T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

PLATFORMLAR ARASI ÇERÇEVELERE GENEL BAKIŞ, FLUTTER VE REACT NATİVE PERFORMANS KARŞILAŞTIRMASI

YÜKSEK LİSANS TEZİ

Cumali TEKSÖZ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

TEMMUZ, 2021

T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

PLATFORMLAR ARASI ÇERÇEVELERE GENEL BAKIŞ, FLUTTER VE REACT NATİVE PERFORMANS KARŞILAŞTIRMASI

YÜKSEK LİSANS TEZİ

Cumali TEKSÖZ

(Y1813.010012)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Adem ÖZYAVAŞ

TEMMUZ, 2021

ONAY BELGESİ

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum “Platformlar Arası Çerçevelere Genel Bakış, Flutter Ve React Native Performans Karşılaştırma” adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya’da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (28/07/2020)

Cumali TEKSÖZ

ÖNSÖZ

Yazılım alanında hızla gelişmekte olan ve önümüzdeki süreçte hayatımızda daha fazla yer edineceğini düşündüğümüz mobil cihaz yazılımları üzerine keyifli bir çalışma olarak ele aldığımız bu çalışmada yapmış olduğum, Araştırmam boyunca deneyimiyle bana destek olan, yardımlarını esirgemeyen, yürütmüş olduğum bu çalışmanın gerçekleşmesinde yol gösterici olan danışmanım Dr. Öğr. Üyesi Adem ÖZYAVAŞ'a tüm içtenliğimle teşekkür ediyorum.

Temmuz 2020 **Cumali TEKSÖZ**

İÇİNDEKİLER

Sayfa

ONAY BELGESİ ..... **i**
YEMİN METNİ ..... **ii**
ÖNSÖZ ..... **iii**
İÇİNDEKİLER ..... **iv**
KISALTMALAR ..... **vi**
TABLO LİSTESİ ..... **vii**
ŞEKİL LİSTESİ ..... **viii**
ÖZET ..... **ix**
ABSTRACT ..... **x**
1.GİRİŞ **1**
1.1. Benzer Çalışma Örnekleri 4
1.2. Kullanılan Yazılım ve Araçlar 6
1.2.1 Yazılım geliştirme kiti (sdk) 6
1.2.2 Uygulama programlama ara yüzü (api) 6
1.2.3 Sdk ve Api arasındaki farklar 7
1.2.4 Dart yazılım dili 7
1.2.5 JavaScript yazılım dili 9
1.2.6 Çapraz yazılım geliştirme türleri 11
2. KULLANILAN UYGULAMARARIN MİMARİ YAPILARI **14**
2.1. Flutter 15
2.1.1 Widget 16
2.1.2 Flutter çalışma mimarisi 17
2.2. React Native 22
2.2.1 React Native çalışma mimarisi 23
2.3 Masaüstü Testleri 25
2.4. Mobil Testler 26
2.4.1 Test 1(Görsellerin listelenmesi ve 360 derecelik dönüşler) 26
2.4.2 Test 2(Animasyon testi) 27
2.4.3 Test3(Görsel ve animasyon testi) 28
2.4.4 Mobil test sonuçlarının değerlendirilmesi 29
3. KULLANICI DENEYİMİ **31**

4. SONUÇ VE ÖNERİLER 34
KAYNAKÇA 36
EKLER 41
ÖZGEÇMİŞ 66

KISALTMALAR

ABI : Application Binary Interface - Uygulama İkili Ara Yüzü
API : Application Programming Interface - Uygulama Programlama Arayüzü
APK : Android Application Package - Android Uygulama Paket Uzantısı
ARM : Acorn RISC Machine - ARM mimarisi RISC tabanlı işlemci mimarisi
RISC : Reduced instruction set computer - Azaltılmış Komut Seti Bilgisayarı
DOM : Document Object Model - Belge Nesnesi Modeli
VDOM : Virtual Document Object Model - Sanal Belge Nesnesi Modeli
FPS : Frames Per Second - Saniyedeki Kare Sayısı
HTML : Hypertext Markup Language - Hiper Metin İşaretleme Dili
IDE : Integrated Development Environment - Tümleşik Geliştirme Ortamı
MWC : Mobile World Congress - Mobil Dünya Kongresi
ODTÜ : Orta Doğu Teknik Üniversitesi
OpenGL : Open Graphics Library - Açık Görsel Kütüphanesi
PHP : Hypertext Preprocessor - Üstünyazı Önişlemcisi
PWA : Progressive Web Apps - İleri Web Uygulaması
SDK : Software Development Kit - Yazılım Geliştirme Kiti
TCMB : Türkiye Cumhuriyet Merkez Bankası
TÜBİTAK: Türkiye Bilimsel ve Teknolojik Araştırma Kurumu
TÜİK : Türkiye İstatistik Kurumu
WebGL : Web Graphics Library - Web Görsel Kitaplığı

TABLO LİSTESİ

Sayfa

Tablo 2.1: Test İçin Kullanılan Telefon Marka Ve Model Özellikleri. 26
Tablo 2.2: Test1, Donanımların Performans Sonuçlarının İncelenmesi. 27
Tablo 2.3: Test2, Donanımların Performans Sonuçlarının İncelenmesi 28
Tablo 2.4: Test3, Donanımların Performans Sonuçlarının İncelenmesi 29
Tablo 3.1: Katılımcı, Bilgisayar Donanımlarının Yük Verilerinin İncelenmesi 32
Tablo 3.2: Katılımcıların, 1.Uygulama Deneyim Sonuçlarının İncelenmesi 33
Tablo 3.3: Katılımcıların, 2.Uygulama Deneyim Sonuçlarının İncelenmesi 33

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1: Hanehalkı Bilişim Teknolojileri (Bt) Kullanımı 1
Şekil 1.2: Sdk İle Api Yazılımlarının Farkı. 7
Şekil 1.3: Dart Programlama Dili, Mikser Çalışma Mantığı. 8
Şekil 1.4: Örnek Bir Dart Programlama Dili Kod Satırı. 9
Şekil 1.5: Javascript Programlama Dili İle Bytecode Oluşturma Süreci. 10
Şekil 1.6: Örnek Bir Javascript Programlama Dili Kod Satırı 10
Şekil 2.1: Son 12 ay içinde yapılan bazı Çapraz Platform arama Sonuçları. 14
Şekil 2.2: Flutter Logo 15
Şekil 2.3: Widgetlar ile oluşturulmuş bir uygulama örneği. 16
Şekil 2.4: Katıştırıcı (Embedder) çalışma dosyaları. 17
Şekil 2.5: Flutter Çerçeve - Yapı (Framework) çalışma dosyaları. 18
Şekil 2.6: Flutter Motoru (Engine) çalışma dosyaları. 18
Şekil 2.7: Tarayıcı (Browser) çalışma dosyaları. 19
Şekil 2.8: Flutter Motoru ve Platform çalışma dosyaları 20
Şekil 2.9: Flutter, windget ve ağaç yapısı. 21
Şekil 2.10: React Native Logo 22
Şekil 2.11: React Native, köprüleme işleminde arayüz oluşturma adımları 23
Şekil 2.12: React Native kod çalıştırma süreci. 24

PLATFORMLAR ARASI ÇERÇEVELERE GENEL BAKIŞ, FLUTTER VE REACT NATİVE PERFORMANS KARŞILAŞTIRMASI

ÖZET

Giderek yaygınlaşan mobil cihazlar (tablet, telefon, vb.) ile birlikte yazılım sektörleri de değişmektedir. Daha doğru bir ifade ile değişime ayak uydurmak durumunda kalmışlardır. Firmalar farklı platformda bulunan müşterilerine/kişilere ulaşabilmek için, her işletim sistemine (Android, ISO, macOS, Windows) ve cihaza uygun yazılımlar geliştirmek durumunda kalmışlardır. Bu sorun, firmaları yazılım alanına ayırmaları gereken maliyeti yükseltmektedir. Farklı platformlara uygun yazılımlar hazırlanarak, bu yazılımları mobil marketlere yüklenmekte ve bu süre içinde zaman kaybı yaşanmaktadırlar. Bu sorunlara çözüm olarak geliştirilen çapraz platform yazılımlar, bu çalışma kapsamında ele alınarak incelenmiştir. Çapraz Platform yazılımlar veya bir diğer adı platform bağımsız yazılımlar farklı işletim sistemlerinde çalışabilen yazılım geliştirme kitleridir. Platform bağımsız yazılımlar ile geliştirilen uygulamalar içindeki kütüphane dosyaları sayesinde tekrarlanabilir kod blokları oluşturarak bunları kullanabilmektedirler. Bu tekrarlama işlemi sayesinde işletmelere maliyetten kazanç sağlarken, yazılım geliştiricileri için ise zamandan tasarruf sağlamaktadır. Gelişmiş birçok Çapraz Platform içerisinde seçilen iki Yazılım Geliştirme Kiti (SDK) Flutter ve React Native yazılımlarının çalışma mimarisi incelenerek farklı cihazlar üzerinde testler yapılmıştır. Yapılan bu testler sonucu, İşlemci, Ram bellek, Batarya, FPS, ve Uygulamaların bellek miktarları tespit edilmiştir. Testlerden elde edilen veriler karşılaştırılıp değerlendirilmiştir. Aynı zamanda bu çalışmada Yazılım Geliştirme Kitleri (Flutter ve React Native) avantaj ve dezavantajları incelenmiştir. Değerlendirmelerde kullanıcı deneyimlerinin de önemi göz önünde bulundurularak üniversitede ilgili bölümlerde (ön lisans, lisan ve yüksek lisans) eğitim gören veya sahada yazılım alanında çalışmakta olan kişilerden iki platform ortamında yapılan yazılımlar ile deneyimlerinin paylaşılması istenerek toplanan veriler üzerinde değerlendirmeler yapılmıştır.

Anahtar Kelimeler: *React Native, Flutter, SDK, Çapraz Platform Yazılımlar*

CROSS-PLATFORM FRAMEWORKS OVERVİEW, FLUTTER AND REACT NATİVE PERFORMANCE COMPARİSON

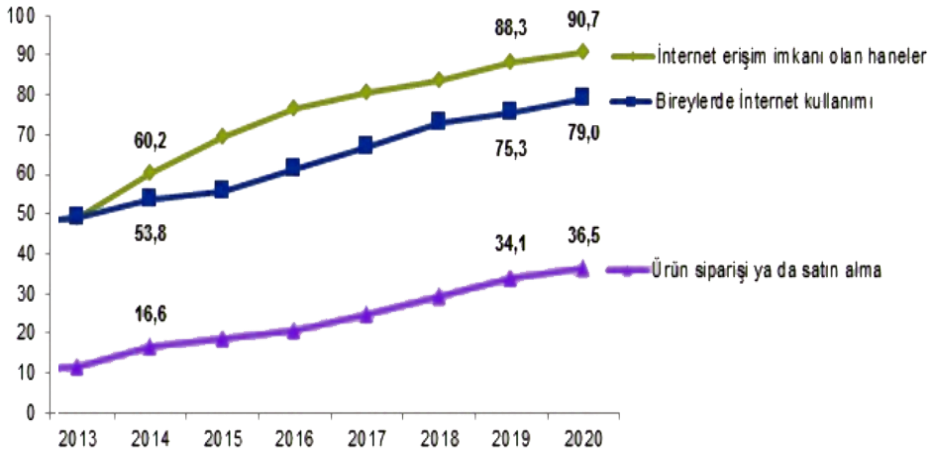
ABSTRACT

Along with the increasingly widespread mobile devices (tablets, phones, etc.), the software industry had to keep up with this change. Companies need to develop software suitable for each operating system (Android, ISO, macOS, Windows) and device in order to reach their customers on different platforms. This problem, in addition to allocating high costs to the software field, also causes loss of time in the process of preparing software suitable for different platforms and uploading it to the market. Cross-platform software which is developed as a solution to this problem has been examined within the scope of this study. Cross-platform software, or platform-independent software, is software development kits that can run on different operating systems. Thanks to the library files in the applications developed with platform-independent software, they can create repeatable code blocks and use them. Thanks to this repetition process, while providing cost savings to businesses, time is saved for software developers. The working architecture of two Software Development Kits (SDKs) Flutter and React Native, selected from among many advanced Cross-Platforms, was examined and tests were carried out on different devices. As a result of the tests, performance tests have been made to compare the amount of CPU, RAM memory, Battery, FPS, and memory of the applications. The data obtained as a result of these tests were compared and evaluated. At the same time, the advantages and disadvantages of the mentioned Software Development Kits (Flutter and React Native) are examined. Considering the importance of user experience in the evaluations, people studying at the relevant departments (associate, undergraduate and graduate) at the university and working in the field of software were asked to share their experiences with the software made in two platforms and included in order to increase objectivity in the evaluation of software development kits.

Key Words: *React Native, Flutter, SDK, Cross-Platform Software.*

1.GİRİŞ

Bulunduğumuz çağın ve özellikle dönemin pandemi (COVİD-19 Salgını) koşulları göz önünde bulundurulduğunda mobil cihaz ve doğal olarak mobil uygulamaların kullanım oranı her geçen gün artmaktadır. Piyasada var olan en büyük iki marketin (Play ve App Store) piyasadaki ekosistemin %90'lık oranın üzerinde hâkimiyeti bulunmaktadır (Url-1, 2021). Daha önce birçok şirket için mobil uygulamalar ve sanal ortamda bulunmak tercih konusu iken, koşulların bu durumu biraz daha mecburiyet boyutuna taşıdığı görülmektedir. Dünyada olduğu gibi bu durum ülkemizde de ODTÜ ve TÜBİTAK ortaklığı ile başlatılan internet hareketi, yirmi yılı aşkın süredir giderek ağını genişletmektedir. İnternet hareketi günümüzde birçok sektör tarafından artık vazgeçmesi zor bir hal almasına neden olduğu tespit edilmiştir (Ünaldı, 2019). Türkiye İstatistik Kurumu tarafından paylaşılan veriler internet kullanım oranının 2013 ile 2020 yılları arasında yapılan, internet ortamında ürün satın alma işlemlerinin, son yedi yılda iki katını aştığını yaptığı araştırmalar sonucu tespit edilmiştir (Url-2, 2020).



Şekil 1.1: Hanehalkı Bilişim Teknolojileri (BT) Kullanımı

Kaynak: Url-2, (2020)

Şekil 1.1'de Türkiye'de internet erişim imkânı olan evlerin 2013-2020 yılında Nüfusa oranla %49 iken %90,7'e, Bireysel internet kullanımı %49'dan %79'a ve internetten ürün satın almanın %11'den %36,5'e çıktığı görülmektedir (Url-2, 2020). İnternet kullanımı arttıkça firmalar bu pazara kayıtsız kalamamakta ve bu pazardan pay almak için birbirleri ile yarışmaktadırlar. Bu pazardan pay sağlamak amacıyla birçok farklı platform araştırmalara konu olmuştur. Bunların başında ise internet tabanlı yazılımlar gelmektedir. İnternet tabanlı yazılımların bugünkü tahmini değerinin 6,5 trilyon dolarlık bir pazar olduğu kabul edilmektedir (Url-3, 2021). İnternet tabanlı yazılımların kendi içinde ayrışmakta ve farklı alanlara hizmet vermektedirler. İnternet sayfaları veya mobil uygulamalar oluşturan yazılımlar bu türün yazılım örneklerinin başında gelmektedir (Url-4, 2020). Fakat kurumsal firmalar genellikle bu iki platformu doğal iki unsur gibi görmekte ve bunları tek çatı altında oluşturma yoluna gitmektedir. Ama bilinmelidir ki bu durum bilinenin dışında birçok yan bağlantı ile oluşturulmaktadır. Akıllı telefonlar, otonom araçlar, tabletler, smart TV gibi birçok farklı platformlar bu yazılım bağlantılarına örnek verilebilinir (Biørn-Hansen, Rieger, Grønli, Majchrzak ve Ghinea, 2020). Çok sayıdaki platform ve bu platformlara hâkim yazılım dillerine hâkim personel çalıştırılması ve yönlendirmesinden oluşacak trafiğin yanı sıra, yapılacak işlemler üzerinde büyük bir kırtasiye sorunu çıkarmaktadır. Örneğin yapılan iş koluna göre bir web sayfası hazırlanması için bir yazılım dili (php,java,html vb.) tercih edilmesi gerekmektedir. Bu yazılım diline hâkim bir kişi (personel) ile anlaşılmalı veya mobil uygulama oluşturmak isteyen kurumların/kişilerin yazılmak istenen uygulamayı oluşturacak yazılım mühendisleri ile anlaşmaları gerekmektedir. Yazılımın oluşturulması için oluşturulan ekibin kurulmasının ardından, karşılarına bu defa farklı yazılım firmaların kullandıkları farklı işletim sistemleri sorunu çıkabilmektedir. Yazılım sektörüne bakıldığında en çok kullanılan, mobil işletim sistemleri arasında yer alan en az üç platform karşılaşılmaktadır. Bunlar Android, IOS ve Windows işletim sistemleri gibi mobil işlem payı yüksek ve çok farklı kullanıcılar tarafından kullanılan işletim sistemi yazılımlardır (Swarna, Purnama ve Anthony, 2020). Tüm bu işlemlerin gerçekleştiği senaryolarda ise; farklı platformlarda yazılım geliştiren geliştiricilerin birbirleri ile iş birliğinde içinde olmaları gerekmektedir. Bu kişilerin birlikte uyum içinde çalışması da bu işin bir başka zorlu boyutunu göstermektedir (Aldayel ve Alnafjan, 2017). Tüm bu zorluklar ve maliyetler kendi içinde küçük işletmelerin büyümesinin önünde birer engel olarak

