

T.C.
ERCİYES ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

ETERNALİB WEB PROJECT

Hazırlayan

Sinan UYĞUN
1030520914
Emre YÖRÜK
1030510202
Onur Karakaya
1030520931

Danışman
Dr. Öğr. Üyesi FEHİM KÖYLÜ

Bilgisayar Mühendisliği
Bitirme Ödevi

Haziran 2024
KAYSERİ

“Eternalib Web Project” adlı bu çalışma, jürimiz tarafından Erciyes Üniversitesi Bilgisayar Mühendisliği Bölümünde Bitirme Ödevi olarak kabul edilmiştir.

03/06/2024

JÜRİ :

Danışman : Dr. Ögr. Üyesi FEHİM KÖYLÜ

Üye :

Üye :

ONAY :

Yukarıdaki imzaların, adı geçen öğretim elemanlarına ait olduğunu onaylarım.

.... / /20...

Prof. Dr. Veysel ASLANTAŞ
Bilgisayar Müh. Bölüm
Başkanı

ÖNSÖZ / TEŞEKKÜR

“ETERNALİB WEB PROJECT” adlı çalışmada içerik seçme sürecine diğer kullanıcıların ve uluslararası değerlendirme kuruluşlarının puanları ve yorumları ile katkıda bulunmaya çalışıldı .

Bu projenin gerçekleşmesi için bize destek ve katkılarından dolayı danışmanımız Asst. Prof. Fehim KÖYLÜ ‘ ye teşekkür ederiz.

ETERNALİB WEB PROJECT**Sinan UYĞUN****Emre YÖRÜK****Onur Karakaya**

**Erciyes Üniversitesi, Bilgisayar Mühendisliği
Danışman : Dr. Ögr. Üyesi FEHİM KÖYLÜ**

ÖZET

Biz geliştirdiğimiz web projesi ile birlikte insanların sevdikleri eserlere yorum yapabilme veya sevmedikleri eserlere eleştiri yapabilmelerini sağlayabilmeyi amaçladık. Bu sayede insanlar ortak veya zıt fikirlerini beyan edebilmeleri sağlamak, insanların birbirleri hakkında fikir sahibi olmalarını sağlamaya çalıştık. Projemizde olumlu dönüşleri ön plana çıkarmak için profillerde beğenilen eserleri görüntülemek ve ana sayfada en çok beğenilen eserleri sergileyerek olumlu dönüşleri ödüllendirmek istedik.

Anahtar Kelimeler: Web,Back-end,Front-end.

ETERNALIB WEB PROJECT**Sinan UYĞUN****Emre YÖRÜK****Onur Karakaya**

**Erciyes University, Computer Engineering
Supervisor: Assistant Professor FEHİM KÖYLÜ**

ABSTRACT

The objective of the web project was to facilitate the expression of opinions regarding the works that users found appealing or disagreeable. This was achieved by enabling users to provide feedback on the works they liked or disliked. In addition, the project aimed to facilitate the expression of opposing opinions and to encourage users to engage in constructive criticism. To highlight the positive feedback, the project displayed the works that had received the most likes on the user profiles. Furthermore, the project rewarded positive feedback by displaying the most liked works on the home page.

Keywords: Web, Back-end, Front-end.

İÇİNDEKİLER

ETERNALİB WEB PROJECT

KABUL VE ONAY	i
ÖNSÖZ / TEŞEKKÜR	ii
ÖZET	iii
ABSTRACT	iv
İÇİNDEKİLER	v
TABLOLAR LİSTESİ	x
ŞEKİLLER LİSTESİ	xi
KISALTMALAR	xiii

GİRİŞ	1
-----------------	---

1. BÖLÜM Back-End

1.1. Django	5
1.1.1. Model-View-Controller (MVC)	6
1.1.2. Veritabanı Bağımsızlığı	6
1.1.3. Yönetim Kolaylığı	6
1.1.4. Güvenlik	6
1.1.5. Modüler yapı	7
1.1.6. Kullanılan Paketler	7
1.1.6.1. UUID	7
1.1.6.2. json	7
1.1.6.3. user	7
1.2. Rest Framework	8
1.2.1. Proje back-end front-end bağlantısı öncesi	8
1.2.2. Proje back-end front-end bağlantısı sonrası	8

1.2.2.1. Rest Framework Response	9
1.2.2.2. Serializer	9
1.2.2.3. apiview	9
1.2.3. Movie App	10
1.2.3.1. Top 10 Film Fonksiyonu	10
1.2.3.2. Film Listeleme Fonksiyonu	10
1.2.3.3. Film Kategori Fonksiyonu	11
1.2.3.4. Film Yorumları Fonksiyonu	11
1.2.3.5. Film Beğeni Fonksiyonu	12
1.2.3.6. ID ile Film Fonksiyonu	12
1.2.3.7. ID ile Film Kategorileri Fonksiyonu	13
1.2.3.8. Urlname ile Film Fonksiyonu	13
1.2.3.9. Film Yorum Yapma	14
1.2.3.10. Film Beğeni Yapma	15
1.2.3.11. ID ile Film Yorumu Fonksiyonu	16
1.2.3.12. ID ile Film Beğeni Fonksiyonu	16
1.2.3.13. Kullanıcı ID ile Beğeni Fonksiyonu	17
1.2.3.14. Kullanıcı ID ile Beğeni Kapsamlı Fonksiyon	17
1.2.3.15. ID ile Kategoriler Kapsamlı Fonksiyon	18
1.2.3.16. Yorum Silme Fonksiyonu	19
1.2.4. Game App	20
1.2.4.1. Top 10 Oyun Fonksiyonu	20
1.2.4.2. Oyun Listeleme Fonksiyonu	20
1.2.4.3. Oyun Kategori Fonksiyonu	20
1.2.4.4. Oyun Yorumları Fonksiyonu	20
1.2.4.5. Oyun Beğeni Fonksiyonu	20
1.2.4.6. ID ile Oyun Fonksiyonu	21
1.2.4.7. ID ile Oyun Kategorileri Fonksiyonu	21

1.2.4.8. Urlname ile Oyun Fonksiyonu	21
1.2.4.9. Oyuna Yorum Yapma	21
1.2.4.10. Oyun Beğeni Yapma	21
1.2.4.11. ID ile Oyun Yorumu Fonksiyonu	22
1.2.4.12. ID ile Oyun Beğeni Fonksiyonu	22
1.2.4.13. Kullanıcı ID ile Beğeni Fonksiyonu	22
1.2.4.14. Kullanıcı ID ile Beğeni Kapsamlı Fonksiyon	22
1.2.4.15. ID ile Kategoriler Kapsamlı Fonksiyon	22
1.2.4.16. Yorum Silme Fonksiyonu	22
1.2.5. Book App	23
1.2.5.1. Top 10 Kitap Fonksiyonu	23
1.2.5.2. Kitap Listeleme Fonksiyonu	23
1.2.5.3. Kitap Kategori Fonksiyonu	23
1.2.5.4. Kitap Yorumları Fonksiyonu	23
1.2.5.5. Kitap Beğeni Fonksiyonu	23
1.2.5.6. ID ile Kitap Fonksiyonu	24
1.2.5.7. ID ile Kitap Kategorileri Fonksiyonu	24
1.2.5.8. Urlname ile Kitap Fonksiyonu	24
1.2.5.9. Kitaba Yorum Yapma	24
1.2.5.10. Kitaba Beğeni Yapma	24
1.2.5.11. ID ile Kitap Yorumu Fonksiyonu	25
1.2.5.12. ID ile Kitap Beğeni Fonksiyonu	25
1.2.5.13. Kullanıcı ID ile Beğeni Fonksiyonu	25
1.2.5.14. Kullanıcı ID ile Beğeni Kapsamlı Fonksiyon	25
1.2.5.15. ID ile Kategoriler Kapsamlı Fonksiyon	25
1.2.5.16. Yorum Silme Fonksiyonu	25
1.2.6. Account App	26
1.2.6.1. Kullanıcı Bilgilerini Veren Fonksiyon	26

1.2.6.2. ID verilen Kullanıcı Fonksiyonu	27
1.2.6.3. Username Verilen Kullanıcı Fonksiyonu	27
1.2.6.4. Kullanıcı Bilgilerini Güncelleme Fonksiyonu	28
1.2.6.5. Kullanıcı Giriş Fonksiyonu	29
1.2.6.6. Kullanıcı Kayıt Fonksiyonu	29
1.2.6.7. Kullanıcı Çıkış Fonksiyonu	30
1.2.6.8. Kullanıcı Silme Fonksiyonu	30
1.2.6.9. Kullanıcı Şifre Değiştirme Fonksiyonu	31
1.2.6.10. Profil Resmi Değiştirme Fonksiyonu	31
1.2.7. Arama Fonksiyonu	32
2. BÖLÜM	
Front-End	
2.1. Angular	34
2.1.1. Html	35
2.1.2. CSS	35
2.1.3. JavaScript	35
2.1.4. TypeScript	36
2.1.5. Kullanılan kütüphaneler ve paketler	36
2.1.5.1. JQuerry	36
2.1.5.2. Popper.js	36
2.1.5.3. Bootstrap	37
2.1.5.4. fontawesome	37
2.1.5.5. ngx-toastr	37
2.1.5.6. ngx-sweetalert2	38
2.1.5.7. ngx-spinner	38
2.1.6. Generic Http Service	38
2.1.7. Generic components	39

3. BÖLÜM
Veritabanı

3.1. Veritabanı	40
3.1.1. MongoDB	40
3.1.2. Sqlite	41
3.1.3. MSsql	41
4. BÖLÜM	
TARTIŞMA, SONUÇ ve ÖNERİLER	
4.1. Tartışma, Sonuç ve Öneriler	42
4.1.1. Tartışma	42
4.1.2. Sonuç	45
4.1.3. Öneriler	45
KAYNAKLAR	46
EKLER	46
ÖZGEÇMİŞ	47

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

Şekil 1.1.	Top 10 movie	10
Şekil 1.2.	Movies	10
Şekil 1.3.	Movie Categories	11
Şekil 1.4.	Movie Comments	11
Şekil 1.5.	Movie Likes	12
Şekil 1.6.	Movie ID	12
Şekil 1.7.	Movie Category ID	13
Şekil 1.8.	Movie Urlname	13
Şekil 1.9.	Movie Add Comment	14
Şekil 1.10.	Movie Add Like	15
Şekil 1.11.	Movie Comments	16
Şekil 1.12.	Movie Like	16
Şekil 1.13.	User Like	17
Şekil 1.14.	User Like	17
Şekil 1.15.	Category	18
Şekil 1.16.	Comment delete	19
Şekil 1.17.	Tüm Kullanıcı Bilgileri	26
Şekil 1.18.	Kullanıcı Bilgileri	27
Şekil 1.19.	Kullanıcı Adı ile Kullanıcı Bilgileri	27
Şekil 1.20.	Kullanıcı Bilgilerini Günceller	28
Şekil 1.21.	Kullanıcı Giriş	29
Şekil 1.22.	Kullanıcı Kaydı	29

Şekil 1.23. Kullanıcı Çıkışı	30
Şekil 1.24. Kullanıcı Silme	30
Şekil 1.25. Kullanıcı Şifre Değiştirme	31
Şekil 1.26. Kullanıcı Profil Resmini Değiştirme	31
Şekil 1.27. Arama Yapma	32
Şekil 2.1. Profil Ekranı	33
Şekil 2.2. Movie Page	34
Şekil 2.3. Movie Detail Page	34
Şekil 2.4. Book Page	39

KISALTMALAR

<i>BE</i>	:	Back-End
<i>FE</i>	:	Front-End
<i>DB</i>	:	Veritabanı
<i>XL</i>	:	x-link
<i>IL</i>	:	instagram-link
<i>FL</i>	:	facebook-link
<i>LL</i>	:	linkedn-link
<i>FN</i>	:	full-name

GİRİŞ

Günümüzde kartopu etkisiyle artan veriler sebebiyle insanların kendi zevklerine göre içerik bulma süreleri giderek artmaktadır . Projemiz , kullanıcıların tek bir çatı altında , kolay ve hızlı bir şekilde tüketmek istedikleri film , oyun ve kitap içeriklerini bulmaları konusunda kaynak niteliğindedir. Projenin temel amacı insanların tüketecekleri içerik konusunda tanıtımları vasıtasyyla temel bir özet bilgisine sahip olmaları , insanların ilgili içeriklerine yaptıkları yorumlar ve puanlamalar sayesinde fikir oluşturmalarıdır. Sitemizde bulunan içeriklerden kullanıcılar açısından en fazla beğenilen içerikler , erişim kolaylığı amaçlandığından anasayfada gözükmektedir . Bu sayede kullanıcı hızlı bir bakış sayesinde , kendi zevkine de uyarsa , tüketmek istediği içeriği hızlı bir şekilde bulabiliyor. Sitede bulunan tüm içeriklerin IMBD ve Metacritic gibi dünya çapında kabul görmüş bilgi bankalarının değerlendirmeleri mevcuttur .Kullanıcı , diğer kullanıcıların değerlendirmeleri yerine bu bilgi bankalarının değerlendirmelerine odaklanarak da içerik arayabiliyor.

Günümüzün hızlı dünyasında daha önce tükettiğimiz ama ileride yeniden tüketmek istediğimiz içerikleri unutuyoruz . Tam bu noktada projenin kullanıcılar açısından diğer büyük bir avantajı , tüketip beğendikleri içerikleri dijital bir ortamda kayıt altında tutabiliyor olmalarıdır. Bu sayede beğendikleri içerikleri tekrardan hatırlayıp izleyebiliyor ve paylaşmak istedikleri insanlarla hızlı bir şekilde paylaşabiliyorlar.

