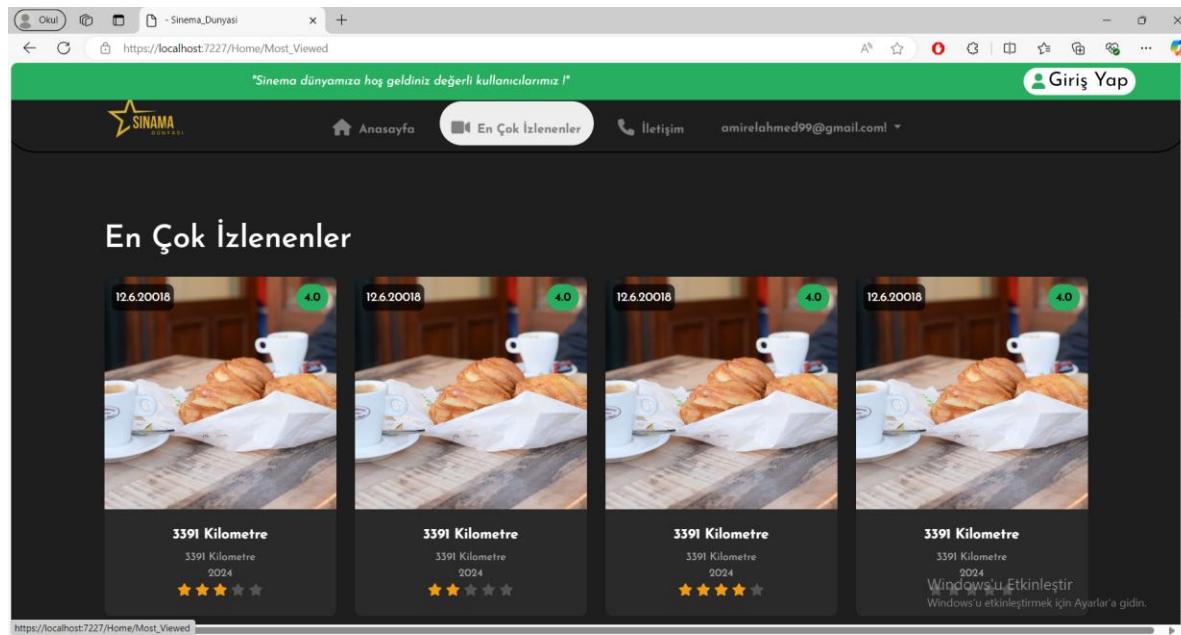
## Ana Sayfa

- **Hoş Geldiniz Mesajı:** Kullanıcıları sıcak bir şekilde karşılayan bir mesaj. "Sinema dünyamıza hoş geldiniz, değerli kullanıcılarımız!"
- **Film Afişleri:** Ana sayfada, güncel ve popüler filmlerin afişleri ve temel bilgileri yer alır. Kullanıcılar bu afişlere tıklayarak daha fazla bilgi alabilir.
- **Arama Çubuğu:** Kullanıcıların film araması yapabilecekleri bir alan.
- **Tüm Filmler Butonu:** Kullanıcıların tüm filmleri listeleyebilmesi için bir buton.
- **Kullanıcı Bilgileri:** Giriş yapmış kullanıcının e-posta adresi görüntülenir.
- **Menü Seçenekleri:** Ana sayfa, en çok izlenenler, iletişim ve kullanıcı türü seçimi gibi menü seçenekleri bulunur.
- **Film Bilgileri ve Fiyatlar:** Kullanıcılar filmlerin türlerini, çıkış tarihlerini ve fiyatlarını görebilir. Fiyatlara tıkladıklarında giriş yapmadılarsa uyarı mesajı alırlar.



## En Çok İzlenenler Sayfası

- **Sayfa Başlığı:** "En Çok İzlenenler" yazısı büyük ve belirgin şekilde sayfanın üst kısmında yer alır.
- **Film/Dizi Listesi:** En çok izlenen filmler ve diziler listeler halinde görüntülenir. Her bir film veya dizi hakkında şu bilgiler yer alır:
  - **Afiş Görseli:** Filmin veya dizinin afişi.
  - **Adı:** Filmin veya dizinin adı. Örneğin, "3391 Kilometre."
  - **Yayınlanma Tarihi:** Filmin veya dizinin yayılanma tarihi. Örneğin, "12.6.2018."
  - **Puan:** Kullanıcıların verdiği puan. Örneğin, "4.0."
- **Değerlendirme Yapma:** Kullanıcılar, izledikleri filmlere ve dizilere puan verebilirler. Puanlama 1 ile 5 arasında yapılır.



## İletişim Sayfası

### Eğer Kullanıcı Giriş Yapmamışsa:

- Kullanıcı İletişim Sayfasına gitmek istediğiinde, "Mesaj göndermek için giriş yapmalısınız" şeklinde bir uyarı mesajı alır.
- "Giriş yapmalısınız" kelimesine tıkladığında, kullanıcı Giriş Sayfasına yönlendirilir.

### Eğer Kullanıcı Giriş Yapmışsa:

- Kullanıcı giriş yaptıysa, İletişim Sayfası aşağıdaki gibi görünür ve kullanıcı gerekli bilgileri doldurabilir:
  - Ad Soyad:** Kullanıcının adını girmesi için alan.
  - E-posta:** Kullanıcının e-posta adresini girmesi için alan.
  - Mesaj:** Kullanıcının mesajını yazması için alan.
- Formu Doldurma:** Kullanıcı formdaki tüm bilgileri doldurup mesaj gönderebilir.
  - Boş alan bırakıldığında "Lütfen tüm alanları doldurunuz" uyarısı alır.
- Mesaj Gönderme:** Kullanıcı formu doldurup mesajı gönderdikten sonra mesaj admin tarafına iletilecektir.

The screenshot shows a web browser window with the following details:

- Title Bar:** "Sinema\_Dünyası" - Sinema\_Dünyası
- Address Bar:** https://localhost:7227/Contact/ContactForm
- Header:** "Sinema dünyamiza hoş geldiniz değerli kullanıcılarımız!" and a "Giriş Yap" button.
- Navigation:** "Anasayfa", "En Çok İzlenenler", "İletişim", and an email link "amirelahmed99@gmail.com".
- Main Content:** "Bize Ulaşın" (Reach Us) heading.
- Form Fields:** "TELEFON :" field containing "1 (234) 567 891", "İLGİLİ YASA :" section with legal text, and "HUKUKSAL :" section with legal text.
- Modal Form:** A modal window titled "Formu doldurun" with fields for "Adsoyad" (Last Name), "E-posta" (Email), and "Mesaj" (Message). The message field contains "Mesajınız".
- System Message:** "Windows'u Etkinleştir" and "Windows'u etkinleştirmek için Ayarlar'a gidin."

## Alışveriş Sepetiniz Sayfası

### Eğer kullanıcı giriş yapmışsa:

- Sayfanın başında "Alışveriş Sepetiniz" başlığı yer alır.
- **Film Listesi ve İşlemler:** Kullanıcı sepetinde bulunan filmleri, fiyatları ile birlikte görebilir ve her bir film için "Sil" butonu ile filmi sepetinden çıkarabilir.
- **Toplam Tutar:** Kullanıcı sepetindeki tüm filmlerin toplam fiyatını görebilir. Örneğin, toplam ₺639,00.
- **Daha Fazla Ürün Ekle Butonu:** Kullanıcı bu butona tıklayarak sepetine daha fazla film ekleyebilir.
- **Ödeme Yap Butonu:** Kullanıcı bu butona tıklayarak ödeme yapabilir.
  - Boş alan bırakırsa "Lütfen tüm alanları doldurunuz" uyarısı alır.

### Eğer kullanıcı giriş yapmamışsa:

- Film fiyatına tıkladığında "film satın almak için giriş yapmalısınız" şeklinde bir uyarı alır.
- "Giriş yapmalısınız" kelimesine tıkladığında kullanıcı giriş sayfasına yönlendirilir.

### Ödeme Yapma

- **Ödeme Yap Butonu:** Kullanıcı ödeme yap butonuna tıkladığında, ödeme ekranına yönlendirilir.

### Daha Fazla Ürün Ekleme

- **Daha Fazla Ürün Ekle Butonu:** Kullanıcı bu butona tıklayarak sepetine birden fazla film ekleyebilir.

Film	Fiyat	İşlemler
Aile Arasında	130	Sil
Şans Eseri	210	Sil
99. Hücre Bloğunda	299	Sil

Toplam: ₺639,00

Daha fazla ürün ekle    Ödeme Yap

- **Film Silme :** Kullanıcı her bir film için "Sil" butonuna tıklayarak filmleri sepetinden çıkarabilir.

The screenshot shows a web browser window for 'Sinema\_Dünyası' at the URL <https://localhost:7227/Cart>. The page displays a shopping cart titled 'Alışveriş Sepetiniz' containing three movies: 'Aile Arasında' (Price: 130), 'Şans Eseri' (Price: 210), and '99. Hücre Bloğunda' (Price: 299). Each item has a 'Sil' (Delete) button in the 'İşlemler' (Actions) column. A modal dialog box is overlaid on the page, asking 'Bu ürünü sepetinizden silmek istediğinizden emin misiniz?' (Are you sure you want to delete this item from your cart?). Below the dialog are 'İptal' (Cancel) and 'Sil' (Delete) buttons. At the bottom of the page, there are buttons for 'Daha fazla ürün ekle' (Add more products) and 'Ödeme Yap' (Make payment). The footer includes a location pin icon, the text 'ISPARTA İSUBÜ', and a note about activating Windows.

Film	Fiyat	İşlemler
Aile Arasında	130	Sil
Şans Eseri	210	Sil
99. Hücre Bloğunda	299	Sil

**Toplam: ₺639,00**

Daha fazla ürün ekle    Ödeme Yap

Giriş Yap

Bu ürünü sepetinizden silmek istediğinizden emin misiniz?

İptal Sil

Windows'u Etkinleştir

Windows'u etkinleştirmek için Ayarlar'a gidin.

## Ödeme Yap Sayfası

Eğer kullanıcı “Ödeme Yap” butonuna tıklandıktan sonra karşısına çıkan sayfa:

- **Kart Numarası:** 16 haneli, yalnızca rakamdan oluşan kart numarası girişi.
- **Son Kullanma Tarihi:** Ay/Yıl formatında (MM/YY) iki hane/iki hane olacak şekilde tarih girişi.
- **CVV:** 3 haneli güvenlik kodu girişi.

## Uyarı Mesajları

- **Boş Alan Uyarısı:** Kullanıcı herhangi bir alanı boş bıraktığında “Lütfen tüm alanları doldurunuz” uyarısı alır.
- **Geçersiz Kart Numarası Uyarısı:** Kullanıcı 16 haneli olmayan veya rakam dışında karakter içeren bir kart numarası girdiğinde “Geçersiz kart numarası” uyarısı alır.
- **Geçersiz Tarih Uyarısı:** Kullanıcı tarih formatına uygun olmayan bir tarih girdiğinde “Geçersiz tarih” uyarısı alır.
- **Geçersiz CVC Uyarısı:** Kullanıcı 3 haneli olmayan bir CVC kodu girdiğinde “Geçersiz CVC” uyarısı alır.

## Onaylama

- Kullanıcı tüm bilgileri doğru ve eksiksiz şekilde girdikten sonra “Ödeme Yap” butonuna tıklayarak ödeme işlemini tamamlar.

The screenshot shows a payment page titled "Ödeme Sayfası - Sinema\_Dünyası". At the top, there's a green header bar with the text "\*Sinema dünyamıza hoş geldiniz değerli kullanıcılarımız !\*" and a "Giriş Yap" button. Below the header, the main content area has a dark background. It features a form with three input fields: "Kart Numarası" containing "2444454454545454", "Son Kullanma Tarihi" containing "09/28", and "CVV" containing "444". Below the form is a large green "Ödeme Yap" button. The URL in the browser address bar is "https://localhost:7227/Checkout".

## Çıkış Yapma

- Kullanıcı hesabı üzerine geldiğinde “Çıkış yap” seçeneği gözükecektir.
- Kullanıcı “Çıkış yap” butonuna tıkladığında, çıkış işlemi gerçekleşecektir ve kullanıcı ana sayfaya yönlendirilecektir.

The screenshot shows a web browser window with the following details:

- Title Bar:** "Sinema\_Dünyası" - Sinema\_Dünyası
- Address Bar:** https://localhost:7227/Contact/ContactForm
- Header:**
  - Sinama DÜNYASI logo
  - Anasayfa (Home)
  - En Çok İzlenenler (Most Viewed)
  - İletişim (Contact)
  - Giriş Yap (Log In) button
  - E-mail: amirelahmed99@gmail.com
  - Çıkış yap (Logout) button
- Main Content:**
  - "Sinema dünyamıza hoş geldiniz değerli kullanıcılarımız!" message
  - Bize Ulaşın** (Get in touch with us) heading
  - TELEFON :** 1 (234) 567 891, 1 (234) 567 891
  - İLGİLİ YASA :** MADDE 5 (1) Yer sağlayıcı, yer sağladığı içeriği kontrol etmek veya hukuka aykırı bir faaliyetin söz konusu olup olmadığını araştırmakla yükümlü değildir.
  - HUKUKSAL :** Yer sağlayıcı, yer sağladığı hukuka aykırı içerikten, ceza sorumluluğu ile ilgili hükümler saklı kalmak kaydıyla, bu Kanunun 8inci ve 9uncu maddelerine göre haberdar edilmesi halinde ve teknik olarak imkân bulunduğu ölçüde hukuka uygun içeriği yayından kaldırır.
- Form Area:** "Formu doldurun" (Fill out the form) title. It contains three input fields:
  - Adsoyad (Last Name): "Adınızı giriniz" placeholder
  - E-posta (Email): "E-posta adresi giriniz" placeholder
  - Mesaj (Message): "Mesajınız" placeholder
- Bottom Right:** A small note: "Windows'u Etkinleştir" (Enable Windows) with the sub-instruction "Windows'u etkinleştirmek için Ayarlar'a gidin."
- Page Bottom:** https://localhost:7227/Contact/ContactForm#

## 2- Admin sayfaları .

### Admin Giriş Sayfası

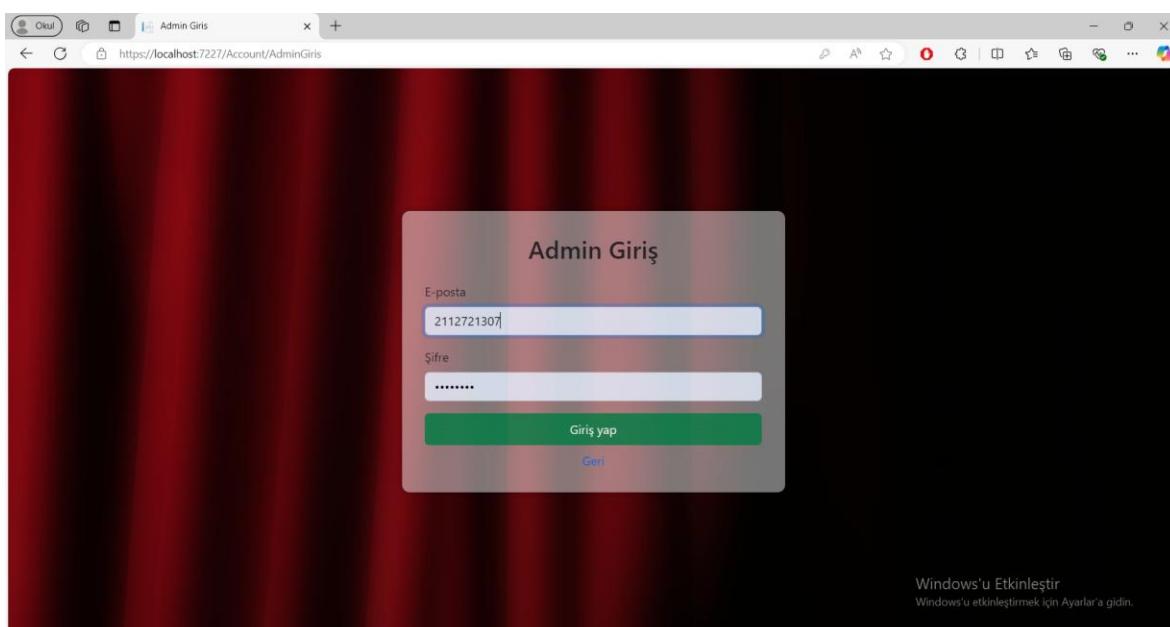
- **TC Kimlik Numarası:** Admin kullanıcılar e-posta yerine TC kimlik numaralarını gireceklerdir. Bu alan yalnızca rakam içermelidir.
- **Şifre:** Admin kullanıcıların veritabanında kayıtlı olan şifresi.
- **Giriş Yap:** Kullanıcı bilgilerini girdikten sonra bu butona basarak sisteme giriş yapabilir.
- **Geri:** Kullanıcı anasayfasına geri dönmek için bu kelimeye basabilir.

### Uyarı Mesajları

- **Boş Alan Uyarısı:** Kullanıcı herhangi bir alanı boş bıraktığında “Lütfen tüm alanları doldurunuz” uyarısı alır.
- **Geçersiz TC Kimlik Numarası:** Kullanıcı geçerli formatta olmayan bir TC kimlik numarası girdiğinde “Geçersiz TC kimlik numarası” uyarısı alır.
- **Yanlış Şifre:** Kullanıcı doğru TC kimlik numarasını girdiği halde yanlış şifre yazdığında “Şifreniz yanlış” uyarısı alır.

### Başarılı Giriş:

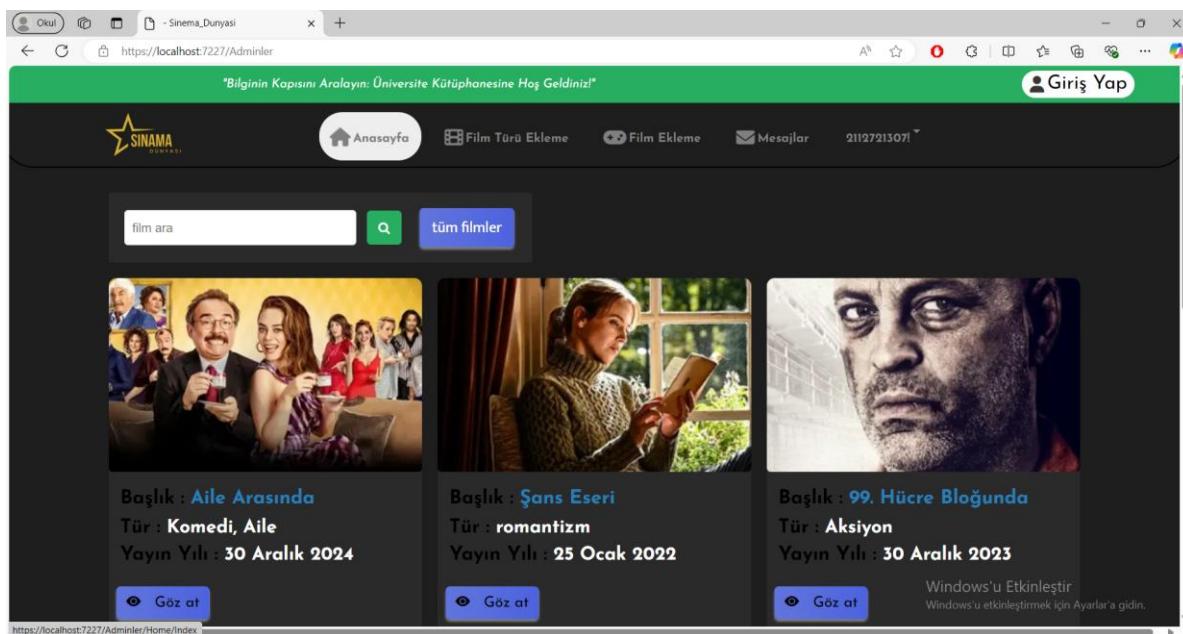
- Admin doğru bilgilerle giriş yaptığında, admin temasına yönlendirilecektir.



## Admin Ana Sayfa

Admin kullanıcı başarılı bir şekilde giriş yaptıktan sonra karşısına çıkan ana sayfa şu özellikleri içerir:

- **Navigasyon Menüsü:**
  - **Anasayfa:** Admin ana sayfasına dönüş.
  - **Film Türü Ekleme:** Yeni film türleri eklemek için bir sayfa.
  - **Film Ekleme:** Yeni filmler eklemek için bir sayfa.
  - **Mesajlar:** Kullanıcılar tarafından gönderilen mesajları görüntülemek için bir sayfa.
- **Arama Çubuğu:** Tüm filmleri arayarak bulmak için bir alan.
- **Tüm Filmler Butonu:** Admin, tüm filmleri listelemek için bu butona tıklayabilir.

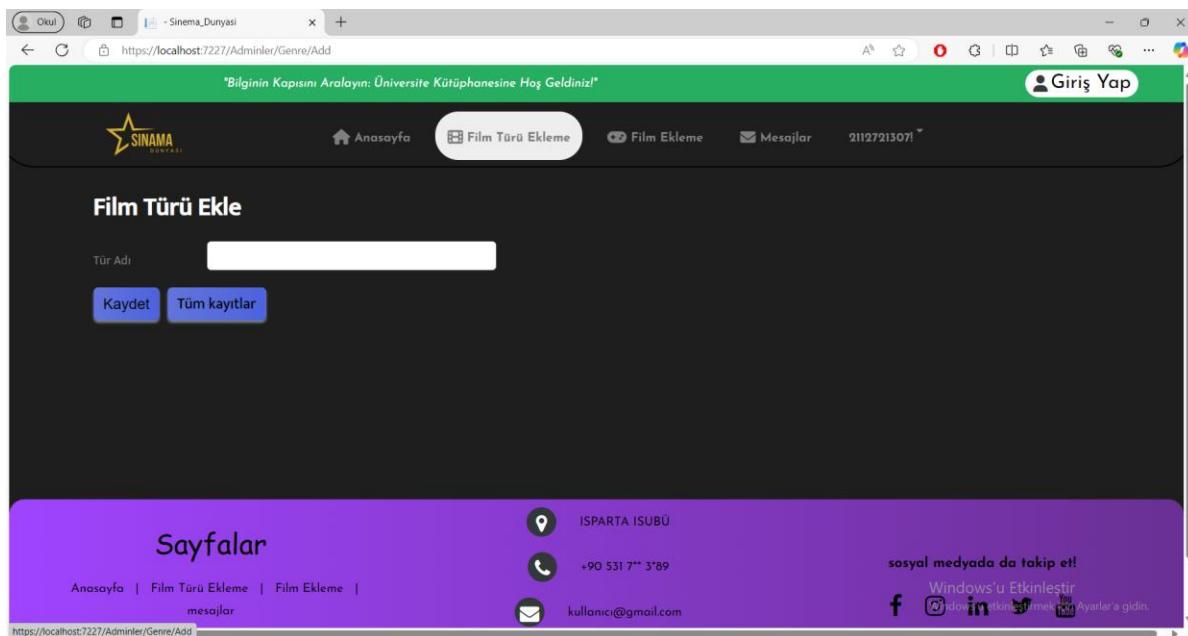


- **Film Türü Ekleme Formu:**

- **Tür Adı:** Admin kullanıcı, eklemek istediği film türünün adını bu alana yazacaktır.
- **Kaydet Butonu:** Admin kullanıcı, girdiği film türünü kaydetmek için bu butona tıklayacaktır.
- **Tüm Kayıtlar Butonu:** Admin kullanıcı, daha önce eklenmiş tüm film türlerini görüntülemek için bu butona tıklayabilir.

## Uyarı Mesajları

- **Boş Alan Uyarısı:** Admin kullanıcı, "Tür Adı" alanını boş bıraktığında "Lütfen tüm alanları doldurunuz" uyarısı alır.



## Tüm Kayıtlar Sayfası

Admin kullanıcı, "Tüm Kayıtlar" butonuna tıkladığında bu sayfa açılacaktır:

### Film Türleri Listesi

- **Tablo Başlıklarları:**

- **Tür Adı:** Film türlerinin adlarını içerir. Örneğin: "Romantik," "Korku," "Komedi, Aile," "Aksiyon," "Çocuk," "Aşk."
- **İşlemler:** Her tür için düzenleme ve silme işlemleri yapılabilir.

### İşlem Butonları

- **Düzenleme Butonu (Kalem İkonu):** Admin kullanıcı, bu butona tıklayarak film türü üzerinde değişiklik yapabilir.
- **Silme Butonu (Çöp Kutusu İkonu):** Admin kullanıcı, bu butona tıklayarak film türünü silebilir.

### Uyarı Mesajları

- **Silme İşlemi Uyarısı:** Kullanıcı, silme butonuna tıkladığında "Emin misiniz?" şeklinde bir uyarı mesajı alır. Eğer silmek istedğini onaylarsa, ilgili film türü silinir.
- **Düzenleme İşlemi:** Admin kullanıcı, düzenleme butonuna tıkladığında, film türü bilgilerini düzenleyebileceğii bir form açılır.

