****

**GÖMÜLÜ SİSTEM ÇÖZÜMLEMELERİ**

**Dr. Öğr. Üyesi Fırat AYDEMİR**

**Mobil Film-Dizi Tanıtma Uygulaması**

**2022-2023 Güz Dönemi**

**28/10/2023**

**İsmail KEYVAN – 201913171030**

**Muhammed Emir GÖZCÜ – 201913171053**

**Giriş**

Yaptığımız araştırmalar ve örnek incelemelerinden sonra mobil uygulama geliştirmede bazı yolları tercih etmemiz gerektiğini anladık. Native veya Cross Platform (Çapraz Platform) uygulama geliştirme arasında bir seçim yapmamız gerekti. Uygulamamızı Flutter ile Cross Platform olarak geliştirmeye karar verdik. Günümüzde geniş kitlelerin kullanıma sunulan uygulamalar genellikle Native olarak yazılıyor. Bu uygulamalar büyük şirketlerin çatısı altından çıkıyor. Biz bu uygulamalara en yakın görünümü ve performansı elde etmek için Flutter ile uygulamamızı geliştirmeyi tercih ettik. Bu raporda Flutter’ın daha derinlerine indik ve uygulamamızda kullanmak durumunda kalacağımız teknolojiler hakkında araştırmalar yaptık.

**Yapılan Çalışmalar**

Mobil uygulama geliştirmek için izlenebilecek tek ve en iyi bir yol bulunamamakta. Teknoloji şirketleri web uygulamalarının ardından mobil uygulamalara da çok hızlı bir şekilde uyum sağladı. Mobil uygulamaları tasarlarken en erken karar verilen konulardan bir tanesi de kullanılacak yazılım dilidir. Uygulamanın kullanılacağı platforma, geliştiricilerin yetkinliğine ve farklı parametrelere bağlı olarak uygulamayı geliştirirken kullanılacak olan yazılım dili veya dilleri belirlenir.

**Objective-C:** Genellikle İPhone ya da İPad benzeri cihazlara yönelik hazırlanan uygulamalarda kullanılır. IOS işletim sisteminde rahatlıkla tercih edilmektedir.

**Kotlin:** Kotlin, Java Virtual Machine (JVM) üzerinde çalışabilen, JavaScript kaynak koduna derlenebilen statik bir programlama dilidir. Çoğu kişinin yanlış bildiği bir konu ise Google tarafından değil, JetBrains şirketi tarafından geliştirilmiş olmasıdır. Özellikle Android uygulamalar geliştirmenizi kolaylaştıran bu dil ile ayrıca Web uygulamaları yazmak da mümkündür. Arayüz tasarımı için daha az kod ve performans sağlayan Jetpack Compose kullanır. Jetpack Compose sağladığı arayüz, state(o anki durum) gibi bir çok değeri kullanabilmek için o fonksiyonun “composable” olmasını gerekir, bu da @Composable anahtar kelimesi kullanılarak kolaylıkla sağlanabilir. Jetpack Compose ile daha kolay bir arayüz tasarımı çok daha az satır kod yazılarak gerçekleştirilebilir. Kotlin dili ile uygulama geliştiren kişiler Android Studio veya VS Code(Visual Studio Code) entegre geliştirme ortamlarını (IDE) kullanabilir.

**Swift:** Apple tarafından ortaya çıkarılan bu dil özellikle IOS işletim sistemine sahip cihazlara yönelik uygulama geliştirmede kullanılır. Apple, IOS işletim sistemindeki verimliliğini bu dile de yansıtmıştır. Swift dilinde otomatik referans sayma özelliği (ARC) yer alır, bu sayede kullanılmayan örnekler otomatik olarak silinip belleğe tekrar kazandırılır. Açık kaynaklı bir yapıya sahip olan bu dil XCode entegre geliştirme ortamlarını (IDE) kullanılarak yazılabilir.

**Dart:** Google tarafından geliştirilip 2017 yılında piyasaya sürülen bu dil henüz daha yeni olmasına rağmen mobil uygulama sektöründe yerini hemen almıştır. En büyük artılarından biri olan Multi-Platform veya diğer bir ismiyle Cross-Platform özelliği ile tek bir kod ile Android, IOS, Windows, Mac, Linux ve Web ortamlarında uygulama geliştirmeye olanak sağlar. Çok maksatlı ve nesneye yönelik bir dildir.