Literatür Özeti

Onur TURGUT (2023) tarafından gerçekleştirilen araştırmada kullanıcıların bir sonraki seyredenekleri filmleri seçerken “letterboxd” gibi film keşif ağlarını ne derece kullandığını ve keşif ağlarının , internetin gelişimi ve geleneksel yöntemlerden dijital sosyal platformlara nasıl kaydıklarını anlatıyor. IMDB , Metacritic , Rotten Tomatoes gibi ağlardaki kullanıcıların yaptıkları görüş , inceleme ve eleştirilerin profesyonel film eleştirmenlerinin yaptıkları incelemelerden daha etkili olduğunu belirtiyor. Sayısal veriler de bu görüşü destekliyor. “letterboxd” dünya çapında film keşif websitesidir ve bu website global olarak en fazla trafik çeken 998. Sitedir , aylık ortalama 33,2 milyon trafik çekmektedir.

“goodreads” isimli site dünya çapında bir kitap keşif ve eleştiri sitesidir. Sayısal verilere bakarak insanların kitap seçimlerinde ne derece etkili olduğunu anlayabiliyoruz . “goodreads” isimli site dünya çapıyla en fazla trafik çeken 367. Sitedir ve aylık ortalama 105.3 milyon kullanıcı çekmektedir.

IMDB bilindiği gibi film derecelendirme sitesidir ve bu sitedeki trafik dünya çapında 71.sıradadır . Aylık ortalama 490 milyon trafik çekmektedir.

Sayısal verilere ve yapılan araştırmalara bakarak insanların içerik (film , kitap ve oyun) seçimlerinde insanların yorum ve puanlandırma yapabilecekleri keşif websitelerinin önemini anlayabiliyoruz. Günümüzde sayısı günden güne artan içerikler kullanıcıların hangisini tüketeceği konusunda kararsızlığa sürüklendekte ve bazen hangi yeni kitabı okunması gereği konusunda onlarca dakikalarımız gitmektedir. Keşif sitelerinin önemi bu noktada devreye girmektedir.

Çalışmanın Amacı

Bu çalışmasının amaçları şu şekilde sıralanabilir:

- Projemizin temel amacı , kullanıcıların yeni bir içerik (film , kitap ve oyun) tüketmek istediklerinde fakat hangi içerik olduğunu bilmedikleri durumlarda , karar verme sürecinde onlara destek olmakdır. Tüketmek istediği içerikleri ararken hızlı karar verebilmek amacıyla , içeriklerin temel bir özeti , websitemizdeki kullanıcıların ve dünya çapında değerlendirme kuruluşlarının verdikleri puanları , sitedeki kullanıcıların yaptıkları yorumları görüp nihayetinde kendi zevkine göre değerlendirip en son nihai kararı vermelerini sağlıyoruz.
- İçerik keşif sürecini kolaylaştırıyoruz.Websitemizde bulunan değerlendirme ve yorumlar sayesinde kullanıcıların içerik bulma süreçlerini hızlandırıyoruz. Oluşturduğumuz kategorizasyon sistemi ve arama sayesinde kullanıcıların ihtiyaçlarına uygun içeriklerinin listelenmesini sağlıyoruz.
- Kullanıcıların etkileşimde bulunmasını sağlıyoruz.Benzer zevke sahip kullanıcıların aynı içerik hakkında değerlendirme yapma ve o içerik hakkında tartışma yapmalarına imkan sağlıyoruz bu sayede bir topluluk oluşturuyoruz. Bu topluluğun oluşturduğu bilgi birikimi , diğer kullanıcıların karar verme süreçlerine olumlu yönde katkı veriyor.
- Güvenilir değerlendirme kaynakları sunuyoruz.Metacritic ve IMDB gibi güvenilir bilgi kaynaklarından alınan değerlendirmeleri kullanıcılar ile paylaşıyoruz. Bu sayede karar verme süreçlerini azaltmayı umuyoruz.
- Kullanıcıların beğenikleri içerikleri kayıt altına almalarını sağlıyoruz.Kullanıcıların beğenikleri her türlü içeriği ilerde hatırlamak ve/veya dostlarıyla paylaşmak için kayıt altına alınmasını sağlıyoruz. Kullanıcılar kullanıcı dostu kolay bir arayüz ile üç farklı kategoride – kitap , oyun , müzik – beğenikleri içerikleri websitemiz üzerinde kayıt altına almalarını sağlıyoruz.

Tezin Organizasyonu

Bu çalışmanın düzenlemesi şu şekildedir.

1. Bölümde Back-end Hakkında,
2. Bölümde Front-end Hakkında,
3. Bölümde Veritabanı Hakkında,

1. BÖLÜM

Back-End

Back-End başlangıç olarak projemizde Django ile geliştirmeye başladık. Proje ilerlemesi ile bazı framework eklemeleri ile projemizi genişlettik. Django kısaca Python ile geliştirilebilen yüksek seviyeli bir web geliştirme aracıdır. Getirdiği bazı kolaylıklar (admin paneli gibi...) ile proje geliştirme sürecini oldukça hızlandırmıştır. Back-end geliştirme sürecinde çevik bir geliştirme modeli kullandık.

1.1. Django

Django python gibi yüksek seviye bir programlama dili kullanması ile popülerite kazanmıştır. Python kolay yazımı ve esnekliği Django için web alanında etki sağlamaşını kolaylaştırdı. Django kullanıcıı için güçlü bir altyapı sunar, kullanıcı ise bu altyapıyı ihtiyaçlarına uygun olarak düzenlemektedir.

1.1.1. Model-View-Controller (MVC)

Bu anlayış Django için kullanımını oldukça kolaylaştırır. Model kısmında veritabanı ile olan ilişkiler düzenlenmektedir. View ise kullanıcıya sunulan web kısmını sağlar. Controller ise gerekli düzenlemeleri ayarladığımız kısımdır. Bu anlayış kullanıcının mevcut yapıyı kavramasını oldukça kolaylaştırır.

1.1.2. Veritabanı Bağımsızlığı

Django kullanıcısını belli bir veritabanını kullanmayı zorunlu kılmaz. Bu anlayış ile bir çok veritabanına desteği bulunmaktadır. Kullanıcının istediği zaman veritabanları arası geçiş yapabilmesi ve bu geçişte migrate yapısı nedeniyle mevcut modelleri kaybetmeden yapabilmektedir.

1.1.3. Yönetim Kolaylığı

Django sağladığı admin paneli sayesinde kullanıcıları istediği gibi yönetebilmemizi sağlamakta. ayrıca gerekli ayarları sağlarsanız modeli serializera bağlayarak admin panelinden veri eklemesi yapabilmektesiniz. Birçok işlemi bir panelde kolay bir şekilde kullanabilmekteyiz.

1.1.4. Güvenlik

Django web alanındaki genel saldırılar için gerekli önlemleri almış durumdadır. Sql Enfection ve XSS gibi saldırırlara karşı önlemleri bulunmaktadır. Ayrıca Token sistemi ve csrf önlemleri ile mevcut yapıda kullanıcısını güvenli bir web sitesi kurulumuna teşvik etmektedir.

1.1.5. Modüler yapı

Django mevcut yapısı projeyi küçük app yapılarına bölerek birçok farklı yapıyı bir arada tutmayı başarır. Bu yapı ayrıca proje karmaşaklığını ve ileride oluşacak yönetim zorluğu, aşırı genişlemiş proje gibi sorunları önlemektedir.

1.1.6. Kullanılan Paketler

Django içerisinde veya dahil edilen paketler ile projemizi güçlendirdik.

1.1.6.1. UUID

Mevcut modellerde otomatik id tanımlanır kullanıcı isterse bu idleri özelleştirmeside sağlanır. Ancak mevcut id tanımı benzersizlik içermez. Bu durum ise modüller arası id çakışmalarına neden olmaktadır. Bu durumu önlemek için id yapısını otomatik bir UUID yapısına getirmek bize benzersiz id'ler sağlamaktadır. Bu şekilde tanımladığımız bir id aramasında modüller arası karmaşayı önlemekteyiz.

1.1.6.2. json

Json, Django içerisinde aslında veri iletimi ve almında sıklıkla kullanmaktayız. REST Framework'ünün çalışmasında kullanılması ile Django içerisinde kullanılması önemli olan bir pakettir.

1.1.6.3. user

Django'da admin paneli gibi user kısmında kendiliğinden gelmektedir. Ancak biz mevcut projemizde user modeli yeterli olmadığı için özelleştirmeye gittik. Mevcut yapısını düzenleyerek kendi projemiz için özel CustomUser modeli kurduk. Temeli hala user olsada mevcut yapısını oldukça genişletmiş bulunmaktayız.

1.2. Rest Framework

Rest Framework bize getirdiği bazı kolaylıklar için projeye dahil edilmiştir. Öncelikle back-end ile front-end projeleri birleştirilmeden önce ve sonrası için proje ihtiyaçlarını karşılamak için eklenmiştir.

1.2.1. Proje back-end front-end bağlantısı öncesi

Bağlantı öncesinde back-end ile geliştirilen kodların testi için static sayfa ve template oluşturmak back-end geliştiricisi için zaman alacağı için Rest'in sağladığı imkanlar ile kodların çalışması test edilmiş ve geliştirme süreci hızlanmıştır. Rest ile ayrıca veri akışındaki durumu kontrol etmek, veri yapısında oluşan olumsuzlukları gidermek ve veri modelini tekrardan ele alınması gereken durumları yönetilmesinde oldukça yardımcı oldu.

1.2.2. Proje back-end front-end bağlantısı sonrası

Front-end geliştirilmesi ile birlikte front-end ile gelen verinin işlenmesi ve güvenlik için token üretimlerinde kullanıldı. Csrf ,sessionID giriş çıkış için token gibi konuları yönetiminde kullanıldı.

1.2.2.1. Rest Framework Response

Response ile kullanıcıya front-end ile gösterilecek hatayı back-end sürecinde tespit edip front-end ile hatayı kullanıcıya bildirmek için kullandığımız yapı.

1.2.2.2. Serializer

Serializer kullanımı ile Django ile tanımlanmış modeli uygun json verisine dönüştürüp veri tabanından gelen veriyi gerekli başlıklar altında toplamamızı sağlıyor.

1.2.2.3. apiview

Bu yapı bize gelen response içeriğine nasıl bir dönüş sağlamamız gerektiğini yönetmemizi sağlıyor. Bazı özel durumlarda iki durum (POST,GET) için POST için ayrı GET için ayrı dönüt ihtiyacını tek fonksiyonda yönetmemizi sağlıyor.

1.2.3. Movie App

Film model,url ve views gibi djangonun kendine has özelliklerine sahip bir şekilde ayrılmış bölüm. Bu ayırma karışıklığı ve temiz kod yazımını sağlamak için yapıldı.

1.2.3.1. Top 10 Film Fonksiyonu

En çok beğenilen 10 filmi getiren metod.Dislike sıralamada etkisi yoktur. GET isteği beklemektedir. Url erişimi 'movies/list/top' ile olmaktadır.



```

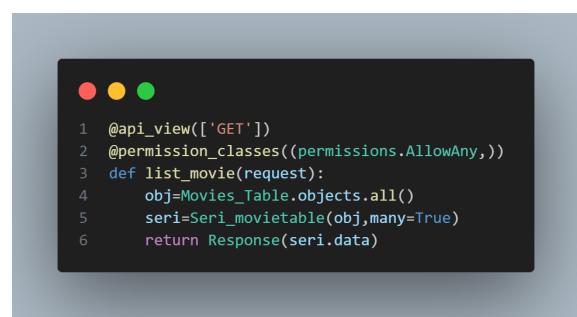
1 @api_view(['GET'])
2 @permission_classes((permissions.AllowAny,))
3 def list_mostmovies(request):
4     top_5_movies = Movies_Table.objects.order_by('-like')[:10]
5     top_5_movie_ids = top_5_movies.values_list('id', flat=True)
6     return JsonResponse(list(top_5_movie_ids), safe=False)

```

Şekil 1.1. Top 10 movie

1.2.3.2. Film Listeleme Fonksiyonu

Tüm filmleri bilgileri ile listeleyen fonksiyondur.GET isteği beklemektedir. Url erişimi '/movies' ile olmaktadır.



```

1 @api_view(['GET'])
2 @permission_classes((permissions.AllowAny,))
3 def list_movie(request):
4     obj=Movies_Table.objects.all()
5     seri=Seri_movietable(obj,many=True)
6     return Response(seri.data)

```

Şekil 1.2. Movies

1.2.3.3. Film Kategori Fonksiyonu

Filmle ait tüm kategori bilgilerini verir. GET isteği beklemektedir. Url erişimi 'movies/category' ile olmaktadır.



```
● ● ●
1 @api_view(['GET'])
2 @permission_classes((permissions.AllowAny,))
3 def list_moviecategory(request):
4     obj=Movies_Category.objects.all()
5     seri=Seri_moviecategory(obj,many=True)
6     return Response(seri.data)
7
```

Şekil 1.3. Movie Categories

1.2.3.4. Film Yorumları Fonksiyonu

Tüm filmlere yapılmış yorumların bilgilerini iletir. GET isteği beklemektedir. Url erişimi 'movies/comment' ile olmaktadır



```
● ● ●
1 @api_view(['GET'])
2 @permission_classes((permissions.AllowAny,))
3 def list_moviecomment(request):
4     obj=Movies_Comment.objects.all()
5     seri=Seri_moviecomment(obj,many=True)
6     return Response(seri.data)
```

Şekil 1.4. Movie Comments

1.2.3.5. Film Beğeni Fonksiyonu

Beğenilen filmler ile ilgili bilgileri verir. GET isteği beklemektedir. Url erişimi 'movies/like' ile olmaktadır.



```

● ● ●
1 @api_view(['GET'])
2 @permission_classes((permissions.AllowAny,))
3 def list_movielike(request):
4     obj=Movies_Like.objects.all()
5     seri=Seri_movielike(obj,many=True)
6     return Response(seri.data)