Ad	Aksiyon
Romance	
horror	
Komedi, Aile	
romantizm	
Aksiyon	
Çocuk	
Aşk	

## Film Ekleme Sayfası

Admin kullanıcı, yeni bir film eklemek istediğiinde bu sayfa açılacaktır:

### Film Ekleme Formu:

- Başlık:** Filmin adı girilir.
- Fotoğraf:** Filmin afiş fotoğrafı yüklenir.
- Tür:** Filmin türü seçilir.
- Yayın Yılı:** Filmin yayınlanma yılı girilir.
- Kadro:** Filmin kadrosu yazılır.
- Yönetmen:** Filmin yönetmeni yazılır.
- Fiyat:** Filmin fiyatı girilir.

### İşlem Butonları:

- Kaydet Butonu:** Girilen bilgileri kaydetmek için.
- Tüm Kayıtlar Butonu:** Daha önce eklenmiş tüm filmleri görüntülemek için.

### Uyarı Mesajları:

- Boş Alan Uyarısı:** Admin kullanıcı, herhangi bir alanı boş bıraktığında “Lütfen tüm alanları doldurunuz” uyarısı alır.
- Geçersiz Veri Uyarısı:** Admin kullanıcı, geçersiz veri girdiğinde ilgili alana göre uygun uyarı alır (örneğin, fiyat alanına rakam dışı karakter girdiğinde “Geçersiz fiyat” uyarısı).

The screenshot shows a web application interface for adding a movie. At the top, there's a navigation bar with links for 'Anasayfa', 'Film Türü Ekleme', 'Film Ekleme' (which is the active tab), 'Mesajlar', and a date '2022/1307'. The main content area has a dark background with white text. It contains input fields for 'Başlık' (Title), 'Fotoğraf' (Photo) with a 'Dosya Seç' (Select file) button, 'Tür' (Genre) with a dropdown menu showing 'select--', 'Romance', 'horror', 'Komedî', 'Aile', 'Yayın Yılı' (Release Year), 'Kadro' (Cast), 'Yönetmen' (Director), and 'Fiyat' (Price). At the bottom, there are two buttons: 'Kaydet' (Save) in blue and 'Tüm kayıtlar' (All records) in grey. The browser status bar at the bottom shows the URL 'https://localhost:7227/Adminler/Movie/Add'.

## Tüm Kayıtlar Sayfası

### Film Listesi

Sayfada admin kullanıcının eklediği tüm filmler listelenir. Her film için aşağıdaki bilgiler yer alır:

### İşlem Butonları

- Düzenle (Kalem İkonu):** Admin kullanıcı, bu butona tıklayarak film bilgilerini düzenleyebilir.
- Sil (Çöp Kutusu İkonu):** Admin kullanıcı, bu butona tıklayarak filmi silebilir.

### Uyarı Mesajları

- Silme İşlemi Uyarısı:** Kullanıcı silme butonuna tıkladığında “Emin misiniz?” şeklinde bir uyarı mesajı alır. Eğer silmek istediğini onaylarsa, ilgili film listeden silinir.
- Düzenleme İşlemi:** Admin kullanıcı, düzenleme butonuna tıkladığında film bilgilerini düzenleyebileceği bir form açılır.

Başlık	Türü	Yayınınylı	Fotoğraf	Kadro	Fiyat	Yönetmen	Aksiyon
Aile Arasında	Komedî, Aile	30 Aralık 2024		Engin Günaydin , Demet Evgar , Erdal Özyaçilar , Gulse Birsel	130	Ozan Açıktan	
Şans Eseri	romantizm	25 Ocak 2022		Lou de Laage , Niels Schneider , Melvil Poupaud ,	210	Woody Allen	
99. Hücre Bloğunda	Aksiyon	30 Aralık 2023		Jennifer Carpenter , Marc Blucas Don Johnson ,	299	S. Craig Zahler	
Ters Yüz 2	Çocuk	22 Aralık 2019		Amy Poehler , Maya Hawke , Kensington Tallman	199	Kelsey Mann	
39 Derecede Aşk	Aşk	2 Aralık 2024		Ayça Ayşin Turan , Furkan Andic , Cem Davran	312	Tunç Sahin	 Windows'u etkinleştirmek için Ayarlar'a gidin.

## Film Düzenleme Sayfası

### Film Düzenleme Formu:

- Başlık:** Filmin mevcut adı otomatik olarak doldurulmuş şekilde gelir.
- Fotoğraf:** Mevcut fotoğraf görseli ve yeni fotoğraf yükleme seçeneği.
- Tür:** Filmin mevcut türü otomatik olarak seçili gelir.
- Yayın Yılı:** Filmin mevcut yayınlanma yılı.
- Kadro:** Filmin mevcut kadrosu.
- Yönetmen:** Filmin mevcut yönetmeni.
- Fiyat:** Filmin mevcut fiyatı.

### İşlem Butonları:

- Kaydet Butonu:** Admin kullanıcı, yapılan değişiklikleri kaydetmek için bu butona tıklayacaktır.
- Tüm Kayıtlar Butonu:** Admin kullanıcı, tüm filmleri listelemek için bu butona tıklayabilir.

### Uyarı Mesajları:

- Boş Alan Uyarısı:** Admin kullanıcı, herhangi bir alanı boş bıraktığında “Lütfen tüm alanları doldurunuz” uyarısı alır.
- Geçersiz Veri Uyarısı:** Admin kullanıcı, geçersiz veri girdiğinde ilgili alana göre uygun uyarı alır (örneğin, fiyat alanına rakam dışı karakter girdiğinde “Geçersiz fiyat” uyarısı).

The screenshot shows a web browser window with the URL <https://localhost:7227/Adminler/Movie/Edit?id=1>. The page title is "Bilginin Kapısını Aralayın: Üniversite Kütüphanesine Hoş Geldiniz!". The header includes a logo for "SİNAMA", navigation links for "Anasayfa", "Film Türü Ekleme", "Film Ekleme", "Mesajlar", and a user ID "2112721307!". A "Giriş Yap" button is also present. The main content area is titled "Filmi Güncelle" and contains the following form fields:

Başlık	Aile Arasında
Fotoğraf	Dosya Seç <small>Seçilen dosya yok</small>
Türü	--select-- Romance horror Komedi, Aile
Yayın Yılı	30 Aralık 2024
Kadro	Engin Günaydın , Demet Evgar , Erdal Özyaçıklar , G
Yönetmen	Ozan Açıktan
Fiyat	130

Below the form, there are two buttons: "Kaydet" and "Tüm kayıtlar". To the right of the form, a copyright notice reads: "Windows'u Etkinleştir" and "Windows'u etkinleştirmek için Ayarlar'a gidin."

## Kullanıcı Tarafından Gelen Mesajlar

### Mesajlar Tablosu

Sayfada admin kullanıcının gelen mesajları görüntüleyebilecegi bir tablo bulunur. Tabloda yer alan bilgiler:

- Ad Soyad:** Kullanıcının adı ve soyadı.
- E-posta:** Kullanıcının e-posta adresi.
- Mesaj:** Kullanıcının gönderdiği mesaj.
- Gönderilme Tarihi:** Mesajın gönderildiği tarih ve saat.

### Arama ve Sıralama

- Mesajlarda Arama:** Admin kullanıcı, arama çubuğuunu kullanarak mesajlar arasında arama yapabilir.
- Sıralama:** Admin kullanıcı, mesajları belirli kriterlere göre (örneğin, tarih veya kullanıcı adı) sıralayabilir.

### Sayfalama

- Gösterim Seçenekleri:** Admin kullanıcı, mesajların sayfa başına kaç adet gösterileceğini seçebilir (örneğin, 10'ar veya 20'ser).

Ad Soyad	E-posta	Mesaj	Genderilme Tarihi
Yasemin Korkmaz	yasemin.korkmaz@hotmail.com	"Gökyüzü Macerası" filmi HD kalitesinde mevcut mu?	2024-12-30 12:19
Mehmet Arslan	mehmet.arslan@yahoo.com	Satin aldığım film dosyasında bir sorun yaşadım. Yardımcı olabilir misiniz?	2024-12-30 12:13

Show 10 entries

Search:

Anasayfa | Film Türü Ekleme | Film Ekleme | Mesajlar | 202721307! | Giriş Yap

Gelen Mesajlar

Showing 1 to 2 of 2 entries (filtered from 22 total entries)

Previous | 1 | Next

Windows'u Etkinleştir  
Windows'u etkinleştirmek için Ayarlar'a gidin.

https://localhost:7227/Adminler/Contact/ContactList

## Admin Çıkış Yapma

- TC Kimlik Üzerine Gelme:** Admin kullanıcı, sayfanın sağ üst köşesindeki TC kimlik numarası üzerinde geldiğinde "Çıkış yap" seçeneği görünür.
- Çıkış Yap:** Admin kullanıcı, "Çıkış yap" butonuna tıkladığında çıkış işlemi gerçekleşir.
- Yönlendirme:** Çıkış yaptıktan sonra admin kullanıcısı, kullanıcı ana sayfasına yönlendirilecektir.

The screenshot shows a web browser window for the 'Cinema\_Dunyasi' application. The URL is <https://localhost:7227/Adminler/Contact/ContactList>. The page title is "Bilginin Kapısını Aralayın: Üniversite Kütüphanesine Hoş Geldiniz!". At the top right, there is a 'Giriş Yap' button. Below it, a message box displays the phone number '2112721307'. The main content area is titled 'Gelen Mesajlar' (Incoming Messages). It features a table with two entries:

Ad Soyad	E-posta	Mesaj	Gönderilme Tarihi
Yasemin Korkmaz	yasemin.korkmaz@hotmail.com	"Gökyüzü Macerası" filmi HD kalitesinde mevcut mu?	2024-12-30 12:19
Mehmet Arslan	mehmet.arslan@yahoo.com	Satın aldığım film dosyasında bir sorun yaşadım. Yardımcı olabilir misiniz?	2024-12-30 12:13

At the bottom of the table, it says "Showing 1 to 2 of 2 entries (filtered from 22 total entries)". There are navigation buttons for 'Previous', 'Next', and a page number '1'. A watermark at the bottom right says "Windows'u Etkinleştir" and "Windows'u etkinleştirmek için Ayarlar'a gidin". The browser address bar shows the full URL again.

## **6. Kullanılan Teknolojiler**

Projemizde, aşağıdaki teknolojiler ve kütüphaneler kullanılmıştır:

1. Proje raporu hazırlanmış mı? (Proje Raporu hazırlanmadıysa proje kabul edilmez.) .

Evet proje raporu hazır.

2.. .Net 8 veya 9 ile mi geliştirilmiş, .Net 9 ile geliştirilmiştir.

3. Yayın Durumu: Site yayınlanmıştır.

4. Uygulama İşleyiği: Uygulama sorunsuz bir şekilde çalışmaktadır.

5. Admin Değişiklikleri: Admin tarafından yapılan değişiklikler public siteye yansımaktadır.

6.MVC Tasarım Kalıbı: Proje, MVC (Model-View-Controller) tasarım kalıbına uygun olarak geliştirilmiştir.

7. Özelliklerin Geliştirilmesi: Projede yapılması istenen tüm özellikler geliştirilmiştir.

8. Html Helper: Projede Html Helper kullanılmıştır.

9. Custom Html Helper: Projede Custom Html Helper kullanılmıştır.

10. Tag Helper: Projede Tag Helper kullanılmıştır.

11. Custom Tag Helper: Projede Custom Tag Helper kullanılmıştır.

12. Server Side Validation: Server side validation kullanılmıştır.

13. Client Side Validation: Client side validation kullanılmıştır.

14. Dependency Injection: Projede Dependency Injection kullanılmıştır.

15. Server Side Paging: Server side paging kullanılmıştır.

16. Jquery Kütüphanesi: Projede Jquery kütüphanesi kullanılmıştır.

17. Jquery DataTables Kütüphanesi: Projede Jquery DataTables kütüphanesi kullanılmıştır.

18. SweetAlert Kütüphanesi: Projede SweetAlert kütüphanesi kullanılmıştır.

19. Admin için Tema: Admin için tema uygulanmıştır.

20. Area Oluşturulması: Projede area oluşturulmuştur.

21. Authentication: Authentication (kimlik doğrulama) uygulanmıştır.

22. Authorization: Authorization (yetkilendirme) uygulanmıştır.

23. Veritabanı Bağlantısı: Proje, MS Sql Server'a bağlanmıştır ve Entity Framework Code First yaklaşımıyla geliştirilmiştir.

24. ViewComponent Kullanımı: Projede ViewComponent geliştirilmiş ve kullanılmıştır.

Bu teknolojiler, projemizin işlevsellliğini ve performansını artırmak için kullanılmıştır. Eğer başka bir konuda yardıma ihtiyacınız olursa, sormaktan çekinmeyin!

- Projemizde kullandığımız crud işlemleri (GET, POST)

### GET İsteği

- [HttpGet] özniteliği (attribute) ile Add metodunun HTTP GET isteklerini işlemesi gerekiği belirtiliyor.
- Add metodu, GET isteği yapıldığında bir görünüm (view) döndürüyor.

### POST İsteği

- [HttpPost] özniteliği ile Add metodunun HTTP POST isteklerini işlemesi gerekiği belirtiliyor.
- Add metodu, bir Genre modelini parametre olarak alır ve ModelState.IsValid ile doğrulama yapar.
- Model durumu geçerli ise, \_genreService.Add metodunu çağırarak türü ekler.
- Sonuca bağlı olarak TempData'da bir mesaj ayarlar ve Add işlemine yönlendirir veya modeli ile birlikte görünümü döndürür.

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `GenreController.cs` file. The code defines a controller for managing genres. It includes a constructor that injects an `IGenreService` dependency. The `Add` action method handles both GET and POST requests. For a GET request, it returns a view. For a POST request, it checks if the model state is valid. If valid, it adds the genre to the service and redirects to the `Add` action. If invalid, it returns the view with the model. The `Genre` view is located in the `Views\Genre` folder. On the right, the Solution Explorer shows the project structure for "Sinema\_Dunyasi". It includes an `Areas\Administrer` folder containing controllers like `ContactController.cs`, `GenreController.cs`, `HomeController.cs`, and `MovieController.cs`. It also contains `Index.cshtml` and `MovieDetail.cshtml` views under `Genre`. Other areas like `Account`, `Cart`, and `Checkout` are also present. The `Properties` and `wwwroot` folders are also visible.

```
Index.cshtml
GenreController.cs
CheckoutController.cs
Layout.cshtml
Index.cshtml
ContactList.cshtml
MovieDetail.cshtml
Most_Viewed.cshtml

Solution Explorer
Solution 'Sinema_Dunyasi' (1 of 1 project)
  -> Sinema_Dunyasi
    -> Connected Services
    -> Dependencies
    -> Properties
    -> wwwroot
    -> Areas
      -> Administer
        -> Controllers
          -> C# ContactController.cs
          -> C# GenreController.cs
          -> C# HomeController.cs
          -> C# MovieController.cs
        -> Data
        -> Models
        -> Views
          -> Contact
            -> C# ContactList.cshtml
          -> C# Genre
          -> Home
            -> C# Index.cshtml
            -> C# MovieDetail.cshtml
          -> Movie
      -> Controllers
        -> C# AccountController.cs
        -> C# CartController.cs
        -> C# CheckoutController.cs
        -> C# ContactController.cs
        -> C# HomeController.cs
      -> Data
      -> Helpers
      -> Migrations

Windows'u etkinleştirmek için Ayarlar'ı gidebilirsiniz.

Item(s) Saved
Add to Source Control
Select Repository
```

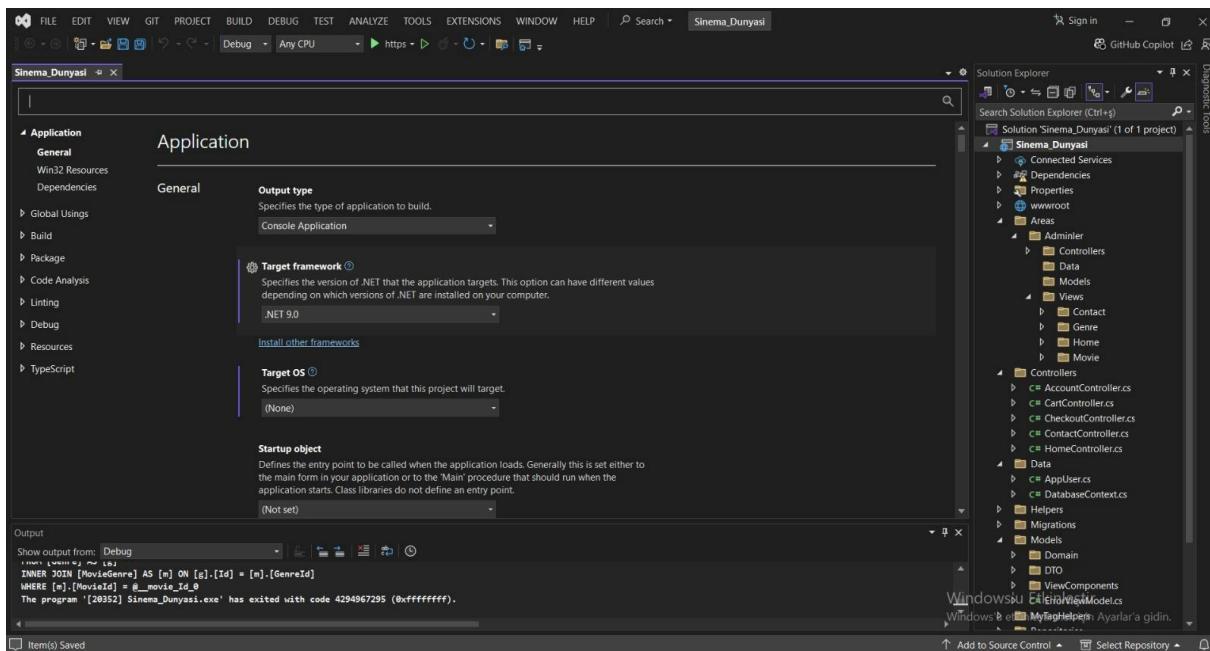
## 7. Proje Gereksinimleri ve Karşlanması

### 7.1 : Proje raporu hazırlanmış mı?

Evet proje raporu hazır.

### 7.2 : .Net 8 veya 9 ile mi geliştirilmiştir?

Aşağıdaki fotoğrafta görüldüğü gibi, target framework kısmında belirlenen projemizin .NET 9 ile sağda projemizin dosyaları yani sinema\_Dunyasi dosyaları yer almaktadır.



### 7.3 : Site yayınlanmış mı?

. Çalıştığımız proje, .NET 9 sürümünü içeriyordu. Ancak, **Somee.com** sitesi bu .NET sürümünü desteklemediği için, bunun yerine boş bir **.NET 8** projesi oluşturarak yayını gerçekleştirdim. İşlemleri şu şekilde gerçekleştirdim :

**1.** İlk olarak, yeni bir **.NET 8** projesi oluşturdum.

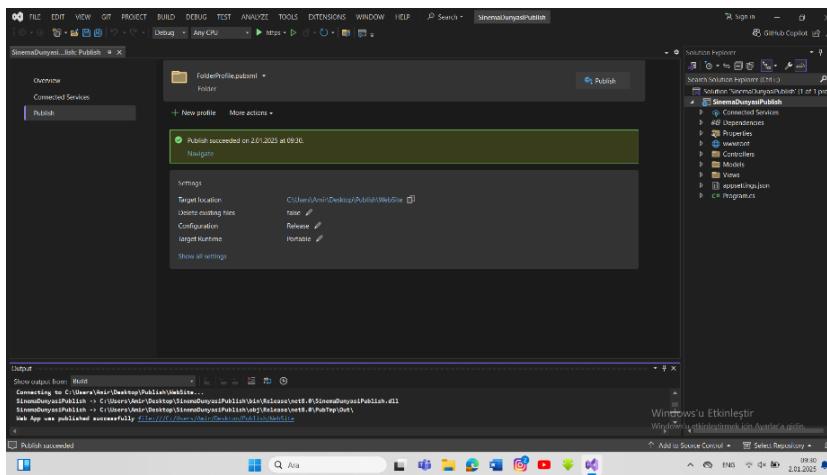
**2.** Daha sonra, projeye sağ tıklayıp **Publish** seçeneğini seçtim ve projeyi bilgisayarımı bir **folder** olarak yayınladım.

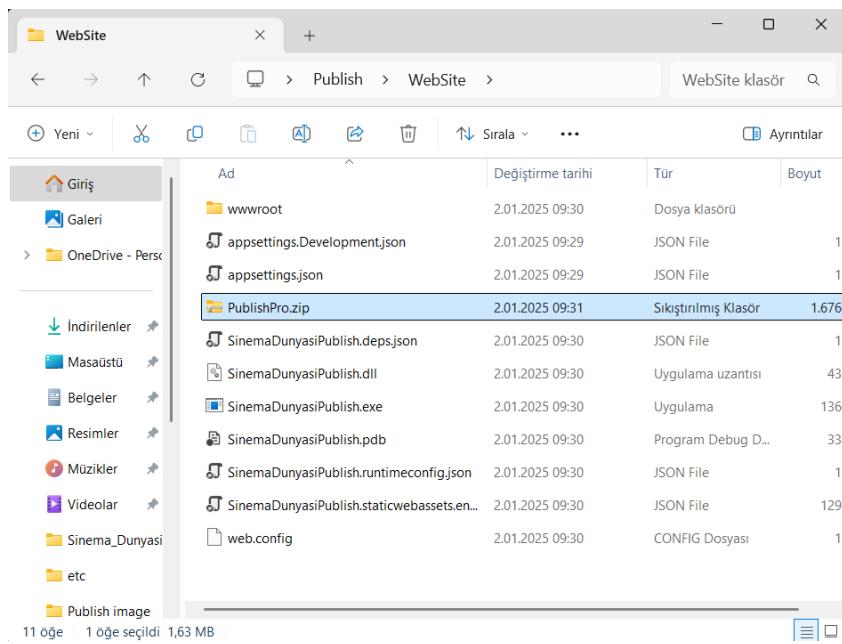
**3.** Ardından, **Somee.com** sitesine giriş yaptım ve daha önce bilgisayarımı kaydettiğim projeyi **zip** formatında sıkıştirdim.

**4. Somee.com** üzerinde **Websites** sekmesine girdim, ardından projeyi yükleyerek yayına aldım.

. Bu şekilde projem internet üzerinde çalışır hale geldi.

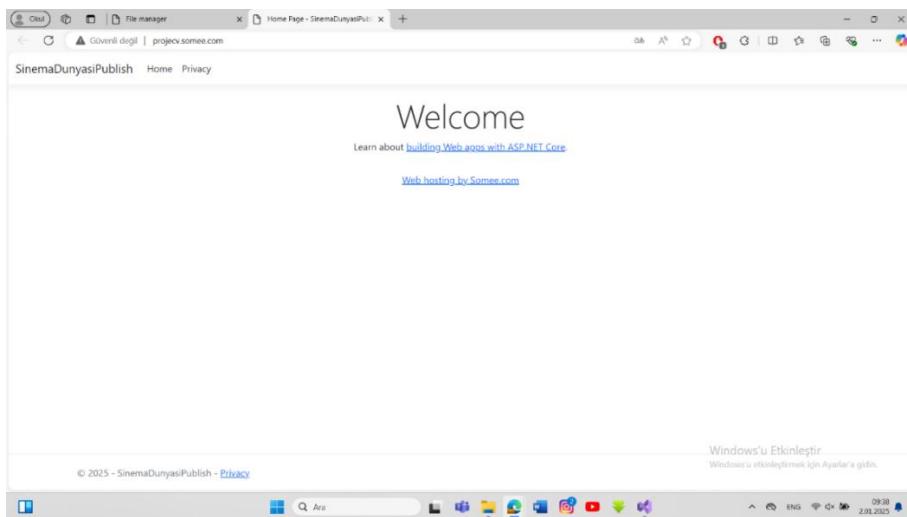
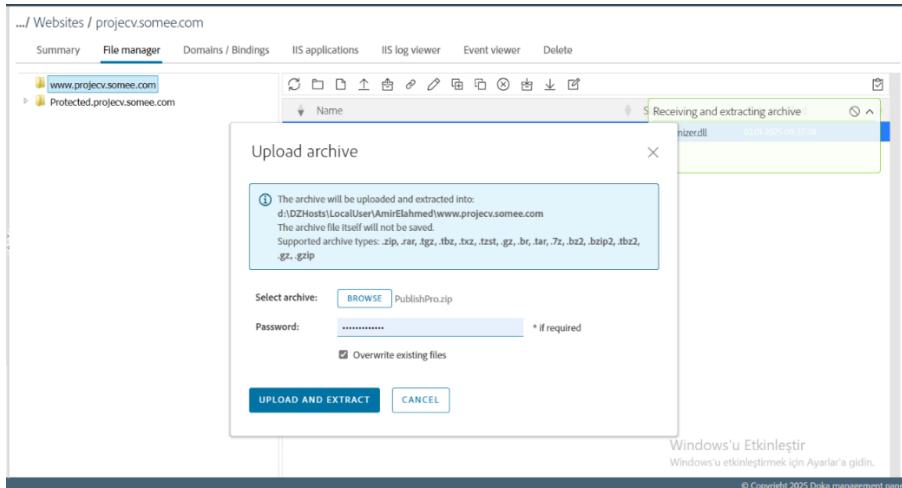
Aşağıdaki görseller, bu adımları daha ayrıntılı olarak göstermektedir.