küçük işletmeler zorlarken, büyük işletmeler için ise bu maliyetler ek yükler doğurmaktadır (Dehlinger ve Dixon, 2011). Tabi ki bu durum yazılım işini meslek edinmiş kişiler tarafından da sorun yaratmaktadır. Bu tür dillere hâkim olmak başlı başına bir zorluk iken, bu dillerin başka yazılım ekibi ile anlaşılarak eşzamanlı bir şekilde ilerlemek daha da büyük bir problem olabilmektedir (Oulasvirta, Wahlström, ve Ericsson, 2011). Bu tezin amacı bu zorlu süreçleri ve engelleri hafifletmek adına üretilmiş yazılımlardan olan Çapraz Platform yazılımları arasında benzerlik ve farklılıkları göstermek ve bunların tercih edilirken nelere dikkat edilmesi gerektiği hakkında bir çözüm önerisinde bunulmaktır. Flutter, React Native, Cordova, Xamarin, PWA gibi SDK'lardan ikisi Flutter, React Native test edilerek karşılaştırılmaktadır. Seçtiğimiz iki SDK uygulaması üzerinde yapılan stres testleri bize uygulamalarımızın yüksek veri trafiği altında kullandıkları donanım kaynakları hakkında bilgiler incelenmiştir. Yapılan stres testler iki platform (Flutter SDK ile React Native SDK) için hazırlanarak, İşlemci, Ram bellek, Batarya, FPS, ve Uygulamaların bellek miktarları karşılaştırılmıştır. Ayrıca farklı cihazlarda farklı sonuçlar elde edileceği düşünülerek Android, IOS, Masaüstü ve Emülatörler ile veriler toplanarak incelenmiştir. Bu testler ile birlikte kullanılan yazılım geliştirme kitlerinin mimari yapısı incelenerek birbirleri ile olan benzerlik ve farklılıklar tespit edilmiştir. Tez çalışması kapsamında gerekli olan anket çalışması için İstanbul Gedik Üniversitesi Rektörlüğünden, bilgisayar alanında uzmanlaşmış, üç danışman tarafından olay alınarak hazırlanan "Çapraz Platform Yazılımlar Kullanıcı Deneyim Anketi" yapılması adına izin alınmıştır (Ek-A). Ankete uygun denekler (Üniversitelerin ön lisans, lisan ve yüksek lisans bölümlerinde tercihen yazılım alanında eğitim gören veya yazılım alanlarına ilgisi olan ve sahada yazılım alanında çalışmakta olan kişilerden) ilgili bölümlerden seçilerek belirlenmiştir. Belirlenen denekler/kullanıcılar Flutter ve React Native uygulamaları ile ikişer aynı özellikle yazılım (Her deneğin aynı bilgiye sahip olmadığı ve daha önce bu uygulamaları kullanmadıkları ihtimalide göz önüne alınarak giriş seviyesinde uygulama geliştirmelerinin uygun olacağı düşünülmektedir. Sonuç itibari ile kullanıcılara yeni bir uygulama geliştirmekten çok geliştirmek istenilen uygulamaları hangi SDK üzerinde geliştirebilecekleri ihtimali üzerinde tercih yapmaları beklenmiştir) yazılması istenilen ve bu işlemlerin ardından deneyimlerini Google Form üzerinden online yapılan ankette, Url-9'linki üzerinden katılarak paylaşmaları sağlanmıştır. Anket sonucunda elde edilen veriler ile yapılan stres testlerinden elde edilen veriler

işlenerek, iki platform arasındaki benzerlik ve farklılıklar belirlenerek, verilerin objektifliği desteklenmiştir.

1.1. Benzer Çalışma Örnekleri

Konu hakkında yapılan bazı araştırmalara bakıldığında ise; Goetz, J., & Li, Y., 2018 yılında "Mobil Uygulamalar için Platformlar Arası Çerçevelerin Değerlendirilmesi" isimli çalışmada Xamarin ve PhoneGap kullanarak bir uygulama geliştirmiştir. Yazılmış olan yazılımlar IOS ve Andorid işletim sistemlerine yüklenerek dört farklı teste üzerinden değerlendirilmiştir. Ön yükleme sürelerinde, aynı cihazlarda Xamarin uygulaması ile geliştirilen sistemin daha hızlı olduğu belirtilmiştir. PhoneGap ile geliştirilen yazılımların boyutunun daha küçük olduğu tespit edilmiştir. Ram bellek ve işlemci kullanım miktarları benzer sonuçlar elde edilmiştir. Phonegap yazılımı ile geliştirilen uygulamaların hata ayıklama işlemlerinde daha başarılı olduğu tespit edilmiştir. Xamarin yazılımının yazılım geliştirmek için daha başarılı bulduklarını fakat Phonegap'ı daha az kod satırı ile aynı işlemleri gerçekleştirile bilineceği sonucuna varılmıştır. Gonsalves, M., 2019 yılında "Mobil Geliştirme Çerçevelerinin Değerlendirilmesi Apache Cordova Ve Flutter Ve Etkileri Gelişim Süreci ve Uygulama Özellikleri" isimli çalışmasında, Flutter SDK'sının; Cordova ve Apache SDK'ları üzerindeki etlilerine bakılmıştır. Flutter uygulamasının bellek kullanım miktarı Cordova ve Apache göre daha fazla olduğunu tespit edilmiştir. Flutter diğer iki yazılıma göre daha yeni olmasına karşılık, yapılan testler sonucu daha başarılı bulunduğunu tespit etmiştir. Çalışmanın tüm yazılımların kendine has özelliklerine değinerek, geliştirilecek yazılımların türüne göre sonuçlarının farklı olabilmesi muhtemel olduğu sonucuna varılmıştır. Dhillon, S., ve Mahmoud, Q. H., 2015 yılında " Platformlar arası mobil uygulama geliştirme araçları için bir değerlendirme çerçevesi" isimli çalışmasında, mobil uygulama geliştirme sürecinde yaşanılan zorluklara değinmektedir. Yapılan testler ile bir çok farklı sonuçla karşılaşıla bilineceği fakat en önemli olarak değerlendirilmesi gereken kısmın işlevsellik olduğu sonucuna vardığını ifade etmektedir. Geliştiricilerin de farklı platformlara yazılım geliştirmelerinde yaşadıkları en büyük sorunun aynı işlemler için harcana zaman kayıpları olduğu ve bu sorunun çözümü için çapraz platformların önemine değinilmektedir. Xanthopoulos, S., ve Xinogalos, S., 2013 yılında yaptıkları "Mobil uygulamalar için platformlar arası geliştirme yaklaşımlarının karşılaştırmalı bir

analizi" isimli çalışmada, batarya, internet bağlantı hızı, geliştirme sürecinde yaşanan sorunlardan daha fazla işlevselliğin önemli bir nokta olduğundan bahsetmektedir. Bu nedenden dolayı çapraz platformların türleri analiz edilmiştir. Üç test işlemi gerçekleştirilerek çapraz platform yazılımları ile mesaj ve multimedya uygulamaları hazırlanmıştır. Yapılan testler ile Android kiti ve Cordova kitleri ile hazırlanmıştır. Yapılan testler ile yapılmak istenen uygulamanın türüne göre yazılım geliştirme kitlerinin seçilmesi gerektiği sonucuna ulaştıklarını bildirmektedirler. Nedyak A.V., Rudzeyt O.U., Zainetdinov A.R. ve Ragulin P.G., 2020 yılında yapmış oldukları "Mobil Platformlar Arası Uygulama Geliştirme Araçları" isimli çalışmasında, React Native, Xamarin ve Flutter platformlar arası yazılım geliştirme gözden geçirmektedir. Bu yazılıları bir birinden bağımsız olarak değerlendirip avantaj ve dezavantajlarında değinmektedir. Sonuç olarak ise; bilgi görüntüleme işlemler, e-ticaret uygulamaları, mesajlaşma uygulamaları vb. işlemler için uygun bulunurken. Daha komplike işlemler gerektiren yazılımlar ve oyunlar için bu yazılımların uygun olmadıklarına değinmektedirler. Nawrocki, P., Wrona, K., Marczak, M., ve Sniezynski, B., 2021 yılında "Mobil Uygulamalar için Yerel ve Platformlar Arası Çerçevelerin Karşılaştırması" isimli çalışmada Native, React Native ve Flutter yazılım geliştirme kitlerini, mobil cihazlar için uygulama geliştirmede kullanılan yöntemlerini incelemişlerdir. İki çapraz platformu analiz ederek ve çeşitli araçlar kullanılarak geliştirilen özdeş uygulamaları ile karşılaştırmaktadırlar. Özgün uygulamaların kullanıcılar tarafından tercih edildiğini fakat maliyet ve uzman eksikliğinden firmaların kullanamadıklarına değinmektedir. Bu kısımda maliyeti azaltmak için çapraz platformların kullanılabilirliğine değinmektedir. Dos Santos, D. S., Nunes, H. D., Macedo, H. T., ve Neto, A. C., 2019 yılında yapmış oldukları "Platformlar Arası Mobil Geliştirme Çerçevesi için Öneri Sistemi" isimli çalışmada, kişilerin mobil uygulamalarını en fazla sayıda platforma ulaştırmayı hedeflediklerini fakat bunun zorluklarından bahsetmektedir. Her platform için aynı özel uygulamayı oluşturmak daha fazla finansal, zaman ve işgücü yatırımı gerekmektedir. Bu sorunları azaltmak için, çoklu platformlar için mobil geliştirme çerçevelerin ortaya çıktığını ve aynı kaynak koddan farklı platformlara uygulamalar oluşturmayanın mümkün olduğunu söylemektedir. Fakat her çözüm kendi içinde yeni sorunlarda yarata bilmektedir. Yapılan testler ve anketler sonucunda yapılmak istenen proje türüne göre yazılım geliştiricilerin farklı mobil çapraz platformları seçmeleri gerektiği sonucuna ulaşmışlardır.

1.2. Kullanılan Yazılım ve Araçlar

Uygulama geliştiricileri uygulama türüne göre farklı yazılım özelliklerine ihtiyaç duymaktadırlar (Gross, Gulliksen, Kotzé, Oestreicher, Oestreicher, Palanque, Prates ve Winckler, 2009). Bu çalışma kapsamında kullanılan yazılım dilleri, yazılım geliştirme kitleri ve yardımcı yazılımlar hakkında bilgi sahibi olmak, ilerleyen konuların anlaşılması açısından daha verimli olacaktır. Bu kısımda kullanılan yardımcı uygulamalar ve yazılım dillerinin neler olduğuna ve çalışma mantığına değinilmektedir.

1.2.1 Yazılım geliştirme kiti (sdk)

Yazılım Geliştirme Kiti (Software Development Kit), yazılım geliştiricileri kodlama yaparken bazı standartlara ve kılavuzlara ihtiyaç duyulmaktadır (Koranne, 2009). Bu standartları ve kolaylıkları sağlayan yazılımlar SDK olarak tanımlanmaktadır. SDK'lar geliştiricileri yönlendirecek olan kütüphaneler, örnek kod blokları ve belgelerden oluşmaktadır. Bu kodlar bir nevi ihtiyaç kiti olarak da değerlendirebiliriz (Gross, Gulliksen, Kotzé, Oestreicher, Oestreicher, Palanque, Prates ve Winckler, 2009). Uygulama geliştirilirken, farklı işlemler ve işletim sistemleri için farklı özelliklerde SDK'lar kullanılmaktadır (Zammetti, 2013). Bir uygulama kullanılırken işlemleri hızlandırmak için bir SDK kullanıla bilineceği gibi birden fazla SDK kullanmak da mümkündür (Fayzullaev, 2018). Bu uygulamalar içindeki API, hata yakalama ve hatayı ayıklama gibi yardımcı yazılımları da içermektedir. Birçoğu sistemimize kurduğumuz Tümleşik Geliştirme Ortamı (IDE) ile birlikte sistemimize kurulurlar (Perchat, Desertot ve Lecomte, 2011). SDK yazılımlarına örnek vermek gerekir ise Android SDK, IOS SDK, Windows SDK, React Native SDK ve Flutter SDK yazılımları en çok bilinen SDK örnekleridir (Dagne, 2019).

1.2.2 Uygulama programlama ara yüzü (api)

Uygulamalarda, bazı eklentiler ve kolaylıklar sağlanması adına veya başka servis sağlayıcıları programımıza entegre edilebilmesi için daha önce oluşturulmuş veya oluşturulabilinen yazılımlara API denilmektedir (Upadrista, 2021). Örneğin, bir web sayfamız veya uygulamamız olduğunu düşünelim bu sayfada anlık döviz verilerini kullanıcılarımız ile paylaşmamız veya satın alınacak bir ürün için döviz işlemleri

yapılması gerekmektedir. Bu işlemi sürekli olarak ülkemizde TCMB *"https://www.tcmb.gov.tr/kurlar/kurlar_tr.html"* (erişim, 2020), internet sayfası üzerinden anlık takibinin gerçekleştirmesi gerekmektedir. Bu şekilde yapılacak işlem oldukça zaman alıcıdır. TCMB web sayfası üzerinden doğrudan anlık veri güncellemelerini dinleyip bunları sayfamızdaki uygun gördüğümüz alana veya hesaplamada kullanmak üzere veriler çekilip sayfamızda kullanabileceğimiz olanaklar sağlayan yazılımlardır.

1.2.3 Sdk ve Api arasındaki farklar

SDK ve API yazılımlarına yeni başlayan veya bu yazılımlar ile çok fazla uğraşmamış kişilerin bu yazılımı sıkça karıştırması veya aynı zannetmesi mümkün olmaktadır. Bu sebeple bu yazılımların birbirinden farklarına kısaca değinmek daha sağlıklı olacaktır.

Şekil 1.2: Sdk ile Api yazılımlarının farkı.

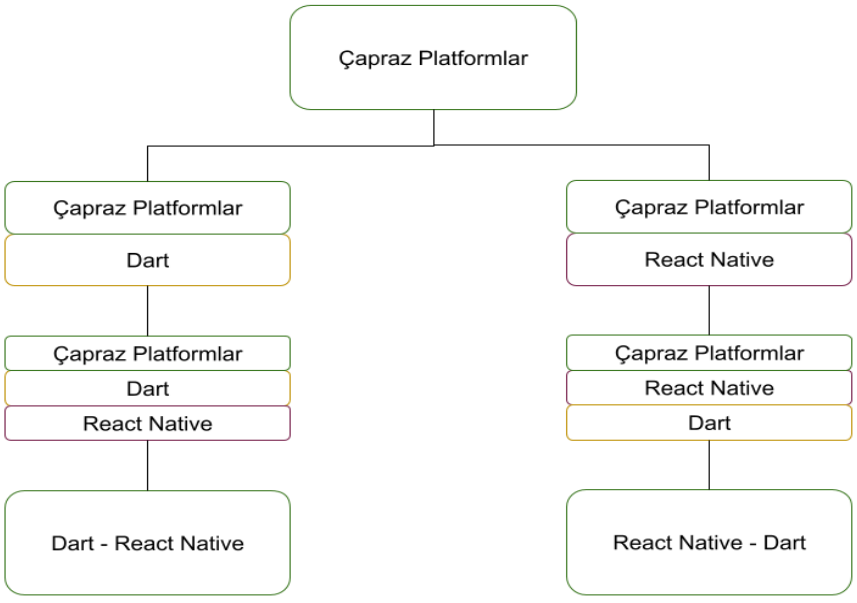
Bu durumun birbirine karışmasının en büyük nedenlerinden biri birçok zaman karşımıza çıkan SDK dosyalarının içindeki API dosyaları olmuştur. Matematiksel bir örnek vermek gerekirse iki küme düşünelim ve bu kümelerimizin biri rasyonel sayılar ile diğer kümemiz ise doğal sayılar ile doldurulmuş olunsun. SDK dosyalarını rasyonel sayılar (…,-70,-63,-25,-13,-1,0,1,13,25,63,70, …) kümesi, API dosyalarına ise doğal sayılar(2,3,13,19,71) kümesi diyelim. Rasyonel sayıların içinde doğal sayılar da bulunmaktadır. Fakat doğal sayılar kümesinde bütün rasyonel sayılar mevcut değildir, daha sınırlıdır ve daha özel bir işlevi mevcuttur. Yani genellikle SDK dosyalarında API dosyaları mevcutken API dosyalarından bir SDK oluşturulması henüz mümkün değildir.

1.2.4 Dart yazılım dili

Dart dili Google tarafından nesne yönelimli programlama dili olarak ilk kez 2011 yılında kullanıcılara tanıtılmış açık kaynak kodlu bir yazılım dilidir. İlk kullanım amacı Google Chrome üzerinde çalışmalarda kullanılmak istenmiş ise de gerekli geliştirmeler sonucu piyasada giderek büyüyen bir pazar payına sahip bir yazılım dili

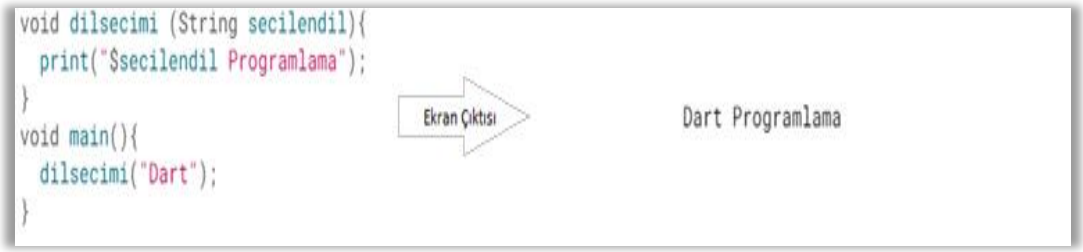
olmayı başarmıştır (Hassan, 2020). Windows, Macos ve Linux işletim sistemlerinde çalışan editörler hazırlayarak, kodların otomatik tahmini, hata ayıklama ve Syntax highlighting özellikleri eklenmiştir (Biessek, 2019). Web uygulamalar, mobil uygulamalar ve diğer alanlarda, kendini ispatlamaya çalışmaktadır. Yeni bir dil olmasına rağmen Flutter yazılım dili ile birlikte çalışmasından dolayı pek çok kişi bu dili merak edip öğrenmeye çalışmaktadır (Bkz. Şekil 1.7). Dart dilinin birçok yazılım dili ile ortak yanları olmasının yanı sıra onu farklı yapan özellikleri ile yazılım programlayan Kişiler arasında tercih gücü artmaktadır (Meiller, 2021).

