****

**GÖMÜLÜ SİSTEM ÇÖZÜMLEMELERİ**

**Dr. Öğr. Üyesi Fırat AYDEMİR**

**Mobil Film-Dizi Tanıtma Uygulaması**

**2022-2023 Güz Dönemi**

**16/12/2022**

**İsmail KEYVAN – 201913171030**

**Muhammed Emir GÖZCÜ – 201913171053**

**Giriş**

Yaptığımız çalışmalardan sonra Flutter kullanarak tasarlayacağımız çini del bir arayüz tasarımı bulduk ve bu tasarımı kendi uygulamamız için entegre ederek kullanmaya başladık. Tasarımı hazırlarken lokal olarak kaydettiğimiz resimleri ve film bilgileri kullanmıştık. Bu rapor döneminde ise bu statik içerikleri TheMovieDB üzerinden çektiğimiz film bilgileri ile değiştirdik. Trend filmler, popüler filmler veya diğer kategorilerde olacak bir değişiklik uygulamamıza gerçek zamanlı olarak yansıyacak. API istekleri için kullanılması gereken istek url’lerini hazırladık ve isteklerde bulunduk. Bize dönen cevapları ham JSON formatından çıkarıp kullanabileceğimiz sınıflar haline getirdik. Bu sınıfları kullanarak istediğimiz içerikleri bir FutureBuilder widget’ı içerisinde kullanıcıya gösterilmesini sağladık. Ayrıca bu durumda ortaya çıkan bazı arayüz hatalarını gidererek uygulamanın geliştirme sürecine devam edildi.

**Yapılan Çalışmalar**

**quicktype:** Kendisine verilen JSON dosyasını parçalayıp seçilen yazılım dili kuralları dahilinde sınıflar, listeler diziler ce diğer değişken tiplerini kullanarak işleyerek kullanıcıya JSON dosyası üzerinde kolay bir şekilde gezme imkanı veren faydalı bir web sitesi.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek olarak verilen JSON dosyasını Dart dilinde bir sınıfa dönüştürelim. Verilen JSON dosyasında bir tanesi String ve de diğeri de List<String> olmak üzere 2 tane verimiz var. Bu sınıfı kullanmak için gerekli olan JSON da bu alanların aynılarını içermeli. FormMap metotlunu kullanarak içine verdiğimiz JSON dosyasını bir Welcome sınıfı olarak elde edip bir değişkene atayabiliriz. Bu değişken ile JSON içerisindeki alanlara sanki bir sınıfın değişkenleri gibi erişip kullanabiliriz. Bu değişkenlerin veri tipleri sınıf içerisinde belirtildiği için her değişkenin kendi veri tipine özgü olan metotları da kullanabiliriz. Yani JSON içerisindeki listeye bir liste olarak erişip listelere özgü olan özellikleri kolaylıkla kullanabiliriz. Bizim kullanacağımız JSON dosyaları daha büyük ve karışık olduğu için oluşturulan sınıf daha büyük olacaktır ve farklı metotlar da kullanılacaktır. Bu süreçte farklı alt sınıfların da oluşturulması mümkündür. Ayrıca içerisinde farklı alanlar içeren JSON dosyaları için farklı bir sınıf oluşturmak gerekir.

Trend filmleri için attığımız isteğe dönen JSON dosyasını bir sınıfa dönüştürelim. Bu sayede JSON dosyası içerisindeki alanlara oluşturulan sınıf objesi ile daha kolay bir şekilde erişebiliriz ve “intellisense(akıllı kod tamamlama)” özelliklerini etkin bir şekilde kullanabiliriz. Oluşacak olan sınıfa, bize gelen JSON dosyasını direkt olarak gönderemeyiz. Bu JSON dosyasını öncelikle bir Map veri tipi olarak decode etmemiz gerekiyor. Bu işlemden sonra elimize gecen veriyi kullanarak bu üzerinden kullanacağımız objeyi oluşturabiliriz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

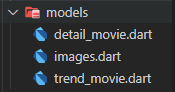
Bize cevap olarak dönen JSON içeriğini “decode” ederek buradaki fromMap metodu içerisine verdiğimizde bize bir Trend sınıfı objesi dönüyor.