```

Şekil 1.5. Movie Likes

1.2.3.6. ID ile Film Fonksiyonu

ID verilen filmin bilgilerini veren fonksiyondur. GET isteği beklemektedir. Url erişimi 'movies/id/<uuid:id>' ile olmaktadır. '/id/' sonrası filmin id bilgisi gelmelidir.



```

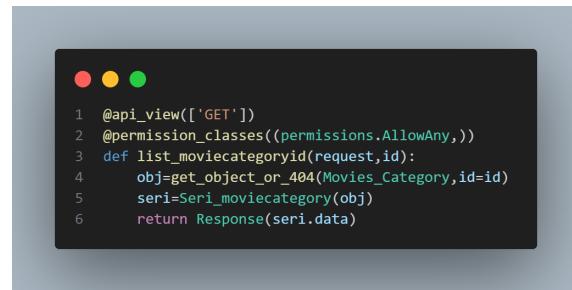
● ● ●
1 @api_view(['GET'])
2 @permission_classes((permissions.AllowAny,))
3 def list_movieid(request,id):
4     obj=get_object_or_404(Movies_Table,id=id)
5     seri=Seri_movietable(obj)
6     return Response(seri.data)
7

```

Şekil 1.6. Movie ID

1.2.3.7. ID ile Film Kategorileri Fonksiyonu

ID verilen film kategorisinin bilgilerini verir. GET isteği beklemektedir. Url erişimi 'movies/get/id/category/<uuid:id>' ile olmaktadır. '/category/' sonrasında film kategorisinin id bilgileri gelmelidir.



```

● ● ●
1 @api_view(['GET'])
2 @permission_classes((permissions.AllowAny,))
3 def list_moviecategoryid(request,id):
4     obj=get_object_or_404(Movies_Category,id=id)
5     seri=Seri_moviecategory(obj)
6     return Response(seri.data)

```

Şekil 1.7. Movie Category ID

1.2.3.8. Urlname ile Film Fonksiyonu

Urlname verilen filme ait bilgileri verir. GET isteği beklemektedir. Url erişimi 'movies/username/<str:username>' ile olmaktadır. '/username/' sonrasında string formatında filme ait url ismi bilgisi gelmelidir.



```

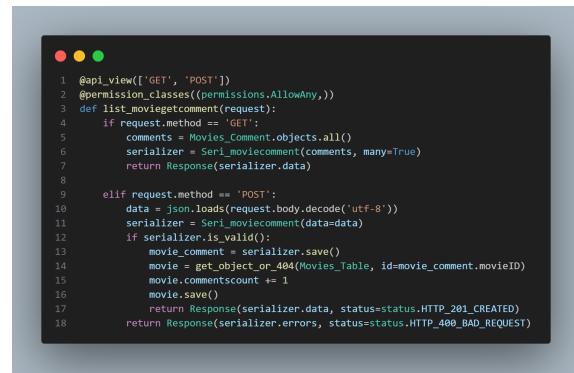
● ● ●
1 @api_view(['GET'])
2 @permission_classes((permissions.AllowAny,))
3 def list_movieusername(request,urlname):
4     obj=get_object_or_404(Movies_Table,urlname=urlname)
5     seri=Seri_movietable(obj)
6     return Response(seri.data)

```

Şekil 1.8. Movie Urlname

1.2.3.9. Film Yorum Yapma

ID verilen filme ait bilgileri bekler. GET isteği alduğunda tüm yorumları döndürür. POST allığında movieID,userID ve comment bilgilerini bekler. Url erişimi 'movies/add/comment' ile olmaktadır.



```

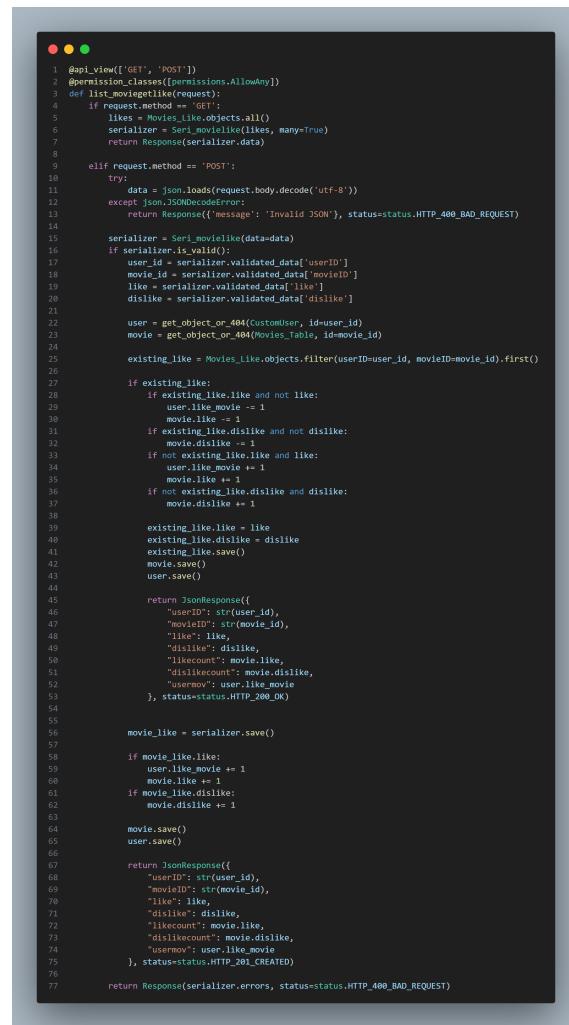
1 @api_view(['GET', 'POST'])
2 @permission_classes([permissions.AllowAny])
3 def list_moviecomment(request):
4     if request.method == 'GET':
5         comments = Movies_Comment.objects.all()
6         serializer = Seri_moviecomment(comments, many=True)
7         return Response(serializer.data)
8
9     elif request.method == 'POST':
10        data = json.loads(request.body.decode('utf-8'))
11        serializer = Seri_moviecomment(data=data)
12        if serializer.is_valid():
13            movie_comment = serializer.save()
14            movie = get_object_or_404(Movies_Table, id=movie_comment.movieID)
15            movie.commentscount += 1
16            movie.save()
17            return Response(serializer.data, status=status.HTTP_201_CREATED)
18        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

Sekil 1.9. Movie Add Comment

1.2.3.10. Film Beğeni Yapma

ID verilen filme ait bilgileri bekler. GET isteği tüm beğenileri döndürür. POST alduğında movieID,userID,like,dislike bilgilerini bekler. Url erişimi 'movies/add/like' ile olmaktadır.



```

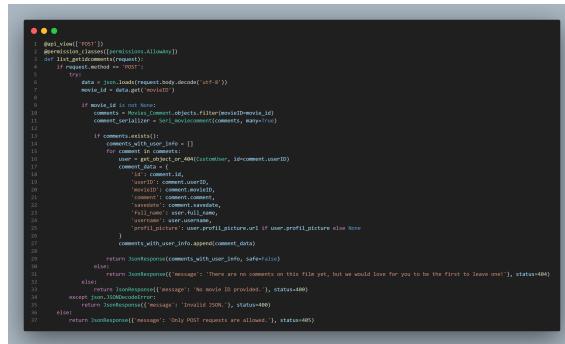
1 @api_view(['GET', 'POST'])
2 @permission_classes([permissions.AllowAny])
3 def list_moviegetlike(request):
4     if request.method == "GET":
5         likes = Movies_Like.objects.all()
6         serializer = Seri_movieLike(likes, many=True)
7         return Response(serializer.data)
8
9     elif request.method == "POST":
10        try:
11            data = json.loads(request.body.decode('utf-8'))
12            except json.JSONDecodeError:
13                return Response({'message': 'Invalid JSON'}, status=status.HTTP_400_BAD_REQUEST)
14
15        serializer = Seri_movieLike(data=data)
16        if serializer.is_valid():
17            user_id = serializer.validated_data['userID']
18            movie_id = serializer.validated_data['movieID']
19            like = serializer.validated_data['like']
20            dislike = serializer.validated_data['dislike']
21
22            user = get_object_or_404(CustomUser, id=user_id)
23            movie = get_object_or_404(Movies_Table, id=movie_id)
24
25            existing_like = Movies_Like.objects.filter(userID=user_id, movieID=movie_id).first()
26
27            if existing_like:
28                if existing_like.like and not like:
29                    user.like_movie -= 1
30                    movie.like -= 1
31                if existing_like.dislike and not dislike:
32                    movie.dislike -= 1
33                if not existing_like.like and like:
34                    user.like_movie += 1
35                    movie.like += 1
36                if not existing_like.dislike and dislike:
37                    movie.dislike += 1
38
39            existing_like.like = like
40            existing_like.dislike = dislike
41            existing_like.save()
42            movie.save()
43            user.save()
44
45            return JsonResponse({
46                "userID": str(user_id),
47                "movieID": str(movie_id),
48                "like": like,
49                "dislike": dislike,
50                "likecount": movie.like,
51                "dislikecount": movie.dislike,
52                "usermov": user.like_movie
53            }, status=status.HTTP_200_OK)
54
55
56            movie_like = serializer.save()
57
58            if movie_like.like:
59                user.like_movie += 1
60                movie.like += 1
61            if movie_like.dislike:
62                movie.dislike += 1
63
64            movie.save()
65            user.save()
66
67            return JsonResponse({
68                "userID": str(user_id),
69                "movieID": str(movie_id),
70                "like": like,
71                "dislike": dislike,
72                "likecount": movie.like,
73                "dislikecount": movie.dislike,
74                "usermov": user.like_movie
75            }, status=status.HTTP_201_CREATED)
76
77        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

Sekil 1.10. Movie Add Like

1.2.3.11. ID ile Film Yorumu Fonksiyonu

ID verilen filme ait tüm yorumları verir. POST isteği movieID verisini bekler. Url erişimi 'movies/get/id/comment' ile olmaktadır.



```

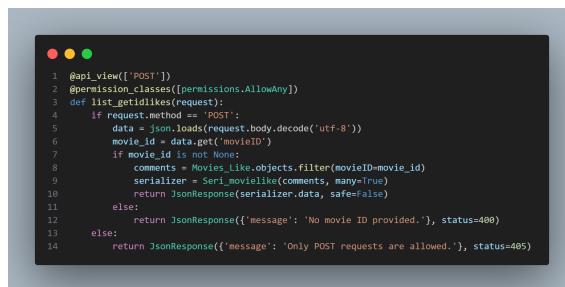
1 @api_view(['POST'])
2 @permission_classes([permissions.AllowAny])
3 def list_getidcomment(request):
4     if request.method == 'POST':
5         data = json.loads(request.body.decode('utf-8'))
6         movie_id = data.get('movieID')
7
8         if movie_id is not None:
9             comments = Movie.objects.filter(movieID=movie_id)
10            comment_serializer = Seri_moviecomment(comments, many=True)
11
12            if comments.exists():
13                comment_data = []
14
15                for comment in comments:
16                    comment_data.append({
17                        'comment_user': comment.user,
18                        'comment_text': comment.comment,
19                        'comment_date': comment.date,
20                        'comment_userID': comment.userID,
21                        'comment_movieID': comment.movieID,
22                        'comment_likes': comment.likes,
23                        'comment_dislikes': comment.dislikes,
24                        'comment_username': comment.username,
25                        'comment_picture': comment.profile_picture.url if user.profile_picture else None
26                    })
27
28                comment_with_user_info = appendcomment_data(comment_data)
29
30                return JsonResponse(comment_with_user_info, safe=False)
31            else:
32                return JsonResponse({'message': 'There are no comments on this file yet, but we would love for you to be the first to leave one!'}, status=404)
33        else:
34            return JsonResponse({'message': 'No movie ID provided.'}, status=400)
35    except:
36        return JsonResponse({'message': 'Invalid JSON.'}, status=400)
37
38    else:
39        return JsonResponse({'message': 'Only POST requests are allowed.'}, status=405)

```

Şekil 1.11. Movie Comments

1.2.3.12. ID ile Film Beğeni Fonksiyonu

ID verilen filme ait beğeni bilgilerini verir. POST isteği movieID verisini bekler. Url erişimi 'movies/get/mid/like' ile olmaktadır.



```

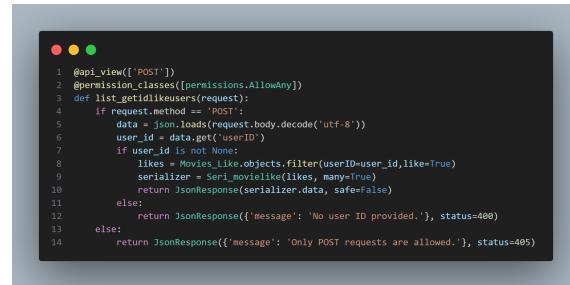
1 @api_view(['POST'])
2 @permission_classes([permissions.AllowAny])
3 def list_getidlikes(request):
4     if request.method == 'POST':
5         data = json.loads(request.body.decode('utf-8'))
6         movie_id = data.get('movieID')
7
8         if movie_id is not None:
9             comments = Movies_Like.objects.filter(movieID=movie_id)
10            serializer = Seri_movieLike(comments, many=True)
11
12            return JsonResponse(serializer.data, safe=False)
13        else:
14            return JsonResponse({'message': 'No movie ID provided.'}, status=400)
15
16    else:
17        return JsonResponse({'message': 'Only POST requests are allowed.'}, status=405)