Dart kodlarını belirli bir yazılım çatısı altında toplamak için **Flutter** SDK (yazılım geliştirme kiti) kullanılır. Flutter, Google tarafından geliştirilen bir kullanıcı arayüzü (UI) geliştirme aracıdır. Kısaca ifade edersek Dart yazılım dili ile Flutter kullanarak Androd, IOS, Linux birden çok platformda kullanılabilen uygulamalar yazmak mümkündür. Flutter’ın rakiplerine karşı en büyük artısı da budur.

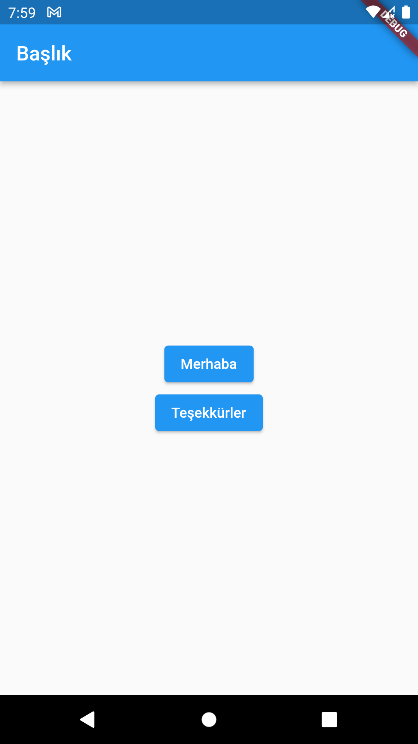
Flutter ile Android uygulama geliştirirken Android’in bize sağladığı güzelliklerden birisi olan Material Design kullanıcı arayüzü (UI) Kullanılır. Flutter kullanılarak hazırlana bir arayüz “Widget” elemanlarından oluşur. Bu widget herhangi bir metin, resim, buton, kullanıcı girdi alanı, başlık veya ikon olabilir. Flutter’ın bize sağlamış olduğu hali hazırda yüzlerle widget ile ileri düzey arayüzler hazırlanabilir. Bunun üstüne kullanıcıların Flutter’ın sağladığı widgetlar ile oluşturdukları hazır paketler kullanılarak işler daha da kolaylaştırılabilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

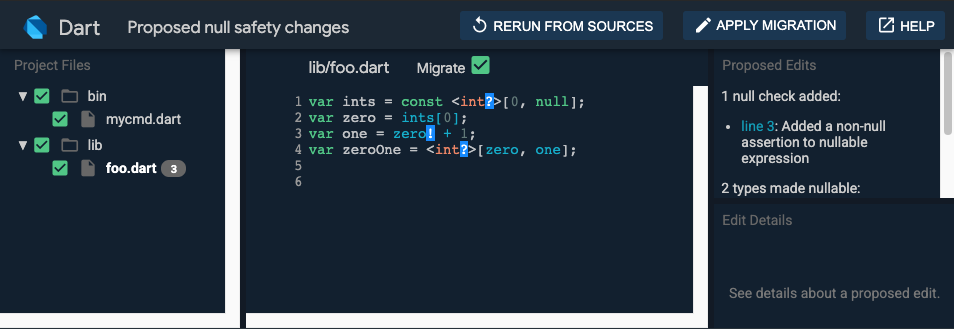
Bütün bu widgetlar bir widget Ağacı (widget Tree) altında bir ağaç yapısı ile tutulur. widget ağacının en üstündeki ebeveyn sınıf ise “ Widget” sınıfıdır.

Ayrıca Flutter ile geliştirilen uygulamalar çok kolay bir şekilde herhangi boyuttaki bir ekrana uyumlu hale getirilebilir.