Dark SDK yazılımı kurulduktan sonra farklı amaçlara hizmet eden derleyici kütüphaneleri, kendine özgü paketleme seçeneği ve diğer yardımcı kütüphaneleri ile birlikte dikkat çekmektedir (Mohanty ve Dey, 2014). Çalışma mimarisine bakılacak olunur ise, içinde derlediği yazılımları, kütüphaneleri aracılığı ile çalışmakta olduğu cihazın yerel işletim sisteminde çalışacak şekilde dönüştürmektedir. Dart ile yazılmış yazılımlar web tarayıcılarında çalıştırıldığında içinde bulunan *"dart2js"* kütüphanesini kullanarak dosyadaki kodları JavaScript dilinde derlemektedir (Url-1, 2021). Dart, miras almak yerine arabirimler mantığı ile hareket etmektedir. Örneğin, üst sınıflara sırası ile ulaşmak yerine daha hızlı sonuç elde etmek için dosyalarını arabirimmiş gibi sınıflandırmaktadır (Hassan, 2020). Bu işlemi Dart mikser olarak isimlendirmektedir. Bu aşamada sınıflar birbirlerine hiyerarşi olarak üstün değildirler (Ahmad, 2020).



Şekil 1.3: Dart programlama dili, mikser çalışma mantığı.

Şekil 1.3'de görüldüğü iki farklı zincirde bulunan verilerimiz haberleşmek istediklerinde sürekli olarak en üst zincire ulaşıp oradan geri dönmek yerine kendine has bir yapı oluşturarak yapmak istediği işlemin süresini kısalttığı gibi daha az donanım kullanmasına da yardımcı olup bekleme süresini azaltmaktadır (Mohanty ve Dey, 2014). Son olarak örnek bir kod bloğu ile dart dilini geçebiliriz.

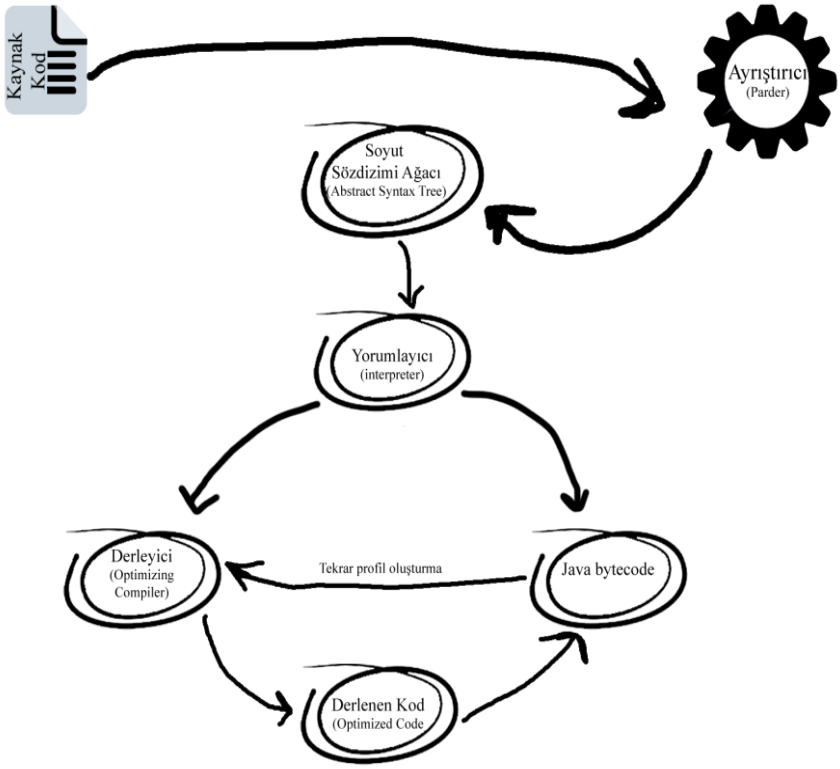


Şekil 1.4: Örnek bir Dart Programlama Dili Kod satırı.

1.2.5 JavaScript yazılım dili

JavaScript(JS) yazılım dili, ilk olarak 1996 yılında yazılmaya başlanmış olan ECMAScript isimli programlama dilinin isim değiştirip JavaScript olarak devam eden oldukça köklü bir dil yapısıdır (Hassan, 2020). İlk amaçları arasında farklı tarayıcılarda çalışmak olsa da ilerleyen yıllarda önce masaüstü daha sonra mobil cihazlarımızda da çalışır şekilde güncellenerek kullanım alanı genişletilmiştir. Yaklaşık olarak çeyrek asırlık sürede piyasadaki web sayfalarının %90 üzerinde kullanılan kod bloklarının tamamında veya bir kısmında yer alarak en çok kullanılan diller arasında yerini korumayı başarmıştır (Mohanty ve Dey, 2014).

JavaScript programlama dili ile tanışan birçok kişi ilk olarak Java dilinin aynı olduğunu veya arasında nasıl bir fark olduğu konusunda karışıklık yaşamıştır. Fakat Java ve JS iki farklı yazılım dilidir. Java, nesne yönelimli ve bağımsız ortamda çalıştırılabilen bir makine programlama dilidir (Mccutchan, Feng, Matsakis, Anderson ve Jensen, 2014). JS ise, betik (Script Language) bir dildir. Betik diller yorumlana bilen ve bir derleyiciye ihtiyaç duymadan doğrudan çalışabilen yazılım dillerine verilen isimdir. JS, C dili ile geliştirilmiş HTML ve CSS ile düzenlenmiş oldukça zengin kütüphanelerden oluşmaktadır (Robbins, 2012). Bu kütüphaneler sayesinde Web sayfaları, FrontEnd Programlama, farklı platformlardan veri alış verişinde bulunma gibi birçok açıdan kullanıcıya kolaylık sağlamaktadır (Scott, 2020).



Şekil 1.5: JavaScript programlama dili ile bytecode oluşturma süreci.

Şekil 1.5'de de görüldüğü üzere JS kod çalışmaya başlandığı anda kod blokları ayrıştırıcıya gönderilip bir ağaç yapısı oluşturulur. Oluşan ağaç yapısı yorumlanarak bytecode oluşturur, bu kod aslında JavaScript motorundan gelen kod işlemidir (Hassan, 2020). Bu adımdan sonra işlemler optimize edici derleyiciye gönderilir yapılan işlem makine kodu oluşturarak arkasından gelen işlemler için işlemi hızlandırmaktır. Yapılan işlemlerde hata tespit edildiğinde işlemlerin tekrar derlenmesi için işlemleri yorumlayıcıya gönderilmektedir. Daha sonra başarılı bir şekilde derlenen işlemler ara yüze aktarılma adımları başlatılır (Novick, 2017). Son olarak örnek bir kod bloğu ile JavaScript dilini geçebiliriz.

```
function dilsecimi(secilendil) {
  alert(secilendil + ' Programlama');
}

dilsecimi('JavaScript');
```

Ekran Çıktısı

JavaScript Programlama

Şekil 1.6: Örnek bir JavaScript Programlama Dili Kod satırı

1.2.6 Çapraz yazılım geliştirme türleri

Bilgisayar programlama dilleri ilk olarak tek bir platform üzerinde çalıştırılmak için planlanmaktaydı fakat zaman içinde gelişen teknolojik hareketler bu alanda da yenilikler yapılmasının gerekliliğini ortaya çıkarmıştır (Blanco ve Lucrédio, 2021). Günümüzde kullanıcılar pek çok farklı platformu (Web (internet Sayfaları), Mobil(uygulamalar), Masaüstü (Programlar) aynı anda kullanmakta ve bundan dolayı hizmet aldıkları sayfalara tüm platformlardan erişmeyi talep etmektedirler. Bu talebi karşılamak isteyen firmalar bu durumdan doğan yüksek maliyet ve güncellemelerin tüm platformlara ayrı ayrı yapılmasını gerekli görmektedirler (Trisnadoli ve Lestari, 2017). Bu durumu fark eden yazılım şirketleri (Google, Facebook, Microsoft vb.) ilk olarak 2008 yılında Çapraz platform yazılımlar (Cross - Platform Software ya da Platform Bağımsız Yazılım) fikri üzerinden yazılım geliştirmeye başlamışlardır (Hartmann, Stead ve Degani, 2011). Çapraz platform yazılımlar tek bir platformda yazılan yazılımların farklı işletim sistemlerinde çalışma olanağı sağlayan bir çeşit yazılım modelidir (Allen, Graupera ve Lundrigan, 2010). Bu tür yazılımlar birden fazla cihazda ve işletim sisteminde çalışabilmektedirler. Bu tür uygulamalar kendi arasında da yaptıkları işlemler veya hitap ettikleri kullanıcılar bakımından farklılık göstermektedir (Raj ve Tolety, 2012). Bunları kendi içinde dört kategoride göstermek gerekirse:

- Çapraz Platform Uygulamaları
- Hibrit Uygulamalar
- Yerel/Derlenmiş Uygulamalar
- Mobil ve Web Uygulamalar

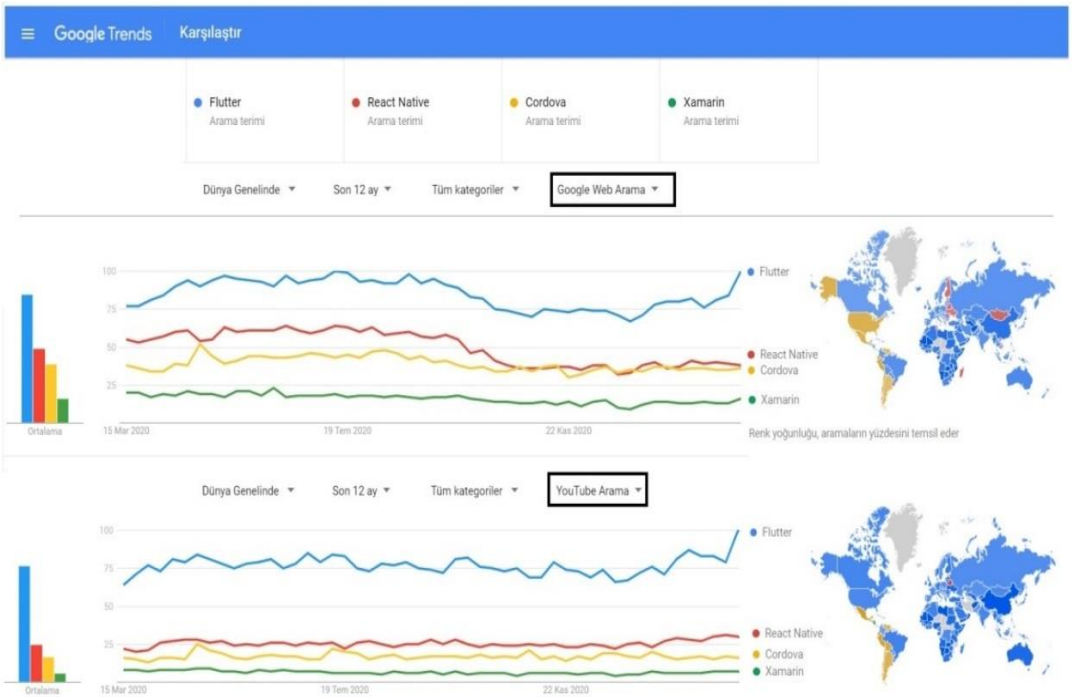
Bu uygulamalar hakkında kısaca bahsetmek konunun bütününe hâkim olmak adına yararlı olacaktır. Çapraz platform uygulamalar, en belirgin özellikleri arasında yazılan kod bloklarının yeniden veya daha anlaşılır bir ifade ile farklı işletim sistemlerinde kullanılmasına izin veren uygulamalardır (Scott, 2020). Bu yazılımlar içinde çalışacak uygulamaları işletim sisteminin dosyalarına gömülü olarak kullanılmaktadır. Uygulamaların kullanılabilmesi için gerekli olan kütüphane dosyalarını ise öncesinde sisteme kurmak gerekmektedir. Çalışma şekilleri içindeki kütüphane dosyalarını doğrudan kullandığı gibi bazı uygulamalarda ise köprü kurarak yazılımları en verimli şeklide çalıştırmayı amaç edinmişlerdir (Adinugroho, Reina ve Gautama, 2015). Bu çalışma konusu kapsamında yer verdiğimiz iki

SDK'da bu uygulamaya örnektir. Hibrit uygulamalar genellikle HTML yazılım dili ile ara yüzleri kodlanmış yazılımlar olarak değerlendirilmektedir (Wang, Liu, Guo, Chen, Zhang, Xu ve Hong, 2017). Peki tüm HTML ile yazılmış uygulamalar, "Hibrit Uygulama olarak değerlendirilebilir mi ?" sorusunun yanıtı, -Hayır olarak verilebilir. Örneğin web tabanlı yazılımlar genellikle HTML yazılım dili kullanmaktadır. Bu yazılım dili ile hazırlanan sayfalar web tarayıcılarında çalışmaktadır. Hazırlanan yazılımın Hibrit olarak değerlendirebilmemiz için aynı zamanda mobil cihazlarımız ile de uyumlu çalışması gerekir. Mobil cihazlarımız ile de uyumlu çalıştırabilen yazılımlar Hibrit uygulama olarak değerlendirilmektedir (Perchat, Desertot ve Lecomte, 2011). JavaScript, Html ve CSS kodları içeren bu uygulamaların diğer web uygulamalarından en büyük farklarından bir diğeri ise diğer HTML ile yazılmış uygulamalarımızı kurulu/çalıştığı cihazın teknik özellikleri hakkında bilgi sunmaz iken bu yazılımlar ile bu tür bilgilere de ulaşmak mümkündür (Trisnadoli ve Lestari, 2017). Yerel/Derlenmiş Uygulamalar, Çapraz Platform Uygulamaların iş mantığına benzer bir çalışma şekli mevcuttur. Çalışırken bir köprüleme işlemleri ile uygulamayı çalıştığı makinenin diline çevirir. Genellikle JavaScript dili ile yazılan uygulamalardır. Kullanılan en eski yöntemlerden biri olan ve kendine has oluşturulmuş yollar ile hızlı ve güvenli olarak değerlendirilebilinmektedir. Fakat kodlama aşamasında tüm adımları ve kısıtlamaları belirlemeniz gerekmektedir. Bu durum farklı sürüme sahip işletim sistemlerinde kısmi sorunlar yaratabilmektedir. Farklı platformlarda geliştirilen yazılımların birbiri ile uyum aşamasında sorun yarabilmektedirler (Shen ve Abraham, 1998). Mobil Web Uygulamalar, Kullanıcı ara yüzleri ile ön plana çıkmaktadırlar. Özellikle farklı yaş ve meslek gruplarına hitap edilen uygulamalarda kullanıcı ara yüzü çok önemli olabilmektedir (Yun, Lee ve Kim, 2007). Kullanıcı uygulamayı öğrenmek değil kullanmak istemektedir. Bunların bilincinde olarak belli standartlardan çıkmadan kullanıcıya daha önce de uygulamamızı kullanmış hissi vermemiz gerekmektedir (Mondal, Pei, Dai, Kabir ve Sahoo, 2020). Bu platformda kullanıcı ara yüzünde, kullanışlılık, alışkanlık görülmektedir. Günümüzde bilgisayarlar kullanıcıların oyun oynamasına hitap ederken, televizyonlar oyun ekranı, sinema ve görsel sosyal medya kullanımında tercih edilirken mobil cihazlarımız alışveriş, yazılı sosyal medya platformları ve mobil oyunlar için daha çok kullanılmaktadır (Ma, Lu, Luo ve Liu, 2014). Tercih ettiğimiz kitlenin belirlenmesi ve onlara uygun ortamların tercih edilmesi mobil web uygulamaları için oldukça önemlidir.

Hibrit Uygulamalar, Mobil Web Uygulamalara benzerdirler. CSS, JavaScript ve HTLM kodları ile yazılan yazılımlardır. Web tabanlı uygulamalar olarak da isimlendirilebilirler (Robbins, 2012). Bunun sebeplerinden biri de tam anlamı ile kullanım için bir internet bağlantısına ihtiyaç duymaktadırlar (Verileri cihazımıza indirip çalışan varyantları da mevcuttur. Örneğin; Bazı oyunların belli bölümleri veya haritaların indirilmesi gibi düşünülebilinir).