```

Şekil 1.12. Movie Like

1.2.3.13. Kullanıcı ID ile Beğeni Fonksiyonu

ID verilen kullanıcıya ait beğeni bilgilerini verir. POST isteği userID verisini beklemektedir. Url erişimi 'movies/get/uid/like' ile olmaktadır.



```

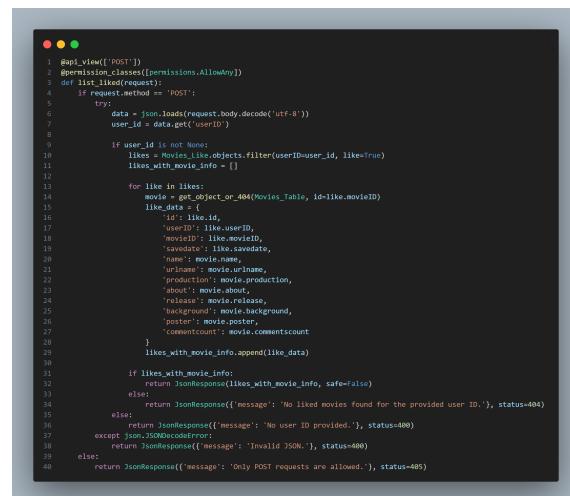
1 @api_view(['POST'])
2 @permission_classes([permissions.AllowAny])
3 def list_getidlikeusers(request):
4     if request.method == 'POST':
5         data = json.loads(request.body.decode('utf-8'))
6         user_id = data.get('userID')
7         if user_id is not None:
8             likes = Movies_Like.objects.filter(userID=user_id, like=True)
9             serializer = Seri_movieLike(likes, many=True)
10            return JsonResponse(serializer.data, safe=False)
11        else:
12            return JsonResponse({'message': 'No user ID provided.'}, status=400)
13    else:
14        return JsonResponse({'message': 'Only POST requests are allowed.'}, status=405)

```

Sekil 1.13. User Like

1.2.3.14. Kullanıcı ID ile Beğeni Kapsamlı Fonksiyon

'Kullanıcı ID ile Beğeni Fonksiyonu' başlığı altında anlattığım fonksiyonun dönüt olarak genişletilmiş hali. Kullanıcı bilgisi ve beğeni bilgileri dışında bu fonksiyon ayrıca film bilgilerini de döndürüyor. Url erişimi 'movies/like/search' ile olmaktadır.



```

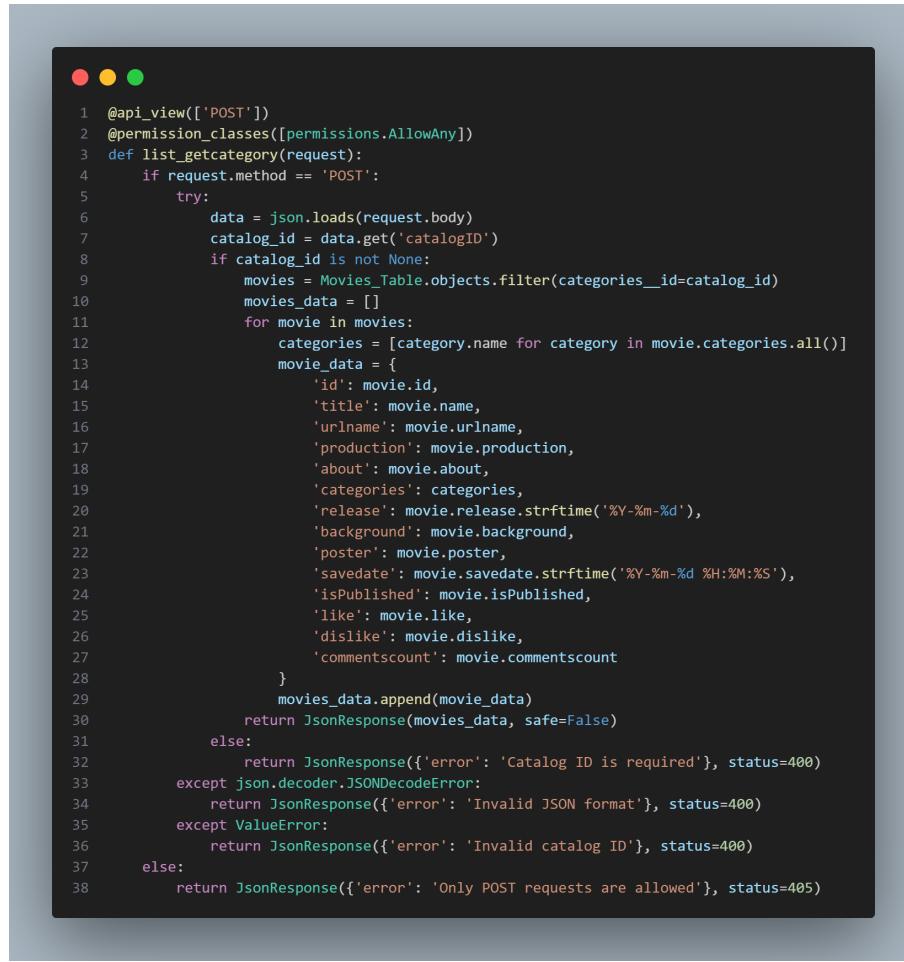
1 @api_view(['POST'])
2 @permission_classes([permissions.AllowAny])
3 def list_likeusers(request):
4     if request.method == 'POST':
5         try:
6             data = json.loads(request.body.decode('utf-8'))
7             user_id = data.get('userID')
8             if user_id is not None:
9                 likes = Movies_Like.objects.filter(userID=user_id, like=True)
10                likes_with_movie_info = []
11
12                for like in likes:
13                    movie = get_object_or_404(Movies_Table, id=like.movieID)
14                    like_data = {
15                        'id': like.id,
16                        'userID': like.userID,
17                        'movieID': like.movieID,
18                        'username': like.username,
19                        'name': movie.name,
20                        'urlname': movie.urlname,
21                        'production': movie.production,
22                        'about': movie.about,
23                        'release': movie.release,
24                        'background': movie.background,
25                        'poster': movie.poster,
26                        'commentCount': movie.commentscount
27                    }
28                    likes_with_movie_info.append(like_data)
29
30                if likes_with_movie_info:
31                    return JsonResponse(likes_with_movie_info, safe=False)
32                else:
33                    return JsonResponse({'message': 'No liked movies found for the provided user ID.'}, status=404)
34
35            else:
36                return JsonResponse({'message': 'No user ID provided.'}, status=400)
37        except json.JSONDecodeError:
38            return JsonResponse({'message': 'Invalid JSON.'}, status=400)
39        else:
40            return JsonResponse({'message': 'Only POST requests are allowed.'}, status=405)

```

Sekil 1.14. User Like

1.2.3.15. ID ile Kategoriler Kapsamlı Fonksiyon

ID verilen Kategori ile ilgili bilgiler ve filmlerin bilgilerini getiriyor. POST isteği catalogID beklemektedir. Url erişimi 'movies/category/get' ile olmaktadır.



```

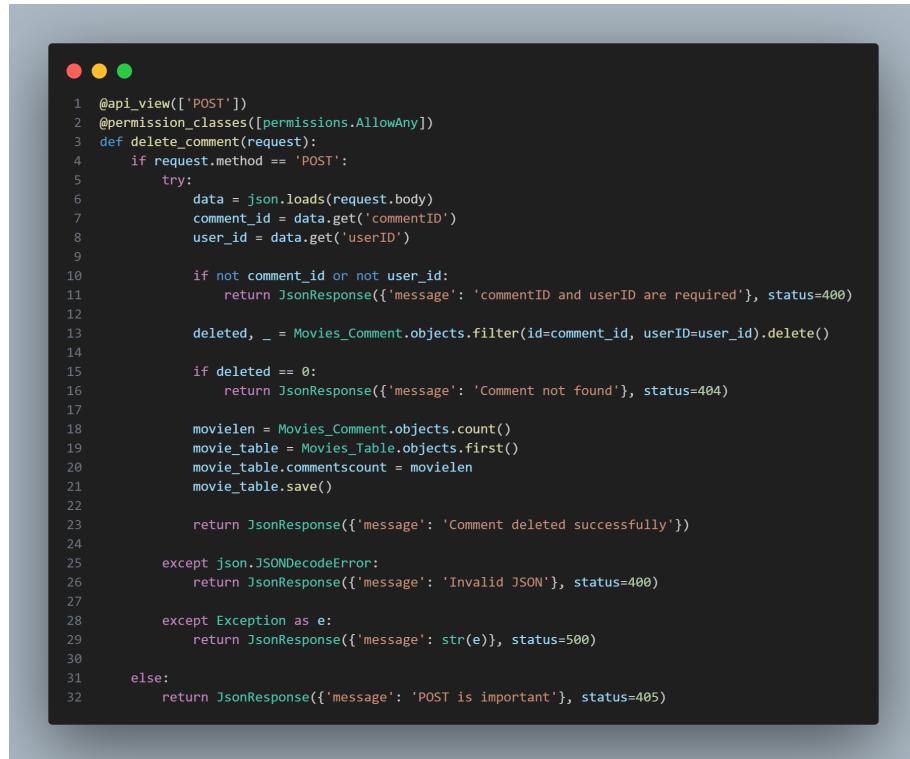
1 @api_view(['POST'])
2 @permission_classes([permissions.AllowAny])
3 def list_getcategory(request):
4     if request.method == 'POST':
5         try:
6             data = json.loads(request.body)
7             catalog_id = data.get('catalogID')
8             if catalog_id is not None:
9                 movies = Movies_Table.objects.filter(categories__id=catalog_id)
10                movies_data = []
11                for movie in movies:
12                    categories = [category.name for category in movie.categories.all()]
13                    movie_data = {
14                        'id': movie.id,
15                        'title': movie.name,
16                        'urlname': movie.urlname,
17                        'production': movie.production,
18                        'about': movie.about,
19                        'categories': categories,
20                        'release': movie.release.strftime('%Y-%m-%d'),
21                        'background': movie.background,
22                        'poster': movie.poster,
23                        'savedate': movie.savedate.strftime('%Y-%m-%d %H:%M:%S'),
24                        'isPublished': movie.isPublished,
25                        'like': movie.like,
26                        'dislike': movie.dislike,
27                        'commentscount': movie.commentscount
28                    }
29                    movies_data.append(movie_data)
30                return JsonResponse(movies_data, safe=False)
31            else:
32                return JsonResponse({'error': 'Catalog ID is required'}, status=400)
33        except json.decoder.JSONDecodeError:
34            return JsonResponse({'error': 'Invalid JSON format'}, status=400)
35        except ValueError:
36            return JsonResponse({'error': 'Invalid catalog ID'}, status=400)
37        else:
38            return JsonResponse({'error': 'Only POST requests are allowed'}, status=405)

```

Sekil 1.15. Category

1.2.3.16. Yorum Silme Fonksiyonu

ID verilen film yorumunu siler. POST isteği commentID beklemektedir. Url erişimi 'movies/comment/delete' ile olmaktadır.



```

1  @api_view(['POST'])
2  @permission_classes([permissions.AllowAny])
3  def delete_comment(request):
4      if request.method == 'POST':
5          try:
6              data = json.loads(request.body)
7              comment_id = data.get('commentID')
8              user_id = data.get('userID')
9
10             if not comment_id or not user_id:
11                 return JsonResponse({'message': 'commentID and userID are required'}, status=400)
12
13             deleted, _ = Movies_Comment.objects.filter(id=comment_id, userID=user_id).delete()
14
15             if deleted == 0:
16                 return JsonResponse({'message': 'Comment not found'}, status=404)
17
18             movieLen = Movies_Comment.objects.count()
19             movie_table = Movies_Table.objects.first()
20             movie_table.commentscount = movieLen
21             movie_table.save()
22
23             return JsonResponse({'message': 'Comment deleted successfully'})
24
25         except json.JSONDecodeError:
26             return JsonResponse({'message': 'Invalid JSON'}, status=400)
27
28         except Exception as e:
29             return JsonResponse({'message': str(e)}, status=500)
30
31     else:
32         return JsonResponse({'message': 'POST is important'}, status=405)

```

Şekil 1.16. Comment delete

1.2.4. Game App

Oyun ile ilgili işlemlerin ve verilerin tutulduğu app.

1.2.4.1. Top 10 Oyun Fonksiyonu

En çok beğenilen 10 oyunu getiren metod. Dislike sıralamada etkisi yoktur. GET isteği beklemektedir. Url erişimi 'games/list/top' ile olmaktadır.

1.2.4.2. Oyun Listeleme Fonksiyonu

Tüm oyunların bilgileri ile listeleyen fonksiyondur. GET isteği beklemektedir. Url erişimi '/games' ile olmaktadır.

1.2.4.3. Oyun Kategori Fonksiyonu

Oyunlara ait tüm kategori bilgilerini verir. GET isteği beklemektedir. Url erişimi 'games/category' ile olmaktadır.

1.2.4.4. Oyun Yorumları Fonksiyonu

Tüm oyunlara yapılmış yorumların bilgilerini iletir. GET isteği beklemektedir. Url erişimi 'games/comment' ile olmaktadır

1.2.4.5. Oyun Beğeni Fonksiyonu

Beğenilen oyunlar ile ilgili bilgileri verir. GET isteği beklemektedir. Url erişimi 'games/like' ile olmaktadır.

1.2.4.6. ID ile Oyun Fonksiyonu

ID verilen oyunun bilgilerini veren fonksiyondur. GET isteği beklemektedir. Url erişimi 'games/id/<uuid:id>' ile olmaktadır. '/id/' sonrası oyunun id bilgisi gelmelidir.

1.2.4.7. ID ile Oyun Kategorileri Fonksiyonu

ID verilen oyun kategorisinin bilgilerini verir. GET isteği beklemektedir. Url erişimi 'games/get/id/category/<uuid:id>' ile olmaktadır. '/category/' sonrasında oyun kategorisinin id bilgileri gelmelidir.

1.2.4.8. Urlname ile Oyun Fonksiyonu

Urlname verilen oyuna ait bilgileri verir. GET isteği beklemektedir. Url erişimi 'games/username/<str:username>' ile olmaktadır. '/username/' sonrasında string formatında oyuna ait url ismi bilgisi gelmelidir.

1.2.4.9. Oyuna Yorum Yapma

ID verilen oyuna ait bilgileri bekler. GET isteği alındığında tüm yorumları döndürür. POST alındığında gameID,userID ve comment bilgilerini bekler. Url erişimi 'games/add/comment' ile olmaktadır.