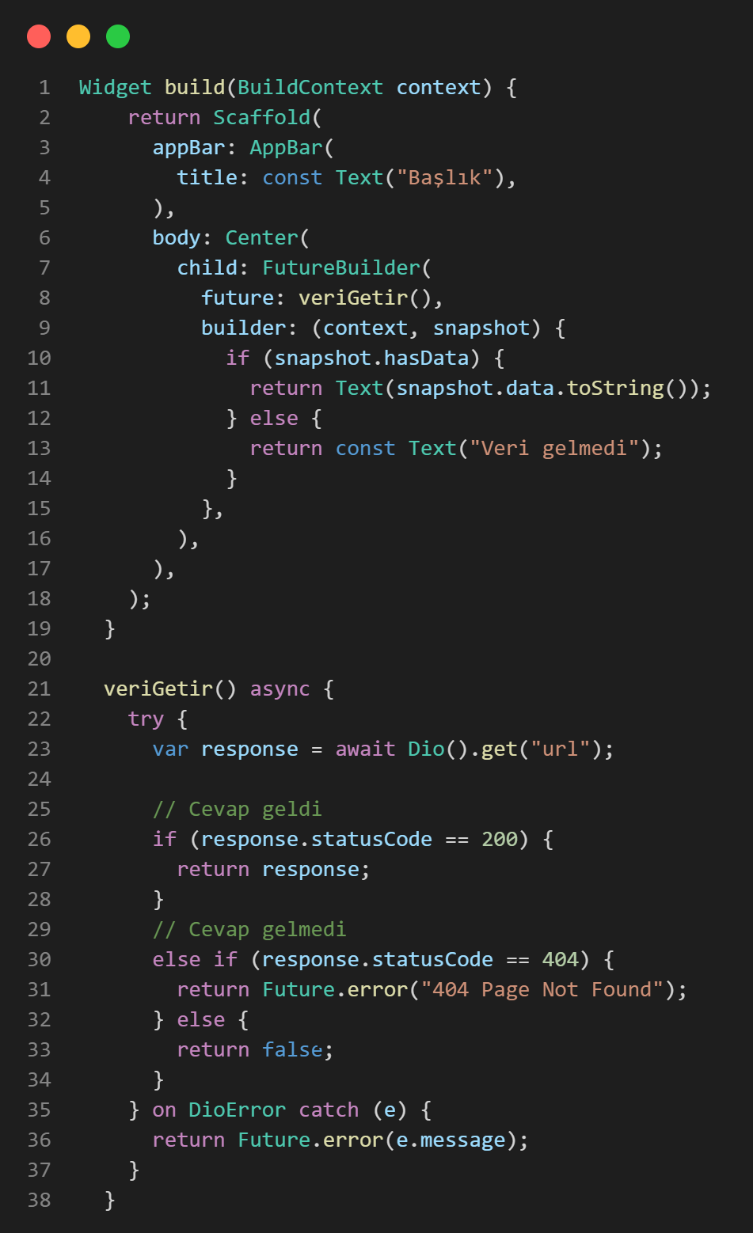
 

Flutterda widgetlar arasında çocuk-ebeveyn (child-parent) ilişkisi vardır. Bir widget kendisinin veya başka bir widget’ın çocuğu veya ebeveyni olur. widget ağacında üst tarafta ebeveynler altta ise çocuklar olacak şekilde bir dizilim vardır. “Column” gibi bazı özel widgetlar birden çok çocuğa ebeveynlik yapabilir. Bu çocuklar bir “childeren” isimli widget listesinin elemanları olabilirler. widgetları içeren sınıflar ise widgetların durumlarının(state) değişip değişmeyeceğine göre ikiye ayrılır. Eğer kullanılan widget üzerindeki metin bir eylem ile değişecek ise bu widget bir “Statefull” sınıfta bulunmalıdır çünkü widgetın o anki durumu bir değişime uğrar. Eğer böyle bir durum söz konusu değil ile widget bir “Stateless” sınıf içinde bulunabilir. Projemizi hazırlarken ihtiyacımızın olacağı bazı Flutter özellikleri aşağıdaki gibidirler:

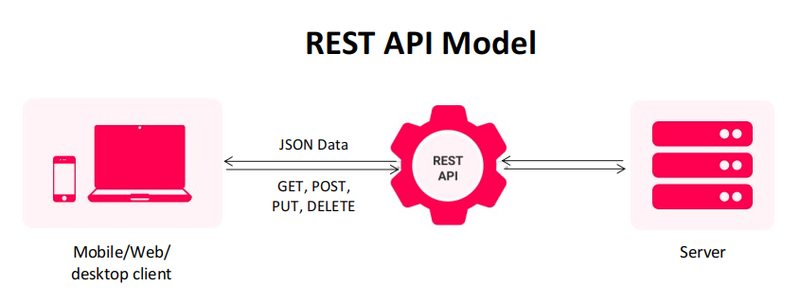
**1- Null Safety:** Flutter’a yakın zamanda gelen bir güncelleme ile Flutter, Null Safety özelliği kazanmıştır. Bu özellik Kotlin ve Swift dillerinde uzun süredir bulunmaktadır ve Flutter da sonunda Null Safety özelliğini bünyesine eklemiştir. Örneğin bir değişken tanımlarken bu değişkene bir değer atayabiliriz veya sonradan atamak üzere değersiz yani null bir şekilde bırakabiliriz. Eğer bu değişkenin değerini bir hata sonucu veya kasıtlı bir şekilde atamayıp değişkeni bir işlemde kullanmayı denersek bize bir null hatası verir. Null Safety tam da burada işe giriyor ve değişkenin tanımlandığı anda herhangi bir değer de verilmesini zorunlu kılıyor. Bu sayede herhangi bir hata ile karşılaşmıyoruz. Tabi her değişken tanımlandığımızda o değişkene bir atamak zorunda olmak biraz can sıkıcı olabiliyor. Değişken tanımlarken en başa “late” anahtar kelimesini ekleyerek bu değişkenin değerini sonradan vermek istediğimizi belirtebiliyoruz.

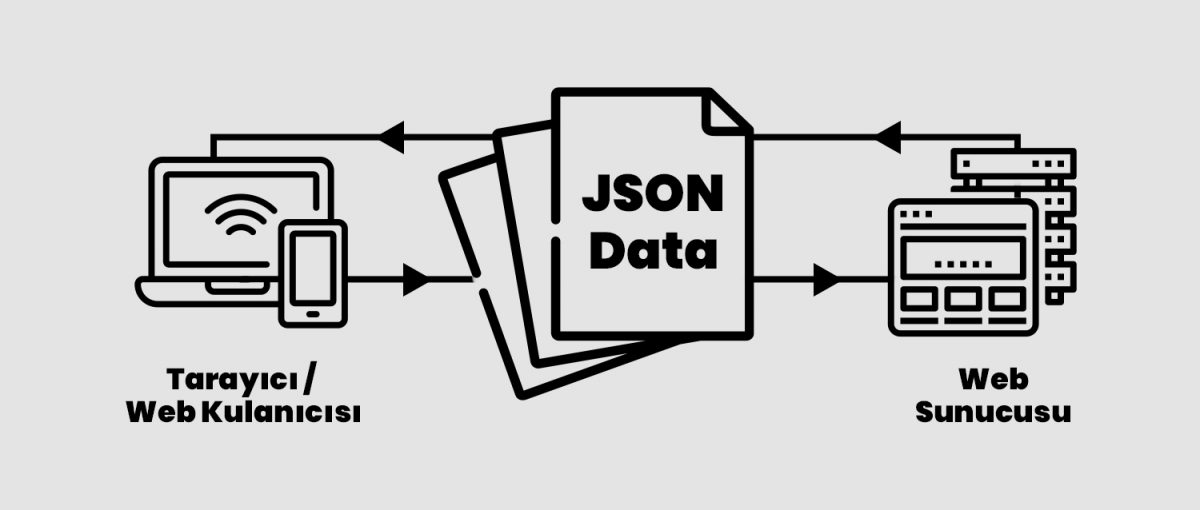


Ayrıca Flutter null olma olasılığı olan bir değişkeni direkt olarak kullanmamıza izin vermiyor. Bu değişkeni kullanırken değişken adının sonuna “!” karakteri ekleyerek bu değişkenin null olmadığını belirtebiliyoruz. Bu durumda değişkenin null olmadığından emin olma işlemi ise geliştiriciye düşüyor. Bu işlemi null bir değişkene uygulanması yine null hatası alınmasıyla sonuçlanabiliyor. Yani Flutter tamamıyla Null Safety bir dil demek bu durumda yanlış oluyor. Flutter, geliştiriciyi güvenli bir kod yazmaya itiyor fakat bunu zorunluluk olarak geliştiriciye