2. KULLANILAN UYGULAMARARIN MİMARİ YAPILARI

Tüm yazılım platformları birbiriyle benzer görünmekle birlikte pek çok açıdan farklılıklar sergilemektedirler (Alseadoon, Ahmad, Alkhalil ve Sultan, 2021). Bu farklılıklar bazen kullanılan yazılım dillerinin kendine özgün mimari yapılarından oluşan avantaj ve dezavantajlar olmakla birlikte kullanılan platformların da bu işlemlerde rollerinin olduğu görülmektedir (Georgantas ve Issarny, 2006). Bu durumlar göz önüne alındığında çapraz yazılım platformların da birbirinden farklı özellikleri görülmektedir (Aldayel ve Alnafjan, 2017). Bu farklılıklar örneklem daraltılmıştır. Örneklemin daraltılmasının sebepleri arasında yazılım geliştirme kitleri üzerinde daha fazla test yapmaktır. Çapraz platform yazılımlar arasında Google ve Youtube üzerinde son 12 ayda Flutter, React Native, Cordova ve Xamarin SDK'larının arama sonuçları çıkarılmıştır. Bu sonuçlar Görsel 7'de görüldüğü gibi arama sonuçlarında en üst sırada bulunan SDK yazılımlarından ikisi Flutter ve React Native araştırmaya konu seçilmiştir.



Şekil 2.1: Son 12 ay içinde yapılan bazı Çapraz Platform arama Sonuçları.

Kaynak: Url-6, (2021) ve Url-6, (2021)

Şekil 2.1'de sırası ile Mavi renk - Flutter, Kırmızı renk - React Native, Sarı renk – Cordova, Yeşil renk - Xamarin ve son olarak Mor PWA (Progressive Web Apps - İleri Web Uygulaması), çapraz platformları olmak üzere son 12 ay içerisinde Google ve Youtube tarafından arama listesi ve dünya üzerinde arama yapılan ülkeleri göstermektedir. Flutter yakın birçok rakibine göre Dünya genelinde yapılan arama sonuçlarında öne çıktığı görülmektedir (Url-5, 2021 ve Url-6, 2021). Bu yazılımlar arasındaki farkların tarafsız bir şekilde tespiti için, incelenen SDK yazılımlarını, tercih etmek isteyen kişilerin daha sağlıklı sonuçlara ulaşması adına mobil (sanal ve gerçek) ve masaüstü bilgisayarlarda testler yapılmıştır. Bu testler programın çalışma esnasında yüklenme hızı, kullandığı RAM bellek miktarı, işlemci gücü, batarya kullanımı, FPS hızı ve hard disklerde kapladığı alanlar üzerinde yapılmıştır. Bu tür özellikler bazı masaüstü yazılımlarda göz ardı edilmekle birlikte mobil yazılım dünyasında oldukça önemli bir rol üstlenmektedir. Mobil uygulamalar kullanıcılar tarafından yazılım kolaylığı ve kullanışlılığı kadar donanımsal yeterlilik ile de tercih edilmektedirler (Nagappan ve Shihab, 2016). Kullanıcıların daha fazla tercih ettiği IOS ve Android işletim sistemleri üzerinde, testler yapılmıştır.

2.1. Flutter

Şekil 2.2: Flutter Logo

Yazılım sektörü ilerleyen her yıl kendini katlayarak büyümekte ve bu büyüme ile yeni iş kolları yaratmaktadır (Napoli, 2019). Yarattığı iş kolları farklı sektördeki ihtiyaçlar doğrultusunda mimarisini şekillendirmektedir. Kullanıcı ve geliştiriciler tasarlayabilecekleri daha basit yaklaşımlar ortaya çıkarmaya çalışmaktadırlar. Oluşan bu yaklaşımların ortaya çıkması için uygulamalar geliştirilmesinde kullanılan yeni ortamlar ve teknolojiler kendini yenileyerek, değiştirerek uyum sağlamışlardır (Jazayeri, 2007). Google tarafında geliştirilen ve Mobil Dünya Kongresi (MWC) sırasında duyurulan Flutter; Windows, Linux, Mac ve yine Google tarafından geliştirilmiş Google Fuchsia işletim sistemleri ile çalıştırılabilen açık kaynak kodlu ve ücretsiz bir mobil uygulama geliştirme kitidir (Wu, 2018). Flutter ile hazırlanan yazılımlar Çapraz Platform adında mimari ile oluşturulmaktadır. Bu mimari yapı

sayesinde yazılan yazılımlar tek bir platform için değil popüler olan pek çok platform için aynı anda yazılım yazma imkânı sağlamaktadırlar (Url-7, 2020). Başka bir ifade ile Flutter SDK'si ile Dart dili kullanılarak yazılan bir yazılım Android, IOS, Web ve Masaüstü platformlarda kullanılmaktadır. Bu çapraz platform sayesinde zamandan ve maliyetten tasarruf sağladığı için hem geliştiriciler hem de kullanıcılar tarafından tercih edilmektedir (Armagan, 2020). Çapraz Platform Yazılımların bir diğer özelliği ise bütün platformlarda birbirinin benzeri ekran görüntüsü alma imkânı sağlamasıdır (Jagiełło, 2019). Bu da kullanıcılar açısından uygulamaları farklı platformlarda da aynı kullanım alışkanlığını yaratmaktadır. Flutter, diğer SDK'ların aksine herhangi bir kurulum paketi indirmenize ihtiyaç duymamaktadır. Genellikle küçük eklenti paketleri ile kütüphanelerini kurmaktadır (Fayzullaev, 2018). Rakiplerine kıyasla daha kesin ve net çözümler ile olaylara yaklaşımı, kendi içindeki nesne kütüphanelerine hızlı erişmesi ve bu kütüphanelerde veya geliştiricinin yaptığı anlık değişimleri daha hızlı uyarlayarak çıktı göstermesi nedeni ile geliştiricilerin dikkatini çekmektedir (Dagne, 2019).

2.1.1 Widget

Programlamada yazılan kodlarda, oluşabilecek hataların azaltılması ve tekrar eden kod bloklarının oluşturacağı hatadan kaçınmanın yanı sıra kullanıcı ara yüzünüde basitleştirir. Widget kullanım kolaylığı sağlaması adına ekranda kullanılan AppBar, Images, Button, NavigationBar, Accessibility, Animation and Motion, Layout, Styling, Text gibi araçların ve hazır çekilen bazı API'ler kullanılarak ekranda yer alan yardımcı öğelerdir. Widgetlar, İngilizcede "WIndow gaDGET" cümlesindeki kelimelerin kısaltılmasından meydana gelmektedir. Yandaki şekilde (Bkz. Şekil 2.3) da görüldüğü gibi ekranda yer alan birçok öge widget yardımı ile oluşturulmaktadır. Kullanmakta olduğumuz Flutter SDK'sında her bir öğe birer widgetden oluşmaktadır. Flutter SDK'sında kolaylıkla widget içinde widget oluşturulması mümkündür (Faust,2020). Popüler kullanılan çoğu SDK uygulamasından widgetlar birer bileşen iken Flutter SDK'sında oluşturmak istediğinin bütün widget örnekleri ve açık kaynaklı kullanım

Şekil 2.3: Widgetlar ile oluşturulmuş bir uygulama örneği.

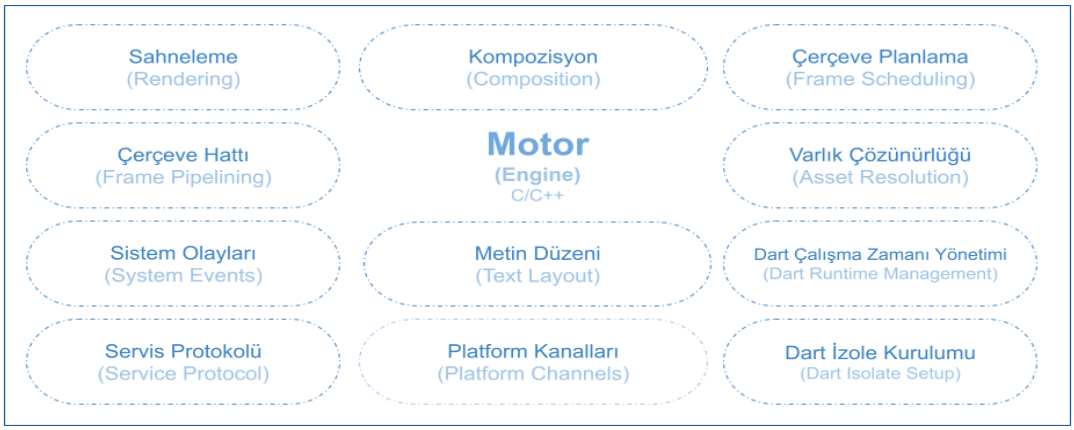
izninin açık olduğu widgetlar Flutter resmi sayfasından rahatlıkla ulaşılabilmektedir (Url-8, 2020).

2.1.2 Flutter çalışma mimarisi

Flutter, açık kaynak kodlu bir yazılım geliştirme kiti olarak Google tarafından piyasaya sürülmüştür. Genel yapısı itibari ile katmanlı bir yapıya sahip olarak tasarlanmış bir sistemdir. Tüm katmanlar temel katmana bağlı olarak birbirinden bağımsız çalışmasından dolayı bağlı olduğu kütüphanelerde birbirinden bağımsız çalışmaktadır (Mainkar ve Giordano, 2019). Katmanların birbirine üstünlüğü olmadığından birçok senaryoda birbirlerine ihtiyaç da duymamaktadırlar (Cairns, 2018). Katmanların birbirinden bağımsız ve istenildiğinde bütün bölümlerin kullanıcının isteğine bağlı olarak farklı amaçlar için kullanılabilmesinin yanı sıra kütüphanelerde isteğe bağlı değişiklikler de birbirini etkilemeyecek şekilde tasarlanmıştır (Url-3, 2021).

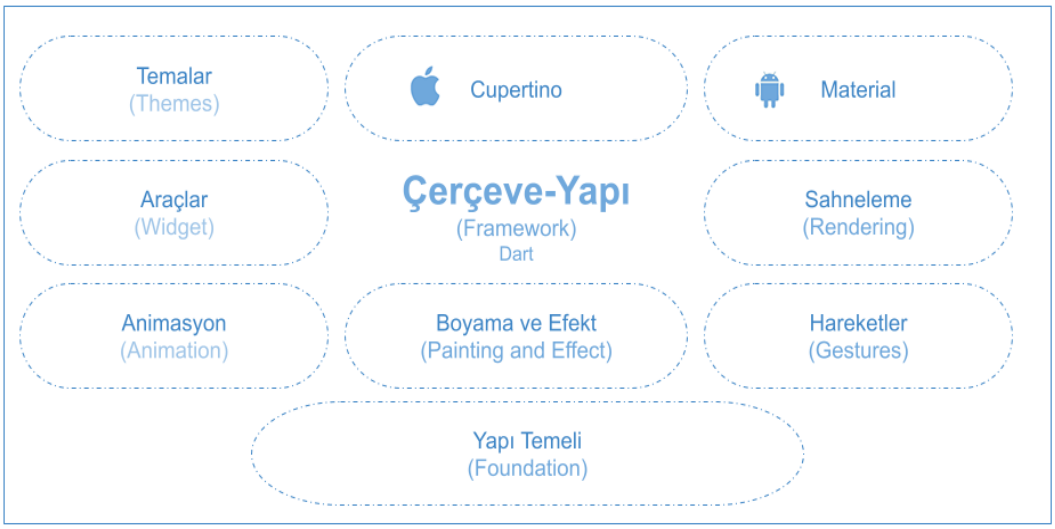
Şekil 2.4: Katıştırıcı (Embedder) çalışma dosyaları.

Flutter kendisi bir işlem oluşturmaz veya yönetmez. Bu durumu Embedder(katıştırıcı) üstlenmektedir. Flutter çerçevesi kullanılacak işletim sistemine bağlı olarak ihtiyaç duyulacak işletim sisteminin dosyalarını da Şekil 2.4'de olduğu gibi yazılan yazılımın içine tek bir dosya şeklinde paketlemektedir (Paul ve Nalwaya, 2019). Bu durum uygulamanın çalıştığı ortama uygun özellikleri barındırmasından dolayı yazılan yazılımın performansını arttırmakta ve her işletim sistemine özgü bir giriş noktası sağlamaktadır (Faust, 2020). Örneğin, kullanıcı ara yüzü, hızlı dosya erişimi, gelen girdi işlemlerinin daha hızlı sağlanması gibi temel işlemlerin, işletim sistemi ile uyumunu bu sayede sağlamaktadır. Gömülü olan kütüphaneler performans olarak diğer çapraz platform yazılımlarından daha seri çalışmasına yardımcı olmakla birlikte rakiplerine göre uygulama boyutlarının daha büyük olması gibi bir dezavantaj da doğurmaktadır (Martins, 2021).



Şekil 2.5: Flutter Çerçeve - Yapı (Framework) çalışma dosyaları.

Flutter, Linux ve Windows işletim sistemi için C++ yazılım dilini kullanırken, Android işletim sistemlerin de ise C++ ile birlikte Java dilini kullanmaktadır. MacOS işletim sistemleri için ise; Objective-C ve Objective-C++ yazılım dillerinin temel dosyaları kurulacak olan uygulamanın içinde gömülü olarak yer almaktadırlar (Martins, 2021). Flutter yazılım geliştirme kitinin büyük bir kısmı C++ yazılım dili ile tasarlanırmıştır. Flutter ile yazılmış uygulama standartlarını sağlaması için ise; içinde gerekli olan Flutter kütüphanesi/motoru bulunmaktadır. Bu kütüphaneler/motor genel olarak diğer parçalar arası işlemlerin gerçekleşmesinde rol oynamaktadırlar (Faust, 2020).



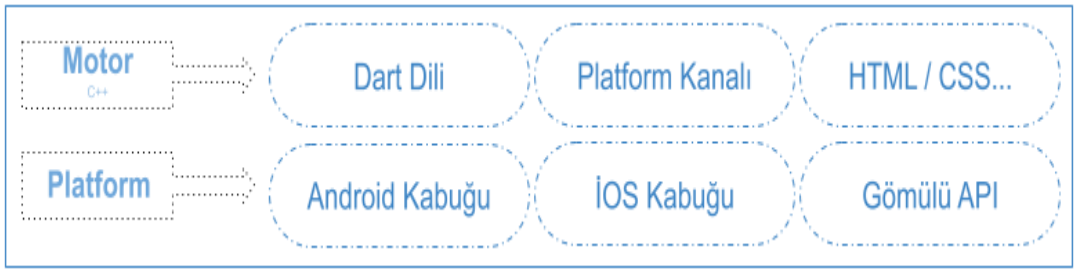
Şekil 2.6: Flutter Motoru (Engine) çalışma dosyaları.

Flutter uygulaması, Cupertino kütüphanesini IOS işletim sistemleri için kullanırken Material kütüphane dosyalarını Android cihazlar için sunmaktadır (Henshaw, 2019). Bu iki kütüphane içindeki kamanlar sayesinde Android ve ISO cihazlarda kullanılabilecek çerçeve ve temalar gibi dosya kütüphanelerini saklamaktadırlar

(Miola, 2020). Widget katmanı ise; tekrar kullanılabilecek sınıfları tanımlamamıza olanak tanıyan katmandır (bu widget(araçlar) kısmına daha detaylı olarak Şekil 2.3'den ulaşabilirsiniz). Rendering (Sahneleme) Katmanı oluşturduğu ağaç yapısı sayesinde mimari yapının düzeni sağlamakla görevlidir (Url-3, 2021). Ağaç yapısından oluşacak anlık değişikliklerden ve güncellemelerden de bu katman sorumludur. Bir alt katmanda ise uygulamanın hareketli resim ve şekilleri yanı sıra diğer temel sınıflar gibi hizmetler yer almaktadır (Chadha, Byalik, Tilevich, ve Rozovskaya, 2016). Foundation (Yapı Temeli) bu katmanda yer alan her nesnenin, widget katmanında bir karşılama sınıfı mevcuttur (Meiller, 2021).

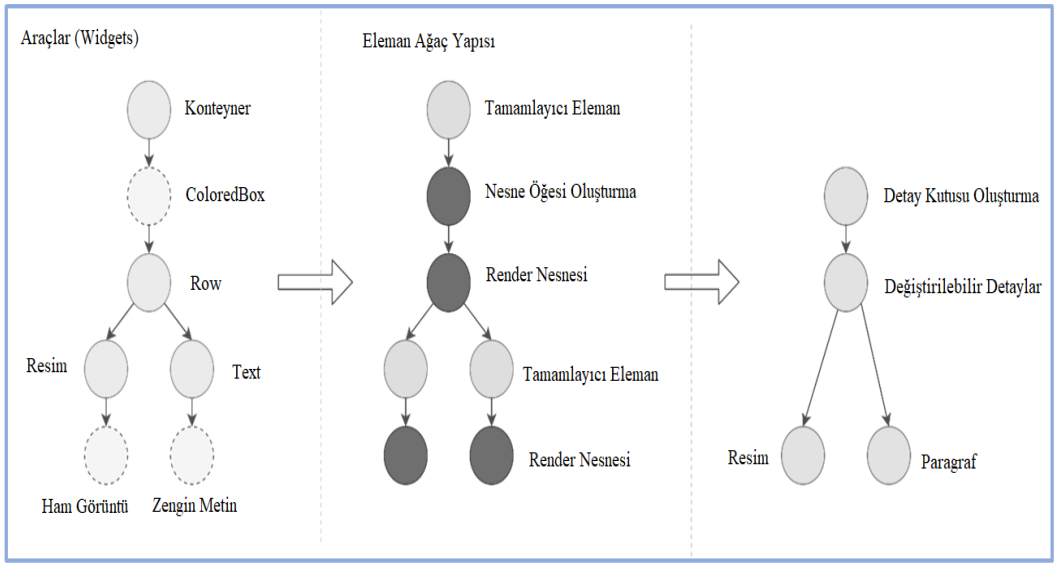
Şekil 2.7: Tarayıcı (Browser) çalışma dosyaları.