Bu dönen objenin içerisindeki alanlar JSON dosyasının içerisindeki alanlar ile aynı oluyor. Bu objeyi kullanarak JSON içerisindeki alanlara çok daha rahat bir şekilde erişebiliyoruz.

Burası ve daha sonrası geliştirici olarak bizi pek ilgilendirmiyor. Oluşturulacak sınıf için gerekli olan diğer veriler tanımlanabiliyor. Burada da farklı alt sınıflar ve enum’lar oluşturulmuş.

metin içeren bir resim

Açıklama otomatik olarak oluşturulduAPI’dan dönen cevabı decode ederek kullanacağımız sınıf model objesini oluşturduk.

 Filmlerin detayları, resimler ve trend filmler için farklı isteklere farklı JSON dosyaları cevap olarak geliyor. Her farklı JSON yapısı için farklı sınıflar oluşturduk ve bu sınıfları ilgili verilere erişmek için kullandık.

Tabi bu sınıfları kullanabilmek için elimizde bir JSON verisi olması lazım. Bu veriyi kullandığımız TheMovieDB API’ına istekler göndererek elde ediyoruz. Bu istekleri Flutter içerisinde “http” paketini kullanarak yapıyoruz. İstekte bulunacağımız url’i girdikten sonra kullandığımız API tarafından verilen anahtar değeri’ni de ekleyerek bir istekte bulunduk. Eğer dönen cevabın statü kodu 200 ise yani işlem başarılı ise dönen cevabı oluşturduğumuz dönüşüm sınıfını kullanarak bir objeye dönüştürüyoruz. Bu obje ile de cevap içerisinde kolayca gezip ilgilendiğimiz kısımlara ulaşabiliyoruz. Aşağıdaki fonksiyon bu işlemler başarılı olduğunda bize JSON dosyasının sadece istediğimiz kısmını süreli bir işlem yani “Future” olarak dönüyor.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bize gelen bu “Future” veri tipindeki içeriği kullanmak için “FutureBuilder” widget’ına ihtiyacımız var. Bu sayede veriler gelirken kullanıcı arayüzünde bir yükleme animasyonu gösterirken veriler geldikten sonra verileri gösterebiliyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu API’a istek atıp dönen verileri bir sınıf haline getiren fonksiyon. Bu zaman alan ve cevap gelip gelmeyeceği belli olan bir işlem olduğu için kodumuzun kalan kısmı bu duruma göre görüntüleniyor.