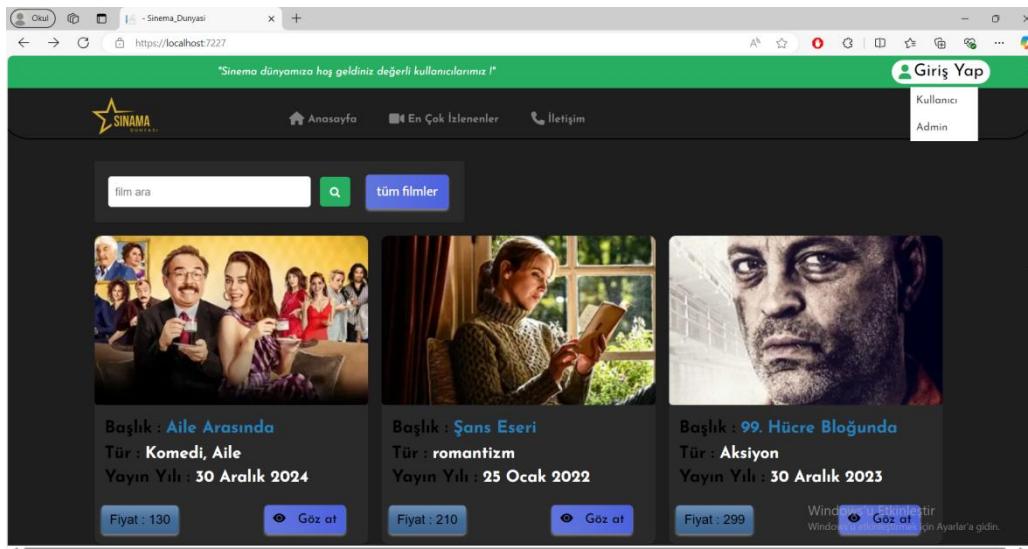
A screenshot of a web-based file manager for the domain "projectv.somee.com". The left sidebar includes "Dashboard", "Support tickets", "Profile", "Billing", "Invoices", "Payment methods", "Subscriptions", "SSL Certificates", "Virtual servers", "Websites", "Create website", and "File manager". The "File manager" tab is selected. The main area shows a table with columns "Name", "Size", and "Date modified". A message at the bottom right says "Windows'u Etkinleştir" and "Windows'u etkinleştirmek için Aşağı'a gitin".

A screenshot of a web-based management panel for the domain "projectv.somee.com". The left sidebar includes "Dashboard", "Support tickets", "Profile", "Billing", "Invoices", "Payment methods", "Subscriptions", "SSL Certificates", "Virtual servers", "Websites", "Create website", and "File manager". The "File manager" tab is selected. The main area shows a summary of the website's configuration, including "Default domain: projectv.somee.com", "Subscription: Free hosting package (SPID1324647)", "Service ID: MPID483348", "Invoices: INVID1430540", "Hosting environment: Windows Server 2022 (IIS 10.0, ASP, ASP.NET v2.0-4.8, ASP.NET Core)", and "Server local path: d:\102\Hosts\LocalUser\AmritLahmed\www.projectv.somee.com". It also displays "URLs" (http://projectv.somee.com, http://www.projectv.somee.com), "FTP" (Addresses: ftp://105.24.244.40/www.projectv.somee.com, Username: AmritLahmed, Password: \* Use the password from this control panel.), and "IIS Application pool" (ASP.NET version: .NET Core (all versions), Managed Pipeline Mode: Integrated). A note at the bottom right says "Windows'u etkinleştirmek için Aşağı'a gitin".



## 7.4 : Uygulama çalışıyor mu?

- Uygulama İşleyişi: Uygulama sorunsuz bir şekilde çalışmaktadır.



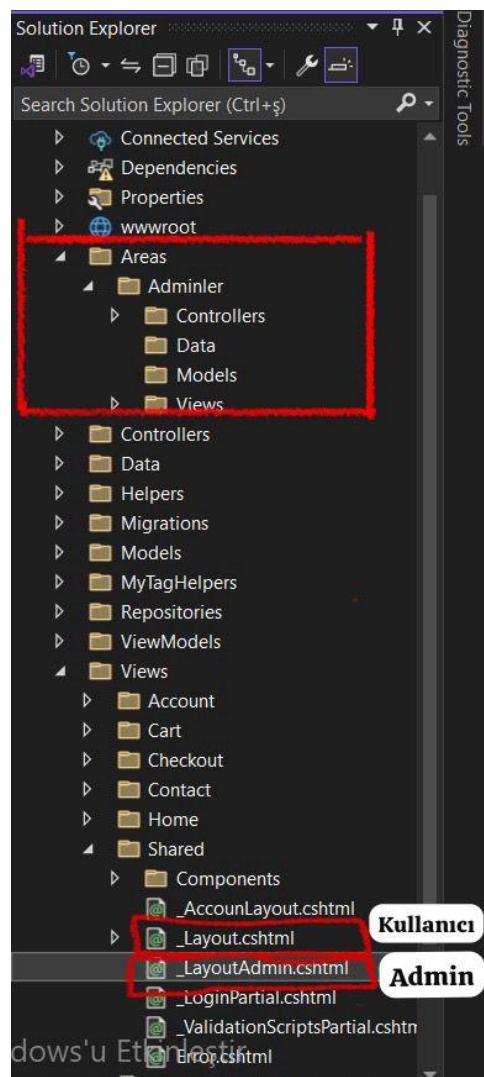
## 7.5 : Admin tarafından yapılan değişiklikler public siteye yansıyor mu?

- Admin tarafından yapılan değişiklikler public siteye yansımaktadır.

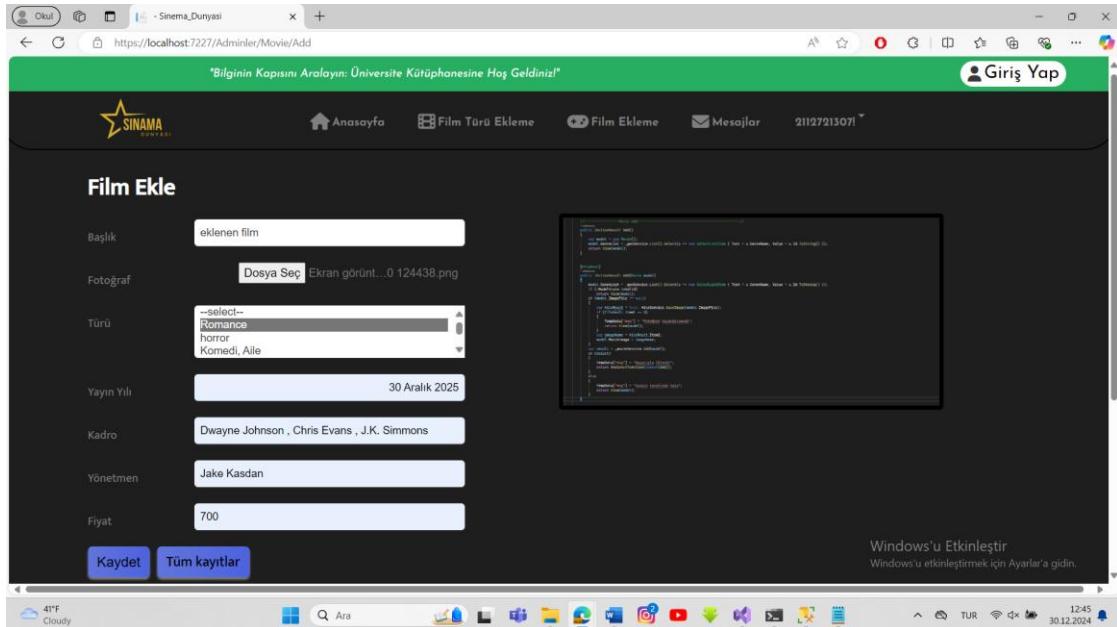
### Proje Yapısı

- Areas Klasörü:** "Areas" klasörünün içinde "Adminler" adında bir alt klasör bulunur. Bu klasörde "Controllers," "Data," "Models," ve "Views" gibi alt klasörler yer alır. Bu yapı, admin modülünün ayrı bir bölüm olarak yapılandırıldığını gösterir.
- Layouts:**
  - Kullanıcı Layout:** "Views" klasörünün içindeki "Shared" alt klasöründe "\_Layout.cshtml" dosyası bulunur. Bu dosya, kullanıcılar için kullanılan genel bir layout (düzen) dosyasıdır.
  - Admin Layout:** Aynı klasörde "\_LayoutAdmin.cshtml" dosyası bulunur. Bu dosya ise admin kullanıcıları için kullanılan özel bir layout dosyasıdır.

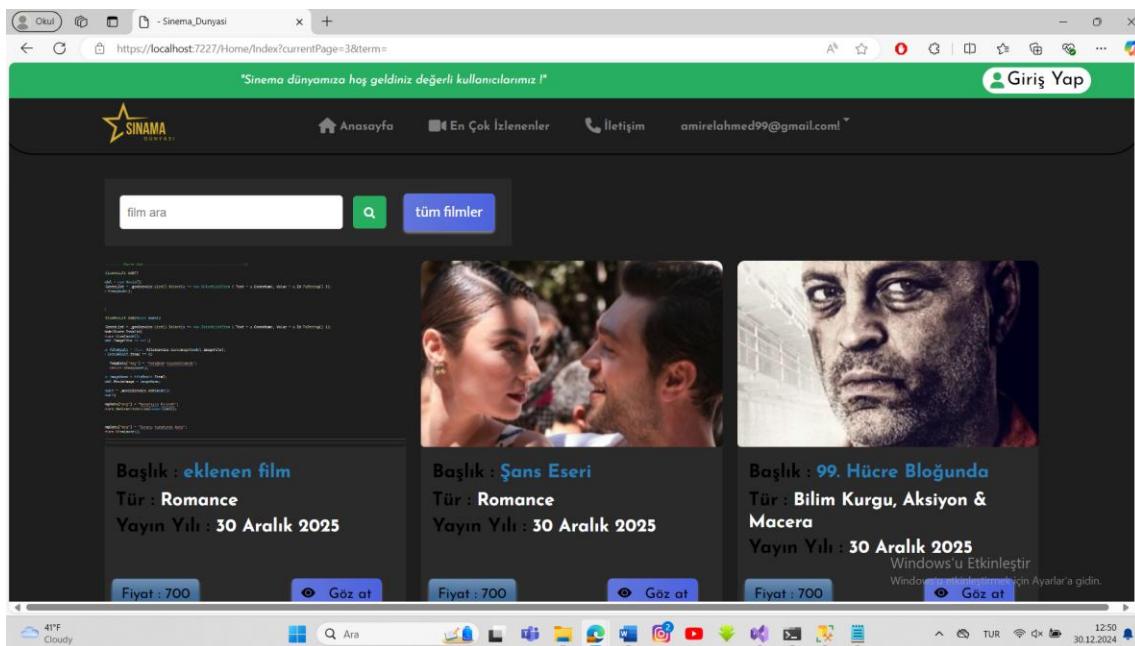
Bu yapı, projemizde kullanıcı ve admin arayüzlerini birbirinden ayırarak farklı kullanıcı deneyimleri sunar. Fotoğrafta gösterildiği gibi, proje dosyaları "sinema\_Dunyasi" klasörü altında organize edilmiştir.



- Admin tarafından film eklemesi ,fotoğrafta gözüken fieldları doldurarak kaydet butonuna tıkladıktan sonra bilgiler kullanıcının sayfalarında gözükebilecek oradan istediğiini yapabilir satın alma vb.



- Kullanıcı tarafından film gözükmesi burada gözüktü ve isterse satın alma veya göz atma gibi işlemler yapabilir

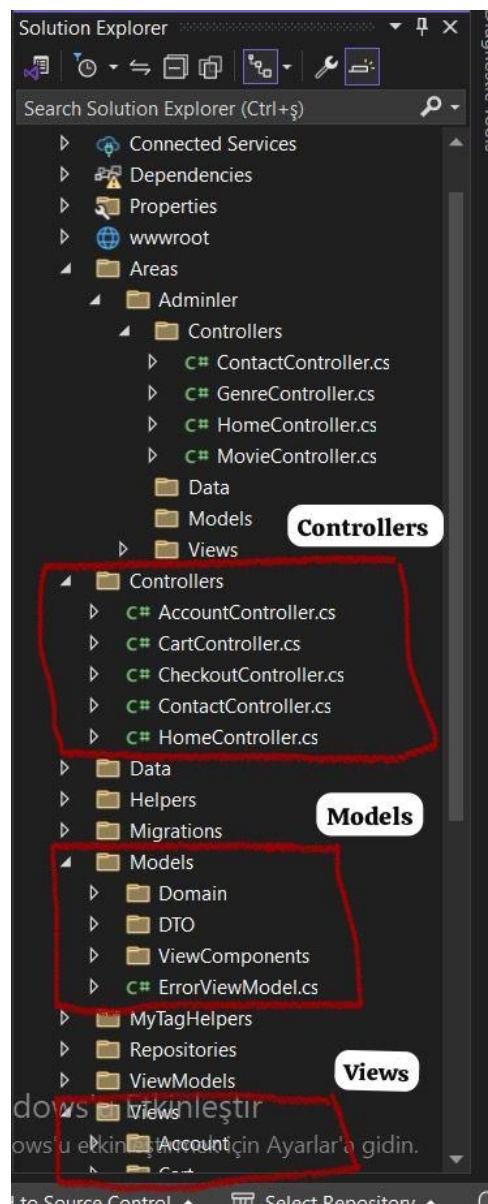


## 7.6 : Proje MVC tasarım kalıbına uygun geliştirilmiş mi?

- Controllers (Denetleyiciler): Bu klasör, uygulamanın iş mantığını ve kullanıcı etkileşimlerini yöneten denetleyici sınıflarını içerir. Fotografta, AccountController.cs, CartController.cs, CheckoutController.cs, ContactController.cs, ve HomeController.cs dosyaları bulunmaktadır.
- Models (Modeller): Bu klasör, uygulamanın veri yapısını ve iş mantığını temsil eden sınıfları içerir. Fotografta, Domain, DTO, ViewComponents alt klasörleri ve ErrorViewModel.cs dosyası bulunmaktadır.
- Views (Görünümler): Bu klasör, kullanıcıya sunulan arayüzleri ve görünümleri içerir. Fotografta, Views klasörü Areas klasörünün altında yer almaktadır.

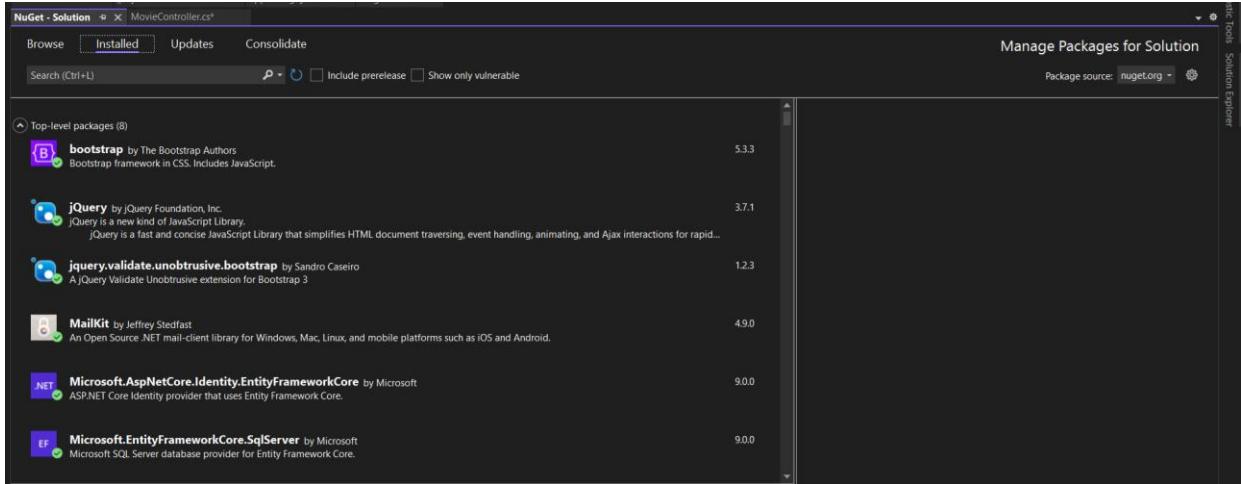
### Proje Dosyaları

Sağ tarafta bulunan `sinema_Dunyasi` klasörü altında projenizin dosyaları yer almaktadır.



## 7.7 : Projede yapılması istenen özellikler geliştirilmiş mi?

- . Özelliklerin Geliştirilmesi: Projede yapılması istenen tüm özellikler geliştirilmiştir.
- . Aşağıdaki gösterilen projemizde kullandığımız package ve kütüphanelerin bazıları



## 7.8 : Html Helper Kullanılmış mı?

. Html Helper : Microsoft, front-end konularına ilgisi veya bilgisi sınırlı olan back-end geliştiricileri için HTML yazmayı daha kolay hale getiren yollar sunmaktadır. Geleneksel HTML yazma yöntemleri yerine, ihtiyaçlarımıza göre HTML Helper'ları kullanarak daha hızlı ve etkili bir şekilde çalışabiliriz.

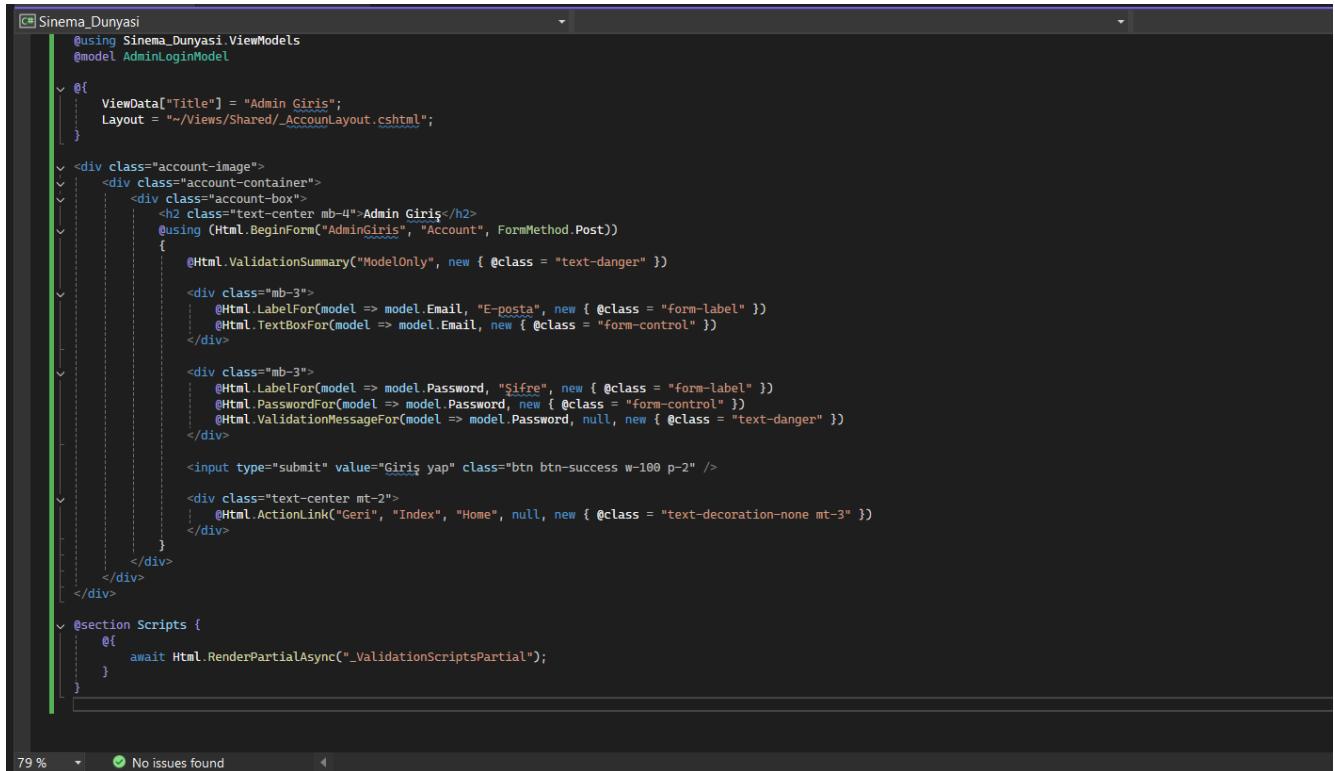
. Bu projede, ihtiyacımız olan özelliklere göre HTML Helper'ları kullanarak form elemanlarını oluşturduk. Örneğin, HTML'de metin kutusu oluşturmak için şu şekilde yazmak yerine:

- <input type="text" name="Email" class="form-control" />

. HTML Helper kullanarak şu şekilde yazdık :

```
@Html.TextBoxFor(model => model.Email, new { @class = "form-control" })
```

. Bu yaklaşım, özellikle front-end bilgi düzeyi düşük olan back-end geliştiricilerin, HTML elemanlarını kolaylıkla ve doğru bir şekilde oluşturmasına olanak sağlar.



The screenshot shows the code editor of Visual Studio with the file 'AdminLogin.cshtml' open. The code uses HTML Helpers like @Html.TextBoxFor and @Html.PasswordFor to generate form elements. The code is as follows:

```
 Sinema_Dunyasi
 @using Sinema_Dunyasi.ViewModels
 @model AdminLoginModel

 @{
     ViewData["Title"] = "Admin Giriş";
     Layout = "~/Views/Shared/_AccountLayout.cshtml";
 }

 <div class="account-image">
     <div class="account-container">
         <div class="account-box">
             <h2 class="text-center mb-4">Admin Giriş</h2>
             @using (Html.BeginForm("AdminGiris", "Account", FormMethod.Post))
             {
                 @Html.ValidationSummary("ModelOnly", new { @class = "text-danger" })

                 <div class="mb-3">
                     @Html.LabelFor(model => model.Email, "E-posta", new { @class = "form-label" })
                     @Html.TextBoxFor(model => model.Email, new { @class = "form-control" })
                 </div>

                 <div class="mb-3">
                     @Html.LabelFor(model => model.Password, "Şifre", new { @class = "form-label" })
                     @Html.PasswordFor(model => model.Password, new { @class = "form-control" })
                     @Html.ValidationMessageFor(model => model.Password, null, new { @class = "text-danger" })
                 </div>

                 <input type="submit" value="Giriş yap" class="btn btn-success w-100 p-2" />
             <div class="text-center mt-2">
                 @Html.ActionLink("Geri", "Index", "Home", null, new { @class = "text-decoration-none mt-3" })
             </div>
         </div>
     </div>
 </div>

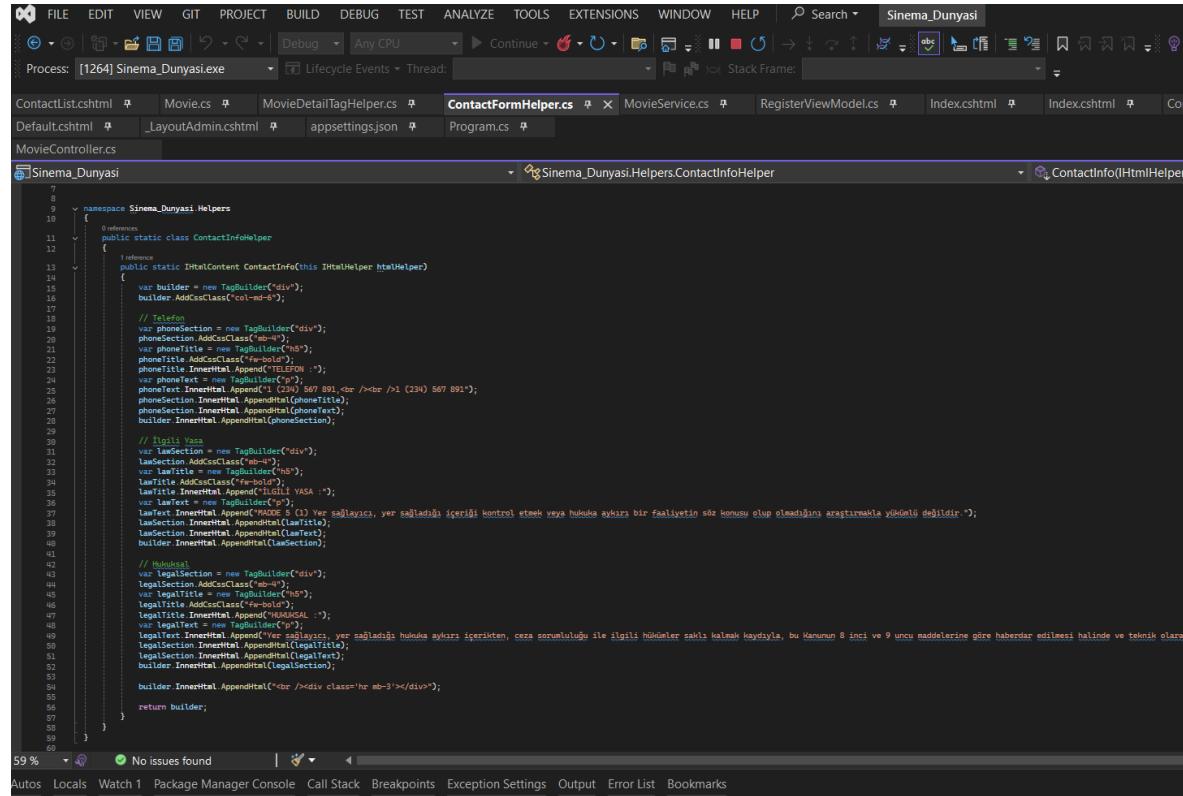
 @section Scripts {
 @{
     await Html.RenderPartialAsync("_ValidationScriptsPartial");
 }
 }
```

## 7.9 : Custom Html Helper Kullanılmış mı?

. Custom Html Helper : Bu projede, ihtiyacımız doğrultusunda mevcut HTML Helpers ile sınırlı kalmak zorunda olmadığımizi, kendi ihtiyaçlarımız doğrultusunda özel HTML Helpers oluşturabileceğimizi göstermek istedik. Gördüğünüz gibi, burada belirtilen özellikleri kullanarak form elemanlarını oluşturduk:

- Html.BeginForm : Veri gönderimi yapacak formu başlatır.
- Html.ValidationSummary : Hata özetini gösterir.
- Html.LabelFor : Belirli bir model özelliği için etiket oluşturur.
- Html.TextBoxFor : Belirli bir model özelliği için metin kutusu oluşturur.
- Html.PasswordFor : Belirli bir model özelliği için şifre kutusu oluşturur.
- Html.ValidationMessageFor : Belirli bir model özelliği için doğrulama mesajını gösterir.
- Html.ActionLink : Bağlantı oluşturur.