Flutter, kullandığı mimari yapısı sayesinde web tabanlı yazılımlar oluşturmak istendiğinde de aynı dosya havuzundan yararlanmaktır (Bracha, 2015). Flutter, çerçevesi ile bu işlemleri diğer dosyalarının üzerine inşa ederek gerçekleştirmektedir (Henshaw, 2019). Yapmakta olduğu işlemlerin gerçekleşmesi için mobil uygulamalarda Dart yazılım dili ile birlikte kullanılan ARM makine dili ile işlemlerini yapmamaktadır. Bu kullanılan makine dili yerine ise; web tabanlı yazılımlarda kullanılan API ve JavaScript standartlarını kullanarak işlemlerini gerçekleştirmektedir. Flutter, performans ve uygulamalarındaki kaliteyi arttırmak adına Şekil 2.7'de bulunan Tuval(Canvas), WebAssembly gibi işlemlerini sırasını gözetmeksizin ihtiyaç duyduğu dosyaya ve alt dosyalarına doğrudan ulaşarak gerçekleştirmektedir (Porto, 2020). Flutter, çekirdek katmanının sınırlarını Dart dili ile çizerek optimize etmekte ve JavaScript derleyicilerini kullanarak web sunucusuna dağıtılabilecek tek bir dosya haline getirebilmektedir (Url-3, 2021). Web desteğini birçok işletim sistemi ve cihazla çalıştırabilmek adına bağımsız, çevrimdışı Kullanıcı Deneyimi (User Experience) sunan ileri web uygulaması (Progressive Web Apps) gibi uygulamaları ile geleneksel web uygulamalarının, internet sayfalarının, mobil cihazlar ile birlikte hibrit bir şekilde kullanılması olanağını da tanımaktadır.



Şekil 2.8: Flutter Motoru ve Platform çalışma dosyaları

Flutter yazılım geliştirme kitinin bir başka özelliği ise 2D görüntü işleyebilen Firefox, Google Chrome, Andoid, Firefox OS, Chrome OS gibi donanım ve yazılımlarda açık kaynak kodlu API kütüphanesi sunan grafik işleme motoru olmasıdır. Bu motor içerisinde Dart diline özgü kütüphaneleri ile bir sanal makine barındırmaktadır (Upadrista, 2021). Bu API dosyalarından herhangi birine ihtiyaç duyulduğunda sistem tarafından uyararak sistemin ihtiyacı olan dosyayı temin etmektedir (Jagiello, 2019). Dart dili ise Flutter motorumuz ile oluşturulan işlemlerin bulunduğu işletim sisteminin yerel koduna derlenme işlemlerini gerçekleştirmektedir. Örneğin IOS sistemleri LLVM ile derlenirken, Android işletim sistemleri C ve C++ yazılım dilleri ile derlenmektedir (Faust, 2020). Bu derleme sonucu uygulamalarımızın çıktılarını tekrar kontrol ederek ISO platformlar için .IPA çıktısı olarak hazırlamaktadır. Android platformuna ise bu derleme işleminin sonucunda .APK olarak aynı anda kodlarını düzenleyerek çıkarmaktadır. Herhangi bir uygulama çalıştırıldığında uygulamanın yüklenmesinden itibaren kontrol ederek kullanıcı ara yüzüne sorunsuz aktarılması ve yüksek FPS işlemleri gibi bir çok işlemi Skia gerçekleşmektedir (Porto, 2020). Flutter genel olarak Widget'lardan oluşmaktadır. Birçok kaynakta "Flutter demek winget demektir. Widget demek Flutter demektir" şeklinde ifadeler yer almaktadır. Bu sebeple widgetlarında çalışma şeklini Flutter mimarisinde yer vermek daha doğru olacaktır. Widgetlar genel itibari ile kullanıcı ara yüzünde görmeye alışık olduğumuz görsellerden, textlerden ve diğer bütün aşamalardan oluşmaktadır (Upadrista, 2021)

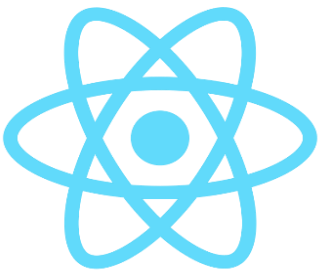


Şekil 2.9: Flutter, windget ve ağaç yapısı.

Widget, kullanıcıya sunulan görselleri oluşturan en önemli kısımda yer almaktadır. Bu sebeple kullanıcı ara yüzünün ana kilit taşı olarak değerlendirilebilir. Widgetlar anlaşmaya dayalı bir hiyerarşi oluştururlar (Swarna, Purnama ve Anthony, 2020). Şekil 2.9'da görüldüğü gibi her çember kendinden önceki çemberler ile iletişim kurabilmektedir. Görsel 15'deki örnekte görülen yapının kök bağlantıları Konteyner açılıp bu Konteynerlerin ekranı ikiye bölünerek bir tarafa resim diğer kısma ise görsel ile ilgili açıklama eklenecek Şekil 2.9'da tasarlandığı görülmektedir. Flutter çalışma hiyerarşisi, Ağaç Yapısındaki değişiklikler elementlerdeki değişiklik ile anlık olarak yer almaktadır. Bu değişikler bütün yapının silinip tekrar oluşmasından ziyade gerekli değişikliğin yapıldığı noktadaki değişkeni değiştirip yapıyı tekrar düzenlenmektedir (Biessek, 2019). Genel yapı kalıcılığı ve önbellekte durması performansı büyük ölçüde etkilemekte ve daha hızlı yüklenme sağlamaktadır (Wooding, 2019). Bahsedilen örnekte tek bir widget yapısı kullanılan basit bir sistemden bahsedilmektedir. Daha büyük projelerde bu durum daha önemli bir rol oynamaktadır. Oluşturulan yapı bir ağacın dallarına benzetilirse RenderObject (nesneyi oluştur) soyut olan objeler oluşturmak için kullanılır. RenderObject sistem ana hatları hakkında genel bilgiye sahip iken, alt katmanları hakkında çok fazla bilgiye sahip değildir. Sadece konum bilgileri içinde yer almaktadır (Miola, 2020). Derleme aşamasında alt katmanlardan gelen bilgiyi işler ve işlemi sonuçlandırmadan genel bir dönüş sağlamaktadır. Bu dönüşlerin ardından her nesne kendi maksimum ve minimum veri taşıma işlemini belirleyerek işlemlerini gerçekleştirmeye başlar bu işlemler sayesinde örneğin bir mobil cihazın ekran boyu ve eni hesaplanarak ekrana

çıkabilecek uygulama kullanıcı ara yüzünü hesaplayarak enini sınırlayıp uzunluk kısmında esneklikler sağlayabilmektedir (Biessek, 2019). Tüm sahne tamamlandıktan sonra işlemleri derleyerek işlem son bulur sonrasın da ise Dart:ui kütüphanesine sonucu aktarmaktadır. Bu örnek üzerinde de görüldüğü üzere Flutter motoru kendi widgetlarını işletim sistemi dosyalarına dönüştürmek yerine Flutter motoru ile oluşturduğu kullanıcı ara yüzü dosyalarını işletim sistemine uygun bir şekilde derleyerek düzenler ve hazırlar. ABI sayesinde platform bağımsız işlemlerini derlemektedir (Swarna, Purnama ve Anthony, 2020). Şekil 2.9'da yer alan adımlar, kurulumun yapıldığı platformda bulunan işletim sistemi ile Flutter uygulaması arasında tam uyum içinde çalışabilecek işlemler gerçekleştirilir ve uygulama çalıştırıldığında IOS ve macOS için UIViewController veya NSViewController elementlerini çağırır. Android işletim sistemlerinde Activity varsayılan olarak çağırır fakat işlemde görüntüde var ise FlutterView paketi içindeki dosyalara da ihtiyaç duymaktadır. Windows işletim sistemlerinde ise; Win32 dosyasını kullanır ve OpenGL API dosyalarından gelen isteklerini okuyup DirectX içinde yer alan DirectInput, DirectSound, DirectMusic, Direct3D, DirectPlay, DirectDraw, DirectShow gibi kitaplıkları dönüştürerek UWP uygulama modeli sunan ANGLE kütüphanesi altında toplamaktadır (Miola, 2020). Flutter motoru diğer diller ile (Kotlin, Swift, C) API üzerinden iletişim kurmak istediğinizde veya başka bir Flutter uygulaması ile haberleşmek isterseniz bu dillerin hepsi ile ortak haberleşme ve veri alış verişine de izin vermektedir (Johnson, 2021). Bu diller ile iletişimin dışında Firebase üzerinden işlemlerimizi, uygulamada yer almasını istediğiniz reklam kodlarını veya cihaz üzerinden istenilen donanım özelliklerini veya kullanımını (Kamera, Bluetooth, Ses giriş çıkışı, Yüz tanıma, parmak izi, …) sağlayacak tüm iletişimlerde eklentileri sayesinde mümkün kılmaktadır (Wooding, 2019).

2.2. React Native



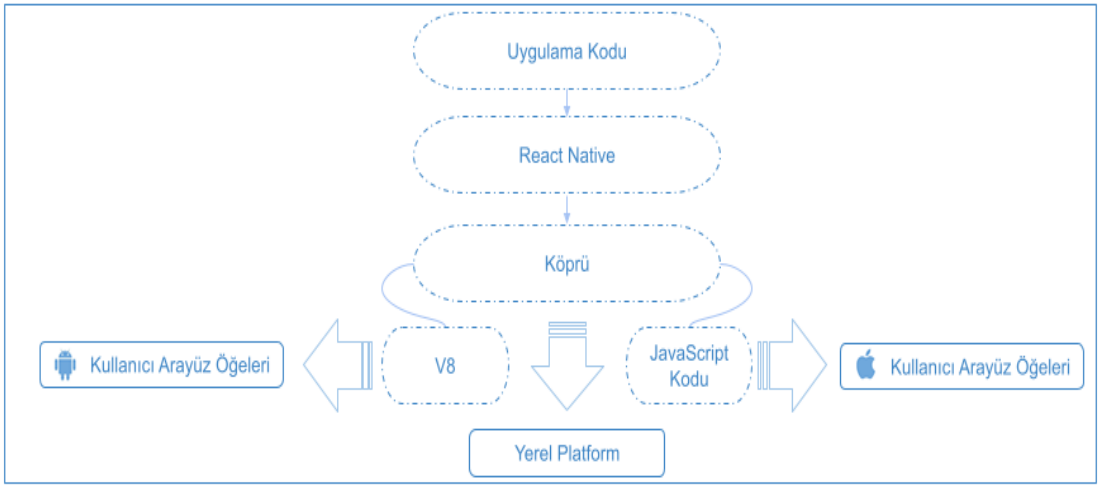
Şekil 2.10: React Native Logo

React Native, Facebook tarafında 2013 yılında beta olarak duyurulan ve 2015 yılında kullanıcıların hizmetine sunulmasıyla birlikte geliştirilme süreci devam etmekte olan açık kaynak kodlu, güncel bir SDK yazılımıdır (Yudin, 2020). React Native ile yazılım geliştirilebilmesi için JavaScript yazılımına

hâkim olmak gerekmektedir (Paul ve Nalwaya, 2019). React Native, Flutter yazılım geliştirme kitinden farklı olarak işlemlerini köprü kurarak gerçekleştirmektedir (Jagiełło, 2019). React Native ile geliştirilmiş yazılımlar bu köprüleme sayesinde Android, IOS, Web, Smart TV(Android, ios) ve MacOS gibi platformlara uygun olarak aynı anda proje geliştirmek mümkündür (Yan, Xiao, Hu, Peng ve Jiang, 2018). İlk piyasaya çıktıktan kısa bir süre sonra büyük şirketlerin dikkatini çekerek birçok uygulamanın mimarisini oluşturmayı başarmıştır (Bkz. React Native mimarisi ile geliştirilmiş yazılımlar).

2.2.1 React Native çalışma mimarisi

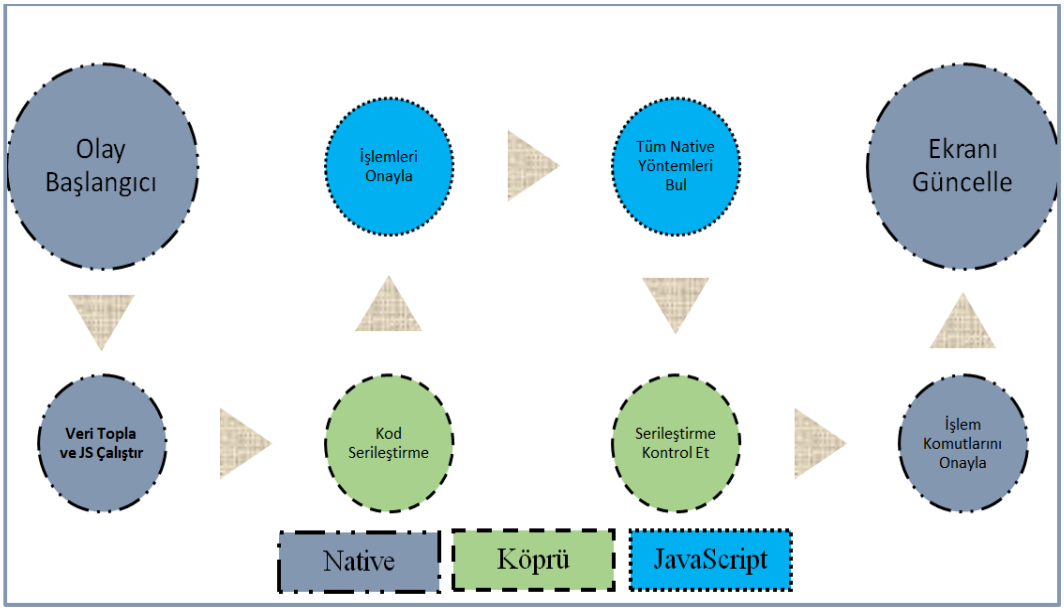
React Native, JavaScript ile hazırlanan kütüphane dosyaları ile oluşturulan ReactJavaScript kütüphanesini kullanarak işlemlerini gerçekleştirmektedir. ReacNative, dosyalarına erişmek için Doküman Nesne Modelini (Document Object Model) kullanmaktadır (Danielsson, 2016). React Native, Flutter SDK'sından farklı olarak işlemlerini köprü kurarak gerçekleştirmektedir. Kullanıcı ara yüzü ile React Native Kütüphanesi arasında JavaScript yazılım dili köprülüme işlemi gerçekleştirmektedir. Bu köpüleme sanal olarak gerçekleşmektedir ve bağlantı köprüsünü kullanan kütüphane dosyaları sayesinde Çapraz Platformların en belirgin özelliği olan farklı işletim sistemi ve cihazları destekleme işlemini gerçekleştirebilmektedir (Byalik, Chadha ve Tilevich, 2015).



Şekil 2.11: React Native, köprüleme işleminde arayüz oluşturma adımları

Köpürüleme işleminde yapılan değişlikler sayesinde platformlar arası yazılım geliştirmeyi mümkün kılan bu mimari yapı, hızlı değişen verilerde (Animasyon gibi) işlem yavaşlamalarına ve bu yavaşlamayı telafi etmek adına daha çok çalışmakta

olduğu cihazdan daha fazla güç talep etmektedir. Bu işlemler sırasında işlem gücü yetersiz cihazlarda bir işlem trafiği yaratmaktadır (Raj ve Tolety, 2012). React Native platformu farklı işletim sistemleri için kendine özgü kütüphanelerini kullanarak tek bir defa yazılan kodlarını farklı işletim sistemlerine dönüştürmektedir. Örneğin Android işletim sistemi için Şekil 2.11'de görüldüğü gibi V8 motorunu kullanmaktadır. V8 motoru Google tarafından C++ yazılım dili ile geliştirilmiş açık kaynak kodlu WebAssembly ve JavaScript kütüphaneleridir. V8 bağımsız çalışabilme özelliğinden dolayı oldukça kullanışlı bir yapıya sahiptir (Axelsson, ve Carlström, 2016). IOS işletim sistemi için ise, C, Swift ve Objective-C çerçevelerini kullanan Core.js kütüphanesi ile oluşturulmaktadır (Novick, 2017). Oluşturulan bu dosyalar köprü vasıtası ile dışarıya aktarılmaktadır. Bu köpürüleme işlemi üzerinden yazılı uygulama cihazlarımızda çalışmaya başlamaktadır. Daha önce de belirtildiği gibi genel itibari ile Yazılım Geliştirme Kitleri (SDK) kütüphane dosyalarımızın çevresini oluşturmakla görevli yazılımlardır. Buradaki kütüphane dosyalarımızı oluşturan JavaScript dosyalarıdır. React Native uygulamamız ise bir çerçeve olarak yapılan işlemleri derlemektedir (Hansson ve Vidhall, 2016). Şekil 2.12'de daha detaylı incelenecek olursa;



Şekil 2.12: React Native kod çalıştırma süreci.

JavaScript dosyasının Native üzerine kurulu bir çerçeveye yerleştirilerek React Native çerçevesinin oluştuğunu ve bu köpürüleme sayesinde birçok platformda çalışan yazılımlar geliştirilebildiğinden bahsetmiştik. Şekil 2.12'de görüldüğü üzere bu işlemleri sekiz adımda tamamlanabilmektedir. İlk olarak yazılım çalıştırılır ve

gelen dosyalar toplanıp JavaScript kütüphanesine bildirilir. JavaScript dosyaları daha önce bahsettiğimiz sanal köprüye aktarılmaktadır. VDOM işlemlerini React Native kütüphanesine aktarılır. React nesne modeline dönüştürülmesi ve anlaşmasını sağlayacak dosyalara çevirir. Bu işlemin yapılmasının nedeni ise kullanıcının beklediği arayüze gönderilecek öğelerin oluşturulması işlemidir. Bu işlemlerin ardından serileştirilmiş JavaScript dosyaları kontrol edilerek, hata ayıklamaları yapılmaktadır. Köprüleme işlemlerinde hatasız olarak çıkan kod emirleri, süzülerek ekrana aktarılmaktadır. Bu adımdan sonra kullanıcı gerçekleştirilen işlemleri cihaz arayüzünde görüntüleyebilmektedir.