**2-Future Builder:** Web üzerinden gelen veya yüklenmesi zaman alacak olan verileri widgetlar içinde anında göstermek mümkün değil. Bazen çok kısa süreler de olsa bu verinin yüklenmesi için bir zaman gerekiyor ve bu veriyi henüz elimize geçmeden ekrana yansıtmaya çalışmak hatalara sebep oluyor. Flutter bu tarz durumlarda bize “Future Builder” widget’ını sunuyor. Zaman alacak işlemler örnekteki gibi asenkron bir metot içinde yapılıp Future Builder widget’ının “future” özelliğine verilince veri gelene kadar istediğimiz farklı bir widget ve veri gelince de eklediğimiz diğer bir widget kullanıcıya gösterilir. “Dio” paketi ile API verileri alınabilir.

Günümüzde ASİS gibi kurumsal şirketler mobil uygulama geliştirme kısmında Android ve IOS geliştirmelerini ayrı ayrı uygulamalardan yapıyor. Bu takip edilebilirliği kolaylaştırıyor. Google’ın arkasında sağlam bir şekilde durduğu Flutter ise yeni yeni yaygınlaşmaya başlamasına rağmen Google Ads, Alibaba ve SpaceX Go! gibi uygulamalarda kendini ön plana koydu.

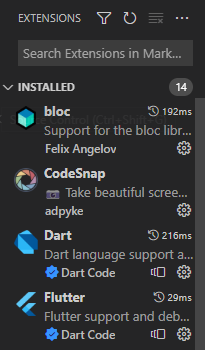
**API:** API yani Uygulama Programlama Arayüzü bir uygulamanın veya platformun paylaştığı verilere izin verilen sınırlar dahilinde dışarıdan erişilebilmesini sağlayan bir arayüzdür. API kullanımında amaç bize API erişimi veren sağlayıcımızın izin verdiği verilere erişim sağlamaktır. Kullanım sürecinde uygulamanın kendisine ihtiyaç duymadan sunulan özellikleri, fonksiyonları, içerikleri edinmek ve verilen izinler dahilinde gönderim, silme, güncelleme gibi işlemleri yapmak da mümkündür. REST API ise HTTP protokolünü (GET, POST, PUT DELETE gibi talep tipleri) kullanarak geliştiricilere pratik bir API kullanımı sunar. REST API, kullanıcı ve sağlayıcı arasında XML, JSON, HTML gibi formatlarda veri taşıması yaparak uygulamaların haberleşmesini sağlar.

**JSON (JavaScript Object Notation):** JSON, verileri depolamak ve farklı platformlar arasında taşıma için kullanılan bir veri formatıdır. Programlama dilinden bağımsız olarak farklı platformlar arasında veri alışverişi için kullanılır. 

**IDE (Integrated Development Kit):** IDE bir yazılım geliştirme projesi için gereken tüm araçları tek bir yerde birleştiren bir yazılım uygulamasıdır. IDE’ler, bir geliştiricinin uçtan uca bir program yazmak ve oluşturmak için ihtiyaç duyduğu neredeyse tüm araçları sağlar. Geliştiriciler, uygulamalarını çalıştırırken kod yazmak, yönetmek ve yürütmek için bir IDE kullanır. IDE, bir düzenleyici, derleyici ve hata ayıklayıcı sağlar ve genellikle kod tamamlama ve genel kod yönetimi gibi görevleri gerçekleştirir. IDE’ler genellikle temel düzey kod üzerine inşa edilecek framework’lerin ve öğe kitaplıklarının eklenmesine sahiptir veya buna izin verir. Yazılım sektöründe yaygın olarak kullanılan bazı IDE’ler : Microsoft Visual Studio, Eclipse, Android Studio, XCode, Visual Studio Code.