. Bu şekilde, projemizde ihtiyaçlarımıza karşılamak için gerekli özellikleri kullanarak kendi HTML Helpers yapıalarımızı oluşturduk. Bu yaklaşım, esneklik sağlayarak projeye değer katmıştır.



```
1  namespace Sinema_Dunyasi.Helpers
2  {
3      public static class ContactInfoHelper
4      {
5          public static IHtmlContent ContactInfo(this IHtmlHelper htmlHelper)
6          {
7              var builder = new TagBuilder("<div>");
8              builder.AddCssClass("col-md-4");
9
10             // Telefon
11             var phoneSection = new TagBuilder("div");
12             phoneSection.AddCssClass("mb-3");
13             phoneSection.AddCssClass("form-group");
14             phoneSection.InnerHtml.Append("TELEFON :");
15             var phoneText = new TagBuilder("p");
16             phoneText.InnerHtml.Append("<br /><br />+90 542 867 891 <br />/>1 (234) 567 891");
17             phoneSection.InnerHtml.AppendAt(phoneText);
18             phoneSection.InnerHtml.AppendAt(builder);
19             builder.InnerHtml.AppendAt(phoneSection);
20
21             // İlgili Yasa
22             var lawSection = new TagBuilder("div");
23             lawSection.AddCssClass("mb-3");
24             lawSection.AddCssClass("form-group");
25             lawSection.InnerHtml.Append("YASA :");
26             var lawText = new TagBuilder("p");
27             lawText.InnerHtml.Append("MADE S (1) Ver sahlayıcı, yer sahipliği içeriği kontrol etmek veya hükmü aykırı bir faaliyetin söz konusu olup olmadığı arastırmağa yüklenir.");
28             lawSection.InnerHtml.AppendAt(lawText);
29             lawSection.InnerHtml.AppendAt(builder);
30             builder.InnerHtml.AppendAt(lawSection);
31
32             // Hukuki
33             var legalSection = new TagBuilder("div");
34             legalSection.AddCssClass("mb-3");
35             var legalTitle = new TagBuilder("h4");
36             legalTitle.AddCssClass("mb-3");
37             legalTitle.InnerHtml.Append("HUKUK :");
38             var legalText = new TagBuilder("p");
39             legalText.InnerHtml.Append("er sahlayıcı, yer sahipliği hukuka aykırı içeriğten, ceza sorumluluğu ile ilgili hükümler saklı kalmak kaydıyla, bu Kanunun 8 inci ve 9 uncu maddelerine göre haberdar edilmesi halinde teknik olarak");
40             legalSection.InnerHtml.AppendAt(legalText);
41             legalSection.InnerHtml.AppendAt(builder);
42             builder.InnerHtml.AppendAt(legalSection);
43
44             builder.InnerHtml.AppendAt("<br /><div class='hr mb-3'></div>");
45
46         }
47     }
48 }
```

- . Bu projede, ihtiyacımız doğrultusunda kendi oluşturduğumuz **ContactInfo Helper**'ı Razor Page'de kullandık ve talebimize göre özelleştirdiğimiz özelliklerden yararlandık. Örneğin, burada **email** ve **şifre** alanları için HTML Helper'ları kullanarak form elemanlarını dinamik ve verimli bir şekilde oluşturduk.

```

@model Sinema_Dunyasi.Models.Domain.Contact
@using Sinema_Dunyasi.Helpers

@{
    ViewComponent 23
}
@Custom_Html_Helper 8
<section id="Contact-section">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-12 col-lg-10">
                <h2 class="text-center mb-2">Bize Ulaşın</h2>
                <div class="hr mb-5"></div>
            </div>
        </div>

        <!-- TempData mesajları -->
        @if (TempData["Success"] != null)
        {
            <div class="alert alert-success">
                @ TempData["Success"]
            </div>
        }
        @if (TempData["Error"] != null)
        {
            <div class="alert alert-danger">
                @ TempData["Error"]
            </div>
        }

        <div class="row justify-content-center">
            <div class="col-md-6">
                @Html.ContactInfo() <!-- Custom Html Helper -->
            </div>
            <div class="col-md-6">
                @await Component.InvokeAsync("ContactForm") <!-- ViewComponent -->
            </div>
        </div>

        <div class="map-container mt-5">
            <iframe src="https://www.google.com/maps/embed?pb=1Im181m121lM31d12613_90611923395412d38_54058857884588813d37_7788707014204512m31f812f013f013m211102412176814f13_113m31w2" style="width: 100%; height: 100%; border: none;">
        </div>
    </div>
</div>

```

79 % No issues found

Autos Locals Watch 1 Package Manager Console Call Stack Breakpoints Exception Settings Output Error List Bookmarks

## 7.10 : Tag Helper Kullanılmış mı?

- . Tag Helper : Bu projede, ihtiyaçlarımız doğrultusunda **Tag Helper**'ları kullanarak form elemanlarını oluşturduk. Örneğin, bir form elemanını şu şekilde oluşturduk:

- <input asp-for="GenreName" class="form-control" />

- . Bu, geleneksel HTML kullanımı yerine Tag Helper'i kullanarak daha basit ve anlaşılır bir kod yazmamızı sağladı.

```

@model Sinema_Dunyasi.Models.Domain.Genre
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

@{
    Layout = "/Views/Shared/_LayoutAdmin.cshtml";
}

<form class="w-40" href="/Adminler/Genre/Add" method="post">
    <h2 class="pb-10">Film Türü Ekle</h2>
    <div class="input-containen">
        <label asp-for="GenreName" class="w-40">Tür Adı:</label>
        <input type="text" class="input" asp-for="GenreName">
    </div>
    <div class="message-box pd-x-28 error">
        <span asp-validation-for="GenreName" ></span>
    </div>

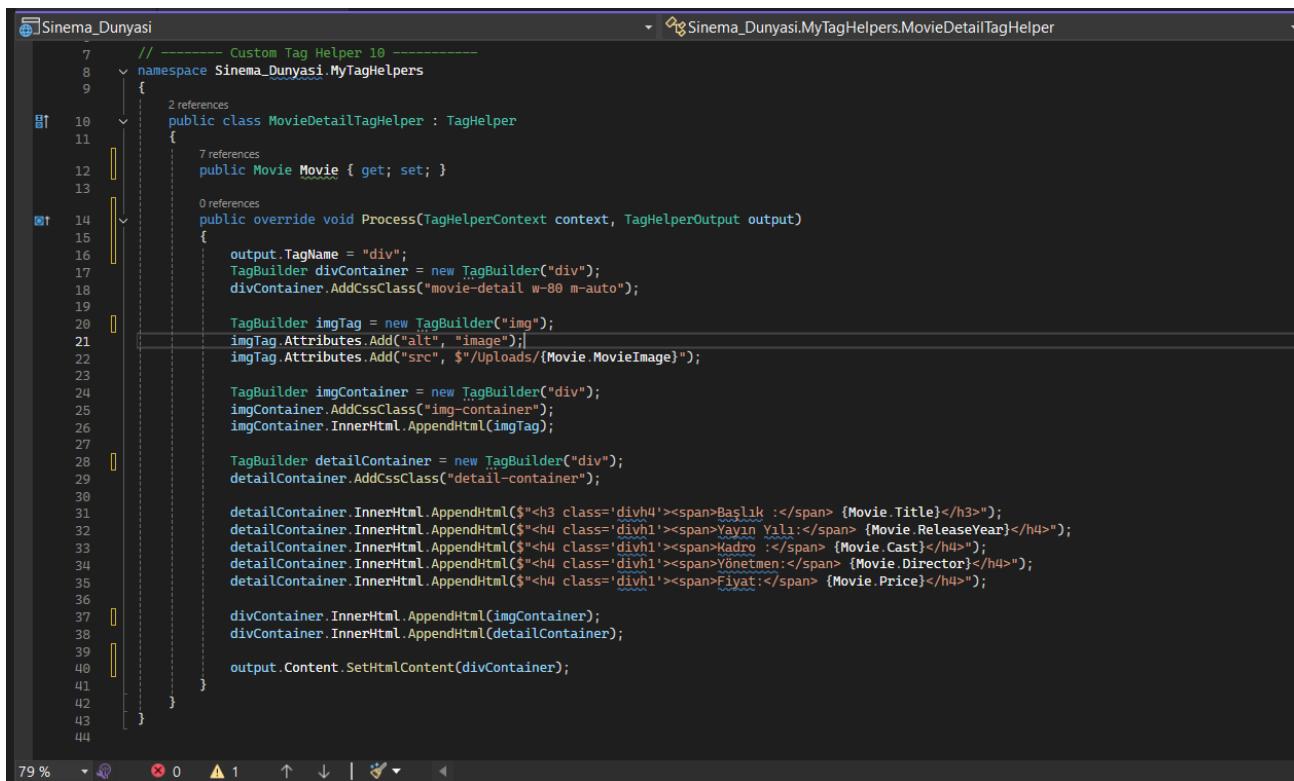
    @if (TempData["msg"] != null)
    {
        <div class="alert mb-1">
            @ TempData["msg"]
        </div>
    }

    <div class="input-containen">
        <button class="btn btn-default" type="submit" value="Kaydet"><button>
        <a class="btn btn-sec" href="/Adminler/Genre/GenreList">Tüm kayıtlar</a>
    </div>
</form>
<br /><br /><br />

```

## 7.11 : Custom Tag Helper Kullanılmış mı?

- . Custom Tag Helper : Bu projede, Microsoft'un sunduğu esneklik sayesinde ihtiyacımız doğrultusunda kendi özel Tag Helper'larımıza oluşturabiliyoruz. Yani, sadece mevcut Tag Helper'larla sınırlı kalmak zorunda değiliz; istediğimiz özelliklere sahip kendi Tag Helper'larımıza oluşturabiliyoruz.
- . Örneğin, bu projede bir **Custom Tag Helper** oluşturduk. Resimde görüldüğü gibi, **MovieDetailTagHelper** adlı özel bir Tag Helper oluşturduk. Bu Tag Helper, belirli bir film hakkında dinamik olarak HTML elemanları oluşturmaktadır.
- . Özellikle, **div**, **img**, ve **span** elemanlarını kullanarak, film başlığı, açıklaması, yönetmeni, fiyatı ve afiş gibi bilgileri içeren HTML yapısını dinamik olarak oluşturur. Bu, bizim ihtiyacımıza göre özelleştirdiğimiz bir yapıdır ve geliştirdiğimiz uygulamalarda daha esnek ve dinamik bir çözüm sunar.



The screenshot shows the Visual Studio code editor with the file `MovieDetailTagHelper.cs` open. The code defines a custom tag helper for movie details. It uses `TagBuilder` objects to construct HTML elements like `<div>` and `<img>`, applying CSS classes such as `movie-detail` and `detail-container`. The code also handles `Movie` objects and their properties like `Title`, `ReleaseYear`, `Cast`, `Director`, and `Price`.

```
// ----- Custom Tag Helper 10 -----
namespace Sinema_Dunyasi.MyTagHelpers
{
    public class MovieDetailTagHelper : TagHelper
    {
        public Movie Movie { get; set; }

        public override void Process(TagHelperContext context, TagHelperOutput output)
        {
            output.TagName = "div";
            TagBuilder divContainer = new TagBuilder("div");
            divContainer.AddCssClass("movie-detail w-80 m-auto");

            TagBuilder imgTag = new TagBuilder("img");
            imgTag.Attributes.Add("alt", "image");
            imgTag.Attributes.Add("src", $"{"/Uploads/{Movie.MovieImage}}");

            TagBuilder imgContainer = new TagBuilder("div");
            imgContainer.AddCssClass("img-container");
            imgContainer.InnerHtml.AppendHtml(imgTag);

            TagBuilder detailContainer = new TagBuilder("div");
            detailContainer.AddCssClass("detail-container");

            detailContainer.InnerHtml.AppendHtml($"<h3 class='divh4'><span>Başlık :</span> {Movie.Title}</h3>");
            detailContainer.InnerHtml.AppendHtml($"<h4 class='divh1'><span>Yayın Yılı:</span> {Movie.ReleaseYear}</h4>");
            detailContainer.InnerHtml.AppendHtml($"<h4 class='divh1'><span>Kadro :</span> {Movie.Cast}</h4>");
            detailContainer.InnerHtml.AppendHtml($"<h4 class='divh1'><span>Yönetmen:</span> {Movie.Director}</h4>");
            detailContainer.InnerHtml.AppendHtml($"<h4 class='divh1'><span>Fiyat:</span> {Movie.Price}</h4>");

            divContainer.InnerHtml.AppendHtml(imgContainer);
            divContainer.InnerHtml.AppendHtml(detailContainer);

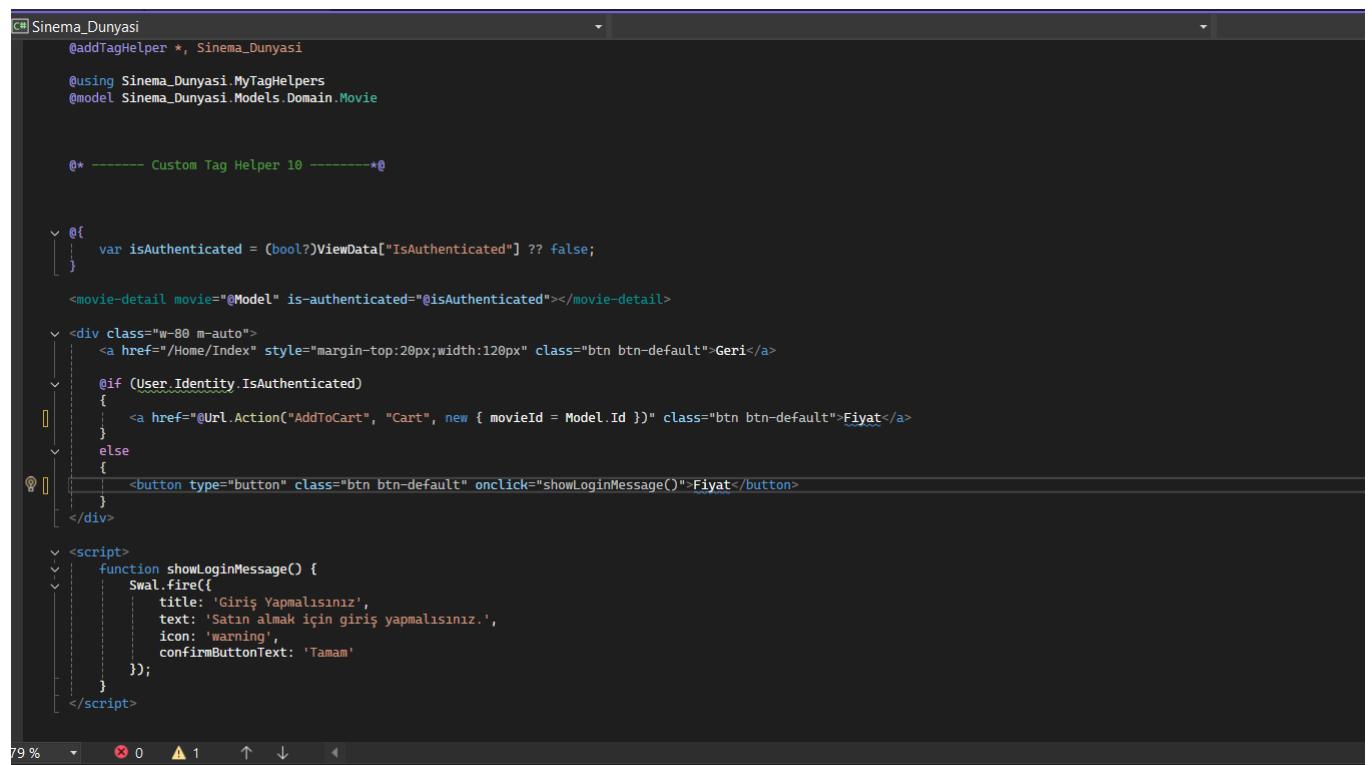
            output.Content.SetHtmlContent(divContainer);
        }
    }
}
```

. Bu projede, kendi oluşturduğumuz **Custom Tag Helper**'ı Razor Page'de kullandık. Bu sayede, ihtiyacımız doğrultusunda özelleştirilmiş özelliklerden yararlandık.

. Örneğin, oluşturduğumuz movie-detail adlı Tag Helper'ı şu şekilde Razor Page'de kullandık:

- <movie-detail movie="@Model" is-authenticated="@isAuthenticated"></movie-detail>

. Bu Custom Tag Helper, film hakkında dinamik olarak HTML elemanları oluşturur. Ayrıca, movie-detail Tag Helper'ı film detaylarını ve kullanıcı doğrulamasına göre dinamik içerikleri yönetir. Kullanıcı doğrulaması yapıldığında, film detayları ve satın alma bağlantısı ( **Fiyat** ) gösterilir. Eğer kullanıcı doğrulaması yapılmamışsa, butona tıklanınca giriş yapma mesajı gösterilir.



The screenshot shows the Visual Studio code editor with the following code:

```
C# Sinema_Dunyasi
@addTagHelper *, Sinema_Dunyasi
@using Sinema_Dunyasi.MyTagHelpers
@model Sinema_Dunyasi.Models.Domain.Movie

/* ----- Custom Tag Helper 10 ----- */

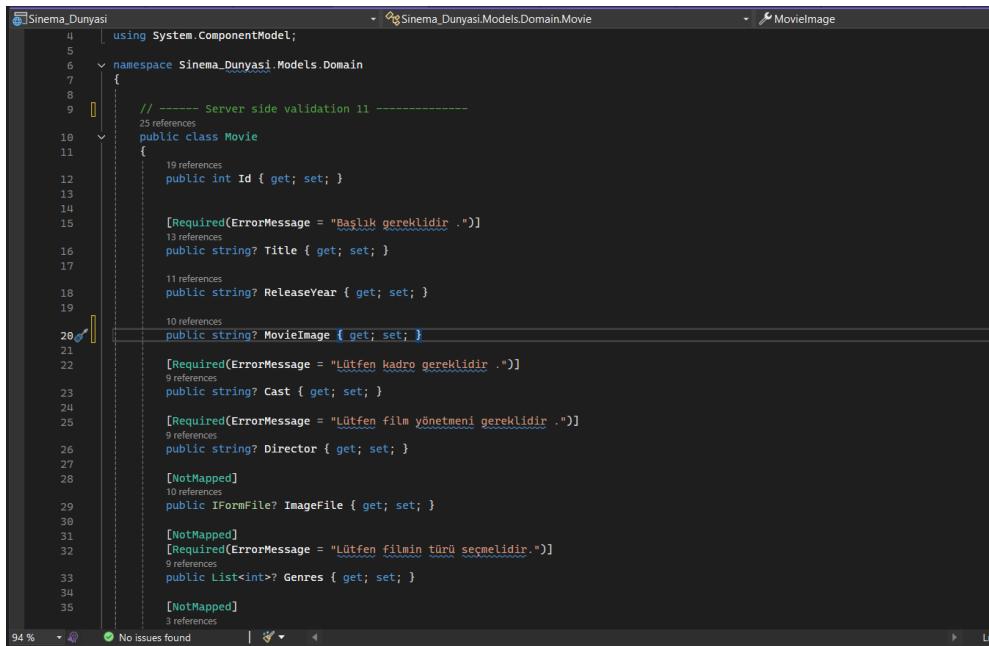
{
    var isAuthenticated = (bool?)ViewData["IsAuthenticated"] ?? false;

    <movie-detail movie="@Model" is-authenticated="@isAuthenticated"></movie-detail>

    <div class="w-80 m-auto">
        <a href="/Home/Index" style="margin-top:20px; width:120px" class="btn btn-default">Geri</a>
        @if (User.Identity.IsAuthenticated)
        {
            <a href="@Url.Action("AddToCart", "Cart", new { movieId = Model.Id })" class="btn btn-default">Fiyat</a>
        }
        else
        {
            <button type="button" class="btn btn-default" onclick="showLoginMessage()">Fiyat</button>
        }
    </div>
    <script>
        function showLoginMessage() {
            Swal.fire({
                title: 'Giriş Yapmalısınız',
                text: 'Satın almak için giriş yapmalısınız.',
                icon: 'warning',
                confirmButtonText: 'Tamam'
            });
        }
    </script>
}
```

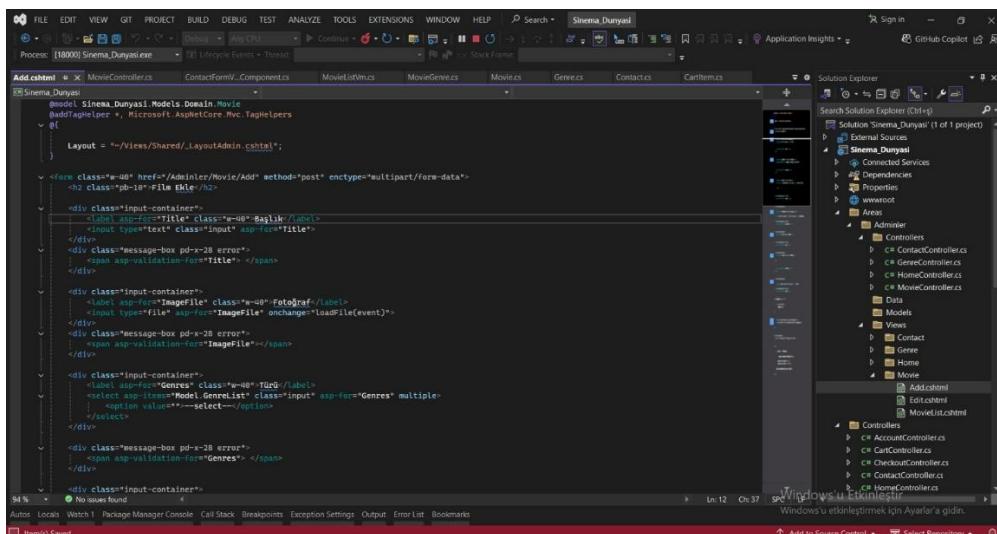
## 7.12 : Server side validation Kullanılmış mı?

- . Server Side Validation: Bu projede, **server-side validation** (sunucu tarafı doğrulama) kullandık. Bu doğrulama yöntemi, verilerin sunucuya gönderilmeden önce doğru ve eksiksiz olmasını sağlar. System.ComponentModel.DataAnnotations kütüphanesini kullanarak belirli alanların boş geçilmemesini sağlamak için doğrulama özelliklerini tanımladık.
- . Bu sayede, sunucuya veri gönderildiğinde boş bırakılan alanlar için özel hata mesajları gösterilir ve verilerin doğruluğu sağlanır.



```
4  using System.ComponentModel;
5
6  namespace Sinema_Dunyasi.Models.Domain
7  {
8
9      // ----- Server side validation 11 -----
10
11     public class Movie
12     {
13
14         [Required(ErrorMessage = "Başlık gereklidir .")]
15         public string? Title { get; set; }
16
17         [Required(ErrorMessage = "Lütfen kadro gereklidir .")]
18         public string? Cast { get; set; }
19
20         [Required(ErrorMessage = "Lütfen film yönetmeni gereklidir .")]
21         public string? Director { get; set; }
22
23         [NotMapped]
24         public IFormFile? ImageFile { get; set; }
25
26         [Required(ErrorMessage = "Lütfen filmin türü seçmelidir .")]
27         public List<int>? Genres { get; set; }
28
29         [NotMapped]
30     }
31
32     9 references
33
34     3 references
35
36
37     10 references
38
39     9 references
40
41     9 references
42
43     9 references
44
45     9 references
46
47     9 references
48
49     9 references
50
51     9 references
52
53     9 references
54
55     9 references
56
57     9 references
58
59     9 references
60
61     9 references
62
63     9 references
64
65     9 references
66
67     9 references
68
69     9 references
70
71     9 references
72
73     9 references
74
75     9 references
76
77     9 references
78
79     9 references
80
81     9 references
82
83     9 references
84
85     9 references
86
87     9 references
88
89     9 references
90
91     9 references
92
93     9 references
94
95     9 references
96
97     9 references
98
99     9 references
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
```

Projemizde, server-side validation (**sunucu tarafı doğrulama**) kullanarak model doğrulamaları yapmaktayız. Bu doğrulamalar, view kısmında **asp-for** kullanılarak çağrılmaktadır ve ilgili alanlarda doğrulama mesajları görüntülenmektedir.