2.3 Masaüstü Testleri

Masaüstü testleri için; *"IntelCore i7-10510U CPU @ 1.80GHz (8CPU24GB RAM) – NVIDIA GeForce MX250 2GB Ekran Kartı"* - Windows 10 işletim sistemi yüklü bilgisayar üzerinde sekiz farklı stres testinin önbelleğe alma ve işlemi gerçekleştirme sürelerinde, Dart (Flutter) dili ortalaması alındığında bir ile yirmi dört saniyelik aralıklarla yüklenmesini gerçekleştirmektedir. JavaScript (React Native) dili ortalaması alındığında iki ila kırk saniye aralığında işlemlerini gerçekleştirmeyi başarmıştır. Genel anlamda bazı testlerde JavaScript yüklenme hızının daha yüksek olmasına rağmen test edilen dillerin genel ortalamaları incelendiğinde, Dart dili daha performanslı bir şekilde işlemlerini gerçekleştirmiştir. Bu durumun sebepleri Flutter ve React Native çalışma mimarilerinin etkili olduğu yapılan testler sonucu elde edilmiştir.

2.4. Mobil Testler

Tablo 2.1: Test için kullanılan telefon marka ve model özellikleri.

Mobil / Donanım	**Galaxy S7 Edge**	**iPhone 7**
Pil Gücü	3600mAh	1960mAh
RAM Kapasitesi	4GB	2GB
İşlemci Kapasitesi	2,3 GHz Quad Core + 1,6 GHz Quad Core	2.34 GHz Quad Core

Çapraz Platform Yazılımlar, hibrit bir Görselde TV, WEB, Masaüstü, Telefon araçları gibi birçok alanda kullanılsa da çalışmanın konusu çerçevesinde bu uygulamaya başlayanların öncelikli olarak tercih etme sebepleri arasında mobil uygulamalar olduğundan testler daha çok mobil platformlar üzerinde yoğunlaştırılmış. Mobil cihazlar üzerinde ise üç farklı stres testi iki farklı mobil işletim sistemi üzerinde Samsung Galaxy S7 Edge ve Apple iPhone 7 cihazları kullanılmıştır. Tüm testler üç defa makine normal çalışma ve ısı düzeyine düştüğü aralıkta tekrar edilip ortalamaları alınarak hesaplanmıştır.

2.4.1 Test 1(Görsellerin listelenmesi ve 360 derecelik dönüşler)

İlk testimizde 3400 listeden oluşan arka planı farklı renklerde, ikili resimler kullanılmış bu görsellerden biri sabit tutularak ikinci görseldeki resimler 360 derecelik açılar ile 15000ms. hızla sonsuz dönüşler yaparak listelendirilmiştir. Amaç mobil cihazımıza ek iş gücü çıkararak mobil cihazın performans özelliklerini daha fazla ortaya çıkarmaktır. Android ve IOS cihazlarda React Native ile Flutter ile hazırlanmış yazılımlar farklı sonuçlar ortaya çıkardılar.

Tablo 2.2: Test1, donanımların performans sonuçlarının incelenmesi.

GalaxyS 7 Edge	React Native	Flutter		React Native	Flutter	iPhone 7
CPU	27.3	12.6		263.6	77.6	CPU
RAM Bellek	324	266		512	371	RAM Bellek
FPS	56	59		60	60	FPS
Bellek	31,17	38.4		31,17	38.4	Bellek
Batarya	164.0mAh	135.28mAh		72.8 mAh	23.9 mAh	Batarya

Android (S7 Edge) cihazımıza ve IOS (iphone7) cihazımıza aynı uygulamayı yükleyip performans karşılaştırmalarını hem Flutter hem de React Native ile hazırlanmış yazılımlar ile çıkan sonuçlarda Flutter gözle görülür farklılıklar ile daha avantajlı sonuçlar ortaya çıkarmıştır. FPS oranlarında büyük bir fark olduğu söylenememektedir. IOS işletim sistemi yüklü olan cihazımızda aynı sonuçlar görülmektedir. Cihazlarımızın RAM kullanımları incelendiğinde ise; Flutter ile de React Native arasında yaklaşık olarak %23'lük bir avantaj ile yine Flutter daha az RAM kullanarak daha önce bitirmiştir. Batarya konusunda IOS cihazımız ile Android cihazımın da büyük farklar ortaya çıkmış olsa da yine Flutter React Native kıyasla test sürecinde daha az güç tüketimi ile testi önde tamamlamıştır. Test1 yüklendiğinde tüm donanım kullanımlarında Flutter daha sağlıklı bir kullanım sunmaktadır. Bunların sonuçların ortaya çıkmasındaki en belirgin etken ise, Flutter çalışma mimarisi başlığı altında detaylı olarak paylaşılmıştır.

2.4.2 Test 2(Animasyon testi)

İkinci stres testsin de, seçilmiş olan hareketli harfler (gif) ile 20 kelime seçilerek (Flutter, React Native, Dart, JavaScript, ….) ekrana yazdırılmıştır. Bunlar yaklaşık olarak 0.5 ms. aralıklarla yinelenmekdir. Seçilen görseller birbirinden farklı animasyonlar yaratmaktadır. Bu da işlemci gücümü ve ram kullanımını oldukça arttırmaktadır. İkinci stres testinde ise, farklı sonuçlar ortaya çıkmış ve bunlar aşağıdaki tabloda listelenerek açıklanmıştır.

Tablo 2.3: Test2, donanımların performans sonuçlarının incelenmesi

GalaxyS7 Edge	React Native	Flutter		React Native	Flutter	iPhone 7
CPU	18.6	15.3		85.9	146.5	CPU
RAM Bellek	326.3	310		158	261.7	RAM Bellek
FPS	49	45		58	60	FPS
Bellek	30,36	37.4		30.36	37.4	Bellek
Batarya	21.6mAh	18.8mAh		21.5mAh	20.2mAh	Batarya

Android ve IOS cihazlarda Test2 yazılımı yüklenmiş ve Flutter ve React Native karşılaştırmaları yapılmıştır. Tabloda görüldüğü üzere Android cihazımızda işlemci kullanımı arasında yaklaşık olarak %18'lik bir fark ile Flutter daha az işlem gücü kullanmaktadır. IOS cihazımızda bu durum yaklaşık %42 gibi büyük bir fark ile React Native daha az işlem gücü kullanarak rakibine fark attığı görülmektedir. FPS kullanımında ise büyük farklar olmamakla birlikte Android cihazımızda React Native, IOS cihazımızda Flutter daha iyi sonuçlar çıkardıkları görülmektedir. Aynı görselde batarya kullanımında da hem cihazlar hem de SDK'larımızın birbirine çok yakın sonuçlar çıkardıkları görülmekle birlikte Flutter az bir farkla da olsa daha az pil enerjisi kullandığı görülmektedir.

2.4.3 Test3(Görsel ve animasyon testi)

Üçüncü test ise gerçekleşme olasılığı daha düşük fakat makine donanım tüketimi konusunda etkili bir test seçilmiştir. Telefon ekranının X,Y düzleminde 16X900 görsel ve animasyon kullanılarak yapılmıştır. Ekranda yer alan görseller, 360 derecelik dönüşler ile sonsuz bir döngü oluşturarak dönmektedir. Bu test mobil cihazlarda oldukça yüksek güç tüketimine neden olmuştur. Cihazların performanslarında oldukça yüksek sonuçlar gözlenmek ile birlikte Flutter SDK'sı yapılan bu test sonucunda React Native SDK daha performanslı olduğu tespit edilmiştir.

Tablo 2.4: Test3, donanımların performans sonuçlarının incelenmesi

GalaxyS7 Edge	React Native	Flutter		React Native	Flutter	iPhone 7
CPU	37.8	34.12		393.3	230.6	CPU
RAM Bellek	1284	492		666.2	559.3	RAM Bellek
FPS	36	42		60	60	FPS
Bellek	32.18	39.64		32.18	39.64	Bellek
Batarya	428.2mAh	406.5mAh		59.1mAh	34.7mAh	Batarya

Android ve IOS cihazlarında test3 yazılımı yüklenmiş ve Flutter ve React Native karşılaştırmaları yapılmıştır. Çıkan sonuçlar tablo üzerinden incelendiğinde işlemci kullanımı Android cihazlarında Flutter ve React Native aralarında yaklaşık olarak %6 ile %9 küçük farklar yaratmış olmak ile birlikte IOS işletim sistemi yüklü cihazlarda bu durum daha farklı bir sonuç ortaya koyarak yaklaşık olarak %40 gibi büyük bir fark tespit edilmiştir. Flutter SDK'sı daha az işlem gücü ile öne çıktığı görülmektedir. RAM bellek kullanımında Android işletim sistemi yüklü cihazlarda Flutter, React Native göre yaklaşık olarak %60'lık bir farkla daha az ram kullanımı sağlayarak şaşırtıcı bir sonuç doğurmuştur. IOS işletim sistemi yüklü cihazlarda ise bu fark yaklaşık olarak %15 gibi daha yakın sonuçlar ortaya koyarak Flutter SDK'sı daha az ram gücü ile işlemi gerçekleştirmiştir. Batarya (pil) kullanımında Android işletim sistemi yüklü cihazlarda SDK'larımız arasında pek fark olmasa da IOS işletim sistemi yüklü cihazlarda yaklaşık olarak %40'a yakın bir fark ile Flutter React Native göre daha az pil gücü kullanarak işlemlerini tamamlamıştır.

2.4.4 Mobil test sonuçlarının değerlendirilmesi

Yapılan testler genel olarak değerlendirildiğinde kullanılan yazılım geliştirme kitlerimizin farklı sonuçlar ortaya çıkardığı görülmektedir. Kabul edilmelidir ki bu yapılan senaryolar biraz olağanüstü durumları ortaya çıkarmak adına yapılmıştır. Fakat bu tür senaryoların bazı uygulamalarda pekâlâ mümkün olduğu da kaçınılmaz bir gerçektir. Yapılan testlerde bu durumların gerçekleşme olasılığının olduğu durumlarda Flutter, işlemci kullanım açısından Android işletim sistemi yüklü cihazlarda daha kararlı performanslar çıkarmış olsa da IOS işletim sistemi yüklü cihazlarda Animasyon testlerinde React Native gerisinde kalarak kararlı bir performans ortaya koyamamıştır. Ram kullanım miktarları göz önüne alındığında

çok büyük farklar olmamakla birlikte Flutter, React Native göre yine Android işletim sistemi yüklü cihazlarda daha düşük kullanım sağlayarak performans bakımından kararlı sonuçlar çıkarmıştır. ISO işletim sistemi yüklü cihazlarda React Native Animasyon yükleme ve çalıştırma konusunda Flutter kitinden daha az Ram kullanımı sağlayarak rakibini geçmiş olmasına rağmen diğer iki testimizde maalesef büyük farklar olmamasına rağmen daha fazla kaynak kullanmıştır. FPS kullanımı yapılan testlerde çok fazla farklar oluşmadığından hem Android hem de ISO işletim sistemi yüklü cihazlarda her iki yazılım kiti benzer sonuçlar ile başarılı bir şekilde testlerimizi tamamlamayı başarmış oldukları görülmektedir. Son olarak Batarya (pil) kulanım durumları incelendiğinde Flutter yazılım kiti hem Android hem de IOS işletim sistemi yüklü cihazlarda daha az güç tüketimi ile daha başarılı olduğu görülmektedir. Mevcut durumlar göz önüne alındığında IOS cihazlar için Animasyon ağırlıklı yazılımlarda React Native kullanımı daha sağlıklı ve performanslı olacakken diğer senaryolar için Flutterın tercih edilmesi mevcut sürümler ile daha sağlıklı olacaktır.

3. KULLANICI DENEYİMİ

Tez kapsamında üniversitelerin Fen Bilimleri Enstitüsü Yüksek Lisans programlarından Yapay Zekâ Mühendisliği, Mühendislik Fakültesinden Yapay Zekâ Mühendisliği ve Bilgisayar Mühendisliği, Meslek Yüksek Okuluna bağlı Teknik Programlarından, Bilişim Güvenliği Teknolojisi ve Bilgisayar Programcılığı programında eğitim gören öğrencilerin yanı sıra sahada yazılım alanında meslek sahibi veya yazılıma ilgi duyan yaklaşık elli kişiye Çapraz Platform yazılımlardan, Flutter ve React Native Performans karşılaştırılma yapılması ve deneyimlerini paylaşmaları adına Ek-B'deki otuz sorudan oluşan anket iletilmiştir (Url-9, 2021).
Ankete katılan kişilere sorduğumuz sorularda hâkim olduklarını düşündükleri yazılım dillerini belirtmelerini istediğimizde ilk sırayı C dil ailesi (%72) alırken, çalışma kapsamında değindiğimiz Dart (%2,3) ve JavaScript (%9,1) dillerine hâkim olan kişi sayısı oldukça düşük görülmekteydi." Bu çalışmadan önce, Cross - Platform Software, Platform Bağımsız Yazılım veya Çapraz Platform Yazımlar gibi isimlerini daha önce duydunuz mu?" Sorusuna yanıt olarak Hayır (%66) diyenlerin oranı Evet (%34) diyenlerden fazla olduğu gözlemlenmiştir. Aynı zamanda Ankete katılan kişilere Flutter (%23), React Native (%16) ve diğer SDK birini duyup duymadıkları sorulduğunda "Hiçbirini Duymadım (%68)" diyenlerin oranı oldukça yüksek çıkmasına rağmen Flutter duyanların sayısının React Native göre daha yüksek olduğu görülmekteydi. React Native, Flutter SDK'sından daha eski olmasına rağmen Flutterın daha fazla kişiye ulaşmayı başardığı görülmekteydi. Anketin içinde dört uygulama yapılması istenmiştir. Bunlar iki tane Flutter SDK ile Dart dili kullanılarak (Hello Flutter & Flutter Counter) yazılacak, diğer iki uygulamamız için ise React Native SDK ile JavaScript dili (Hello React – Native & React – Native Counter) kullanılarak oluşturulması istenmiştir. Oluşturulma sürecindeki deneyimlerinin de paylaşılması istenmiştir. "Hello Flutter" ve "Hello React Native" uygulamaları karşılaştırıldığında Hello React Native (%50) uygulamasını oluştururlarken Hello Flutter (%28) uygulamasına oran ile daha fazla zorlandıklarını ifade etmişlerdir. Bu uygulamanın ardından diğer uygulamayı geliştirmeleri istenmiş ve orada da aynı

oran da olmasa da React – Native Counter (%55) oluşturma sürecinde Flutter Counter (%23) uygulamasını yazmaya göre daha fazla zorlandıklarını belirlenmiştir. Kullanışlılık bakımından hangi Arayüz ve dili beğendiklerini sorduğumuzda katılımcılar, Flutter (%62) ve React Native (%28) göre daha çok kullanışlı bulduklarını belirtmişlerdir.

Katılımcılarımızdan uygulamaları hazırladıkları sırada bilgisayarlarındaki performans durumlarını not etmelerini ve iki Yazılım Geliştirme Kitini (SDK) İşlemci Gücünü (GHz), Ekran Kartı, Ram Bellek kullanım senaryolarında hangi SDK'nın daha fazla güç kullandığını iletmeleri konusunda talepte bulunuldu.

Tablo 3.1: Katılımcı, bilgisayar donanımlarının yük verilerinin incelenmesi

***Düşük olmaları daha olumlu bir durumdur.**	React Native SDK	Flutter SDK
İşlemci Gücü(GHz)	%63,60	%36,40 ☑
Ekran Kartı	%59,10	%40,90 ☑
Ram Bellek	%50,00 ☑	%50,00 ☑

Sonuçlara bakıldığında RAM Bellek kullanım miktarında bir ayırt edicilik görülememektedir. İşlemci Gücü (GHz) ve Ekran Kartı kullanımında ise React Native SDK'nın, Flutter SDK göre daha fazla güç kullandığını belirtmişlerdir.

Ankete katılan kişilerin SDK kurulumunu tamamladıktan sonra istenen uygulamaları yazıp mobil cihazlarda FPS (Frame Per Second/Saniyelik Görüntü Sayısı), İşlemci Gücü, Ram Bellek, Batarya kullanımı ve Uygulamanın boyutu ile ilgili sonuçlarda daha başarılı bulduğu özellikleri girmelerini beklediğimizde gelen sonuçlar aşağıdaki gibidir.

Tablo 3.2: Katılımcıların, 1.uygulama deneyim sonuçlarının incelenmesi

Hello Flutter - Hello React Native	React Native SDK	Flutter SDK
FPS	%38,63	%61,36 ☑
İşlemci Gücü(GHz)	%45,45	%54,54 ☑
Ram Bellek	%40,90	%59,09 ☑
Batarya	%36,36	%63,63 ☑
Uygulama Boyutu	%31,81	%68,18 ☑

Ankete katılan kişiler yüzdelik olarak farklılık gösterse de tüm donanım kullanımlarında Flutter SDK'ı daha başarılı buldukları belirlenmiştir. Bir diğer uygulama üzerinde de aynı testleri yapıp test sonucunu göstermeleri istenmiştir. Oradan gelen veriler ise aşağıda belirtildiği gibidir.