1.2.4.10. Oyun Beğeni Yapma

ID verilen oyuna ait bilgileri bekler. GET isteği tüm beğenileri döndürür. POST alındığında gameID,userID,like,dislike bilgilerini bekler. Url erişimi 'games/add/like' ile olmaktadır.

1.2.4.11. ID ile Oyun Yorumu Fonksiyonu

ID verilen oyuna ait tüm yorumları verir. POST isteği gameID verisini bekler. Url erişimi 'games/get/id/comment' ile olmaktadır.

1.2.4.12. ID ile Oyun Beğeni Fonksiyonu

ID verilen oyuna ait beğeni bilgilerini verir. POST isteği gameID verisini bekler. Url erişimi 'games/get/mid/like' ile olmaktadır.

1.2.4.13. Kullanıcı ID ile Beğeni Fonksiyonu

ID verilen kullanıcıya ait beğeni bilgilerini verir. POST isteği userID verisini beklemektedir. Url erişimi 'games/get/uid/like' ile olmaktadır.

1.2.4.14. Kullanıcı ID ile Beğeni Kapsamlı Fonksiyon

'Kullanıcı ID ile Beğeni Fonksiyonu' başlığı altında anlattığım fonksiyonun dönüt olarak genişletilmiş hali. Kullanıcı bilgisi ve beğenisi bilgileri dışında bu fonksiyon ayrıca oyun bilgilerini de döndürüyor. Url erişimi 'games/like/search' ile olmaktadır.

1.2.4.15. ID ile Kategoriler Kapsamlı Fonksiyon

ID verilen Kategori ile ilgili bilgiler ve oyunların bilgilerini getiriyor. POST isteği catalogID beklemektedir. Url erişimi 'games/category/get' ile olmaktadır.

1.2.4.16. Yorum Silme Fonksiyonu

ID verilen oyunun yorumunu siler. POST isteği commentID beklemektedir. Url erişimi 'games/comment/delete' ile olmaktadır.

1.2.5. Book App

Kitap hakkında işlemlerin tutulduğu app kısmı.

1.2.5.1. Top 10 Kitap Fonksiyonu

En çok beğenilen 10 kitabı getiren metod. Dislike sıralamada etkisi yoktur. GET isteği beklemektedir. Url erişimi 'books/list/top' ile olmaktadır.

1.2.5.2. Kitap Listeleme Fonksiyonu

Tüm kitapların bilgileri ile listeleyen fonksiyondur. GET isteği beklemektedir. Url erişimi '/books' ile olmaktadır.

1.2.5.3. Kitap Kategori Fonksiyonu

Kitaplara ait tüm kategori bilgilerini verir. GET isteği beklemektedir. Url erişimi 'books/category' ile olmaktadır.

1.2.5.4. Kitap Yorumları Fonksiyonu

Tüm kitaplara yapılmış yorumların bilgilerini iletir. GET isteği beklemektedir. Url erişimi 'books/comment' ile olmaktadır

1.2.5.5. Kitap Beğeni Fonksiyonu

Beğenilen kitaplar ile ilgili bilgileri verir. GET isteği beklemektedir. Url erişimi 'books/like' ile olmaktadır.

1.2.5.6. ID ile Kitap Fonksiyonu

ID verilen oyunun bilgilerini veren fonksiyondur. GET isteği beklemektedir. Url erişimi 'books/id/<uuid:id>' ile olmaktadır. '/id/' sonrası kitabın id bilgisi gelmelidir.

1.2.5.7. ID ile Kitap Kategorileri Fonksiyonu

ID verilen kitap kategorisinin bilgilerini verir. GET isteği beklemektedir. Url erişimi 'books/get/id/category/<uuid:id>' ile olmaktadır. '/category/' sonrasında oyun kategorisinin id bilgileri gelmelidir.

1.2.5.8. Urlname ile Kitap Fonksiyonu

Urlname verilen kitaba ait bilgileri verir. GET isteği beklemektedir. Url erişimi 'books/username/<str:username>' ile olmaktadır. '/username/' sonrasında string formatında kitaba ait url ismi bilgisi gelmelidir.

1.2.5.9. Kitaba Yorum Yapma

ID verilen kitaba ait bilgileri bekler. GET isteği alduğunda tüm yorumları döndürür. POST allığında bookID,userID ve comment bilgilerini bekler. Url erişimi 'books/add/comment' ile olmaktadır.

1.2.5.10. Kitaba Beğeni Yapma

ID verilen kitaba ait bilgileri bekler. GET isteği tüm beğenileri döndürür. POST allığında bookID,userID,like,dislike bilgilerini bekler. Url erişimi 'books/add/like' ile olmaktadır.

1.2.5.11. ID ile Kitap Yorumu Fonksiyonu

ID verilen kitaba ait tüm yorumları verir. POST isteği bookID verisini bekler. Url erişimi 'books/get/id/comment' ile olmaktadır.

1.2.5.12. ID ile Kitap Beğeni Fonksiyonu

ID verilen kitaba ait beğeni bilgilerini verir. POST isteği bookID verisini bekler. Url erişimi 'books/get/mid/like' ile olmaktadır.

1.2.5.13. Kullanıcı ID ile Beğeni Fonksiyonu

ID verilen kullanıcıya ait beğeni bilgilerini verir. POST isteği userID verisini beklemektedir. Url erişimi 'books/get/uid/like' ile olmaktadır.

1.2.5.14. Kullanıcı ID ile Beğeni Kapsamlı Fonksiyon

'Kullanıcı ID ile Beğeni Fonksiyonu' başlığı altında anlattığım fonksiyonun dönüt olarak genişletilmiş hali. Kullanıcı bilgisi ve beğeni bilgileri dışında bu fonksiyon ayrıca oyun bilgilerini de döndürüyor. Url erişimi 'books/like/search' ile olmaktadır.

1.2.5.15. ID ile Kategoriler Kapsamlı Fonksiyon

ID verilen Kategori ile ilgili bilgiler ve oyunların bilgilerini getiriyor. POST isteği catalogID beklemektedir. Url erişimi 'books/category/get' ile olmaktadır.

1.2.5.16. Yorum Silme Fonksiyonu

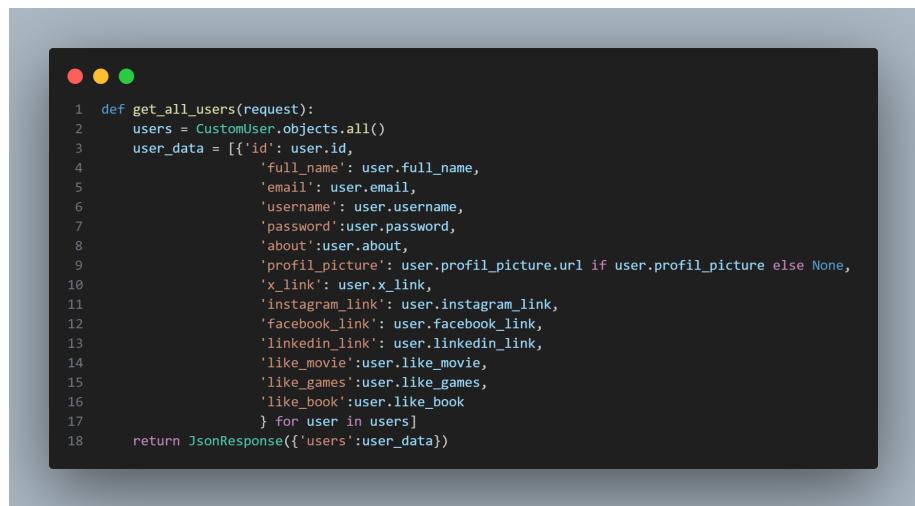
ID verilen kitabın yorumunu siler. POST isteği commentID beklemektedir. Url erişimi 'books/comment/delete' ile olmaktadır.

1.2.6. Account App

Kullanıcı ile ilgili işlemlerin yapıldığı App kısmı. Normalde Djangoda tanımlı olan user modelini özelleştirerek kendi ihtiyaçlarımıza karşılayan yeni bir model oluşturduk.

1.2.6.1. Kullanıcı Bilgilerini Veren Fonksiyon

GET istegidir. Tüm kullanıcıların bilgilerini döndürür. Url erişimi '/accounts' ile olmaktadır.



```

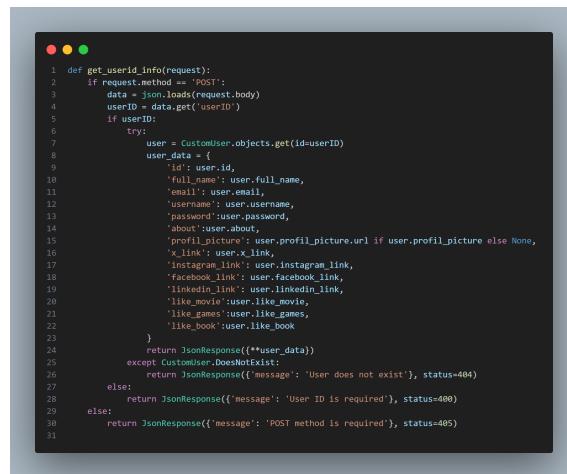
1 def get_all_users(request):
2     users = CustomUser.objects.all()
3     user_data = [{ 'id': user.id,
4                   'full_name': user.full_name,
5                   'email': user.email,
6                   'username': user.username,
7                   'password': user.password,
8                   'about': user.about,
9                   'profil_picture': user.profil_picture.url if user.profil_picture else None,
10                  'x_link': user.x_link,
11                  'instagram_link': user.instagram_link,
12                  'facebook_link': user.facebook_link,
13                  'linkedin_link': user.linkedin_link,
14                  'like_movie': user.like_movie,
15                  'like_games': user.like_games,
16                  'like_book': user.like_book
17                  } for user in users]
18
19 return JsonResponse({ 'users': user_data})

```

Şekil 1.17. Tüm Kullanıcı Bilgileri

1.2.6.2. ID verilen Kullanıcı Fonksiyonu

POST ile userID verisi json formatında beklenmektedir.Url erişimi '/account' ile olmaktadır.



```

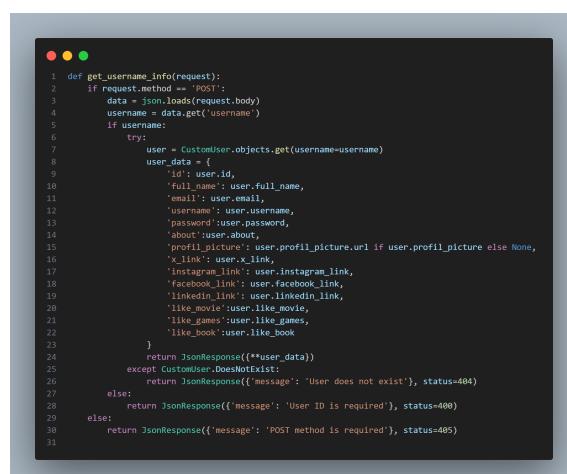
1 def get_userid_info(request):
2     if request.method == "POST":
3         data = json.loads(request.body)
4         userID = data.get('userID')
5         if userID:
6             try:
7                 user = CustomUser.objects.get(id=userID)
8                 user_data = {
9                     'id': user.id,
10                    'full_name': user.full_name,
11                    'email': user.email,
12                    'username': user.username,
13                    'password': user.password,
14                    'about': user.about,
15                    'profil_picture': user.profil_picture.url if user.profil_picture else None,
16                    'x_link': user.x_link,
17                    'instagram_link': user.instagram_link,
18                    'facebook_link': user.facebook_link,
19                    'linkedin_link': user.linkedin_link,
20                    'like_movie': user.like_movie,
21                    'like_games': user.like_games,
22                    'like_book': user.like_book
23                }
24             return JsonResponse(**user_data)
25         except CustomUser.DoesNotExist:
26             return JsonResponse({'message': 'User does not exist'}, status=404)
27     else:
28         return JsonResponse({'message': 'User ID is required'}, status=400)
29     else:
30         return JsonResponse({'message': 'POST method is required'}, status=405)
31