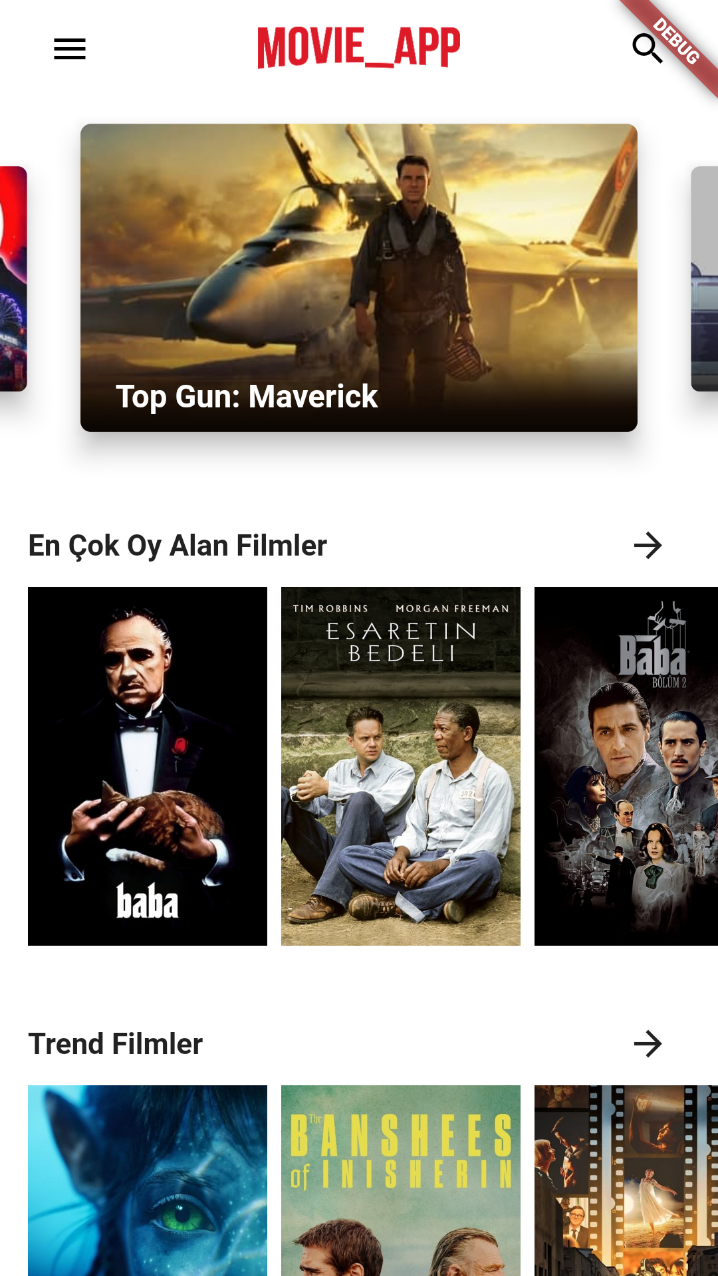
Eğer bağlantı statüsü uygunsa, gelen cevap bir veri içeriyorsa ve bu veri boş değilse bu kısım çalışacak. Kullanıcıya göstermek istediğimiz içeriğin bulunduğu kısım da sadece burada kullanılabilir. Burada elimize geçecek olan verinin tipini bildiğimiz için veri tipini değere veriyoruz. Bu sayede intellisense özelliklerinden yararlanabileceğiz.

metin içeren bir resim

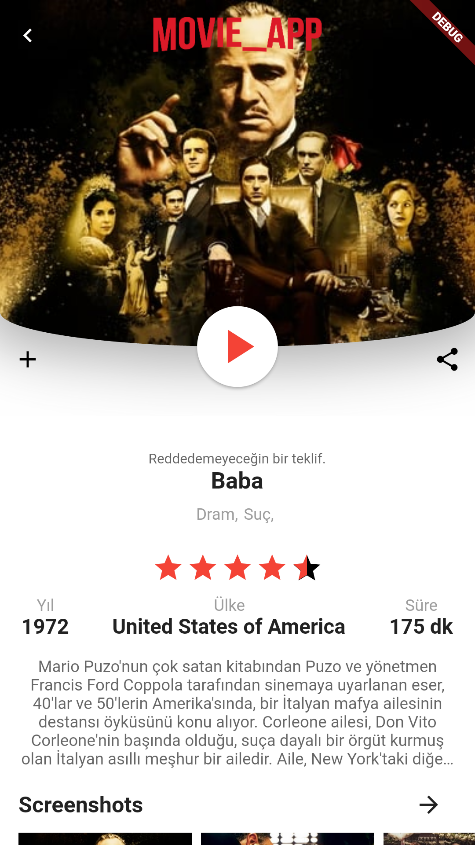
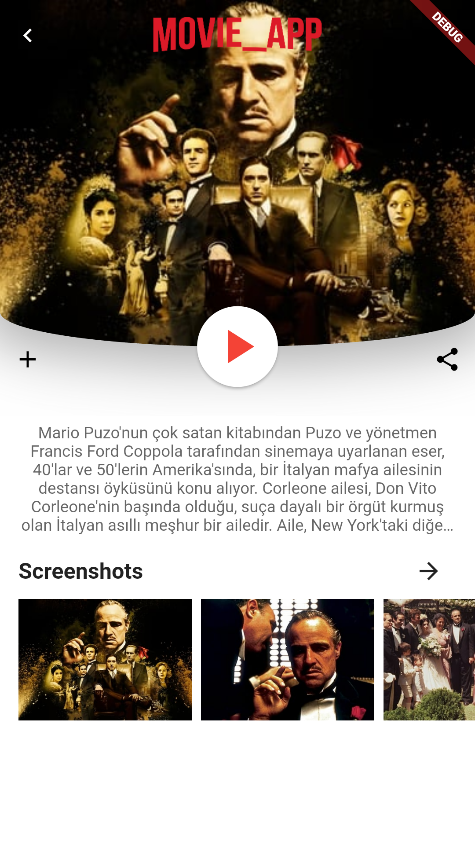
Açıklama otomatik olarak oluşturulduBize dönen verinin yani filmin içerisindeki “id” değerini “Detay” sayfasında da kullanmamız gerekiyor. Bu sebeple bunun için farklı istek atmaktansa burada elimize geçem bu değeri “Detay” sayfasına direkt olarak gönderdik.