**Android Studio:** Android işletim sistemlerine özel uygulamalar geliştirmek için Google tarafından kullanıma sunulan Android Studio, C++, Java ve Kotlin gibi Native dilleri kullanma olanağı sağlar. Diğer IDE’lere göre sistem gereksinimleri ve RAM kullanımı biraz daha yüksek olan bu program, geliştirilen uygulamanın en ince detayına kadar inceleme ve düzenleme desteğini geliştiricilere sağlar.

**XCode:** XCode, Apple tarafından geliştirilmiş bir yazılım geliştirme kitidir. Hem PowerPC hem de Intel tabanlı platformlarda çalışabilen bir altyapıya sahiptir. IOS işletim sistemine sahip cihazlar için Swift dilli ile uygulama yazabilme imkanını sunar. Native uygulama geliştirme desteğini geliştiricilere sağlayan bu uygulama sadece MacOS işletim sistemlerine sahip cihazlarda çalışır.

 **Visual Studio Code:** Visual Studio Code, bir kaynak kodu düzenleyicisinin basitliğini IntelliSense kod tamamlama ve hata ayıklama gibi güçlü geliştirici araçlarıyla birleştirir. İçinde yerleşik gelen dillerin haricinde geniş eklenti kütüphanesi sayesinde kişiselleştirilebilir ve farklı teknolojileri de destekleyebilir. Git desteğini içeren yerleşik kaynak kodu kontrolünü içerisinde bulundurur. İlgili Github deposu ile ilişkilendirilerek Github’ın sağladığı hizmetlerden bu IDE içeresinden erişilebilir. İçerisinden komut satırına erişim sağlanabilir. Dikkat çeken bir özellik de kodu analiz eden, statik analiz için linters ve araçlar gibi uzantıları oluşturma olanağıdır.

**Sonuç**: Dart, Google destekli ve Flutter gibi kaliteli bir yazılım geliştirmek kitine (SDK) sahip Cross-Platform (Çapraz-Platform) bir geliştirme dilidir. Sunduğu widgetlar ile arayüz tasarımları daha profesyonel ve kolay bir şekilde yapılabiliyor. Null Safety özelliği sayesinde alınabilecek null hatalarını düşük bir seviyeye indiriyor. Paket desteği sayesinde diğer kullanıcıların hazırladıkları kodları ve widgetları kendi projelerimizde kullanabiliyor veya kendi paketlerimizi oluşturabiliyoruz. Çapraz platform olması sayesinde yazılan tek parça kod ile Android, IOS, Windows, Mac, Linux ve Web platformlarında uygulamalar geliştirilebiliyor. Web üzerinden “Dio” paketini kullanarak API’ların sağladığı verilere erişilebiliyor ve bu veriler “Future Builder” widgetı ile kullanılabiliyor. widgetlar bir ağaç yapısı içinde bulunuyor ve aralarında ebeveyn-çocuk ilişkisi bulunuyor. Widgtelar o anki durumlarının (state) değişip değişmemesine “Statefull” veya “Stateless” sınıflar içinde bulundurulabiliyor. Bir yazılımcı için kullanması oldukça zevkli ve kolay bir dil. Arayüz tasarımında Jetpack Compose benzeri bir yapı ile geliştiriciye kolaylıklar sağlıyor ve tasarımı daha profesyonel bir hale getiriyor. Yapılabilecek en basit tasarım bile Material widgetlar sayesinde güzel bir görünüme sahip oluyor.

**Başlık:** Flutter in General Usage and Interface Development

**Yapılacak Olan Çalışmala**r**:** Bir sonraki rapor döneminde yapmak istediğimiz Film-Dizi tanıtma uygulaması için hangi paketleri ve widgetları kullanacağımıza ve hangi geliştirme ortamını kullanacağımıza karar verilecektir. Arayüz tasarımının ana hatları genel olarak belirlenecek ve tasarım çalışmalarına başlanacaktır.

**Kaynakça**

[1]Kullanılacak API:<https://www.themoviedb.org/documentation/api>

[2] <https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/>

[3] <https://flutter.dev/>

[5] <https://tr.wikipedia.org/wiki/Visual_Studio_Code>