```

Sekil 1.18. Kullanıcı Bilgileri

1.2.6.3. Username Verilen Kullanıcı Fonksiyonu

POST olarak string formatında bir kullanıcı adı beklenmektedir.Url erişimi 'account/username' ile olmaktadır.



```

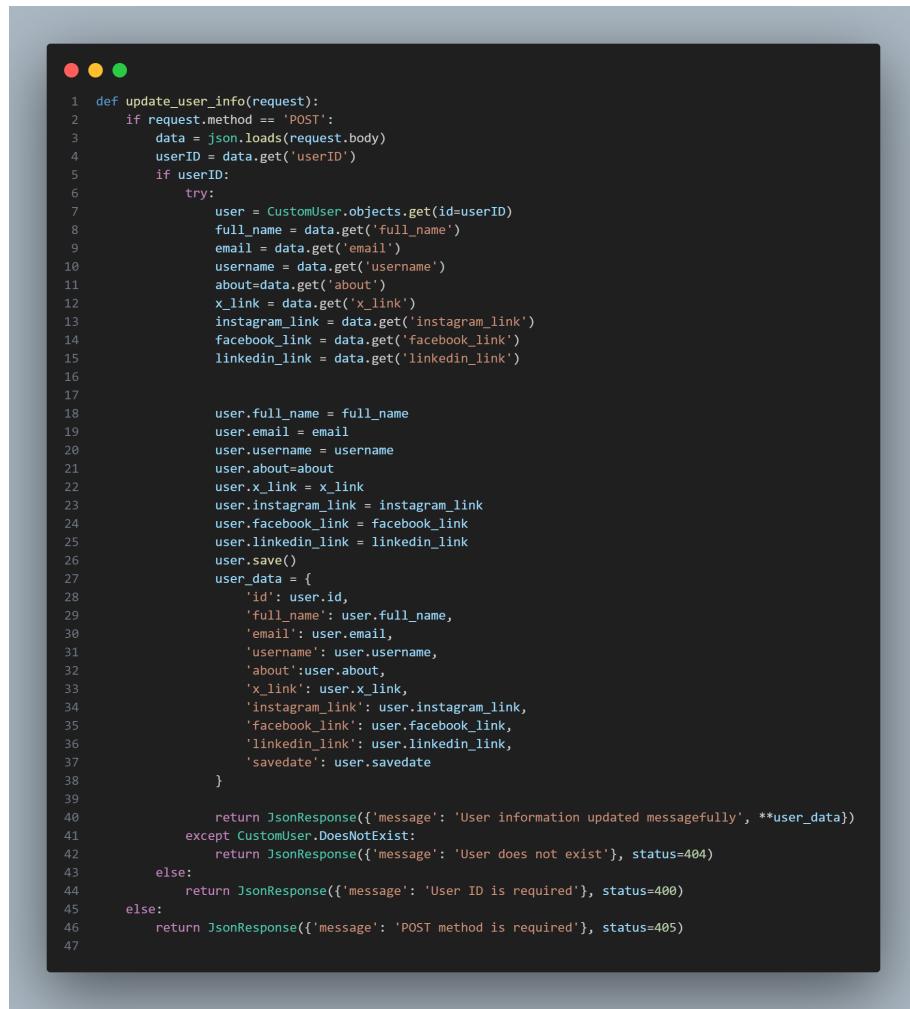
1 def get_username_info(request):
2     if request.method == "POST":
3         data = json.loads(request.body)
4         username = data.get('username')
5         if username:
6             try:
7                 user = CustomUser.objects.get(username=username)
8                 user_data = {
9                     'id': user.id,
10                    'full_name': user.full_name,
11                    'email': user.email,
12                    'username': user.username,
13                    'password': user.password,
14                    'about': user.about,
15                    'profil_picture': user.profil_picture.url if user.profil_picture else None,
16                    'x_link': user.x_link,
17                    'instagram_link': user.instagram_link,
18                    'facebook_link': user.facebook_link,
19                    'linkedin_link': user.linkedin_link,
20                    'like_movie': user.like_movie,
21                    'like_games': user.like_games,
22                    'like_book': user.like_book
23                }
24             return JsonResponse(**user_data)
25         except CustomUser.DoesNotExist:
26             return JsonResponse({'message': 'User does not exist'}, status=404)
27     else:
28         return JsonResponse({'message': 'User ID is required'}, status=400)
29     else:
30         return JsonResponse({'message': 'POST method is required'}, status=405)
31

```

Sekil 1.19. Kullanıcı Adı ile Kullanıcı Bilgileri

1.2.6.4. Kullanıcı Bilgilerini Güncelleme Fonksiyonu

POST olarak alınan userID, *FN*, email, username, about, *XL*, *IL*, *FL*, *LL* verilerini beklemektedir. Alınan veriler ile veritabanındaki veriler değiştirilir. Url erişimi 'account/update' ile olmaktadır.



```

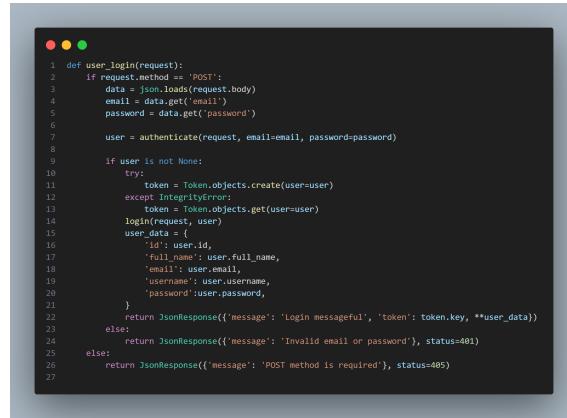
1  def update_user_info(request):
2      if request.method == 'POST':
3          data = json.loads(request.body)
4          userID = data.get('userID')
5          if userID:
6              try:
7                  user = CustomUser.objects.get(id=userID)
8                  full_name = data.get('full_name')
9                  email = data.get('email')
10                 username = data.get('username')
11                 about=data.get('about')
12                 x_link = data.get('x_link')
13                 instagram_link = data.get('instagram_link')
14                 facebook_link = data.get('facebook_link')
15                 linkedin_link = data.get('linkedin_link')
16
17                 user.full_name = full_name
18                 user.email = email
19                 user.username = username
20                 user.about=about
21                 user.x_link = x_link
22                 user.instagram_link = instagram_link
23                 user.facebook_link = facebook_link
24                 user.linkedin_link = linkedin_link
25                 user.save()
26                 user_data = {
27                     'id': user.id,
28                     'full_name': user.full_name,
29                     'email': user.email,
30                     'username': user.username,
31                     'about':user.about,
32                     'x_link': user.x_link,
33                     'instagram_link': user.instagram_link,
34                     'facebook_link': user.facebook_link,
35                     'linkedin_link': user.linkedin_link,
36                     'savedate': user.savedate
37                 }
38
39
40             return JsonResponse({'message': 'User information updated messagefully', **user_data})
41         except CustomUser.DoesNotExist:
42             return JsonResponse({'message': 'User does not exist'}, status=404)
43         else:
44             return JsonResponse({'message': 'User ID is required'}, status=400)
45     else:
46         return JsonResponse({'message': 'POST method is required'}, status=405)
47

```

Sekil 1.20. Kullanıcı Bilgilerini Günceller

1.2.6.5. Kullanıcı Giriş Fonksiyonu

POST olarak alınan email ve password bilgileri ile girişü sağlar. Url erişimi 'login' ile olmaktadır.



```

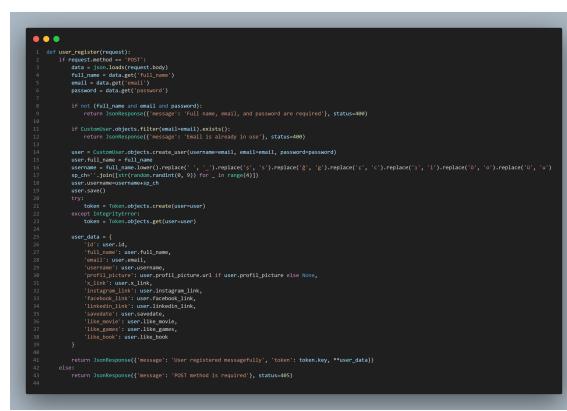
1 def user_login(request):
2     if request.method == 'POST':
3         data = json.loads(request.body)
4         email = data.get('email')
5         password = data.get('password')
6
7         user = authenticate(request, email=email, password=password)
8
9         if user is not None:
10             try:
11                 token = Token.objects.create(user=user)
12             except IntegrityError:
13                 token = Token.objects.get(user=user)
14             login(request, user)
15             user_data = {
16                 'id': user.id,
17                 'full_name': user.full_name,
18                 'email': user.email,
19                 'username': user.username,
20                 'password': user.password,
21             }
22             return JsonResponse({'message': 'Login successful', 'token': token.key, **user_data})
23         else:
24             return JsonResponse({'message': 'Invalid email or password'}, status=401)
25     else:
26         return JsonResponse({'message': 'POST method is required'}, status=405)
27

```

Şekil 1.21. Kullanıcı Girişİ

1.2.6.6. Kullanıcı Kayıt Fonksiyonu

POST olarak alınan FN,email,password bilgileri ile kullanıcı kaydını gerçekleştirir. Url erişimi 'register' ile olmaktadır.



```

1 def user_register(request):
2     if request.method == 'POST':
3         data = json.loads(request.body)
4         first_name = data.get('first_name')
5         last_name = data.get('last_name')
6         email = data.get('email')
7         password = data.get('password')
8
9         if not all([first_name, last_name, email, password]):
10             return JsonResponse({'message': 'Full name, email, and password are required'}, status=400)
11
12         if Customer.objects.filter(email=email).exists():
13             return JsonResponse({'message': 'Email is already in use'}, status=400)
14
15         user = Customer.objects.create_user(first_name=first_name, email=email, password=password)
16         user.first_name = first_name.title()
17         user.last_name = last_name.title()
18         user.username = user.first_name.lower().replace(' ', '_').replace('&', '_').replace('<', '_').replace('>', '_').replace('&gt;', '_').replace('&lt;', '_')
19         user.password = user.password.replace('&', '_').replace('<', '_').replace('>', '_').replace('&gt;', '_').replace('&lt;', '_')
20         user.save()
21
22         token = Token.objects.create(user=user)
23         token = Token.objects.get(user=user)
24
25         user_data = {
26             'id': user.id,
27             'full_name': user.full_name,
28             'email': user.email,
29             'username': user.username,
30             'profile_picture': user.profile_picture.url if user.profile_picture else None,
31             'x_list': user.x_list,
32             'y_list': user.y_list,
33             'rating_x_list': user.rating_x_list,
34             'rating_y_list': user.rating_y_list,
35             'linked_in': user.linkedin_link,
36             'facebook': user.facebook_link,
37             'line_movie': user.line_movie,
38             'line_music': user.line_music,
39             'line_book': user.line_book
40         }
41
42         return JsonResponse({'message': 'User registered successfully', 'token': token.key, **user_data})
43     else:
44         return JsonResponse({'message': 'POST method is required'}, status=405)
45

```

Şekil 1.22. Kullanıcı Kaydı

1.2.6.7. Kullanıcı Çıkış Fonksiyonu

POST isteği ile kullanıcı çıkışını gerçekleştirir. Url erişimi 'logout' ile olmaktadır.



```

1 @csrf_exempt
2 def user_logout(request):
3     if request.method == 'POST':
4         logout(request)
5         return JsonResponse({'message': 'Logout messageful'})
6     else:
7         return JsonResponse({'message': 'POST method is required'}, status=405)

```

Şekil 1.23. Kullanıcı Çıkışı

1.2.6.8. Kullanıcı Silme Fonksiyonu

POST ile userID verilen kullanıcıyı siler. Url erişimi 'delete' ile olmaktadır.



```

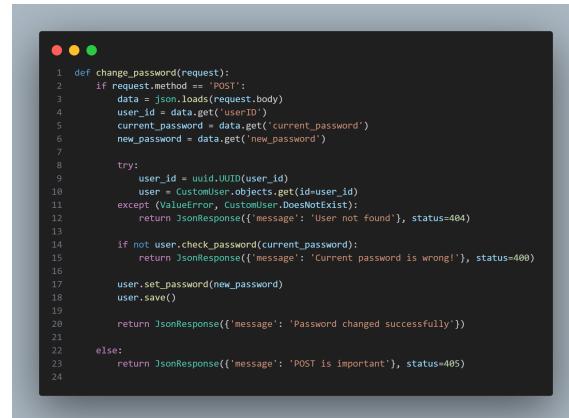
1 def delete_user(request):
2     if request.method == 'POST':
3         try:
4             data = json.loads(request.body)
5         except json.JSONDecodeError:
6             return JsonResponse({'message': 'Invalid JSON'}, status=400)
7
8         userID = data.get('userID')
9         if userID:
10            try:
11                user = CustomUser.objects.get(id=userID)
12                Movies_Comment.objects.filter(userID=userID).delete()
13                Game_Comment.objects.filter(userID=userID).delete()
14                Book_Comment.objects.filter(userID=userID).delete()
15                Book_Like.objects.filter(userID=userID).delete()
16                Movies_Like.objects.filter(userID=userID).delete()
17                Game_Like.objects.filter(userID=userID).delete()
18                user.delete()
19
20                movcomlen = Movies_Comment.objects.count()
21                movlikelen = Movies_Like.objects.filter(like=True).count()
22                movdislikelen = Movies_Like.objects.filter(dislike=True).count()
23
24                Movies_Table.commentscount = movcomlen
25                Movies_Table.like = movlikelen
26                Movies_Table.dislike = movdislikelen
27                Movies_Table.save()
28
29                gamelen = Game_Comment.objects.count()
30                gamelikelen = Game_Like.objects.filter(like=True).count()
31                gamedislikelen = Game_Like.objects.filter(dislike=True).count()
32
33                Games_Table.commentscount = gamelen
34                Games_Table.like = gamelikelen
35                Games_Table.dislike = gamedislikelen
36                Games_Table.save()
37
38                booklen = Book_Comment.objects.count()
39                booklikelen = Book_Like.objects.filter(like=True).count()
40                bookdislikelen = Book_Like.objects.filter(dislike=True).count()
41
42                Book_Table.commentscount = booklen
43                Book_Table.like = booklikelen
44                Book_Table.dislike = bookdislikelen
45                Book_Table.save()
46
47                return JsonResponse({'message': 'Kullanıcı başarıyla silindi'})
48            except CustomUser.DoesNotExist:
49                return JsonResponse({'message': 'Kullanıcı mevcut değil'}, status=404)
50        else:
51            return JsonResponse({'message': 'Kullanıcı ID gereklidir'}, status=400)
52    else:
53        return JsonResponse({'message': 'POST yöntemi gereklidir'}, status=405)

```

Şekil 1.24. Kullanıcı Silme

1.2.6.9. Kullanıcı Şifre Değiştirme Fonksiyonu

POST olarak aldığı userID kullanıcısının şifresini değiştirir. Url erişimi 'changepassword' ile olmaktadır.



```

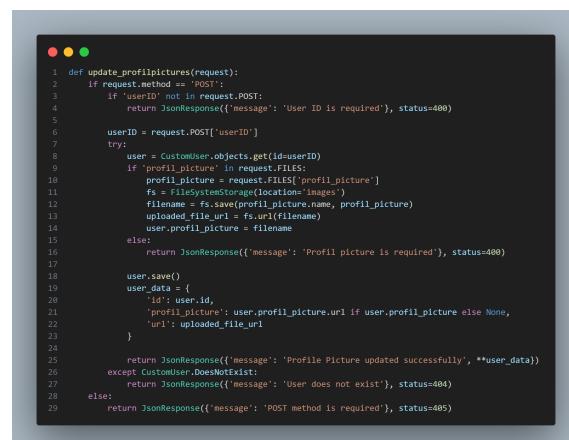
1 def change_password(request):
2     if request.method == "POST":
3         data = json.loads(request.body)
4         user_id = data.get('userID')
5         current_password = data.get('current_password')
6         new_password = data.get('new_password')
7
8         try:
9             user_id = uuid.UUID(user_id)
10            user = CustomUser.objects.get(id=user_id)
11            except (ValueError, CustomUser.DoesNotExist):
12                return JsonResponse({'message': 'User not found'}, status=404)
13
14            if not user.check_password(current_password):
15                return JsonResponse({'message': 'Current password is wrong!'}, status=400)
16
17            user.set_password(new_password)
18            user.save()
19
20            return JsonResponse({'message': 'Password changed successfully'})
21
22        else:
23            return JsonResponse({'message': 'POST is important'}, status=405)
24