Tablo 3.3: Katılımcıların, 2.uygulama deneyim sonuçlarının incelenmesi

Flutter Counter - React Native Counter	**React Native SDK**	**Flutter SDK**
FPS	%56,81	%43,18 ☑
İşlemci Gücü(GHz)	%50,00 ☑	%50,00 ☑
Ram Bellek	%41,86	%58,13 ☑
Batarya	%39,53	%60,46 ☑
Uygulama Boyutu	%39,53	%60,46 ☑

Bu senaryoda da ankete katılan kişiler işlemci tüketimi hariç tüm senaryolarda Flutter SDK ile geliştirdikleri uygulamamızın daha başarılı sonuçlar verdiği belirlenmiştir. Sonuca ulaşırken işlemci kullanımında sayı eşit çıkmıştır ve değerlendirme sonuçlandırılmıştır.

4. SONUÇ VE ÖNERİLER

Genel olarak çalışma kapsamında, Çapraz Platformlar hakkında bilgi toplamak ve bu bağlamda performans, ara yüz ve Çapraz Platformları öğrenmek isteyen kişilerin yaşadığı problemlerin ana hatlarını belirlemek olmuştur. Örneklemin daraltılması adına iki çapraz platform seçilerek bunların karşılaştırılması sonucunda ortaya çıkan veriler değerlendirilmiştir. Değerlendirmelerin objektifliğinin artırılması adına anket hazırlanarak bu ankete katılan kişilerden de bu alanda yaşadıkları tecrübeleri paylaşmaları istendi.

Öncelikli olarak Flutter ve React Native SDK'larımızın çalışma mimarisi incelenmiştir. Aralarında benzerlikler ve farklılıklar gözden geçirilmiştir. Yapılan işlemler sonucunda Flutter ile React Native köklerinde C dilinden yararlanarak kendine has mimariler oluşturdukları görülmüştür. Oluşturulan mimaride en büyük fark ise köprüleme sistemi olduğu görülmüştür.

Flutter yapmak istediği işlemleri kendi içindeki kütüphanelerde derleyerek doğrudan çalıştırdığı için dosya boyutlarının daha büyük olmasına neden olurken, React Native içindeki kütüphaneleri sanal köprüleme kullanarak derlemek istediği işletim sistemi için özel dosya üretmektedir. Bu işlem sanal ortamda gerçekleştiği için Flutterda olduğu gibi uygulamalarımızın dosya boyutunu yükseltmemektedir. Bazı durumlar avantaj sağlarken başka işlemler için dezavantaja neden olabilmektedir. Bu durum yapılan testlerde çıkan sonuçlara etki ettiği için bilinmesinde fazda yarar olacaktır. Bölüm II içinde yapılan masaüstü bilgisayar testlerinde yazılan uygulamanın çalıştırılması sırasında tepki süreleri ölçülmüştür. Genel olarak Yazılım Geliştirme Kitlerinden tepki alım süreci ölçümü yapılırken, yazılan yazılımların tekrar çalıştırma (run) esnasında yükleme hızı ölçülmüştür.

Ölçümler sırasında Flutter 1-24 sn. aralığında ön yüklemeleri gerçekleştirmiş, React Native 1-42 aralığında tepki süreleri ile uygulamaların testlerini tamamlamışlardır. Daha sonra yapılan gerçek cihaz testlerinde Android ve ISO platformları farklı sonuçlar meydana çıkarmıştır. Genel olarak Android işletim sistemi yüklü cihazımızda Flutter daha başarılı olmasına rağmen "Test2" de yapılan animasyon

testinde ISO işletim sistemi yüklü cihazlarda React Native daha başarılı sonuçlar çıkarmıştır. Bulunan sonuçlar Ankete katılan kişilerin verdikleri sonuçlar ile karşılaştırıldığında birbirine benzer sonuçlar ortaya çıkmıştır. (Ankete katılanlar ile çalışma kapsamında yapılan testlerde kullanılan örnekler faklıdır). Kaynak konusunda Flutter – Dart ve React Native – JavaScript ikilileri konusunda React Native – JavaScript ikilileri hakkında daha fazla örnek kod blokları bulunmaktaydı. Bu işlem yeni başlayanlar için bir avantaj olarak değerlendirilebilir. Ankete katılan kişilerinde ise Flutter ile ilgili daha çok veri elde ettiklerini açıklamışlardır (Ek- C ve Url-10, 2021).

Genel anlamda değerlendirilirse proje geliştirilmek istenildiğinde iki SDK ile de güncel örnekler elde etmek mümkün olduğu söylenebilir. Yine Ankete katılan kişiler Dart dilinin öğrenimini JavaScript'e göre daha kolay olduğu konusunda çoğunluğa ulaşmış olsalar da JavaScript hakkında olumlu görüşlerin oranı oldukça yüksekti (Ek-C ve Url-10, 2021). Çıkan görüşler test edilen uygulamalar özelinde olduğunda ve hala iki kitin de güncellendiği ve açıklarını kapatmaya ve daha iyiyi hedefledikleri düşünüldüğünde zaman içinde sonuçların değişiklik göstermesi ve yapılmak istenen işe uygun kitin o gününün şartları göz önünde bulundurularak tekrar değerlendirilmesi ve seçilmesi oldukça sağlıklı olacaktır.

Yaptığımız tüm testlerde Flutter daha iyi sonuçlar vermesine rağmen React Native ile arasında çok büyük farklar olmadığı gerçeğini göz önünde bulundurmamız gerekmektedir. Çalışmanın son değerlendirmesi yapıldığında ve nihai bir sonuç üzerinde karar alınması gerekirse her iki SDK da birçok alanda kendilerini ispatlamış arkasında büyük yazılım şirketleri olmasının yanı sıra üzerinde geliştirilen uygulamalar dünyada en çok kullanılan uygulamalar arasında bulunmaktadır. Bunları göz ardı etmeden her iki SDK üzerinde de geliştirecekleri uygulamaların başarılı olma olasılığının, yazılımı geliştiren kişinin yeteneğine ve fikrin özgün olması ile paralel olduğunu kabul edebiliriz.

KAYNAKÇA

Adinugroho, T. Y., & Gautama, J. B. (2015). Review of multi-platform mobile application development using WebView: Learning management system on mobile platform. *Procedia Computer Science*, *59*, 291-297.

Ahmad, R. (2020, May). Design and Development of Automotive Workshop Application Based on Android and IOS Using Dart Programming Language. In Journal of Physics: Conference Series (Vol. 1539, No. 1, p. 012016). IOP Publishing.

ldayel, A., & Alnafjan, K. (2017, May). Challenges and Best Practices for Mobile Application Development. In *Proceedings of the International Conference on Compute and Data Analysis* (pp. 41-48).

Allen, S., Graupera, V., & Lundrigan, L. (2010). *Pro smartphone cross-platform development: iPhone, blackberry, windows mobile and android development and* distribution. Apress publishing, 1st edition.

Alseadoon, I., Ahmad, A., Alkhalil, A., & Sultan, K. (2021). Migration of existing software systems to mobile computing platforms: a systematic mapping study. *Frontiers of Computer Science*, *15*(2), 1-25.

Armagan, F. (2020). Vergleich von nativer App-und Cross-Platform-Entwicklung (Facebook React Native und Google Flutter).Bachelor-Arbeit im Studiengang " Mediendesigninformatik",Hochschule Hannover University Of Applied Sciences And Arts.

Axelsson, O., & Carlström, F. (2016). Evaluation targeting react native in comparison to native mobile development. H2 - Master's Degree, Lund University.

Biessek, A. (2019). *Flutter for Beginners: An Introductory Guide to Building Cross-platform* Mobile *Applications with Flutter and Dart 2*. Packt publishing, 1st edition.

Biørn-Hansen, A., Rieger, C., Grønli, T. M., Majchrzak, T. A., & Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, *25*, 2997-3040.

Blanco, J. Z., & Lucrédio, D. (2021). A holistic approach for cross-platform software development. *Journal of Systems and Software*, *179*, 110985.

Bracha, G. (2015). *The Dart programming language*. Addison-wesley professional publishing, 1st edition.

Byalik, A., Chadha, S., & Tilevich, E. (2015). Native-2-Native: Automated cross-platform code synthesis from web-based programming resources. *ACM SIGPLAN Notices*, *51*(3), 99-108.

Cairns, B. (2018). Flutter : A Beginners Course, Packt Publishing, Limited.

Chadha, S., Byalik, A., Tilevich, E., Rozovskaya, A. (2016). Facilitating The Development Of Cross-Platform Software Via Automated Code Synthesis From Web-Based Programming Resources. Computer Languages, Systems & Structures, 48, 3–19.

Dagne, L. (2019). Flutter for cross-platform App and SDK development. Bachelor's Thesis, Metropolia University of Applied Sciences.

Danielsson, W. (2016). React Native application development: A comparison between native Android and React Native. Master Thesis, Linköping University

Dehlinger, J., & Dixon, J. (2011, October). Mobile application software engineering: Challenges and research directions. In *Workshop on mobile software engineering* (Vol. 2, pp. 29-32).

Dhillon, S., & Mahmoud, Q. H. (2015). An evaluation framework for cross-platform mobile application development tools. *Software: Practice and Experience*, *45*(10), 1331-1357.

Dos Santos, D. S., Nunes, H. D., Macedo, H. T., & Neto, A. C. (2019, May). Recommendation System for Cross-Platform Mobile Development Framework. In *Proceedings of the XV Brazilian Symposium on Information Systems* (pp. 1-8).

Eskola, R. (2018). React Native Performance Evaluation, Master Thesis Work, Aalto University.

Faust, S. (2020). Using Google´ s Flutter Framework for the Development of a Large-Scale Reference Application. Bachelor Thesis, Technical University of Cologne.

Fayzullaev, J. (2018). Native-like Cross-Platform Mobile Development: Multi-OS Engine & *Kotlin* Native vs Flutter. Master Thesis, Southeast Finland University of Applied Sciences.

Georgantas, N., Issarny, P. I. (2006). Software Platforms. In: Aarts E., Encarnação J. (Eds) *True* Visions., Springer, Berlin, Heidelberg.

Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R. O., & Winckler, M. (Eds.). (2009). *Human-Computer Interaction-INTERACT 2009: 12th IFIP TC 13 International Conference, Uppsala, Sweden, August 24-28, 2009, Proceedigns* (Vol. 1). Springer Science & Business Media.

Hansson, N., & Vidhall, T. (2016). Effects on performance and usability for cross-platform *application* development using React Native. Final thesis, Linköping University.

Hartmann, G., Stead, G., & DeGani, A. (2011). Cross-platform mobile development. *Mobile Learning Environment, Cambridge*, *16*(9), 158-171.

Hassan, A. M. (2020). JAVA and DART programming languages: Conceptual *comparison*. *Indonesian Journal of Electrical Engineering and Computer Science*, *17*(2), 845-849.

Henshaw, N. (2019) Real-world projects with Flutter, Packt Publishing, 1st Edition.

Jagiello, J. (2019). Performance comparison between React Native and Flutter.

Jazayeri, M. (2007, May). Some trends in web application development. In *Future of Software Engineering (FOSE'07)* (pp. 199-213). IEEE.

Johnson, A. E. (2021, March). To GitHub or Not to GitHub?. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 1331-1331).

Koranne, S. (2009). An Overview of the SDK. *Practical Computing on the Cell Broadband Engine*, 133-143.

Ma, Y., Lu, X., Luo, Y., & Liu, X. (2014, June). A Graph-Based Approach to Assisting *Creation* of Mobile Web Applications. In *2014 IEEE International Conference on Web Services* (pp. 728-729). IEEE.

Mainkar, P., & Giordano, S. (2019). *Google Flutter Mobile Development Quick Start Guide: Get Up and Running with IOS and Android Mobile App Development*. Packt Publishing Ltd.

Martins, F. (2021). Flutter And Dart The Complete Guide: Create Cross-Platform Mobile Apps with Google's Latest Open-Source SDK Through Flutter And Dart, Independently published (March 14, 2021)

McCutchan, J., Feng, H., Matsakis, N., Anderson, Z., & Jensen, P. (2014, February). A SIMD programming model for Dart, JavaScript, and other dynamically typed scripting languages. In *Proceedings of the 2014 Workshop on Programming models for SIMD/Vector processing* (pp. 71-78).

Meiller, D. (2021). Modern App Development with Dart and Flutter 2. In *Modern App* Development *with Dart and Flutter 2*. De Gruyter Oldenbourg.

Miola, A. (2020). Flutter Complete Reference: Create Beautiful, Fast And Native Apps For Any Device, Kindle Edition.

Mohanty, S., & Dey, S. R. (2014) .Dart evolved for web-a comparative study with javascript. *International Journal of Computer Applications*, *975*, 8887.

Mondal, S. K., Pei, Y., Dai, H. N., Kabir, H. D., & Sahoo, J. P. (2020, December). Boosting UI Rendering in Android Applications. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (pp. 285-286). IEEE.

Nagappan, M., & Shihab, E. (2016, March). Future trends in software engineering research for mobile apps. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)* (Vol. 5, pp. 21-32). IEEE.

Napoli, M. L. (2019). Introducing Flutter and Getting Started. *Beginning Flutter: A Hands-on Guide to App Development; Wrox: Indianapolis, IN, USA*, 3-25.

Nawrocki, P., Wrona, K., Marczak, M., & Sniezynski, B. (2021). A Comparison of Native and Cross-Platform Frameworks for Mobile Applications. *Computer*, *54*(3), 18-27.

Nedyak, A., Rudzeyt, O., Zainetdinov, A., & Ragulin, P. (2020). Mobile cross-platform app development tools. *Russian Journal of Resources, Conservation and Recycling*, *7*(4).

Novick, V. (2017). *React Native-Building Mobile Apps with JavaScript*. Packt Publishing Ltd.

Oulasvirta, A., Wahlström, M., & Ericsson, K. A. (2011). What does it mean to be good at using a mobile device? An investigation of three levels of experience and skill. *International journal of human-computer studies*, *69*(3), 155-169.

Paul, A., & Nalwaya, A. (2019). React Native for Mobile Development. *React Native for Mobile Development. California: Apress, Berkeley, CA. https://doi. org/10.1007/978-1-4842-4454-8*.

Perchat, J., Desertot, M., & Lecomte, S. (2013). Component based framework to create mobile cross-platform applications. *Procedia Computer Science*, *19*, 1004-1011.

Porto, J. (2020) Flutter: A practical approach, Independently Published,1st Edition.

Raj, C. R., & Tolety, S. B. (2012, December). A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In *2012 Annual IEEE India Conference (INDICON)* (pp. 625-629). IEEE.

Robbins, J. N. (2012). *Learning web design: A beginner's guide to HTML, CSS, JavaScript, and web graphics*. " O'Reilly Media, Inc.", 4 Edition.

Scott, A. D. (2020). *JavaScript everywhere: building cross-platform applications with GraphQL, React, React Native, and Electron*. O'Reilly Media, first st. edition.

Shen, J., & Abraham, J. A. (1998, October). Native mode functional test generation for processors with applications to self test and design validation. In *Proceedings International Test Conference 1998 (IEEE Cat. No. 98CH36270)* (pp. 990-999). IEEE.

Swarna, I., Purnama, J., & Anthony, R. (2020). Cross-Platform Analysis and Development of Online Catering Platform (Kunyahku). *Journal of Applied Information, Communication and Technology*, *7*(2), 79-89.

Trisnadoli, A., & Lestari, I. (2017, November). Software requirement analysis of "family tracking mobile application" on cross platform with hybrid approach. In *2017 International Conference on Data and Software Engineering (ICoDSE)* (pp. 1-6). IEEE.

Upadrista, V. (2021). *Formula 4.0 for Digital Transformation: A Framework Using Digital Enablers from Industry 4.0*. Productivity Press, 1st edition.

Ünaldı A. (2019). İnternetin Kısa Tarihi, ISBN:9781492786412, 2. Baskı

Wang, H., Liu, Z., Guo, Y., Chen, X., Zhang, M., Xu, G., & Hong, J. (2017, April). An explorative study of the mobile app ecosystem from app developers' perspective. In *Proceedings of the 26th International Conference on World Wide Web* (pp. 163-172).

Wooding, V. (2019). Flutter Tips, Tricks, and Techniques, Packt Publishing, Limited.

Xanthopoulos, S., & Xinogalos, S. (2013, September). A comparative analysis of cross-platform development approaches for mobile applications. In *Proceedings of the 6th Balkan Conference in Informatics* (pp. 213-220).

Yun, M., Lee, J., Kim, W., & Kim, S. (2007, February). UI Player Framework for Mobile Devices. In *The 9th International Conference on Advanced Communication Technology* (Vol. 1, pp. 151-154). IEEE.

Zammetti, F. (2013). *Learn Corona SDK Game Development*. Apress, 1st edition.

İnternet Adresleri

Url-1 "dart2js: Dart-to-JavaScript compiler", dart.dev, (03.02.2021) <https://dart.dev/tools/dart2js>

Url-2 “Hanehalkı Bilişim Teknolojileri (BT) Kullanım Araştırması, 2020”, data.tuik.gov.tr, (12.10.2020)
<https://data.tuik.gov.tr/Bulten/Index?p=Hanehalki-Bilisim-Teknolojileri-(BT)-Kullanim-Arastirmasi-2020-33679>
Url-3 “Introduction to widgets”,flutter.dev,(05.02.2021)
<https://flutter.dev/docs/development/ui/widgets-intro>
Url-4 “Desktop vs Mobile vs Tablet vs Console Market Share Worldwide”, gs.statcounter.com,(13.10.2020)
<https://gs.statcounter.com/platform-market-share#monthly-201301-202011>
Url-5 “Google trendler, zamana ve bölgelere göre karşılaştırmalı döküm”, trends.google.com, (18.02.2021)
<https://trends.google.com/trends/explore?q=flutter,react%20native,cordova,xamarin,pwa>
Url-6“Youtube trendler, zamana ve bölgelere göre karşılaştırmalı döküm”,trends.google.com, (18.02.2021)
<https://trends.google.com/trends/explore?q=flutter,react%20native,cordova,xamarin,pwa>
Url-7 “Flutter Doküman”, flutter.dev, (21.10.2020)
<https://flutter.dev/>
Url-8 “Widget catalog”,flutter.dev,(26.12.2021)
<https://flutter.dev/docs/development/ui/widgets>
Url-9 “Çapraz Platform Deneyim Anketi”,docs.google.com,(28.07.2021)
<https://docs.google.com/forms/d/e/1FAIpQLSdOzuKBN2nxwRZnCalZ7MdoQMKi6GjTTQGcMN1SrBf0569PgA/viewform>
Url-10 “Çapraz Platform Deneyim Anket verileri”,docs.google.com, (28.07.2021)
<https://docs.google.com/forms/d/1NPu0AZfcW_Lm9Cfi5ZuTxIfIlTiU0Bx_wWQoLvCvv6A/viewanalytics?usp=form_confirm >

EKLER

EK A: Yazışmalar.