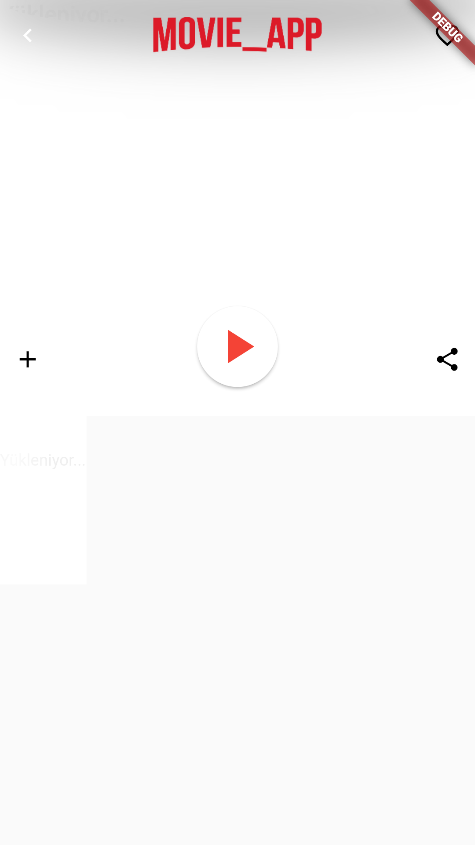
metin içeren bir resim

Açıklama otomatik olarak oluşturulduEğer bir bağlantı sorunu olursa, hatalı bir istekte bulunulursa, API kaynaklı bir hata olursa veya veriler çekilirken bekleme durumundayken “else” durumuna düşeriz. Bu durumda kullanıcıya göstermek istediğimiz Widget’ı burada tekrar döndürmeliyiz. Biz bu durumda halka şeklinde dönen bir yükleniyor simgesi ekledik. Arayüzün iyileştirilmesi için daha farklı tasarımlar da eklenebilir. Bu durumdayken “FutureBuilder” sürekli dinleme durumundadır. Veriler elimize geçtiği anda ekranda bu kısım için hazırladığımız arayüz gösterilir.

API’dan gelen veriler yukarıda anlatıldığı şekilde işlenip kullanıcı arayüzüne yansıtıldığında uygulamamız yandaki gibi gözükmektedir. İçeriklere tıklanıldığında içerik ile ilgili olan detay sayfasına yönlendirme yapılıyor. Yönlendirme yapılırken bu filmin id’sini de detay sayfasına göndermek gerekiyor çünkü detay sayfasında bu id ile film detaylarını elde etmek için farklı bir istekte bulunmak gerekiyor.