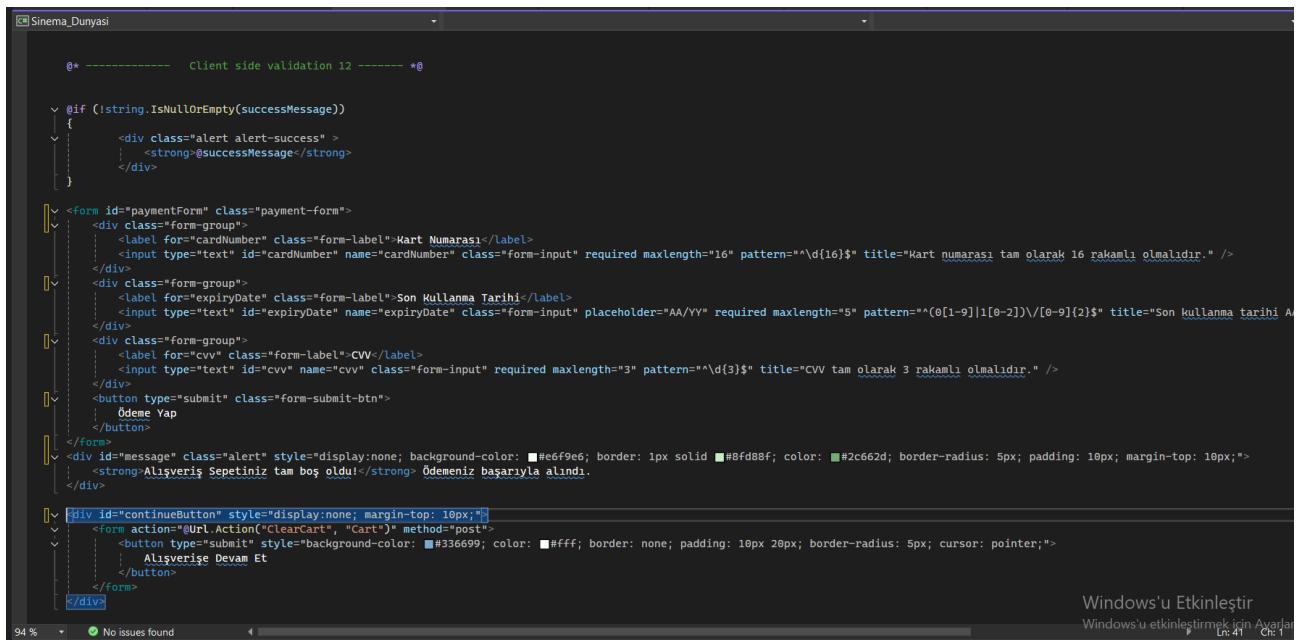


## 7.13 : Client side validation Kullanılmış mı?

Bu fotoğrafta, bir HTML formunun kodu gösteriliyor. Form, kredi kartı bilgilerini toplamak için tasarlanmış ve client-side validation (istemci tarafı doğrulama) kullanılıyor. İşte kullanılan doğrulama özellikleri:

1. required: Bu özellik, ilgili alanın doldurulmasının zorunlu olduğunu belirtir. Örneğin, kart numarası alanında required kullanılarak bu alanın doldurulması zorunlu kılmış.
2. maxlength: Bu özellik, ilgili alanın maksimum karakter uzunluğunu belirtir. Aynı örnekte, maxlength="16" kullanılarak kart numarasının maksimum 16 karakter olabileceği belirtilmiş.
3. pattern: Bu özellik, ilgili alanın belirli bir düzeni takip etmesini sağlar. Örneğin, pattern="\d{16}" kullanılarak kart numarasının tam olarak 16 rakamdan oluşması gerekiği belirtilmiştir.

Bu doğrulama özellikleri, kullanıcıların formu doğru ve eksiksiz bir şekilde doldurmasını sağlamak için kullanılır.



The screenshot shows an HTML file with client-side validation logic. The code includes an if statement to check if a success message is null or empty, displaying it in an alert if true. The main form contains fields for card number, expiration date, and CVV, each with its own validation rules (required, maxlength, and pattern). A submit button is present, and a message div at the bottom indicates a successful purchase. A continue button is also shown at the bottom.

```
/*
----- Client side validation 12 -----
*/
@if (!string.IsNullOrEmpty(successMessage))
{
    <div class="alert alert-success" >
        |   <strong>@successMessage</strong>
    </div>
}

<form id="paymentForm" class="payment-form">
    <div class="form-group">
        <label for="cardNumber" class="form-label">Kart Numarası</label>
        <input type="text" id="cardNumber" name="cardNumber" class="form-input" required maxlength="16" pattern="\d{16}" title="Kart numarası tam olarak 16 rakamlı olmalıdır." />
    </div>
    <div class="form-group">
        <label for="expiryDate" class="form-label">Son Kullanma Tarihi</label>
        <input type="text" id="expiryDate" name="expiryDate" class="form-input" placeholder="AA/YY" required maxlength="5" pattern="(0[1-9]|1[0-2])/(0[0-9]{2})$" title="Son kullanma tarihi AA/YY formatında olmalıdır." />
    </div>
    <div class="form-group">
        <label for="cvv" class="form-label">CVV</label>
        <input type="text" id="cvv" name="cvv" class="form-input" required maxlength="3" pattern="\d{3}$" title="CVV tam olarak 3 rakamlı olmalıdır." />
    </div>
    <button type="submit" class="form-submit-btn">
        Ödeme Yap
    </button>
</form>
<div id="message" class="alert" style="display:none; background-color: #e6f9e6; border: 1px solid #8fd88f; color: #2c662d; border-radius: 5px; padding: 10px; margin-top: 10px;">
    <strong>Alışveriş Sepetiniz tam boş oldu!</strong> Ödemeniz başarıyla alındı.
</div>

<div id="continueButton" style="display:none; margin-top: 10px;">
    <form action="@Url.Action("ClearCart", "Cart")" method="post">
        <button type="submit" style="background-color: #336699; color: #fff; border: none; padding: 10px 20px; border-radius: 5px; cursor: pointer;">
            Alışverişe Devam Et
        </button>
    </form>
</div>
```

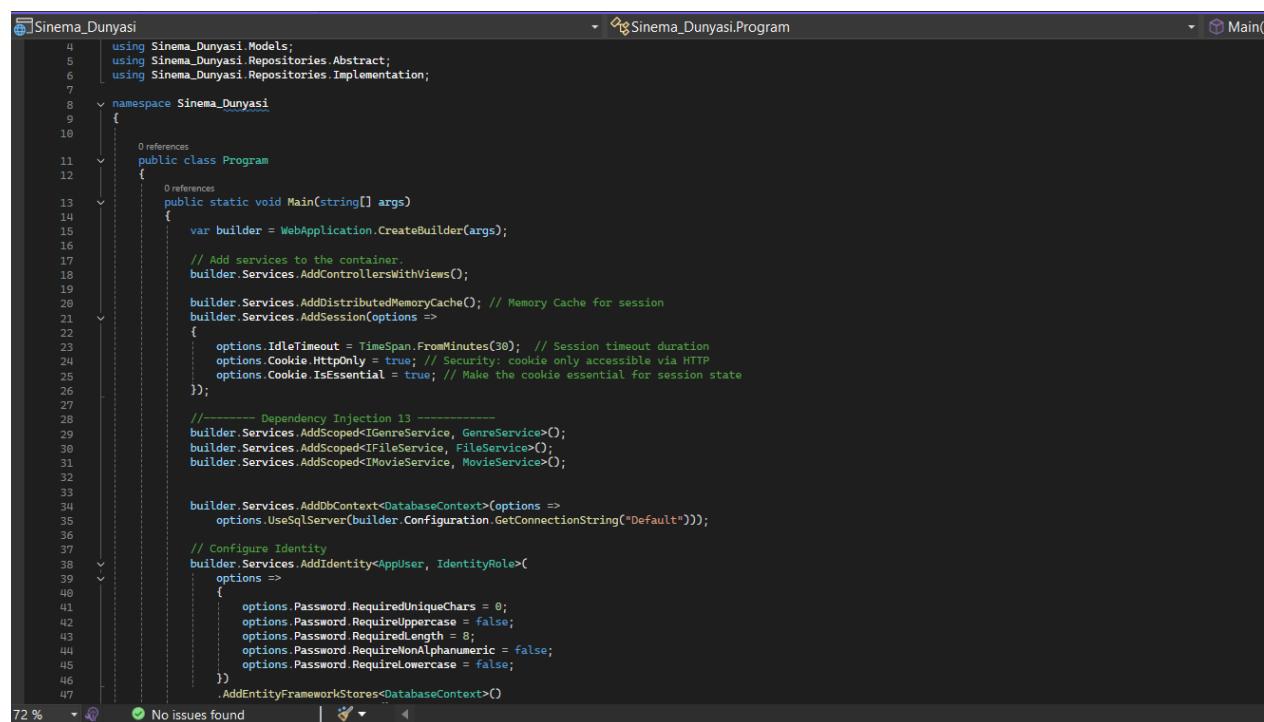
## 7.14 : Dependency Injection Kullanılmış mı?

. Dependency Injection: Bu projede **Dependency Injection** (Bağımlılık Enjeksiyonu) yöntemini kullanarak, projeye uygun olarak sınıflarımızı tanımladık ve kullanım sürelerine göre (scoped veya transient) yapılandırdık.

. Dependency Injection, uygulama bileşenlerinin birbirine bağımlılığını azaltmak ve kodun daha modüler ve test edilebilir olmasını sağlamak için kullanılan bir tekniktir. Örneğin, aşağıdaki gibi sınıfları ve hizmetleri scoped olarak tanımladık:

- builder.Services.AddScoped<IKisiService, KisiService>();
- builder.Services.AddScoped<IGenreService, GenreService>();
- builder.Services.AddScoped<IMovieService, MovieService>();
- builder.Services.AddScoped<IFileService, FileService>();

. Bu şekilde, projede ihtiyaç duyduğumuz hizmetleri ve sınıfları uygun kullanım süreleri ile yapılandıracak, daha esnek ve yönetilebilir bir mimari oluşturduk.



The screenshot shows the Visual Studio IDE with the 'Program.cs' file open in the main editor. The code is written in C# and demonstrates the configuration of a dependency injection container. It includes sections for adding services (like controllers, session options, and various service types like IGenreService, IFileService, IMovieService), setting up a database context, and configuring identity settings. The code is well-organized with comments explaining the purpose of each section. The status bar at the bottom indicates 'No issues found'.

```
1  using Sinema_Dunyasi.Models;
2  using Sinema_Dunyasi.Repositories.Abstract;
3  using Sinema_Dunyasi.Repositories.Implementation;
4
5  namespace Sinema_Dunyasi
6  {
7
8      public class Program
9      {
10
11         public static void Main(string[] args)
12         {
13             var builder = WebApplication.CreateBuilder(args);
14
15             // Add services to the container.
16             builder.Services.AddControllersWithViews();
17
18             builder.Services.AddDistributedMemoryCache(); // Memory Cache for session
19             builder.Services.AddSession(options =>
20             {
21                 options.IdleTimeout = TimeSpan.FromMinutes(30); // Session timeout duration
22                 options.Cookie.HttpOnly = true; // Security: cookie only accessible via HTTP
23                 options.Cookie.IsEssential = true; // Make the cookie essential for session state
24             });
25
26             //----- Dependency Injection 13 -----
27             builder.Services.AddScoped<IGenreService, GenreService>();
28             builder.Services.AddScoped<IFileService, FileService>();
29             builder.Services.AddScoped<IMovieService, MovieService>();
30
31
32             builder.Services.AddDbContext<DatabaseContext>(options =>
33                 options.UseSqlServer(builder.Configuration.GetConnectionString("Default")));
34
35             // Configure Identity
36             builder.Services.AddIdentity<AppUser, IdentityRole>(
37                 options =>
38                 {
39                     options.Password.RequiredUniqueChars = 0;
40                     options.Password.RequireUppercase = false;
41                     options.Password.RequiredLength = 8;
42                     options.Password.RequireNonAlphanumeric = false;
43                     options.Password.RequireLowercase = false;
44                 })
45                 .AddEntityFrameworkStores<DatabaseContext>()
46
47         }
48     }
49 }
```

Bu görüntüde, bir ASP.NET Core MVC uygulamasının MovieController.cs adlı dosyasının kodu gösteriliyor. Bu kod, filmle ilgili işlemleri yönetmek için bir denetleyici içerir ve dependency injection (bağımlılık enjeksiyonu) kullanır. İşte kodun daha detaylı bir

#### **1- Namespace ve Usings:**

```
using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.AspNetCore.Mvc.Rendering;  
using Microsoft.EntityFrameworkCore;  
using Sinema_Dunyasi.Models;  
using Sinema_Dunyasi.Models.Domain;  
using Sinema_Dunyasi.Repositories.Abstract;
```

#### **2- Namespace Deklarasyonu:**

```
namespace Sinema_Dunyasi.Areas.Adminler.Controllers
```

#### **3- Controller Sınıfı:**

```
[Area("Adminler")]  
[Authorize(Roles = "Admin")]  
public class MovieController : Controller
```

#### **4-Özel Alanlar (Private Fields):**

```
private readonly IMovieService _movieService;  
private readonly IGenreService _genreService;  
private readonly IFileService _fileService;  
private readonly DatabaseContext _databaseContext;
```

#### **5-Constructor ile Dependency Injection:**

```
public MovieController(IGenreService genreService, IMovieService movieService, IFileService  
fileService, DatabaseContext databaseContext)  
{
```

```

        _genreService = genreService;
        _movieService = movieService;
        _fileService = fileService;
        _databaseContext = databaseContext;
    }
}

```

```

1  using Microsoft.AspNetCore.Authorization;
2  using Microsoft.AspNetCore.Mvc;
3  using Microsoft.AspNetCore.Mvc.Rendering;
4  using Microsoft.EntityFrameworkCore;
5  using Sinema_Dunyasi.Data;
6  using Sinema_Dunyasi.Models.Domain;
7  using Sinema_Dunyasi.Repositories.Abstract;
8
9  namespace Sinema_Dunyasi.Areas.Adminler.Controllers
10 {
11     [Area("Adminler")]
12     [Authorize(Roles = "Admin")]
13     public class MovieController : Controller
14     {
15         private readonly IMovieService _movieService;
16         private readonly IFileService _fileService;
17         private readonly IGenreService _genService;
18         private readonly DatabaseContext _databaseContext;
19         public MovieController(IGenreService genService, IMovieService MovieService, IFileService fileService, DatabaseContext databaseContext)
20         {
21             _movieService = MovieService;
22             _fileService = fileService;
23             _genService = genService;
24             _databaseContext = databaseContext;
25         }
26
27         //-----Movie Add-----
28         public IActionResult Add()
29         {
30             var model = new Movie();
31             model.GenreList = _genService.List().Select(a => new SelectListItem { Text = a.GenreName, Value = a.Id.ToString() });
32             return View(model);
33         }
34
35         [HttpPost]
36         [ValidateAntiForgeryToken]
37         public IActionResult Add(Movie model)
38         {
39             model.GenreList = _genService.List().Select(a => new SelectListItem { Text = a.GenreName, Value = a.Id.ToString() });
40             if (!ModelState.IsValid)
41                 return View(model);
42             if (model.ImageFile != null)

```

72 % | No issues found

## **7.15 : Server side paging Kullanılmış mı?**

sunucu tarafında sayfalama (server-side paging) işlemlerini açıklıyor. Kod, veritabanından belirli sayıda film verisi çekerken sayfalama ve arama işlevselligi ekliyor. İşte detaylı bir açıklama:

Kullanılan Değişkenler ve Fonksiyonlar

- term: Kullanıcının arama yapmak için girdiği anahtar kelime.
- paging: Sayfalamanın etkin olup olmadığını belirten boolean değişken.
- currentPage: Kullanıcının o anki sayfa numarası.
- pageSize: Her sayfada gösterilecek kayıt sayısı. Burada sabit olarak 10 belirlenmiş.
- list: Filmler listesini tutan değişken.
- data: MoviesListVM türünde, sonuçları saklayan ve döndüren değişken.

### **1-Veritabanından Veri Çekme:**

```
var list = _databaseContext.Movie.ToList();
```

İlk olarak tüm filmler veritabanından çekilir.

### **2-Arama Filtreleme:**

```
if (!string.IsNullOrEmpty(term))  
{  
    term = term.ToLower();  
  
    list = list.Where(a => a.Title.ToLower().StartsWith(term)).ToList();  
}
```

Eğer arama terimi verilmişse, filmler bu terime göre filtrelenir. Filtreleme işlemi, film başlığının arama terimi ile başlaması koşuluyla gerçekleştirilir.

### **3-Sayfalama (Paging):**

```
if (paging)  
{  
    int pageSize = 10;  
  
    int listCount = list.Count;  
  
    int totalPages = (int)Math.Ceiling((count / (double)pageSize));
```

```

list = list.Skip((currentPage - 1) * pageSize).Take(pageSize).ToList();

data.PageSize = pageSize;

data.CurrentPage = currentPage;

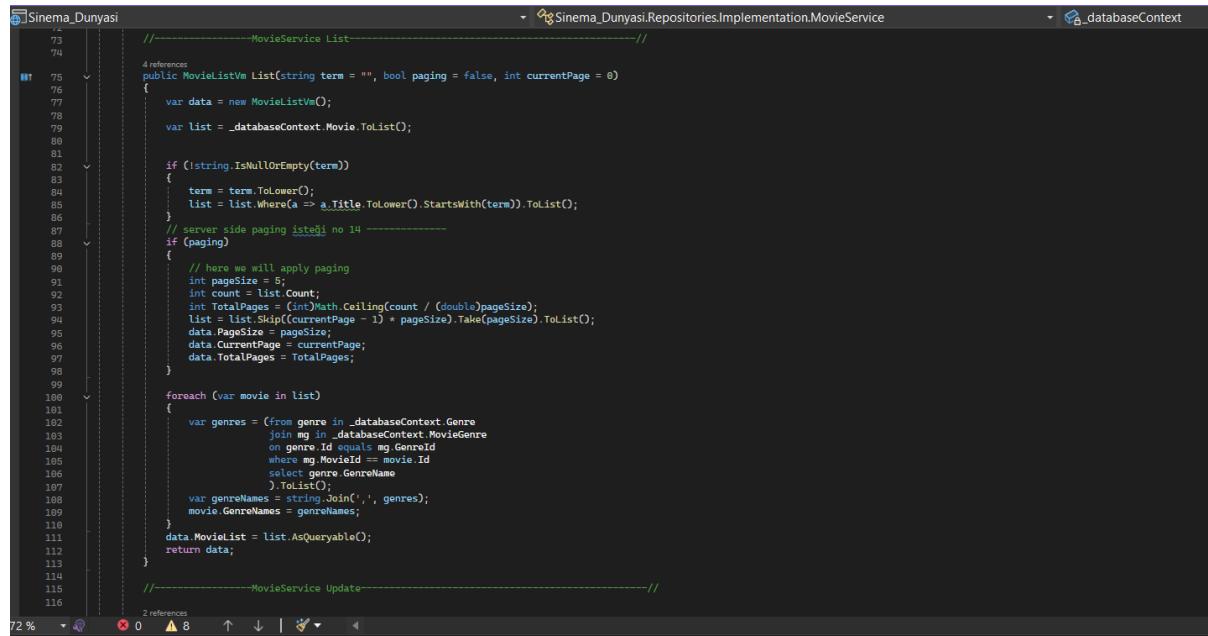
data.TotalPages = totalPages;

}

```

Skip: Skip((currentPage - 1) \* pageSize) ifadesi, belirtilen sayıda kayıt atlanarak sayfalamaya başlanması sağlar.

Take: Take(pageSize) ifadesi, belirtilen sayıda kaydın alınmasını sağlar. Burada, her sayfada 5 kayıt alınıyor.



```

//-----MovieService List-----//
73
74
75    public MovieListVm List(string term = "", bool paging = false, int currentPage = 0)
76    {
77        var data = new MovieListVm();
78
79        var list = _databaseContext.Movie.ToList();
80
81        if (!string.IsNullOrEmpty(term))
82        {
83            term = term.ToLower();
84            list = list.Where(a => a.Title.ToLower().StartsWith(term)).ToList();
85        }
86        // server side paging isteği no 14 -----
87        if (paging)
88        {
89            // here we will apply paging
90            int pageSize = 5;
91            int count = list.Count;
92            int TotalPages = (int)Math.Ceiling(count / (double)pageSize);
93            list = list.Skip((currentPage - 1) * pageSize).Take(pageSize).ToList();
94            data.PageSize = pageSize;
95            data.CurrentPage = currentPage;
96            data.TotalPages = TotalPages;
97        }
98
99        foreach (var movie in list)
100        {
101            var genres = (from genre in _databaseContext.Genre
102                         join mg in _databaseContext.MovieGenre
103                         on genre.Id equals mg.GenreId
104                         where mg.MovieId == movie.Id
105                         select genre.GenreName
106                         ).ToList();
107            var genreNames = string.Join(',', genres);
108            movie.GenreNames = genreNames;
109        }
110        data.MovieList = list.AsQueryable();
111        return data;
112    }
113
114    //-----MovieService Update-----//
115
116

```

## 7.16 : Jquery kütüphanesi Kullanılmış mı?

. Jquery Kütüphanesi: Bu projede, **jQuery kütüphanesini** kullanarak dinamik ve etkileşimli web sayfaları oluşturduk. jQuery, JavaScript kodlarını daha kolay ve daha kısa yazmamıza olanak sağlar. Özellikle, buton tıklamaları gibi olayları yönetmek ve sunucu ile iletişim kurmak için AJAX isteklerini kullanmak üzere jQuery'den yararlandık.

. Örneğin, bir film sepetinden öğe kaldırma ve kullanıcıya doğrulama modalları göstermek gibi işlemler için jQuery kullanarak bu işlemleri basitleştirdik.