```

Şekil 1.25. Kullanıcı Şifre Değiştirme

1.2.6.10. Profil Resmi Değiştirme Fonksiyonu

POST olarak aldığı userID hesabın profil resmini değiştirir. Url erişimi 'updateprofilepicture' ile olmaktadır.



```

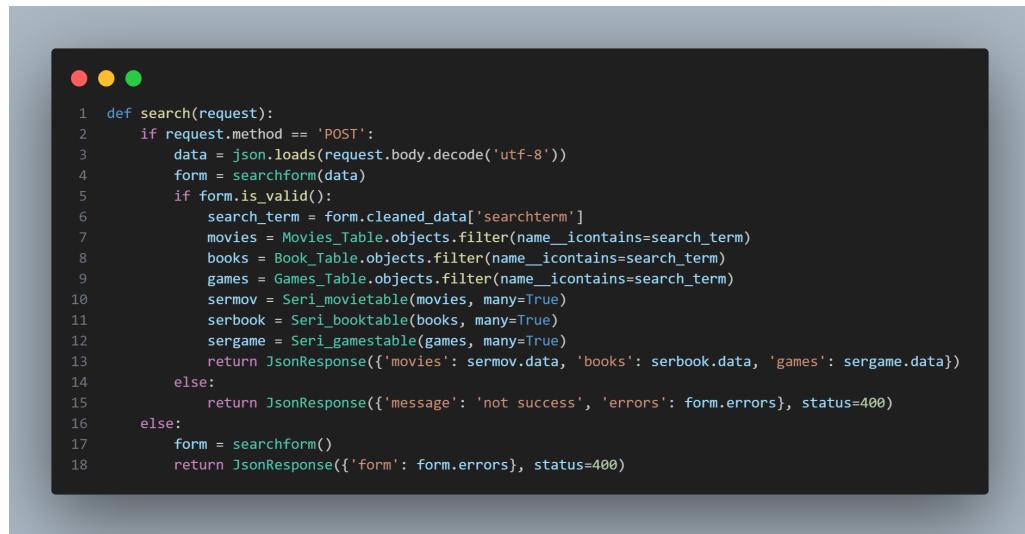
1 def update_profilepicture(request):
2     if request.method == "POST":
3         if 'userID' not in request.POST:
4             return JsonResponse({'message': 'User ID is required'}, status=400)
5
6         userID = request.POST['userID']
7
8         try:
9             user = CustomUser.objects.get(id=userID)
10            if 'profil_picture' in request.FILES:
11                profil_picture = request.FILES['profil_picture']
12                fs = FileSystemStorage(location='images')
13                filename = fs.save(profil_picture.name, profil_picture)
14                uploaded_file_url = fs.url(filename)
15                user.profil_picture = filename
16            else:
17                return JsonResponse({'message': 'Profil picture is required'}, status=400)
18
19            user.save()
20            user_data = {
21                'id': user.id,
22                'profil_picture': user.profil_picture.url if user.profil_picture else None,
23                'url': uploaded_file_url
24            }
25
26            return JsonResponse({'message': 'Profile Picture updated successfully', **user_data})
27        except CustomUser.DoesNotExist:
28            return JsonResponse({'message': 'User does not exist'}, status=404)
29
30        return JsonResponse({'message': 'POST method is required'}, status=405)

```

Şekil 1.26. Kullanıcı Profil Resmini Değiştirme

1.2.7. Arama Fonksiyonu

Arama Yapmak için geliştirilen sınıf. POST olarak searchterm beklemektedir. oluşturulan tüm app bilgilerinden arama yapar.



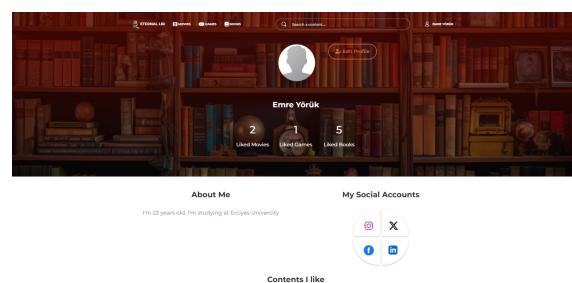
```
● ● ●
1 def search(request):
2     if request.method == 'POST':
3         data = json.loads(request.body.decode('utf-8'))
4         form = searchform(data)
5         if form.is_valid():
6             search_term = form.cleaned_data['searchterm']
7             movies = Movies_Table.objects.filter(name__icontains=search_term)
8             books = Book_Table.objects.filter(name__icontains=search_term)
9             games = Games_Table.objects.filter(name__icontains=search_term)
10            sermov = Seri_movietable(movies, many=True)
11            serbook = Seri_booktable(books, many=True)
12            sergame = Seri_gametable(games, many=True)
13            return JsonResponse({'movies': sermov.data, 'books': serbook.data, 'games': sergame.data})
14        else:
15            return JsonResponse({'message': 'not success', 'errors': form.errors}, status=400)
16    else:
17        form = searchform()
18        return JsonResponse({'form': form.errors}, status=400)
```

Şekil 1.27. Arama Yapma

2. BÖLÜM

Front-End

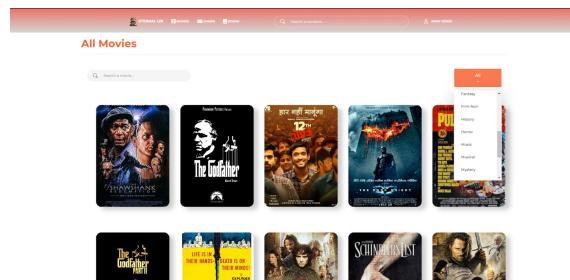
Projemizin front-end kısmını geliştirebilmek için temel web geliştirme araçlarının (HTML, CSS, JavaScript) yanı sıra daha efektif ve hızlı bir şekilde web sitesi geliştirebilmek için yaygın javascript frameworklerinden biri olan Angular framework’unu kullandık. Typescript tabanlı çalışan bu framework’ü kullanırken hem typescript hem de javascript kodları yazabilmekteyiz. Biz de projemizi geliştirirken her ikisini de kullandık. Tabi ki Angular’ın yanı sıra bize sunmuş olduğu bileşenleri (ngx-toastr, ngx-sweetalert2, ngx-spinner vs.) de uygulamamıza dahil ettik. Web sitemizin tasarımları için de yaygın ve responsive bir tasarım olanağı sunan Bootstrap ve benzeri kütüphanelerden yararlandık. Bunların yanı sıra web sitemize ikonlar ekleyebilmek için fontawesome kütüphanesini de uygulamamıza dahil ettik. Web site bileşenlerimize daha çok özellik ekleyebilmek için de Jquery ve Popper.js kütüphanelerini kullandık.



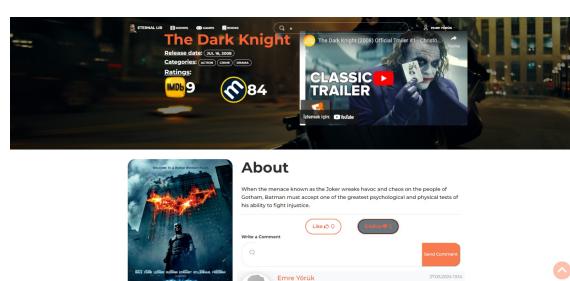
Şekil 2.1. Profil Ekranı

2.1. Angular

Angular, Google'in geliştirmiş olduğu ve sürekli getirdiği yeniliklerle güncel tuttuğu güçlü bir web framework'üdür. Angular'ın en önemli özelliklerinden biri olan router özelliğini de uygulamamızda kullandık. Angular'ın bize sağlamış olduğu bu özellik sayesinde tek sayfa uygulama (single-page application) geliştirmemize olanak sağlamasıdır. Yani uygulama içinde sayfa geçisi yaptığımız zaman sayfayı yeniletmeden sanki tek bir sayfadaymış gibi hızlı ve dinamik bir geçiş yapmamızı sağlıyor. Projemizi geliştirirken Angular'ın en son sürümü olan Angular 17 kullandık. Angular 17'nin önceki sürümlerden en önemli özelliği artık standalone yapısına geçmiş olmasıdır. Angular 15 ile gelen bu özellik Angular 17'de artık varsayılan uygulama yapısı olarak ayarlanmıştır. Standalone uygulama yapısı sayesinde web sitemiz ilk yüklenliğinde kullanmadığımız sayfalar da arka planda gereksiz yere yüklenmiyor. Sadece biz o sayfayı açtığımız zaman yükleniyor. Bu sayede uygulamamızın disk ve ram kullanımını azaltarak daha hızlı bir deneyim sağlanıyor.



Şekil 2.2. Movie Page



Şekil 2.3. Movie Detail Page

2.1.1. Html

HTML, web sayfalarının hazırlanmasında kullanılan sistemdir. HTML aslında bir programlama dili değildir ancak bilgisayarlarımıza kullandığımız web sitelerinin oluşturulmasında kullanılır. Chrome, Brave ve Microsoft Edge gibi tarayıcılar HTML kodlarını işler ve bu kodları web sayfasına dönüştürür. Kısaca HTML web sitelerinin temel iskeletidir. Biz de web sitemizi geliştirebilmek için tek seçenekimiz ve temel web site yapısı olan HTML kullandık.

2.1.2. CSS

CSS bir HTML yapısının masaüstü veya mobil cihazlarda ve çalıştırıldığında nasıl görüneceğini belirlemek için kullanılmaktadır. CSS sayesinde aynı uygulamayı sayısız farklı görünümlere sokabiliriz. Biz de web sitemizin gerekli kısımlarında web site bileşenlerimizi konumlandırmak ve şekillendirmek için CSS kullandık

2.1.3. JavaScript

JavaScript, kullanıcıların etkileşimli web siteleri oluşturmak için kullandığı bir programlama dilidir. JavaScript işlevleri, uygulama akışlarını yenilemekten animasyonlar ve etkileşimli bileşenler göstermeye kadar, bir web sitesi kullanıcısının deneyimini iyileştirebilir. Javascript hem front-end hem de back-end tarafında kullanılan geniş ve güçlü bir programlama dilidir. Jacascript'i back-end yani sunucu tarafında kullanabilmek için Nodejs devreye girer. Bu sayede tek bir programlama diliyle bütün (full-stack) bir web sitesi geliştirebiliriz. Biz web sitemizi geliştirirken javascript'i back-end olarak değil yalnızca front-end taradında kullandık.

2.1.4. TypeScript

TypeScript bir programlama dili değildir ancak TypeScript'e JavaScript'in makyajlı hali de denebilir. TypeScript kodlarını yazmak, kod yapısı olarak diğer nesne tabanlı dillere benzerliği sebebiyle JavaScript kodu yazmaktan çok daha kolaydır. Tabi ki TypeScript JavaScript tabanlı olduğu için herhangi bir JavaScript kodunu da anlayıp çalıştırabilir. Ayrıca Angular mevcut sürümlerinde TypeScript tabanlı olduğundan web sitemiz için front-end tarafında gerekli olan metodları yazarken TypeScript kullandık. Gerektiği yerlerde JavaScript kodları da yazdık.

2.1.5. Kullanılan kütüphaneler ve paketler

Django içerisinde veya dahil edilen paketler ile projemizi güçlendirdik.

2.1.5.1. JQuerry

jQuery'nin temel amacı, çok uzun ve karmaşık kod satırlarını tek bir koda sıkıştırmak ve böylece yüzlerce kodu yeniden yazmak zorunda kalmamaktır. Daha az kod satırıyla yüksek performanslı JavaScript uygulamaları geliştirmemize yarayan bu kütüphaneyi de biz de web projemize entegre ederek gerekli yerlerde kullandık.

2.1.5.2. Popper.js

Popper.js, ,lerlemeli ve kolayca özelleştirilebilen açılır penceler ve tooltip'ler oluşturmak için kullanılan güclü ve yaygın olarak kullanılan bir JavaScript kütüphanesidir. Popper.js ayrıca düzensiz düzenler oluşturmak için de kullanılır. Biz de web sitemizde açılır penceler ve tooltip'ler gibi bileşenleri kullanabilmek için Popper.js'i uygulamamıza dahil ederek kullandık

2.1.5.3. Bootstrap

Bootstrap'ın sahip olduğu CSS ve JavaScript taslakları, web sitelerinin ve mobil uygulamaların kullanıcılarına görünen bileşenleri için kullanılır. HTML, CSS, Less, Sass ve JavaScript ile yazılmış olan Bootstrap, tamamen etkileşimli ve duyarlı web uygulamaları geliştirmek için kullanılabilcek güçlü bir kütüphanedir.. Biz de web sitemizin bileşenlerini kodlarken hem CSS kodlarından tasarruf etmek hem de responsive yapıda bir site tasarlamak için Bootstraap kullandık.

2.1.5.4. fontawesome

Fontawesome ile birbirinden farklı ücretli ve ücretsiz ikonları alıp uygulamamıza dahil edebiliyoruz. Biz de web sitemizde kullanılan ikonlar için güçlü bir ikon kütüphanesi olan fontawesome kütüphanesini kullanarak uygulamamızda gerekli yerlerde ilgili ikonları ekledik.