Detay sayfasında ilgili içerik ile genel bilgiler gösteriliyor. Bu bilgilere ulaşmak için farklı bir istek atmak, bu istekten dönen cevabı farklı bir sınıf içerisinde işleyip “FutureBuilder” widget’ını kullanarak bu verilerin gelip gelmeme durumlarını da ele almak gerekiyor.





Veriler beklenirken kullanıcıya boş bir ekran gösteriliyor. Burada alternatif olarak yükleniyor algısını yaratacak halka şeklinde dönen bir widget veya bir yükleme indikatörü de gösterilebilir. Fakat yükleme işleme hızlı gerçekleştiği için bu widget’ların oluşturulmasına gerek duymadık. Yükleme tamamlandıktan sonra ise gelen verile ile oluşturulmuş ekran kullanıcıya sunuluyor. API bize filmin 10 üzerinden aldığı puanı veriyor. Bu değeri 5 yıldız halinde göstermek için ikiye bölüyoruz. Geri kalan bilgiler ilgili kısımlarda gösteriliyor. Bu sayfa her içerk içiin farklı veriler kullanılarak tekrar oluşturuluyor, bu yüzden esnek bir tasarım yapmaya önem verdik.

**Sonuç**: Kullandığımız API bize uygulamamızda kullanacağımız bilgileri yapacağımız isteğe karşılık bir JSON dosyası formatında veriyor. Bu dosyayı Flutter içerisinde daha kolay kullanabilmek için “quicktype” web sitesi üzerinde bir Dart sınıfı haline getirdikten sonra kodumuza ekleyebiliyoruz. Kullandığımız “http” paketi sayesinde isteğimizi gönderip dönen cevabı oluşturduğumuz sınıfın ilgili metotları ile o sınıfın bir objesi olarak alabiliyoruz. Bu işlemden önce, dönen dosyayı gerekli “Map” veri tipinde decode etmemiz gerekiyor. Bütün bu işlemlerden sonra bu verleri uygulamamızda kullanabiliriz. İnternet üzerinden veri almak hatalara müsait ve bağlantı hızından etkilenen işlemler olduğu için Flutter bu işlemlerin ne zaman sonuç vereceğini bilmiyor. Sonuçları geldiği durumda gösterip gelen veri olmadığı durumda ise farklı arayüz öğesi göstermek için “FutureBuilder” widget’ını kullanmamız gerekiyor. Bu widget içerisine parametre olarak zaman gerektiren işlemi veriyoruz e bu işlemin başarı durumu sonuçlarına göre olacak işlemleri yazıyoruz. Değişen durumlarda bu widget dinleme durumunda olduğu için veri geldiğinde arayüze istediğimiz widget’lar yansıyor. Ayrıca bu widget kendi içerisinde bir cache(önbellek) sistemine sahip olduğu için veriler geldikten sonra bir sorun olsa dahi veriler kullanıcıya gösterilmeye devam ediliyor.

**Yapılacak Olan Çalışmala**r**:** Bir sonraki rapor döneminde yapmak istediğimiz uygulama içerisindeki bazı butonlara işlevsellik kazandırmak ve farklı kategorilerin görüntüleneceği sayfa, kullanıcı kayıt sayfası, yan açılır menünün arayüz tasarımlarını yapmak.

**Kaynakça**

**Github**: <https://github.com/emirgzc/movie_app>

[1] <https://app.quicktype.io/>

[2]<https://www.themoviedb.org/documentation/api>

[3] <https://jsonbeautifier.org/>

[4] <https://dribbble.com/shots/5026483-Netflix-Mobile-App-Redesign>

[5] <https://developers.themoviedb.org/3/movies/get-movie-details>