The screenshot shows the Visual Studio IDE interface. The code editor displays C# code for a Razor page named 'Index.cshtml'. The code includes HTML for a modal dialog and a script block that uses jQuery to handle delete operations. The Solution Explorer on the right shows the project structure, including 'Contact' and 'Movie' controllers, 'Genre' and 'Home' views, and various shared files like 'Index.cshtml' and 'MovieDetail.cshtml'.

```
<a href="#" id="checkout-button" class="btn btn-success">Ödeme Yap</a>
<div id="error-message" style="display: none; margin-top: 20px;"></div>
<div class="modal fade" id="confirmDeleteModal" tabindex="-1" aria-labelledby="confirmDeleteModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="confirmDeleteModalLabel">Ürünü Sil</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <p>Bu Ürünü sepetinizden silmek istediğiniz emin misiniz?</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">iptal</button>
        <button type="button" class="btn btn-danger" id="confirmDeleteBtn">Sil</button>
      </div>
    </div>
  </div>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
  $(document).ready(function () {
    const $totalPrice = $('#total');
    const checkoutButton = $('#checkout-button');
    const errorMessageDiv = $('#errorMessage');
    const successMessageDiv = $('#successMessage');
    let currentDeleteMovieId = null;

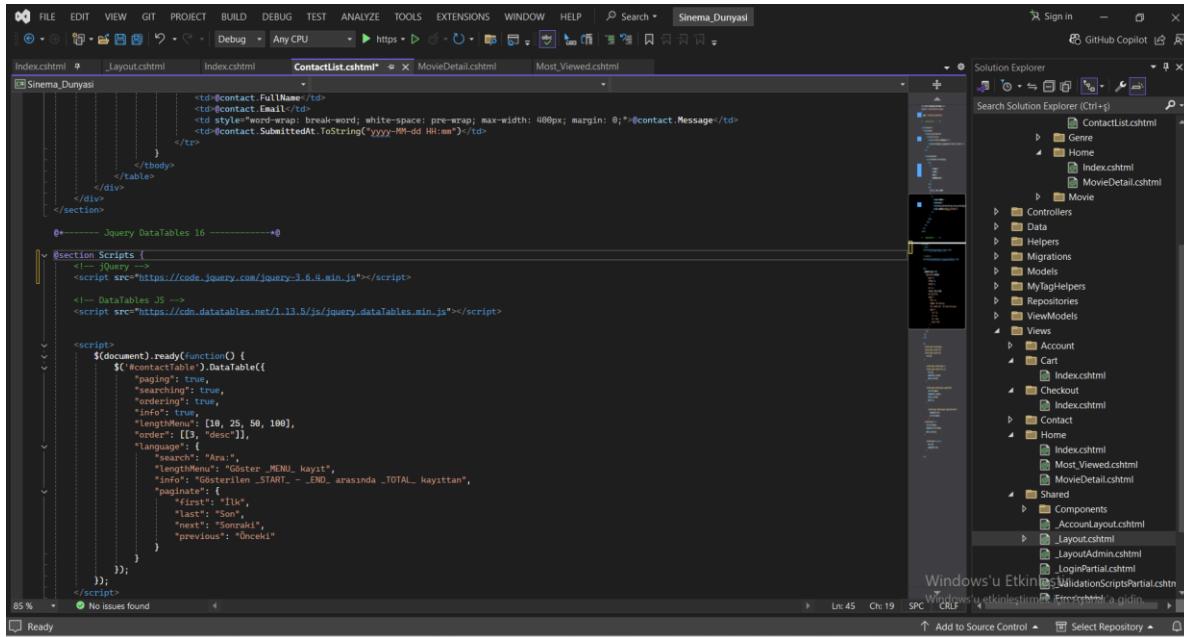
    $('#remove-link').on('click', function (event) {
      event.preventDefault();
      const $movieItem = $(this).closest('tr');
      const movieId = $movieItem.data('movie-id');

      $('#confirmDeleteModal').modal('show');
    });

    $('#confirmDeleteBtn').on('click', function () {
      const movieId = currentDeleteMovieId;
      $.ajax({
        type: 'DELETE',
        url: `/Movie/Delete/${movieId}`,
        success: function (response) {
          if (response.success) {
            $totalPrice.text(response.total);
            $movieItem.remove();
            successMessageDiv.show();
            successMessageDiv.text(`Ürün başarıyla silindi.`);
            setTimeout(() => successMessageDiv.hide(), 3000);
          } else {
            errorMessageDiv.show();
            errorMessageDiv.text(`Silme işlemi başarısız oldu.`);
            setTimeout(() => errorMessageDiv.hide(), 3000);
          }
        },
        error: function (error) {
          console.error(error);
        }
      });
    });
  });
</script>
```

## 7.17 : Jquery DataTables kütüphanesi Kullanılmış mı?

- Jquery DataTables Kütüphanesi: Bu projede, **DataTables kütüphanesini** kullanarak dinamik ve kullanıcı dostu veri tabloları oluşturduk. DataTables, büyük veri kümelerini sayfalama, arama ve sıralama gibi işlemlerle yönetmeyi kolaylaştıran güçlü bir JavaScript kütüphanesidir.
- Kodumuzda, sayfalama, arama, sıralama ve bilgi göstergesi gibi özellikleri etkinleştirdik. Bu, kullanıcıların verileri daha kolay ve etkili bir şekilde incelemelerine olanak tanır.



The screenshot shows the Visual Studio IDE interface with the ContactList.cshtml file open in the main editor. The code includes the following:

```
Index.cshtml _Layout.cshtml Index.cshtml ContactList.cshtml MovieDetail.cshtml Most_Viewed.cshtml
Sinema_Dunyasi
<tbody>
    <tr>
        <td>@contact.FullName</td>
        <td>@contact.Email</td>
        <td style="word-wrap: break-word; white-space: pre-wrap; max-width: 400px; margin: 0;">@contact.Message</td>
        <td>@contact.SubmittedAt.ToString("yyyy-MM-dd HH:mm")</td>
    </tr>
</tbody>
</table>
</div>
</section>
@----- JQuery DataTables 16 -----@
@section Scripts {
    <!-- jQuery -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <!-- DataTables JS -->
    <script src="https://cdn.datatables.net/1.13.5/js/jquery.dataTables_min.js"></script>

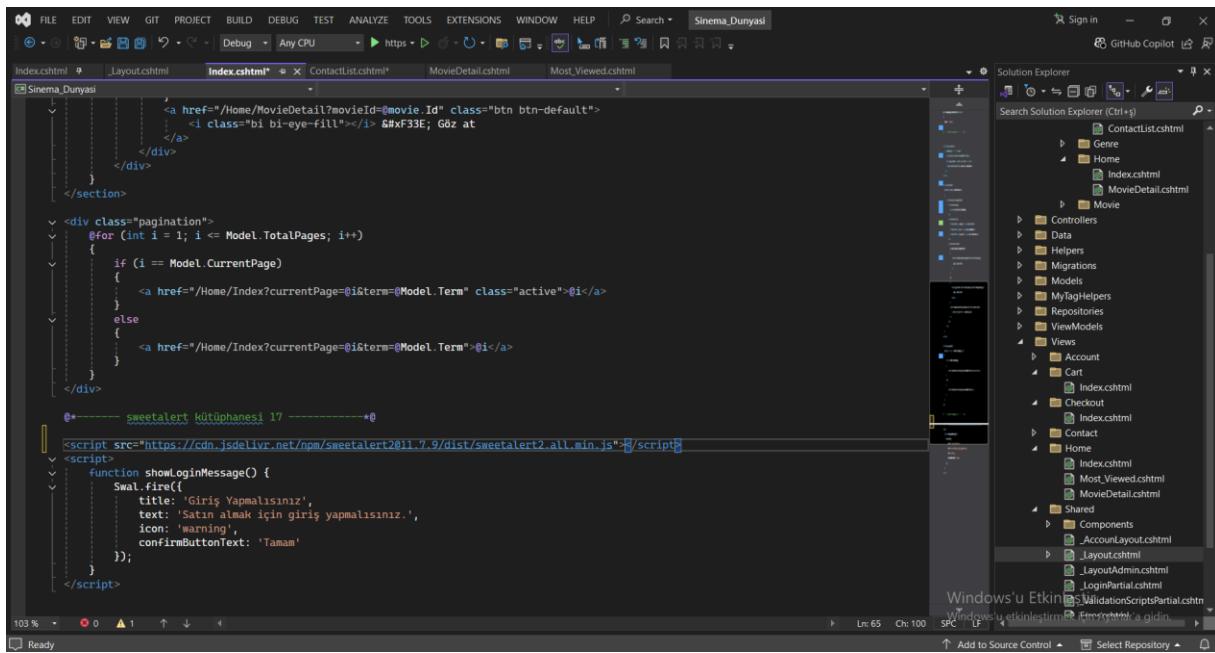
    <script>
        $(document).ready(function() {
            $('#contactTable').DataTable({
                "paging": true,
                "lengthChange": false,
                "ordering": true,
                "info": true,
                "lengthMenu": [10, 25, 50, 100],
                "order": [[0, "desc"]],
                "language": {
                    "search": "Ara:",
                    "lengthMenu": "Göster _MENU_ kayıt",
                    "info": "Gösterilen _START_ - _END_ arasında _TOTAL_ kayıttan",
                    "paginate": {
                        "first": "ilk",
                        "last": "Son",
                        "next": "Sonraki",
                        "previous": "Önceki"
                    }
                }
            });
        });
    </script>
}

```

The Solution Explorer on the right shows the project structure, including the ContactList.cshtml file under the Views/Contact folder.

## 7.18 : SweetAlert kütüphanesi Kullanılmış mı?

- . SweetAlert Kütüphanesi: Bu projede, **SweetAlert** kütüphanesini kullanarak kullanıcıya dostu ve estetik uyarılar gösterdik. Gördüğünüz gibi, SweetAlert ile kullanıcıya, örneğin giriş yapmadan bir işlem yapmak istediklerinde, anlamlı ve göz alıcı uyarılar verebiliyoruz.
- . SweetAlert, kullanıcı deneyimini iyileştirerek, işlemleri daha etkileşimli ve çekici hale getiriyor. Kodumuzda, Swal.fire() fonksiyonunu kullanarak çeşitli mesajlar ve uyarılar oluşturduk.

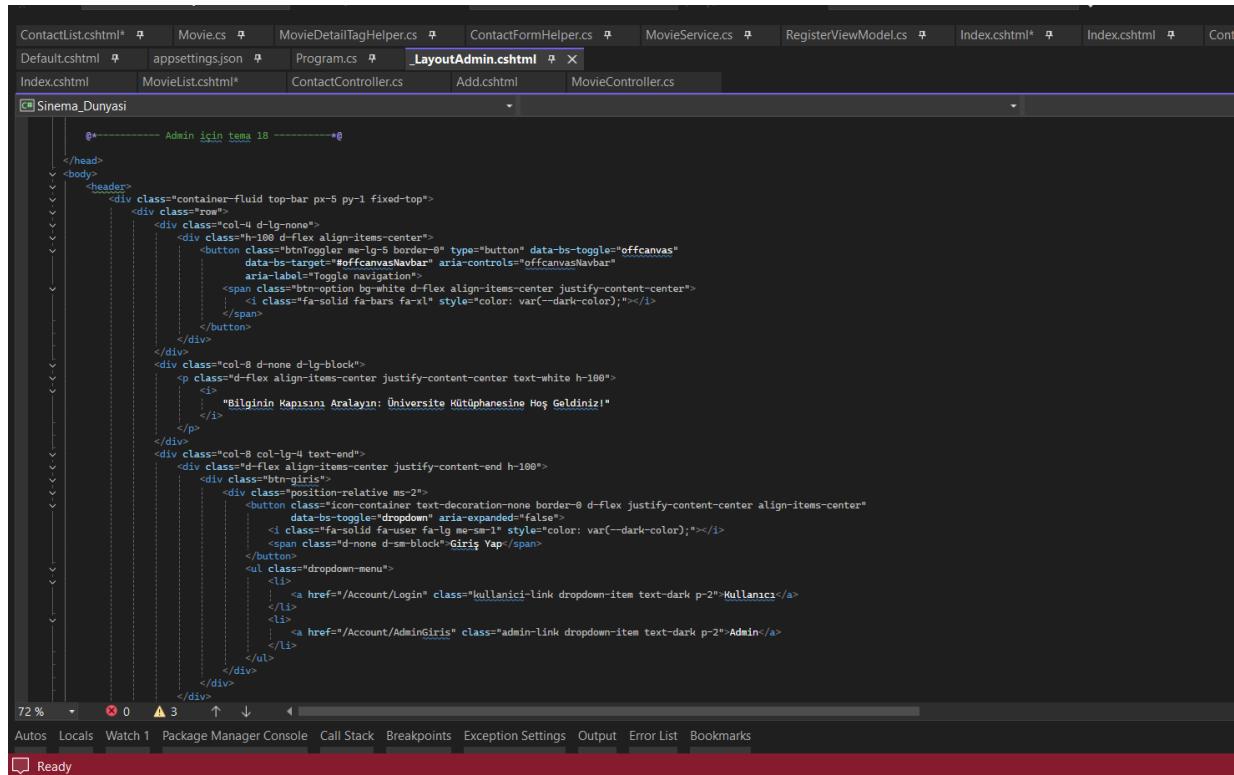


The screenshot shows the Visual Studio IDE interface with the 'Index.cshtml' file open in the editor. The code includes a section for pagination and a script block containing SweetAlert code. The script block is titled 'sweetalert kütüphanesi 17' and uses the Swal.fire() function to display a warning message about login requirements. The Solution Explorer on the right shows the project structure, including controllers, data, helpers, migrations, models, repositories, view models, and views for various controllers like Contact, Home, and Account.

```
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11.7.9/dist/sweetalert2.all.min.js"></script>
<script>
    function showLoginMessage() {
        Swal.fire({
            title: 'Giriş Yapmalısınız',
            text: 'Satın almak için giriş yapmalısınız.',
            icon: 'warning',
            confirmButtonText: 'Tamam'
        });
    }
</script>
```

## 7.19 : Admin için tema uygulanmış mı?

. Admin için Tema: Bu projede, kullanıcı ve admin için ayrı temalar oluşturduk. Gördüğünüz gibi, kullanıcılar için `_Layout.cshtml` sayfasını, adminler için ise `_LayoutAdmin.cshtml` sayfasını kullandık. Bu, kullanıcı ve admin arayüzlerini ayırarak, her iki tarafın da ihtiyaçlarına özel, özelleştirilmiş bir deneyim sunmamıza olanak sağlar.



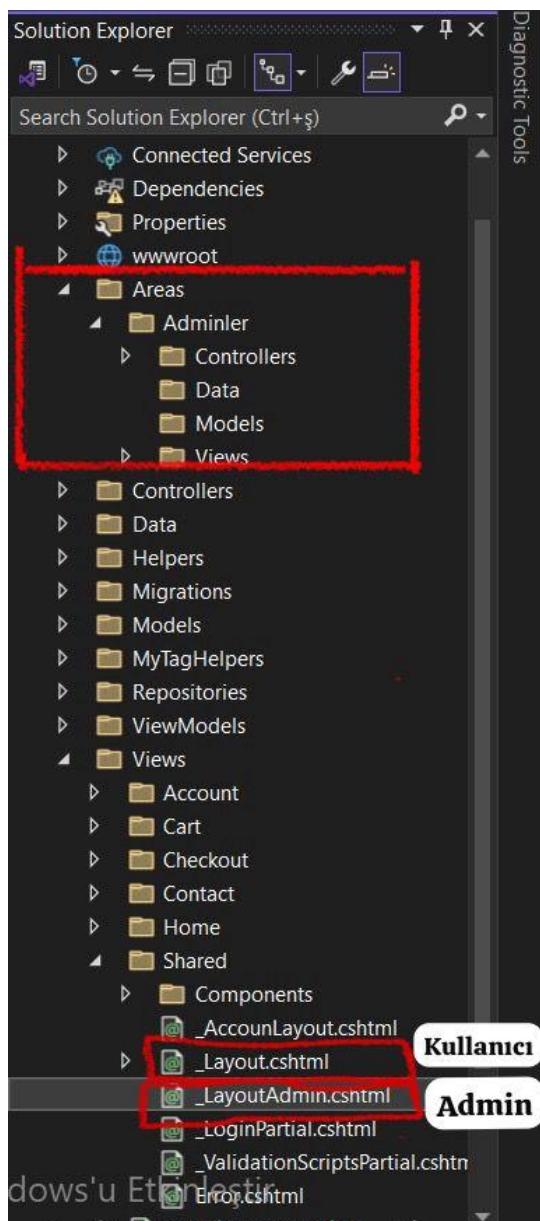
The screenshot shows the Visual Studio code editor with the `_LayoutAdmin.cshtml` file open. The code is a template for an administrator's layout page, featuring a header with a navigation menu and a sidebar with login links. The code uses Bootstrap classes like `container-fluid`, `row`, and `col` to structure the layout. It includes a sidebar with a title "Sinema Dünyası" and a message "Bilginin Kapısını Aralayan: Üniversite Kütüphanesine Hoş Geldiniz!". The sidebar also contains a dropdown menu for user and admin login.

```
<!-- Admin için tema 18 -->
</head>
<body>
    <header>
        <div class="container-fluid top-bar px-5 py-1 fixed-top">
            <div class="row">
                <div class="col-4 d-lg-none">
                    <div class="d-flex align-items-center">
                        <button class="icon icon-justify w-100 border-0 type="button" data-bs-toggle="offcanvas"
                               data-bs-target="#offcanvasNav" aria-controls="offcanvasNav">
                            <i class="fa-solid fa-bars fa-2x" style="color: var(--dark-color);"></i>
                        </button>
                    </div>
                </div>
                <div class="col-8 d-none d-lg-block">
                    <p class="d-flex align-items-center justify-content-center text-white h-100">
                        <i>
                            <i>Bilginin Kapısını Aralayan: Üniversite Kütüphanesine Hoş Geldiniz!</i>
                        </i>
                    </p>
                </div>
                <div class="col-8 col-lg-4 text-end">
                    <div class="d-flex align-items-center justify-content-end h-100">
                        <div class="btn-Login">
                            <div class="position-relative ms-2">
                                <button class="icon icon-login text-decoration-none border-0 d-flex justify-content-center align-items-center"
                                       data-bs-toggle="dropdown" aria-expanded="false">
                                    <i class="fa-solid fa-user fa-w-1" style="color: var(--dark-color);"></i>
                                    <span class="d-none d-sm-block">Giriş Yap</span>
                                </button>
                                <ul class="dropdown-menu">
                                    <li>
                                        <a href="/Account/Login" class="kullanici-link dropdown-item text-dark p-2">Kullanıcı</a>
                                    </li>
                                    <li>
                                        <a href="/Account/AdminGiris" class="admin-link dropdown-item text-dark p-2">Admin</a>
                                    </li>
                                </ul>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </header>
    <div class="container-fluid px-5 py-5">
        <div class="row">
            <div class="col-4 d-lg-none">
                <div class="d-flex align-items-center">
                    <button class="icon icon-justify w-100 border-0 type="button" data-bs-toggle="offcanvas"
                           data-bs-target="#offcanvasNav" aria-controls="offcanvasNav">
                        <i class="fa-solid fa-bars fa-2x" style="color: var(--dark-color);"></i>
                    </button>
                </div>
            </div>
            <div class="col-8 d-none d-lg-block">
                <p class="d-flex align-items-center justify-content-center text-white h-100">
                    <i>
                        <i>Bilginin Kapısını Aralayan: Üniversite Kütüphanesine Hoş Geldiniz!</i>
                    </i>
                </p>
            </div>
            <div class="col-8 col-lg-4 text-end">
                <div class="d-flex align-items-center justify-content-end h-100">
                    <div class="btn-Login">
                        <div class="position-relative ms-2">
                            <button class="icon icon-login text-decoration-none border-0 d-flex justify-content-center align-items-center"
                                   data-bs-toggle="dropdown" aria-expanded="false">
                                <i class="fa-solid fa-user fa-w-1" style="color: var(--dark-color);"></i>
                                <span class="d-none d-sm-block">Giriş Yap</span>
                            </button>
                            <ul class="dropdown-menu">
                                <li>
                                    <a href="/Account/Login" class="kullanici-link dropdown-item text-dark p-2">Kullanıcı</a>
                                </li>
                                <li>
                                    <a href="/Account/AdminGiris" class="admin-link dropdown-item text-dark p-2">Admin</a>
                                </li>
                            </ul>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

- **Areas Klasörü:** "Areas" klasörünün içinde "Adminler" adında bir alt klasör bulunur. Bu klasörde "Controllers," "Data," "Models," ve "Views" gibi alt klasörler yer alır. Bu yapı, admin modülünün ayrı bir bölüm olarak yapılandırıldığını gösterir.
- **Layouts:**
  - **Kullanıcı Layout:** "Views" klasörünün içindeki "Shared" alt klasöründe "\_Layout.cshtml" dosyası bulunur. Bu dosya, kullanıcılar için kullanılan genel bir layout (düzen) dosyasıdır.
  - **Admin Layout:** Aynı klasörde "\_LayoutAdmin.cshtml" dosyası bulunur. Bu dosya ise admin kullanıcıları için kullanılan özel bir layout dosyasıdır.

Bu yapı, projemizde kullanıcı ve admin arayüzlerini birbirinden ayırarak farklı kullanıcı deneyimleri sunar. Fotoğrafta gösterildiği gibi, proje dosyaları "sinema\_Dunyasi" klasörü altında organize edilmiştir.

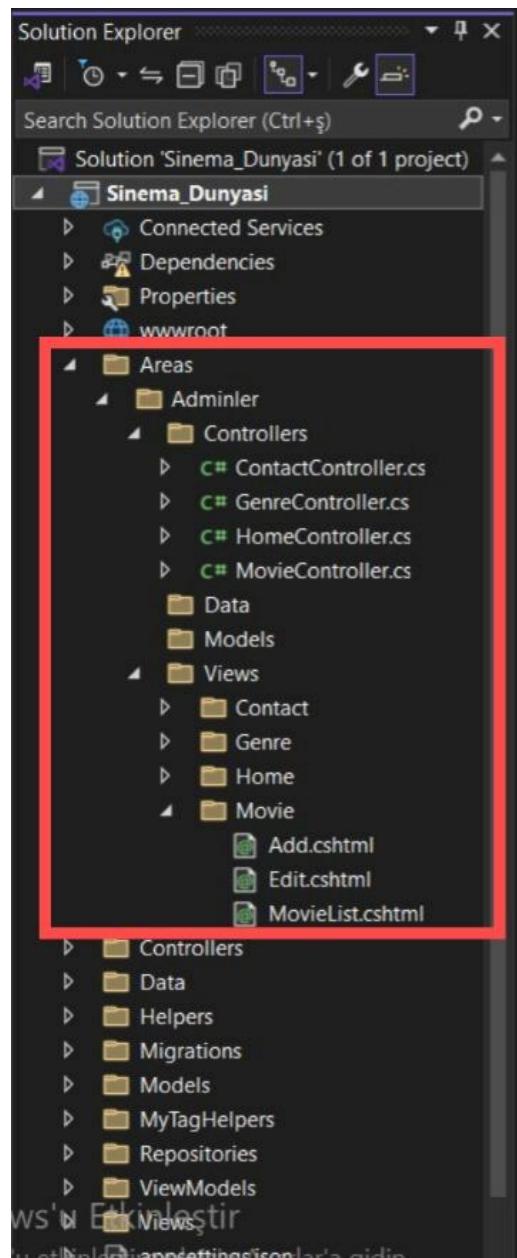


## 7.20 : Area oluşturulmuş mu?

Bu görüntü, Visual Studio'da açık olan bir ASP.NET Core MVC projesinin Solution Explorer penceresini gösteriyor. Proje adı "Sinema\_Dunyasi" ve burada alanlar (**Areas**) kullanılarak oluşturulmuş bir yapı mevcut.

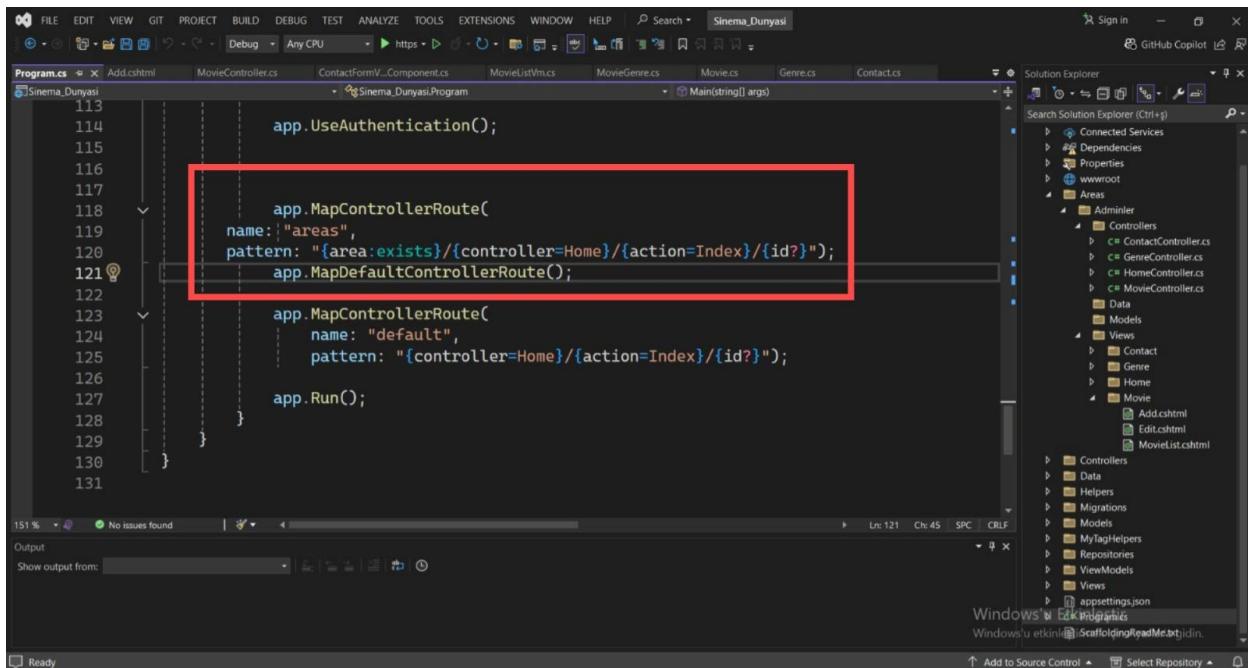
### Alanlar (**Areas**) Kullanımı

ASP.NET Core'da alanlar (**Areas**), büyük uygulamaları böümlere ayırmak ve her bölümün kendi controller, view ve model yapılarını barındırmamasını sağlamak için kullanılır. Bu, projeyi daha yönetilebilir ve düzenli hale getirir.



Bu görüntü, Visual Studio'da açık olan bir C# projesinin Program.cs dosyasını göstermektedir. Görüntüde, ASP.NET Core uygulaması için rota yapılandırması yapılmaktadır. Kodun belirli bir kısmı kırmızı bir kutu ile vurgulanmıştır. Bu kısımda, "areas" adında bir rota tanımlanmıştır.

Bu kod, ASP.NET Core uygulamasında alanlar (areas) için bir rota yapılandırması yapmaktadır. "areas" adında bir rota tanımlanmış ve bu rota için bir desen (pattern) belirlenmiştir. Desen, {area:exists}/{controller=Home}/{action=Index}/{id?} şeklindedir. Bu desen, alanın (area) var olup olmadığını kontrol eder ve eğer varsa, varsayılan olarak "Home" kontrolcüsünü ve "Index" aksiyonunu kullanır. Ayrıca, opsiyonel bir "id" parametresi de bulunmaktadır. Bu rota yapılandırması, uygulamanın farklı alanlar için farklı kontrolcüler ve aksiyonlar kullanmasına olanak tanır.