2.1.5.5. ngx-toastr

ngx-toastr, web sitesinde kullanıcıya anlamlı ve güzel görünen bir mesaj vererek yaptığı işlemin sonucunu görmesini sağlayan bildirimler verilmesini sağlayan güçlü bir kütüphanedir. Biz de web sitemizi geliştirirken kullanıcıya geri bildirim vermek için ngx-toastr kütüphanesini kullandık. Bu sayede mesela kullanıcı giriş yaptığında, çıkış yaptığında ve bilgilerini düzenlediğinde vs. işlemin sonucunu görmesini sağladık. ngx-toastr ile yalnızca başarılı işlemlerde değil başarısız/hatalı işlemlerde de geri bildirim verdirebiliyoruz. Bu sayede kullanıcı yaptığı işlemin neden başarısız olduğunu anlayıp ona göre düzelterek işlemi uygun bir şekilde tekrar yapabilecektir.

2.1.5.6. ngx-sweetalert2

ngx-sweetalert2, web sitesinde kullanıcıya sık bir şekilde soru sordurmak veya ngx-toastr gibi geri bildirim verdirmek için kullanılan güclü ve yaygın olarak kullanılan bir kütüphanedir. Biz de web sitemizi geliştirirken kullanıcıya yapacağı işlemde emin olup olmadığını sordurmek için ngx-sweetalert2 kullandık. Bu sayede mesela web sitemizde gezinen bir kişi eğer oturum açmadan bir içeriği beğenmeye veya yorum yapmaya çalıştığında karşısına hesabının olmadığına dair bir geri bildirim çıkar ve eğer kullanıcı o geri bildirimde hesap açma seçeneğini seçerse kayıt ekranına yönlendirilir. Ve ardından kullanıcı hesap açtıktan sonra beğenmeye veya yorum yapmaya çalıştığı içeriye geri yönlendirilir. Eğer kullanıcı çıkan geri bildirimde hesap açmayı değil de iptal etmeyi seçerse işlem iptal edilir ve web sitesi kaldığı yerden yürütülmeye devam edilir.

2.1.5.7. ngx-spinner

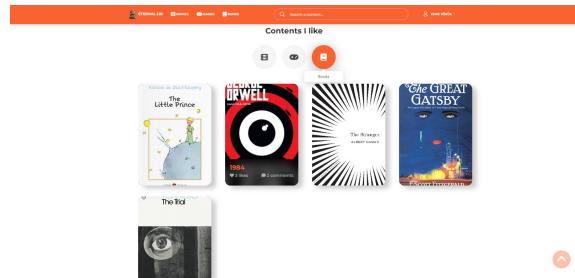
ngx-spinner, web sitelerinde güclü ve güzel animasyonlu yükleme ekranları sunan güclü ve yaygın olarak kullanılan bir kütüphanedir. Biz de web sitemizi geliştirirken yükleme ekranlarını yapabilmek için ngx-spinner kütüphanesinden yararlandık. Bu sayede kullanıcı bir sayfayı açtığında sunucuya istek atılır ve bu isteğin sonucu gelene kadar kullanıcının karşısına güzel animasyonlu bir yükleme ekranı çıkar.

2.1.6. Generic Http Service

Web sitemizde back-end yani sunucu tarafına istek atarken her seferinde gerekli kodları teker teker yazmak yerine tüm uygulamamızda kullanabileceğimiz bir Generic http Service oluşturduk. Ve bu sayede uygulamamızda gereksiz kod tekrarlarından kaçınmış olduk. Bu sayede web sitemizde kullanılan tüm API isteklerini tek bir Generic http Service üzerinden yönetmiş olduk.

2.1.7. Generic components

Web sitemizi tasarlerken yapısı birbirine benzeyen sayfaların temel yapısını alıp generic bir component oluşturarak bu component'a yerleştirdik. Bu sayede sayfalarımızda gereksiz kod tekrarlarından kaçınarak daha temiz ve tasarruflu kod blokları yazmış olduk.



Sekil 2.4. Book Page

3. BÖLÜM

Veritabanı

3.1. Veritabanı

Veritabanı ilk geliştirmeye başlama sürecimizde mongoDB ile büyük verileri işleminin daha kolaylık sağladığı için her sosyal medya gibi mongoDB ile geliştirmenin mantıklı olacağını düşündük. Bu bölümde her denemelerimizi ve hatalarımızdan dolayı değiştirmek zorunda kaldığımız veritabanlarını anlatacağız mongoDB ile başlayıp sqlite ile biten sürecimiz...

3.1.1. MongoDB

MongoDB, geliştirme sürecinin başında temel yapılar back-end geliştirme süreci başlamadan başlattık daha sonra mongoDB ile Django için ek paket dışında dahili olmadığı anlaşıldı. Bundan dolayı Back-end içerisinde Django harici bağlantı için python kodları oluştururdu. Kodlar daha sonra Django isteklerini cevap verecek şekilde dahil edildi. Sürecin sıkıntı olan süreci cloud ile tuttuğumuz için local yerine front-end aşamasında çok fazla gecikmelere sebeb oluyordu. MongoDB local kurulumuna geçmeyi denedik ama bu süreçte çok fazla sorun oldu python ile yazılan kodlar cloud düşünerek yazıldığı için local için sıfırdan ayarlanması gerekiyor hemde Django için tekrar entegre edilmesi gerekiyordu. Bunun için ekstra zamana ihtiyacımız olacaktı ancak projenin genel hatları bile yazılmadığı için mongoDB ile veritabanı geliştirme sürecini sonrası için durdurduk.

3.1.2. Sqlite

Sqlite, Django ile defualt ayarları ile gelen bir sql bu yüzden enegrasyon gibi bir süreci veya ekstra bir ayara ihtiyacımız olmayacağı. Sqlite modellerle birlikte oldukça entegre bir şekilde çalıştığı için Sqlite ile geliştirme ve testleri yapmayı sonradan sql veya mongo ile devam etmeyi kararlaştırdık.

3.1.3. MSsql

MSsql geliştirme süreci back-end sürecinden ayrı tutuldu back-end sqlite ile geliştirilmeye devam ettirilecek front-end den bir arkadaş ise MSsql geliştirilmesine başlamıştı. Süreçler paralel bir şekilde başarılı ilerlemektedir. Ancak Mongodaki sorunlar bu sefer MSsql ile başımıza geldi ve back-end ile birleşimi başarısız oldu. Bu süreçden sonra MySql ile Sqlite ile devam etme konusunda tartıştık. Sonuç olarak zaman faktöründe düşünülerek yeni bir database geçiş riski yerine halihazırda çalışan ve entegre olan sqlite ile geliştirilmeye devam edildi.

4. BÖLÜM

TARTIŞMA, SONUÇ ve ÖNERİLER

4.1. Tartışma, Sonuç ve Öneriler

4.1.1. Tartışma

Yapılan daha önceki çalışmalara (referans) ve yapılan sayısal incelemere bakılarak insanların içerik keşif süreçlerini eski geleneksel yöntemlere göre yapmadıkları , insanların içerik keşif alışkanlıklarının bu projede olduğu gibi daha modern projelere evrildiği görülmüştür. Bu durumun nedeni , artan insan nüfusunun ve buna bağlı olarak gelişen sürekli artan verinin , içerik seçimlerinde kullanıcılar için zorluk teşkil etmesi ve zaman almasıdır . Aynı zamanda günümüz hızlı dünyasında insanların vakitleri kısıtlı olduğundan karar süreçlerinin azalması gerekmektedir. Bu durumlar insanların bu proje gibi teknolojilere yönelmelerini sağlamaktadır. Bunlar , şüphesiz için bu projenin daha fazla trafik çekmesine olumlu yönde katkı sağlayacaktır. Daha önceden yapılmış “letterboxd” ve “goodreads” gibi sitelerde bu sonuçları görülmüyör.

Projede karşılaşılan birçok sorundan en temel olanı kullanacağımız veritabanı oldu. Projenin plan aşamasında MongoDB veritabanı üzerinde karar kılımıştık fakat daha sonra NoSQL bir veritabanının bizim projemizde uygun olmadığını anladık ve ilişkisel veritabanı olan MsSQL'e geçtik. Fakat burada karşılaştığımız zorluk sunucumuzun olmamasıydı. Kişisel bilgisayarın sunucu gibi kullanılmasını denedik fakat bu yöntemin optimal sonuç doğurmayacağını anladık . Kişisel bilgisayarımız sürekli olarak kullanılmalıydı ve burada güvenlik sorunları gözlemlenebilirdi. Son olarak yerel dosya tabanlı bir veritabanı motoru olan LiteSQL kullanmaya karar verdik. Bu veritabanı bizim projemiz için tutarlı , stabil ve hızlı çalışması bakımından uygun oldu .

Projenin benzer kaynaklara göre avantajlı olduğu birçok konu vardır . Temel avantajı , şüphesiz kitap , oyun ve film içeriklerinin tek bir çatı altında kolay bir şekilde değerlendirilmesi , yorumlanabilmesi ve beğenilen içeriklerin kullanıcı tarafında takibinin yapılabilmesidir.

Projenin önyüz (frontend) kısmında temel önyüz teknolojilerinin (HTML ,CSS , JavaScript) yanı sıra angular framework teknolojisini kullandık . Angular framework teksayfa gibi çalıştığından hız açısından oldukça tatminkar sonuçlar sağladı . Responsive bir layout kütüphanesi olan bootstrap kullandığımız için kullanıcı dostu arayüz geliştirebildik. Arkaplanda (backend) ise Python djangonun yazılım esnekliği , zengin kütüphane seçenekleri sayesinde metodların güvenilir (realible) ve stabil çalışmasını sağladık . Veritabanı kısmında LiteSQL kullandığımız için ekstra bir sunucuya gerek kalmadan yüksek performans ve yüksek güvenilirlik ile verilerimizi tutabildik. LiteSQL'in çapraz platform uyumluluğu sayesinde her işletim sisteminde verileri görüntüleyebiliyoruz . Bu kullandığımız teknolojiler bize birçok avantaj sağlamıştır.

Projemizde gelişime açık birçok alan bulunmaktadır. Daha fazla trafik çekmek amacıyla kullanıcılara yaptıkları değerlendirme ve yorum sayılarına göre puan , rozet vs. gibi ödüllendirici ve şevklendirici birtakım dereceler verilebilir . Bu sayede kullanıcılar yaptıkları yorum ve değerlendirme karşılığında bir ödül kazandıklarını düşüneceklerdir ve daha çok yorum yapacaklardır . Sitenin daha çok işlevsel olması daha çok veri bulunmasına bağlıdır ve bu durum site işlevselliği bakımından oldukça yararlı olacaktır. Ayrıca kullanıcılar isterlerse (halka açık veya gizli) sosyal medya bilgilerini kendi profillerinde paylaşabilir ve diğer kullanıcılar ile websitesi üzerinden iletişim kurabilir . Bunun gibi birçok fonksiyon ile websitesi tam anlamıyla bir sosyal medya platformuna dönüşebilir.

4.1.2. Sonuç

Bu projede kullanıcıların içerik seçim konusunda birbirleri ile etkileşimde bulunulması için bir film , kitap ve oyun keşif websitesi yapılmıştır. Bu site sayesinde kullanıcı içerik keşif süresi düşürülmüştür.

4.1.3. Öneriler

- 1) Websitesi her ne kadar esnek (responsive) bir yapıda da tasarlanmış olsa da böyle bir projenin mobil versiyonu olmalıdır.
- 2) Kullanıcıların birbirleriyle daha fazla etkileşim içerisinde girmesi için sosyal medya entegrasyonu yapılabilir.
- 3) Daha gelişmiş kategorizasyon sistemi kullanılabilir . Bu sayede daha spesifik istekler daha kolay bulunur.
- 4) Kullanıcıların geçmişteki tercihlerine göre öneride bulunan bir yapay zeka asistanı eklenebilir.

KAYNAKLAR

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı : Sinan UYĞUN
Uyruğu : Türkiye (T.C.)
Doğum Tarihi ve Yeri : 19.11.1999 - AMASYA
Telefon : 0 545 787 5314
Belgegeçer :
E-posta : 1030520914@erciyes.edu.tr
Adres : ARTIKABAT Mah.
ATALAY Sok.
05700, Gümüşhacıköy AMASYA TÜRKİYE

EGİTİM

Derece	Kurum	Mez. Yılı
Lise	İrfanlı Anadolu Lisesi, AMASYA	2018
Ortaokul	Ülkü Ortaokulu, AMASYA	2014

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı : Onur KARAKAYA

Uyruğu : Türkiye (T.C.)

Doğum Tarihi ve Yeri : 26.09.2000 - İSTANBUL

Telefon : 0 549 170 47 78

Belgegeçer :

E-posta : 1030520931@erciyes.edu.tr

Adres : Hürriyet Mah.

Takdir Sok.

34870, Kartal İSTANBUL TÜRKİYE

EĞİTİM

Derece	Kurum	Mez. Yılı
---------------	--------------	------------------

Lise	Cevizli Anadolu Lisesi, İstanbul	2018
------	----------------------------------	------

Ortaokul	Şeyh Şamil Ortaokulu, İstanbul	2014
----------	--------------------------------	------

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı : Emre YÖRÜK
Uyruğu : Türkiye (T.C.)
Doğum Tarihi ve Yeri : 13.02.2001 - Kahramanmaraş
Telefon : 0 554 168 46 16
Belgegeçer :
E-posta : 1030510202@erciyes.edu.tr
Adres : Haydarbey Mah.
Tekerek Sok.
46000, Onikişubat KAHRAMANMARAŞ TÜRKİYE

EĞİTİM

Derece	Kurum	Mez. Yılı
Lise	Açık Öğretim Lisesi, Ankara	2018
Ortaokul	Aysel Çalık Ortaokulu, Kahramanmaraş	2014