```
113
114     app.UseAuthentication();
115
116
117
118     app.MapControllerRoute(
119         name: "areas",
120         pattern: "{area:exists}/{controller=Home}/{action=Index}/{id?}");
121     app.MapDefaultControllerRoute();
122
123     app.MapControllerRoute(
124         name: "default",
125         pattern: "{controller=Home}/{action=Index}/{id?}");
126
127     app.Run();
128 }
129 }
130 }
131 }
```

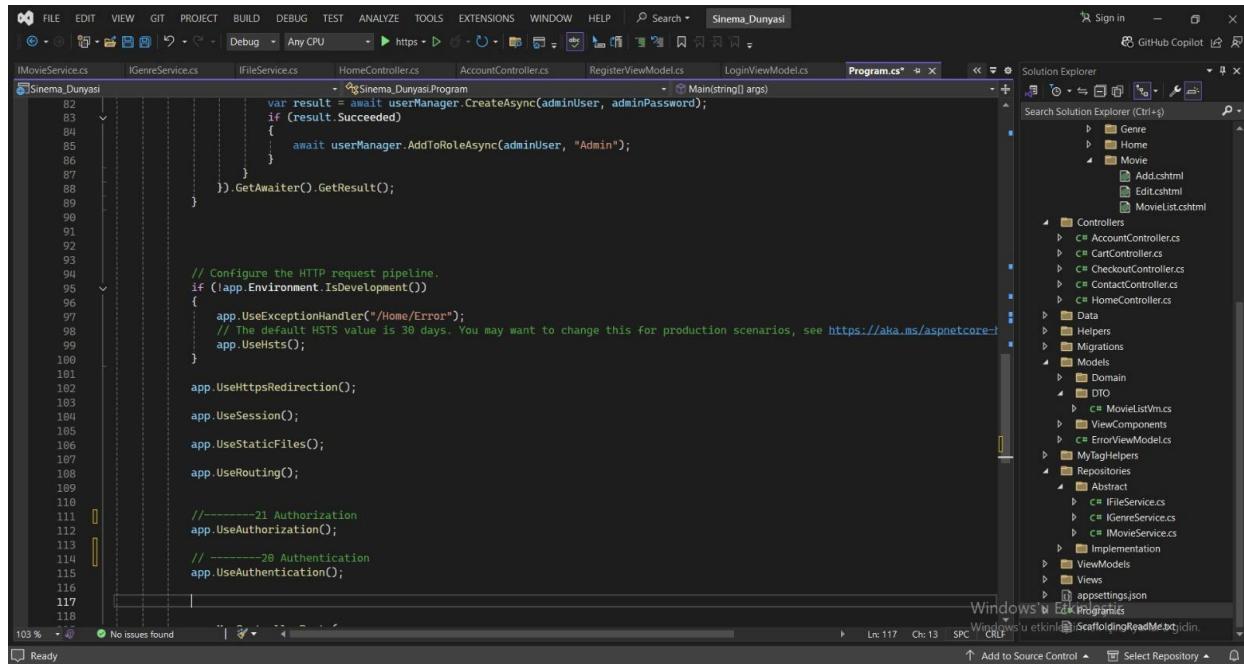
The code snippet shows the `Program.cs` file from a .NET Core application named `Sinema_Dunyasi`. The `app.MapControllerRoute` call at line 118 is highlighted with a red box. This route is specifically designed for areas. It uses the `{area:exists}` placeholder to check if an area exists, and if so, it sets the controller to `Home` and the action to `Index`, with an optional `{id?}` parameter. If no area exists, it falls back to the default route defined below it.

## 7.21 : Authentication uygulanmış mı?

Authentication yetkilendirme ile birlikte kullanım.

Açıklama:

app.UseAuthorization() kimlik doğrulama işleminden sonra, kullanıcının belirli kaynaklara erişim yetkisi olup olmadığını kontrol eder.



The screenshot shows the Visual Studio IDE with the 'Program.cs' file open in the main editor. The code is part of an ASP.NET Core application. It includes logic for creating a new user and adding them to the 'Admin' role. The 'UseAuthorization()' method is highlighted, indicating its placement in the pipeline. The Solution Explorer on the right shows the project structure with controllers like AccountController, HomeController, and ContactController, and various view files like Add.cshtml, Edit.cshtml, and MovieList.cshtml.

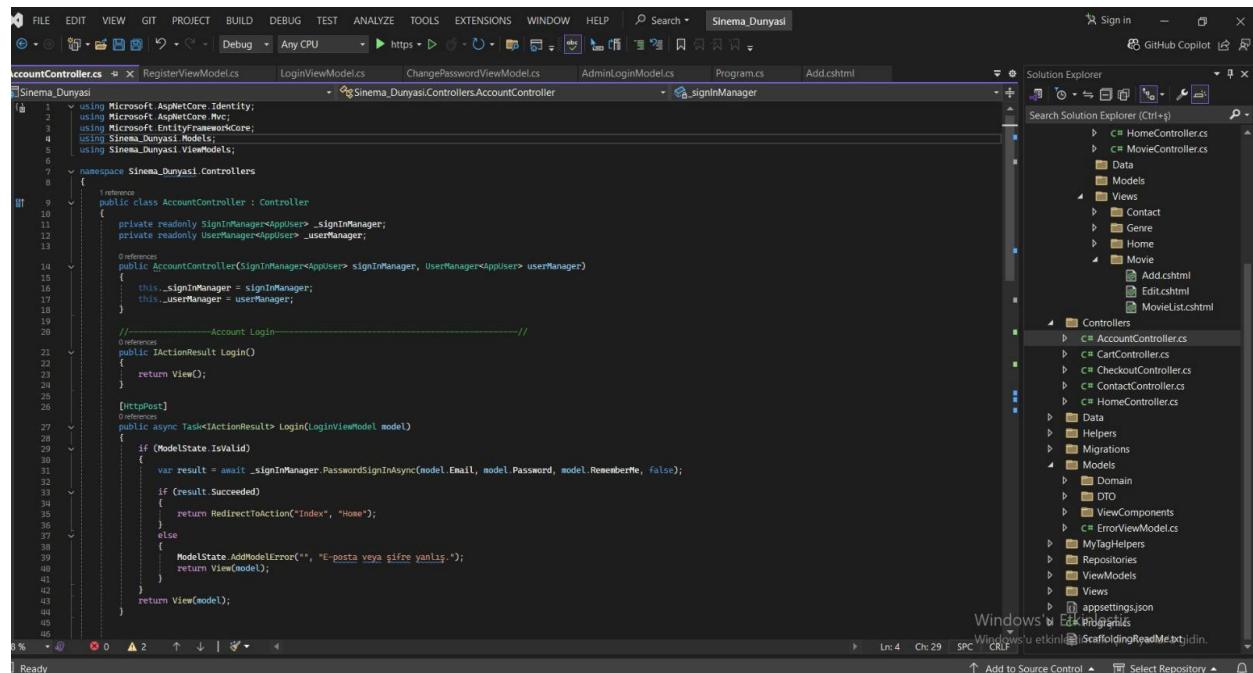
```
82     var result = await userManager.CreateAsync(adminUser, adminPassword);
83     if (result.Succeeded)
84     {
85         await userManager.AddToRoleAsync(adminUser, "Admin");
86     }
87   }).GetAwaiter().GetResult();
88 }
89
90
91
92
93
94 // Configure the HTTP request pipeline.
95 if (app.Environment.IsDevelopment())
96 {
97     app.UseExceptionHandler("/Home/Error");
98     // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
99     app.UseHsts();
100
101
102     app.UseHttpsRedirection();
103
104     app.UseSession();
105
106     app.UseStaticFiles();
107
108     app.UseRouting();
109
110
111 //-----21 Authorization
112 app.UseAuthorization();
113
114 // -----20 Authentication
115 app.UseAuthentication();
116
117
118 }
```

ASP.NET Core'da Login metodunu kullanarak kimlik doğrulama işlemi gerçekleştirilir. Bu metod, kullanıcının e-posta ve şifresini alır ve ASP.NET Core Identity sisteminin bir parçası olan \_signInManager

nesnesi ile kullanıcının kimliğini doğrular. Eğer kimlik doğrulama başarılı olursa, kullanıcı ana sayfaya yönlendirilir. Eğer başarısız olursa, kullanıcıya bir hata mesajı gösterilir.

Login metodu, kullanıcının giriş bilgilerini içeren bir model alır ve bu modelin geçerli olup olmadığını kontrol eder. Eğer model geçerli değilse, giriş formu tekrar gösterilir. Eğer model geçerliyse ve kimlik doğrulama başarılı olursa, kullanıcı ana sayfaya yönlendirilir. Eğer kimlik doğrulama başarısız olursa, kullanıcının e-posta veya şifresinin yanlış olduğunu belirten bir hata mesajı eklenir.

Bu metod, kimlik doğrulama işlemlerini güvenli bir şekilde yönetir ve kullanıcının doğru bilgileri girdiğinden emin olmak için gerekli kontrolleri yapar.



The screenshot shows the Visual Studio IDE interface. The left pane displays the `AccountController.cs` file, which contains the `Login` method. The right pane shows the `Solution Explorer` with project files like `HomeController.cs`, `MovieController.cs`, `Data`, `Models`, and `Views` for `Contact`, `Genre`, `Home`, and `Movie`.

```
1  using Microsoft.AspNetCore.Identity;
2  using Microsoft.AspNetCore.Mvc;
3  using Microsoft.EntityFrameworkCore;
4  using Sinema_Dunyasi.Models;
5  using Sinema_Dunyasi.ViewModels;
6
7  namespace Sinema_Dunyasi.Controllers
8  {
9      [Authorize]
10     public class AccountController : Controller
11     {
12         private readonly SignInManager< ApplicationUser > _signInManager;
13         private readonly UserManager< ApplicationUser > _userManager;
14
15         public AccountController(SignInManager< ApplicationUser > signInManager, UserManager< ApplicationUser > userManager)
16         {
17             this._signInManager = signInManager;
18             this._userManager = userManager;
19         }
20
21         //----- Account Login -----
22         public IActionResult Login()
23         {
24             return View();
25         }
26
27         [HttpPost]
28         public async Task< IActionResult > Login(LoginViewModel model)
29         {
30             if (ModelState.IsValid)
31             {
32                 var result = await _signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);
33
34                 if (result.Succeeded)
35                 {
36                     return RedirectToAction("Index", "Home");
37                 }
38                 else
39                 {
40                     ModelState.AddModelError("", "E-posta veya şifre yanlış.");
41                 }
42             }
43             return View(model);
44         }
45     }
46 }
```

## 7.22 Authorization kullanılmış mı?

app.UseAuthorization() kullanımı, ASP.NET Core uygulamasında yetkilendirme işlemlerini yürütmek için kullanılır. Bu middleware, uygulamanın belirli kaynaklara veya işlevlere erişim için gerekli izinleri kontrol etmesini sağlar. app.UseAuthorization() ifadesi, kullanıcıların gerekli yetkilere sahip olup olmadığını denetler.

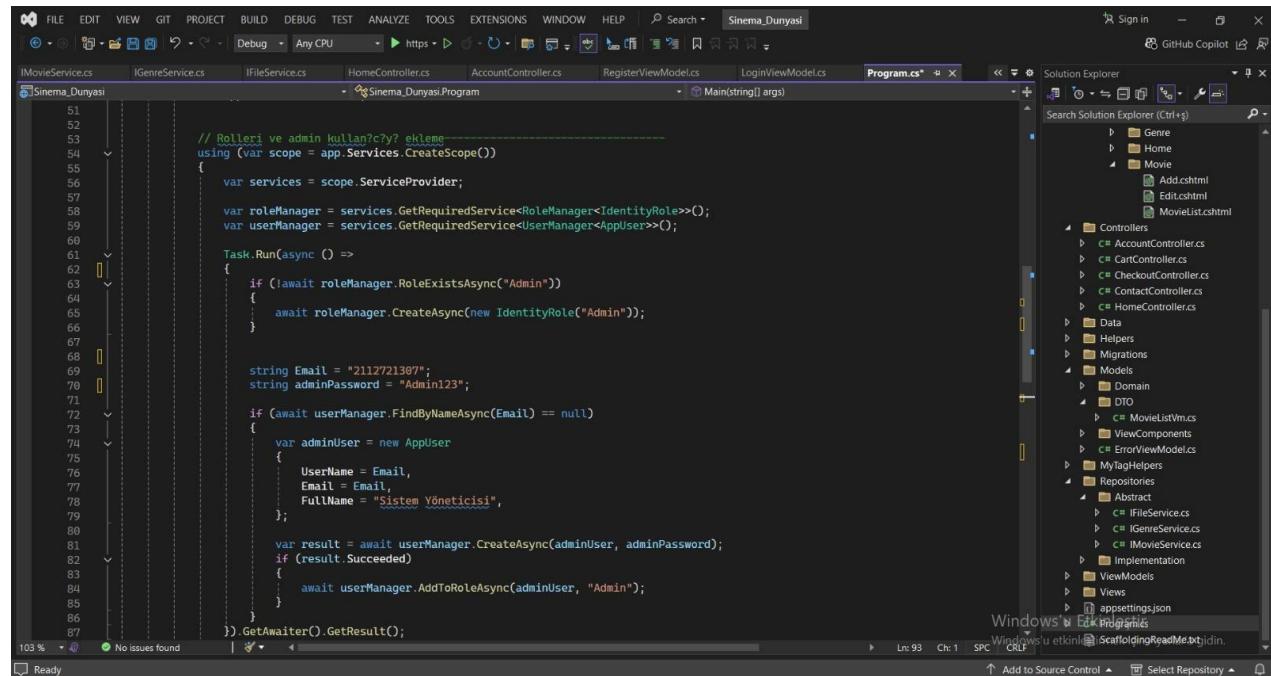
The screenshot shows the Visual Studio IDE interface with the 'Program.cs' file open in the main editor. The code in the file includes several sections of configuration for an ASP.NET Core application. Notably, around line 111, there is a comment indicating the start of the Authorization section, followed by the call to 'app.UseAuthorization();'. This is preceded by comments for sections 21 and 20, which likely refer to other configuration steps like HTTPS redirection and static files.

```
82     var result = await userManager.CreateAsync(adminUser, adminPassword);
83     if (result.Succeeded)
84     {
85         await userManager.AddToRoleAsync(adminUser, "Admin");
86     }
87 }).GetAwaiter().GetResult();
88
89 // Configure the HTTP request pipeline.
90 if (app.Environment.IsDevelopment())
91 {
92     app.UseExceptionHandler("/Home/Error");
93     // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
94     app.UseHsts();
95 }
96
97 app.UseHttpsRedirection();
98 app.UseSession();
99 app.UseStaticFiles();
100 app.UseRouting();
101
102 //-----21 Authorization
103 app.UseAuthorization();
104
105 // -----20 Authentication
106 app.UseAuthentication();
107
108
109
110
111 //-----21 Authorization
112 app.UseAuthorization();
113
114
115
116
117
118
```

Bu kod parçası, ASP.NET Core uygulamasında bir "Admin" rolünü nasıl oluşturduğunu ve bu role bir kullanıcı eklediğini gösteriyor. İşlem şu adımlarla gerçekleşir:

1. Servislerin Alınması: roleManager ve userManager gibi gerekli servisler elde edilir. Bu servisler, rollerin ve kullanıcıların yönetilmesi için kullanılır.
2. Kullanıcı Kontrolü ve Oluşturma: Belirtilen e-posta adresine sahip bir kullanıcı daha önce oluşturulmadıysa, yeni bir kullanıcı oluşturulur ve bu kullanıcıya "Admin" rolü atanır.

Bu sayede, uygulama başlatıldığında otomatik olarak bir "Admin" rolü ve belirli bir kullanıcıya bu rol atanır.



The screenshot shows the Visual Studio IDE interface with the following details:

- MenuBar:** FILE, EDIT, VIEW, GIT, PROJECT, BUILD, DEBUG, TEST, ANALYZE, TOOLS, EXTENSIONS, WINDOW, HELP.
- Toolbar:** Search, GitHub Copilot.
- Solution Explorer:** Shows the project structure for "Sinema\_Dunyasi".
- Code Editor:** Displays the `Program.cs` file with the following code:

```
51
52
53    // Rolleri ve admin kullanıcısını ekleme
54    using (var scope = app.Services.CreateScope())
55    {
56        var services = scope.ServiceProvider;
57
58        var roleManager = services.GetRequiredService<RoleManager<IdentityRole>>();
59        var userManager = services.GetRequiredService<UserManager<AppUser>>();
60
61        Task.Run(async () =>
62        {
63            if (!await roleManager.RoleExistsAsync("Admin"))
64            {
65                await roleManager.CreateAsync(new IdentityRole("Admin"));
66            }
67
68            string Email = "2112721307";
69            string adminPassword = "Admin123";
70
71            if (await userManager.FindByNameAsync(Email) == null)
72            {
73                var adminUser = new AppUser
74                {
75                    UserName = Email,
76                    Email = Email,
77                    FullName = "Sistem Yöneticisi",
78                };
79
80                var result = await userManager.CreateAsync(adminUser, adminPassword);
81                if (result.Succeeded)
82                {
83                    await userManager.AddToRoleAsync(adminUser, "Admin");
84                }
85            }
86        }).GetAwaiter().GetResult();
87    }
88
89 }
```

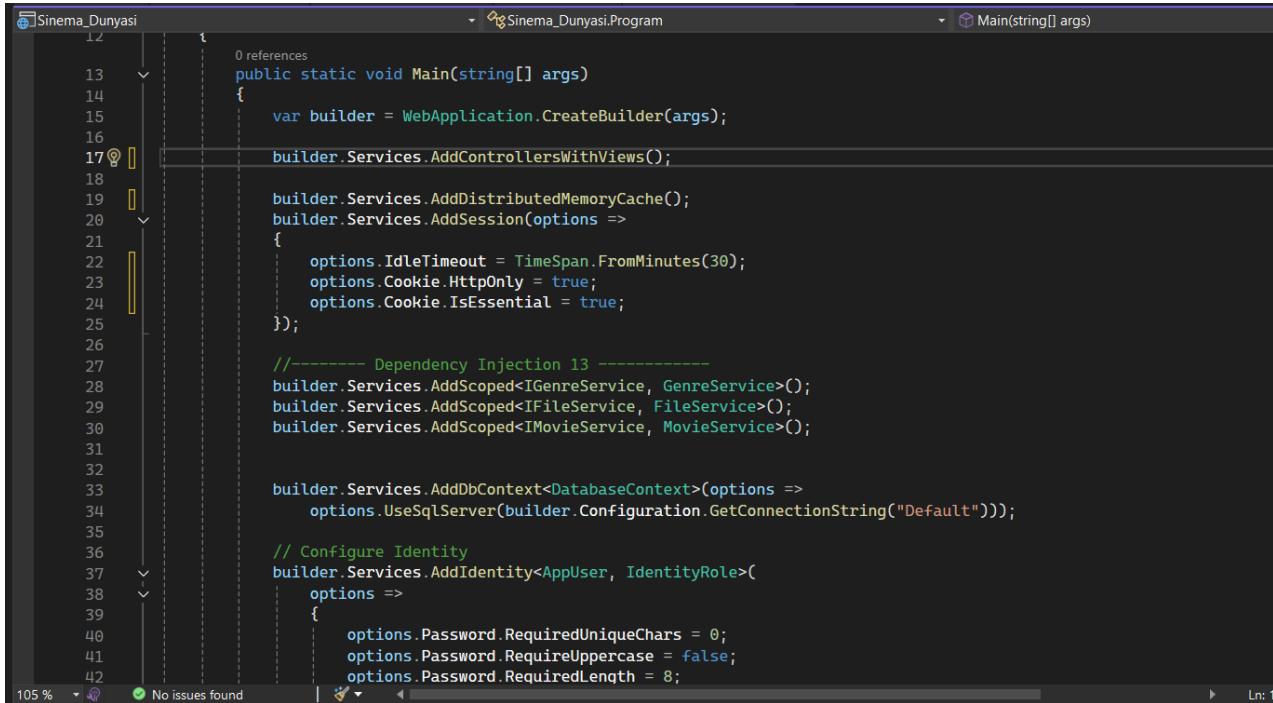
The code performs the following steps:

- Creates a service scope.
- Gets the `RoleManager` and `UserManager` services.
- Creates an "Admin" role if it doesn't exist.
- If the user with the email "2112721307" does not exist, creates a new `AppUser` with the given email and full name, and adds the "Admin" role to the user.

## 7.23 : MS Sql Server'a bağlanmış mı? Entity Framework Code First yaklaşımıyla geliştirilmiş mi?

. Veritabanı Bağlantısı: Proje, MS Sql Server'a bağlanmıştır ve Entity Framework Code First yaklaşımıyla geliştirilmiştir.

. İlk olarak program.cs tanımlanması gereken package'ları tanımladık

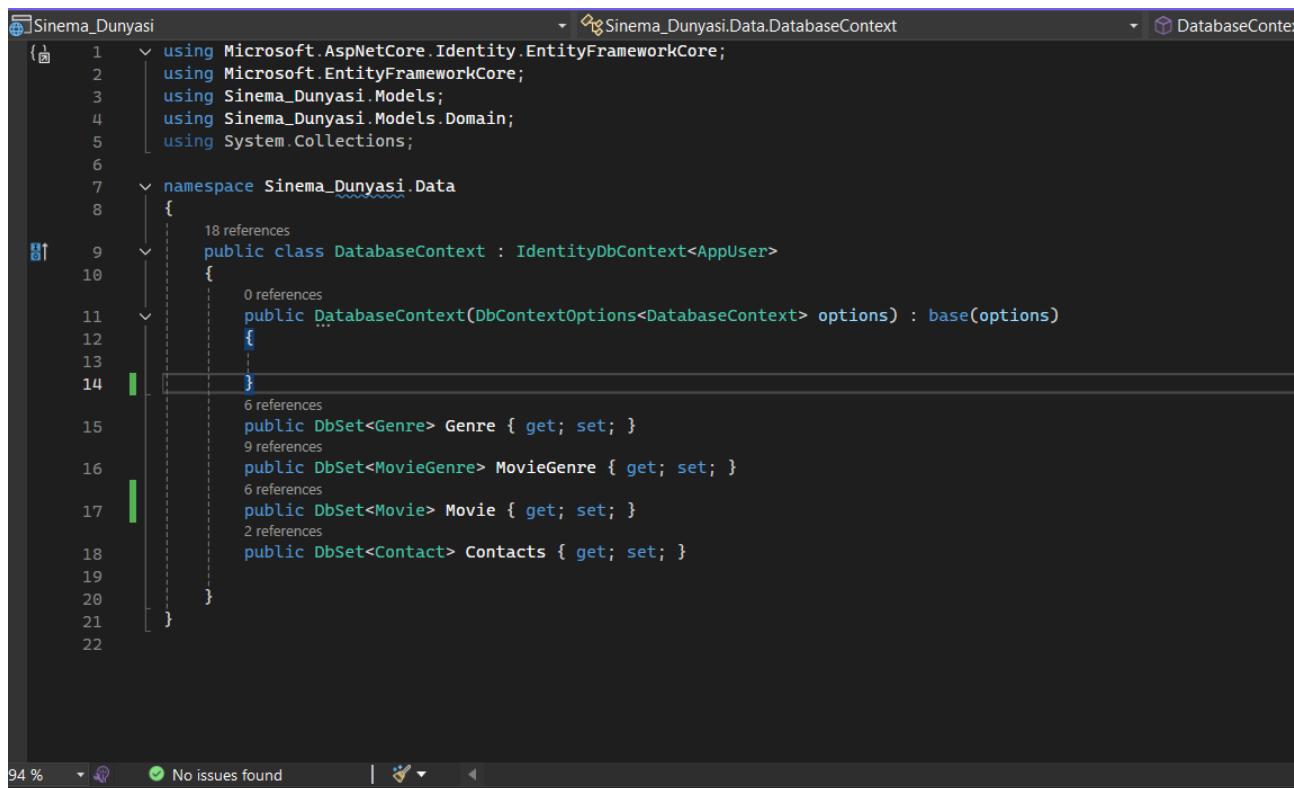


The screenshot shows the code editor window of Visual Studio with the file "Sinema\_Dunyasi\Program.cs" open. The code is as follows:

```
12
13     public static void Main(string[] args)
14     {
15         var builder = WebApplication.CreateBuilder(args);
16
17         builder.Services.AddControllersWithViews();
18
19         builder.Services.AddDistributedMemoryCache();
20         builder.Services.AddSession(options =>
21             {
22                 options.IdleTimeout = TimeSpan.FromMinutes(30);
23                 options.Cookie.HttpOnly = true;
24                 options.Cookie.IsEssential = true;
25             });
26
27         //----- Dependency Injection 13 -----
28         builder.Services.AddScoped<IGenreService, GenreService>();
29         builder.Services.AddScoped<IFileService, FileService>();
30         builder.Services.AddScoped<IMovieService, MovieService>();
31
32
33         builder.Services.AddDbContext<DatabaseContext>(options =>
34             options.UseSqlServer(builder.Configuration.GetConnectionString("Default")));
35
36         // Configure Identity
37         builder.Services.AddIdentity<AppUser, IdentityRole>(
38             options =>
39                 {
40                     options.Password.RequiredUniqueChars = 0;
41                     options.Password.RequireUppercase = false;
42                     options.Password.RequiredLength = 8;
43                 });
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105 %
```

The status bar at the bottom left indicates "No issues found". The status bar at the bottom right shows "Ln: 1".

- . Bu projede, kimlik doğrulama ve yetkilendirme işlemleri için **IdentityDbContext**'ten türetme yöntemini kullandık. **DatabaseContext** sınıfımız, **IdentityDbContext<AppUser>** sınıfından miras alarak ASP.NET Core Identity'nin yerleşik işlevselligidinden faydalandı. Bu yapı, kullanıcı kimlik doğrulama ve yetkilendirme işlemlerini yönetmemizi sağladı.
- . Ayrıca, **DbSet** özelliklerini tanımlayarak veritabanı tablolarıyla iletişim kurduk. **DbSet** özellikleri, uygulamamızda veritabanındaki tabloları temsil eder ve SQL Server ile verilerin okunmasını, yazılmasını ve güncellenmesini sağlar.
- . Bu yaklaşım, projede kimlik doğrulama ve yetkilendirme süreçlerini etkili bir şekilde yönetmemize ve SQL Server veritabanıyla entegre olmamıza olanak tanrıdı.



The screenshot shows the code editor with the file `DatabaseContext.cs` open. The code defines a `DatabaseContext` class that inherits from `IdentityDbContext<AppUser>`. It includes DbSet properties for `Genre`, `MovieGenre`, `Movie`, and `Contacts`.

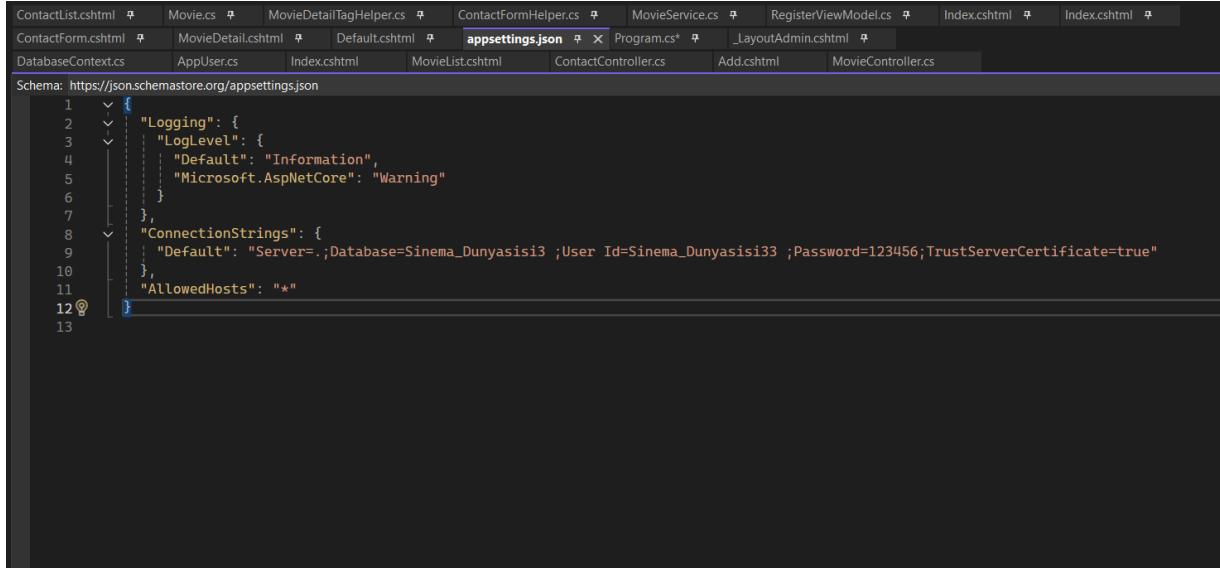
```

1  using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
2  using Microsoft.EntityFrameworkCore;
3  using Sinema_Dunyasi.Models;
4  using Sinema_Dunyasi.Models.Domain;
5  using System.Collections;
6
7  namespace Sinema_Dunyasi.Data
8  {
9      public class DatabaseContext : IdentityDbContext<AppUser>
10     {
11         public DatabaseContext(DbContextOptions<DatabaseContext> options) : base(options)
12         {
13         }
14     }
15     public DbSet<Genre> Genre { get; set; }
16     public DbSet<MovieGenre> MovieGenre { get; set; }
17     public DbSet<Movie> Movie { get; set; }
18     public DbSet<Contact> Contacts { get; set; }
19
20     }
21 }
22

```

The status bar at the bottom indicates 94% completion and shows "No issues found".

. Bu projede, SQL Server'a bağlanmak için gerekli bilgileri **appsettings.json** dosyasına ekledik. Oluşturduğumuz veritabanına bağlanmak için sunucu adı, veritabanı adı, kullanıcı adı ve parola gibi bilgileri bu dosyada belirttiğimiz gibi, bağlantı bilgilerini burada doldurduk. Bu bilgiler, uygulamanın MSSQL Server ile başarılı bir şekilde iletişim kurmasını sağlar.

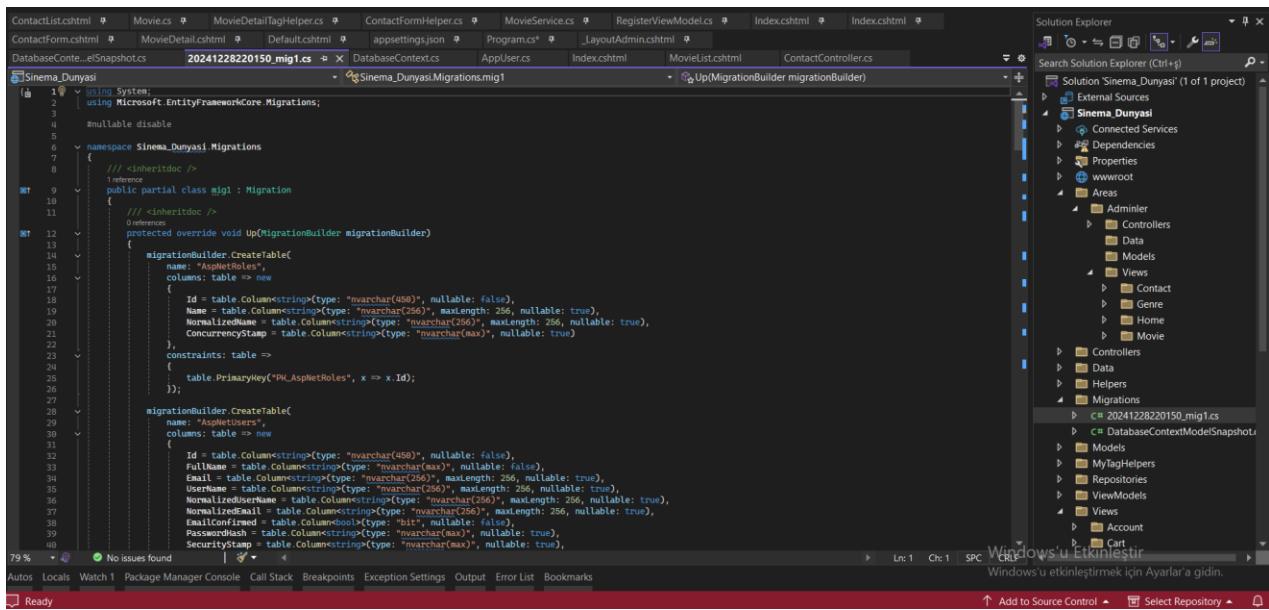


```

1  "Logging": {
2    "LogLevel": {
3      "Default": "Information",
4      "Microsoft.AspNetCore": "Warning"
6    }
7  },
8  "ConnectionStrings": {
9    "Default": "Server=.;Database=Sinema_Dunyasi3;User Id=Sinema_Dunyasi3;Password=123456;TrustServerCertificate=true"
10 },
11 "AllowedHosts": "*"
12 }
13

```

. Sonrasında, Entity Framework Code First yöntemini kullanarak oluşturduğumuz tabloların oluşturulması için terminalde add-migration komutunu yazdık. Bu komut, Entity Framework'ün oluşturduğumuz tabloları SQL sorgularına dönüştürmesini ve projede otomatik olarak bir **Migrations** klasörü oluşturmasını sağlar. Bu klasörün içinde, oluşturduğumuz tabloların SQL sorgusu şeklindeki temsilcileri bulunmaktadır.

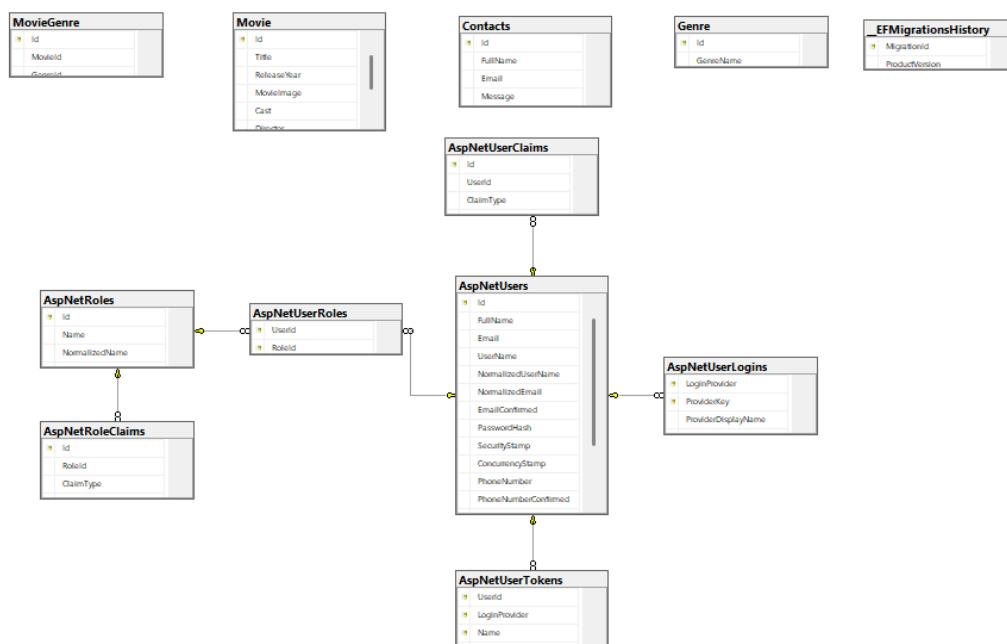


```

1  using System;
2  using Microsoft.EntityFrameworkCore.Migrations;
3
4  namespace Sinema_Dunyasi.Migrations
5  {
6    // Inheritance
7    public partial class mig1 : Migration
8    {
9      // Inheritance
10     protected override void Up(MigrationBuilder migrationBuilder)
11     {
12       migrationBuilder.CreateTable(
13         name: "AspNetRoles",
14         columns: table => new
15         {
16           Id = table.Column<string>(type: "nvarchar(450)", nullable: false),
17           Name = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
18           NormalizedName = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
19           ConcurrencyStamp = table.Column<string>(type: "nvarchar(max)", nullable: true)
20         },
21         constraints: table =>
22         {
23           table.PrimaryKey("PK_AspNetRoles", x => x.Id);
24         });
25
26       migrationBuilder.CreateTable(
27         name: "AspNetUsers",
28         columns: table => new
29         {
30           Id = table.Column<string>(type: "nvarchar(450)", nullable: false),
31           FullName = table.Column<string>(type: "nvarchar(max)", nullable: false),
32           Email = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
33           UserName = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
34           NormalizedEmail = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
35           NormalizedUserName = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
36           EmailConfirmed = table.Column<bool>(type: "bit", nullable: false),
37           PasswordHash = table.Column<string>(type: "nvarchar(max)", nullable: true),
38           SecurityStamp = table.Column<string>(type: "nvarchar(max)", nullable: true),
39         },
40         constraints: table =>
41         {
42           table.PrimaryKey("PK_AspNetUsers", x => x.Id);
43         });
44     }
45   }

```

- . Burada, oluşturduğumuz tabloları **SQL Server** üzerinden diyagramları ve verileri ile birlikte görüntüleyebiliriz. Aşağıdaki resimde görüldüğü gibi, Entity Framework kullanarak oluşturduğumuz tabloların diyagramlarını ve içeriği verileri SQL Server Management Studio gibi araçlar ile inceleyebiliriz.
- . Bu sayede, veritabanı yapısını görsel olarak değerlendirebilir ve verilerin doğru şekilde yapılandırıldığından emin olabilir



## 7.24 : ViewComponent geliştirilmiş ve Kullanılmış mı?

- . ViewComponent Kullanımı: Bu projede, ihtiyacımız doğrultusunda bir **View Component** oluşturduk. Aşağıdaki resimde görüldüğü gibi, **ContactFormViewComponent** adlı bu bileşen, kullanıcılarımızın iletişim formunu görüntülemek için kullanıldı. Bu bileşen, Razor Page'de kolayca çağrılabılır ve form verilerini yönetebilir.

```

ContactList.cshtml
Movie.cs
MovieDetailTagHelper.cs
ContactFormHelper.cs
MovieService.cs
RegisterViewModel.cs
Index.cshtml
Index.cshtml
ContactForm.cshtml
MovieDetail.cshtml
Default.cshtml
appsettings.json
Program.cs
_LayoutAdmin.cshtml
ContactForm..Component.cs
DatabaseContext.cs
AppUser.cs
Index.cshtml
MovieList.cshtml
20241228220150_mig1.cs

Sinema_Dunyasi
Sinema_Dunyasi.Models.ViewComponents.ContactFormViewComponent.cs

using Microsoft.AspNetCore.Mvc;
using Sinema_Dunyasi.Models.Domain;

namespace Sinema_Dunyasi.Models.ViewComponents
{
    public class ContactFormViewComponent : ViewComponent
    {
        public IViewComponentResult Invoke()
        {
            return View(new Contact());
        }
    }
}
  
```

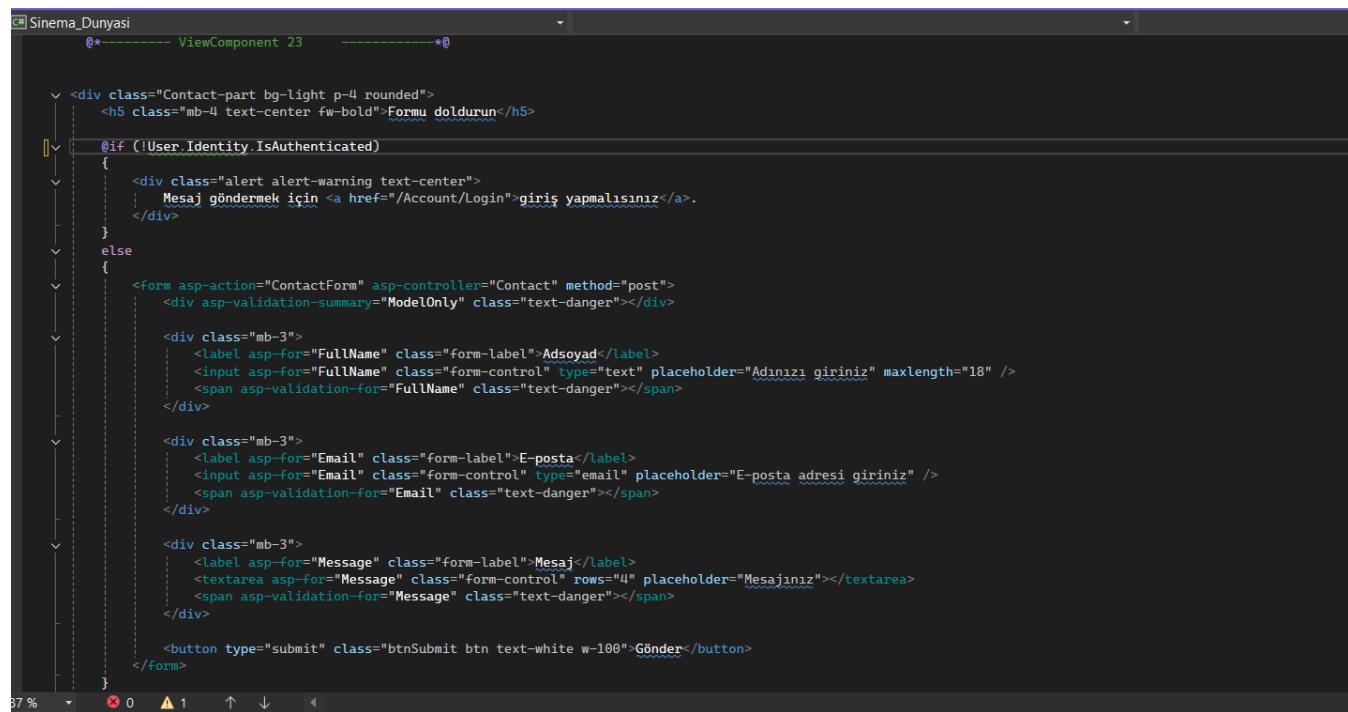
The screenshot shows the ContactForm..Component.cs file in the Visual Studio code editor. The file defines a ContactFormViewComponent class that implements the ViewComponent interface. The Invoke() method returns a View result, which displays a new Contact object. The code editor shows the file's context within the Sinema\_Dunyasi project, including other files like ContactList.cshtml, Movie.cs, and various views.

. Bu projede oluşturduğumuz View Component'in özelliklerini belirledik. Aşağıda görüldüğü gibi, bu View Component, bir iletişim formunu dinamik olarak oluşturmak ve yönetmek için kullanıldı.

. Form Özellikleri:

- FullName: Kullanıcının tam adını girmesi için bir metin alanı.
- Email: Kullanıcının e-posta adresini girmesi için bir e-posta alanı.
- Message: Kullanıcının mesajını yazması için bir metin alanı.

. Bu özellikler, formun dinamik ve kullanıcı dostu olmasını sağlar. Yakında bu bileşeni nasıl kullandığımıza dair bir görüntü paylaşacağım.



The screenshot shows the Visual Studio code editor with the file 'ContactForm.cshtml' open. The code is a View Component for a contact form. It starts with a header section and then checks if the user is authenticated. If authenticated, it displays a warning message about sending messages via email. If not authenticated, it displays a standard contact form with fields for FullName, Email, and Message, each with validation logic. The code uses Bootstrap classes for styling and ASP.NET Core validation attributes like 'asp-validation-for' and 'type="text"' or 'type="email"'.

```
<div class="Contact-part bg-light p-4 rounded">
    <h5 class="mb-4 text-center fw-bold">Formu doldurun</h5>

    @if (!User.Identity.IsAuthenticated)
    {
        <div class="alert alert-warning text-center">
            Mesaj göndermek için <a href="/Account/Login">giriş yapmalısınız</a>.
        </div>
    }
    else
    {
        <form asp-action="ContactForm" asp-controller="Contact" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>

            <div class="mb-3">
                <label asp-for="FullName" class="form-label">Ad soyad</label>
                <input asp-for="FullName" class="form-control" type="text" placeholder="Adınızı giriniz" maxlength="18" />
                <span asp-validation-for="FullName" class="text-danger"></span>
            </div>

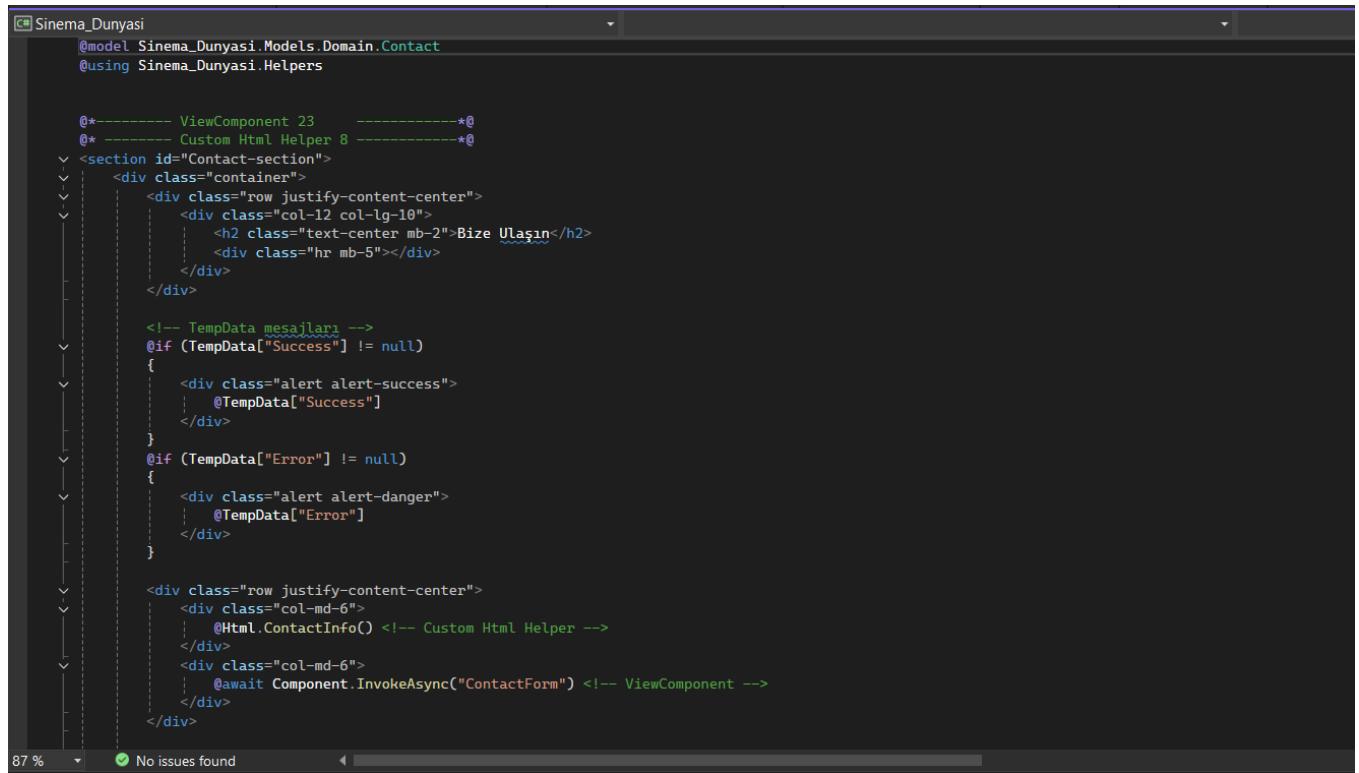
            <div class="mb-3">
                <label asp-for="Email" class="form-label">E-posta</label>
                <input asp-for="Email" class="form-control" type="email" placeholder="E-posta adresi giriniz" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>

            <div class="mb-3">
                <label asp-for="Message" class="form-label">Mesaj</label>
                <textarea asp-for="Message" class="form-control" rows="4" placeholder="Mesajınız"></textarea>
                <span asp-validation-for="Message" class="text-danger"></span>
            </div>

            <button type="submit" class="btnSubmit btn text-white w-100">Gönder</button>
        </form>
    }

```

. Bu projede, oluşturduğumuz **View Component**'i Razor Page'de **InvokeAsync** kelimesi ile çağrırdık. Gördüğünüz gibi, **ContactFormViewComponent**'i dinamik ve kullanıcı dostu bir şekilde kullanmak için bu yöntemden yararlandık.



The screenshot shows the code editor window for a Razor page named Contact.cshtml. The code is written in C# and Razor syntax. It includes imports for `@model Sinema_Dunyasi.Models.Domain.Contact` and `@using Sinema_Dunyasi.Helpers`. The page structure uses Bootstrap classes like `row justify-content-center` and `col-12 col-lg-10`. It contains sections for contact information and a message area. A conditional block checks `TempData["Success"]` and `TempData["Error"]` to display success or error alerts. At the bottom, it includes `@Html.ContactInfo()` and `@await Component.InvokeAsync("ContactForm")`, which is annotated with `<!-- ViewComponent -->`.

```
 Sinema_Dunyasi
 @model Sinema_Dunyasi.Models.Domain.Contact
 @using Sinema_Dunyasi.Helpers

 @*----- ViewComponent 23 -----*@
 @*----- Custom Html Helper 8 -----*@
<section id="Contact-section">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-12 col-lg-10">
                <h2 class="text-center mb-2">Bize Ulaşın</h2>
                <div class="hr mb-5"></div>
            </div>
        </div>

        <!-- TempData mesajları -->
        @if (TempData["Success"] != null)
        {
            <div class="alert alert-success">
                @ TempData["Success"]
            </div>
        }
        @if (TempData["Error"] != null)
        {
            <div class="alert alert-danger">
                @ TempData["Error"]
            </div>
        }

        <div class="row justify-content-center">
            <div class="col-md-6">
                @Html.ContactInfo() <!-- Custom Html Helper -->
            </div>
            <div class="col-md-6">
                @await Component.InvokeAsync("ContactForm") <!-- ViewComponent -->
            </div>
        </div>
    </div>
</section>

87 %  No issues found
```