EK B: Çapraz Platform Deneyim Anketi.

EK C: Çapraz Platform Deneyim Anket Yanıtları.

EK A: Yazışmalar

30.03.2021

T.C.

İstanbul Gedik Üniversitesi Rektörlüğü'ne

Üniversitemiz bünyesinde, Uzaktan Eğitim Uygulama ve Araştırma Merkezi'nde (UZEM) görev almaktayım. Bilgisayar Mühendisliği alanında Yüksek Lisans yapmaktayım. Tez konusu olarak, yeni nesil programlama dili olan *"Çapraz Platform Yazılımları"* üzerine çalışmamı yürütmekteyim. Tez çalışmamda, Çapraz Platform yazılımlarından, Flutter ve React Native Performans karşılaştırılması yapılması planlanmıştır. Bu bağlamda, Üniversitemizin; Fen Bilimleri Enstitüsü Yüksek Lisans programlarından Yapay Zekâ Mühendisliği, Mühendislik Fakültesinden Yapay Zekâ Mühendisliği ve Bilgisayar Mühendisliği, Meslek Yüksek Okuluna bağlı Teknik Programlarından, Bilişim Güvenliği Teknolojisi ve Bilgisayar Programcılığı programında eğitim gören öğrencilerimize, mesleki farkındalık uyandırmak ve yeni geliştirilmiş olan Cross - Platform Software (Platform Bağımsız Yazılım) alanında, Google Trendlerinde üst sıralarda yer alan Flutter (Google şirketi, tarafından oluşturulan açık kaynaklı yazılım geliştirme kiti), React Native (Facebook şirketi, tarafından oluşturulan açık kaynaklı yazılım geliştirme kiti) ve SDK (Yazılım geliştirme kiti) yazılımlarının tanıtımı yapmak ve giriş seviyesi bir uygulama geliştirmek istemekteyim. Yazılımların tanıtım ve uygulama aşamasından sonra öğrencilere, Çapraz Platform yazılımları hakkındaki bir anket çalışması (EK1) yapılacaktır. Anket çalışmasında, öğrencilerin yazılımlar hakkındaki sübjektif görüşleri ve düşünceleri alınacaktır. Elde edilen sonuçlar, Yüksek Lisans Tez çalışmamda kullanılacaktır. Belirtilen uygulamanın ve anket çalışmasının, öğrenciler ile gerçekleştirilebilmesi için gerekli izinlerin tarafıma verilmesini arz ederim.

gerekli duyuru yapılmıştır 31/03

Cumali TEKSÖZ
UZEM İçerik Tasarım & Geliştirme

EK1: Anket Formu

EK B: Çapraz Platform Deneyim Anketi

Çapraz Platform Yazılımları

Değerli Katılımcı,
Bu anket, Çapraz Platformalar arası performans ve kullanım kolaylığı arasında kullanıcıların kişisel deneyimleri sonucu ortaya çıkacak farklılık verilerinin toplanıp işlenerek, karşılaştırması adına kullanılacaktır. Bu anketin temel amacı, katılımcılara Capraz Platform yazılımlarının tanitilmasidir. Ayrıca bu alanda fikir sahibi olmaniz, iki çapraz platform uzerinde deneyim kazanmaniz ve deneyimlerinizi bizlerle paylasmaniz hedeflenmektedir. Kimliğinizi direkt olarak ortaya çıkaracak kimlik bilgilerinden kaçınılmış ve kimlik numarası, isim soyisim, telefon numarası, açık adres vb. bilgiler İSTENMEMEKTEDİR. Anketteki tüm soruları doğru ve eksiksiz tamamlamanız gerekmektedir. Cevaplamak istemediğiniz zaman anketi kapata bilir ve katılmaktan vaz geçebilirsiniz.
Ankette bulunan Çapraz Platformlar sorularına verdiğiniz cevapların Doğru veya Yanlış gibi sonuçlar doğuracak yanıtları bulunmamakta tamamen sübjektif cevaplarınız üzerinden Objektif bir sonuç elde etmek amacı ile kullanılacaktır. Ankette vermiş olduğunuz cevaplar kişisel hiçbir amaçla kullanılmamakta ve tüm cevaplarınızın yer aldığı bir şekilde hiçbir alanda paylaşılmayacaktır. Sadece bilimsel yayın ve/veya toplantı sunumlarında verilen genel cevapların karşılaştırılmasında kullanılacaktır.
Bu anketi doldurarak yukarıda yazanları okuduğunuzu kabul etmiş ve vereceğiniz bilgilerin araştırmacı tarafından analiz edilip kullanılmasına izin verdiğinizi kabul etmiş olursunuz.

Sorularınız için cumali.teksoz@gedik.edu.tr adresinden ulaşabilirsiniz.
Çapraz Platformlar Hakkında Bilgi ve Kurulum Adımları;
Çapraz platform nedir?
Çapraz platformlar, uygun yazılım dilleri ve uygun SDK(Software Development Kit - Yazılım Geliştirme Kiti) yazılımları kullanılarak bir defa yazılan kod bloğunun farklı işletim sistemleri ve platformlarda kullanılmasını sağlayan SDK'lar olarak adlandırılmaktadır.
Bu ankette sizden deneyimlemeniz istenilen Çapraz Platformlardan;
Birincisi, Flutter SDK Google firması tarafından oluşturulmuş ve Dart yazılım dilini kullanmaktadır.
Flutter SDK yazılımını https://flutter.dev/docs/get-started/install adresinden indirebilir ve kurulum adımlarını aynı adres üzerinden takip edebilirsiniz.
İkincisi, React Native SDK Facebook firması tarafından oluşturulmuş ve JavaScript yazılım dilini kullanmaktadır.
React Native SDK yazılımını https://reactnative.dev/docs/environment-setup adresinden indirebilir ve kurulum adımlarını aynı adres üzerinden takip edebilirsiniz.

* Gerekli

1. E-posta *

2. Lütfen yaş aralığınızı belirtir misiniz? *

Yalnızca bir şıkkı işaretleyin.

- ◯ 14-18 Aralığında
- ◯ 19-23 Aralığında
- ◯ 24-28 Aralığında
- ◯ 29-33 Aralığında
- ◯ 34 ve Üstü

3. Cinsiyetinizi belirtir misiniz? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Kadın
- ◯ Erkek

4. Yaşadığınız şehrin plaka kodunu belirtir misiniz?(Örneğin, Adana ili için 01 şeklinde girilmelidir.) *

_____

5. Eğitim durumunuz nedir? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Lise
- ◯ Ön Lisans
- ◯ Üniversite
- ◯ Yüksek Lisans veya Üstü
- ◯ Diğer

6. Bu güne kadar herhangi bir yazılım dili ile kodlama yaptınız mı? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Evet
- ◯ Hayır

7. Hakim olduğunuzu düşündüğünüz yazılım dillerini seçer misiniz?(Bu alanda 2020 yılında popüler olan yazılım dillerinden bazıları sıralanmıştır. Birden fazla seçim yapabilirsiniz?) *

Uygun olanların tümünü işaretleyin.

- ☐ Python
- ☐ PHP
- ☐ Java
- ☐ C / C ++
- ☐ JavaScript
- ☐ Golang (Go)
- ☐ R
- ☐ C #
- ☐ Swift
- ☐ Dart
- ☐ Diğer
- ☐ Hiçbiri

8. Yaptığınız yazılımlar daha çok hangi platformlarda kullanıldı? *

Yalnızca bir şıkkı işaretleyin.

- ◯ World Wide Web(WEB)
- ◯ Mobil(Android, IOS, Windows,Diğer)
- ◯ Masaüstü(Windows, MacOS, UNIX, Linux, Pardus, Diğer)
- ◯ Diğer
- ◯ Hiçbiri

9. Aynı anda, birden fazla platforma yazılım yazmak zorunda kaldığınız durumlar oldu mu? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Evet
- ◯ Hayır

10. Birden fazla yazılım yazmak zorunda kaldığınız durumlarda hangi yazılım dillerini kullandınız ve nedenini kısaca açıklarmısınız? (Birden fazla yazılım yazmayan kişiler "Kullanmadım" yazmaları yeterlidir.) *

11. Bu çalışmadan önce, Cross - Platform Software, Platform Bağımsız Yazılım veya Çapraz Platform Yazımlar gibi isimlerini daha önce duydunuz mu? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Evet
- ◯ Hayır

12. Bu çalışmadan önce, Aşağıda belirtilen Yazılım Geliştirme Kitlerinden (SDK) herhangi biri/birilerini daha önce duydunuz mu?(Bu alanda birden fazla seçim yapabilirsiniz?) *

Uygun olanların tümünü işaretleyin.

- ☐ Flutter
- ☐ React Native
- ☐ Cordova
- ☐ Xamarin
- ☐ Hiçbirini Duymadım

13. Anket kapsamında sizeden inceleminiz istenilen Flutter ve React Native SDK'larından hangisinin arayüzünü diğerinden daha kullanışlı buldunuz? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Flutter SDK
- ◯ React Native SDK
- ◯ Hiçbiri

14. Anket kapsamında sizden istenilen ilk uygulama "Hello Flutter" ve "Hello React Native" yazılımlarından hangisini oluştururken daha fazla zorlandınız? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Flutter SDK
- ◯ React Native SDK
- ◯ Hiçbiri

15. Anket kapsamında sizden istenilen ikinci uygulama "Flutter Counter" ve "React - Native Counter" yazılımlarından hangisini oluştururken daha fazla zorlandınız? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Flutter SDK
- ◯ React Native SDK
- ◯ Hiçbiri

16. Flutter SDK yazılımını ve Dart dilini kullanmaya çalışırken yaşadığınız zorluğu puanlar mısınız, lütfen? *

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Kolay	◯	◯	◯	◯	◯	Zor

17. Dart dili ile yazılım oluştururken yaşadığınız zorluğu puanlar mısınız, lütfen? *

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Kolay	◯	◯	◯	◯	◯	Zor

18. React Native SDK yazılımını kullanmaya çalışırken yaşadığınız zorluğu puanlar mısınız, lütfen? *

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Kolay	◯	◯	◯	◯	◯	Zor

19. JavaScript dili ile yazılım oluştururken yaşadığınız zorluğu puanlar mısınız, lütfen? *

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Kolay	◯	◯	◯	◯	◯	Zor

20. Aşağıda belirtilen SDK yazılımlarını test ederken bilgisayarınızın cihaz özelliklerini göz önünde bulundurarak, bilgisayarınızda daha fazla İşlemci Gücünü(GHz) meşgul ettiğini gözlemlediniz? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Flutter
- ◯ React Native

21. Aşağıda belirtilen SDK yazılımlarını test ederken bilgisayarınızın cihaz özelliklerini göz önünde bulundurarak, bilgisayarınızda daha fazla Ekran Kartı bellek miktarını meşgul ettiğini gözlemlediniz? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Flutter
- ◯ React Native

22. Aşağıda belirtilen SDK yazılımlarını test ederken bilgisayarınızın cihaz özelliklerini göz önünde bulundurarak, bilgisayarınızda daha fazla Ram Bellek miktarını meşgul ettiğini gözlemlediniz? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Flutter
- ◯ React Native

23. Aşağıda belirtilen SDK yazılımları hakkında yaptığınız araştırmalarda hangisi ile ilgili daha hızlı ve daha fazla örnek içeriklere ulaşabildiniz? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Flutter
- ◯ React Native

24. Aşağıda belirtilen Yazılım Dilleri hakkında yaptığınız araştırmalarda hangisi ile ilgili daha hızlı ve daha fazla fazla örnek içeriklere ulaşabildiniz? *

Yalnızca bir şıkkı işaretleyin.

- ◯ Dart
- ◯ JavaScript

25. Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın daha fazla FPS(Frame Per Second/Saniyelik Görüntü Sayısı) daha yüksek FPS verdiğini gözlemlediniz? *

Her satırda yalnızca bir şıkkı işaretleyin.

	Flutter - Dart	React Native - JavaScript
Hello Flutter - Hello React Native	◯	◯
Flutter Counter - React Native Counter	◯	◯

26. Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın daha az İşlemci Gücü kullandığını gözlemlediniz? *

Her satırda yalnızca bir şıkkı işaretleyin.

	Flutter - Dart	React Native - JavaScript
Hello Flutter - Hello React Native	◯	◯
Flutter Counter - React Native Counter	◯	◯

27. Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın daha az Ram Bellek kullandığını gözlemlediniz? *

Her satırda yalnızca bir şıkkı işaretleyin.

	Flutter - Dart	React Native - JavaScript
Hello Flutter - Hello React Native	◯	◯
Flutter Counter - React Native Counter	◯	◯

28. Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın daha az Batarya kullandığını gözlemlediniz? *

Her satırda yalnızca bir şıkkı işaretleyin.

	Flutter - Dart	React Native - JavaScript
Hello Flutter - Hello React Native	◯	◯
Flutter Counter - React Native Counter	◯	◯

29. Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın boyutu daha az yer(HDD) tutmaktaydı? *

Her satırda yalnızca bir şıkkı işaretleyin.

	Flutter - Dart	React Native - JavaScript
Hello Flutter - Hello React Native	◯	◯
Flutter Counter - React Native Counter	◯	◯

30. Yeni bir Çapraz Programlama dili öğrenmek istediğinizde aşağıda belirtine hangi SDK ve Programlama dillini öğrenmeyi tercih ederdiniz?

Yalnızca bir şıkkı işaretleyin.

- ◯ Flutter - Dart
- ◯ Recat Native - JavaScript
- ◯ Diğer: _____

Bu içerik Google tarafından oluşturulmamış veya onaylanmamıştır.

Google Formlar

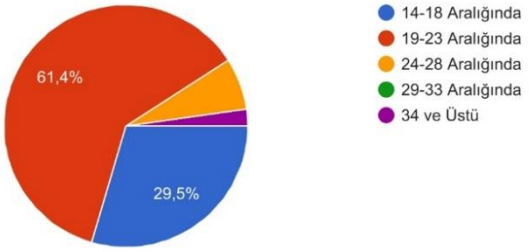
EK C: Çapraz Platform Deneyim Anket Yanıtları

Çapraz Platform Yazılımları

44 yanıt

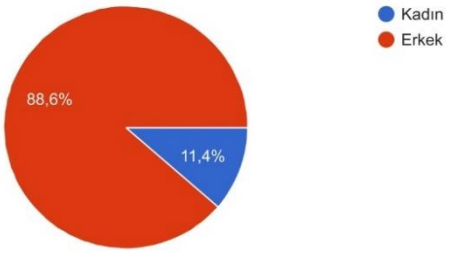
Lütfen yaş aralığınızı belirtir misiniz?

44 yanıt



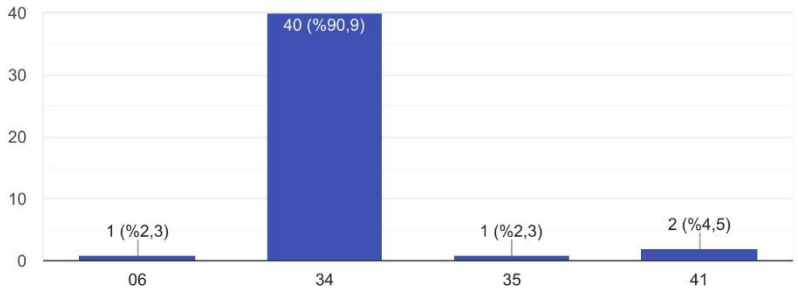
Cinsiyetinizi belirtir misiniz?

44 yanıt



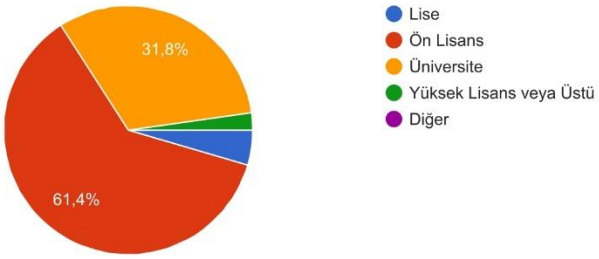
Yaşadığınız şehrin plaka kodunu belirtir misiniz?(Örneğin, Adana ili için 01 şeklinde girilmelidir.)

44 yanıt



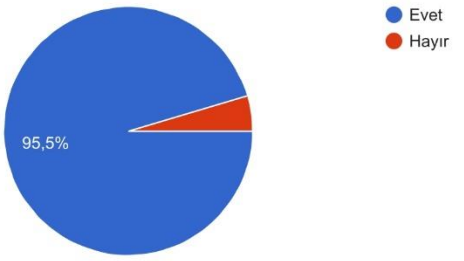
Eğitim durumunuz nedir?

44 yanıt



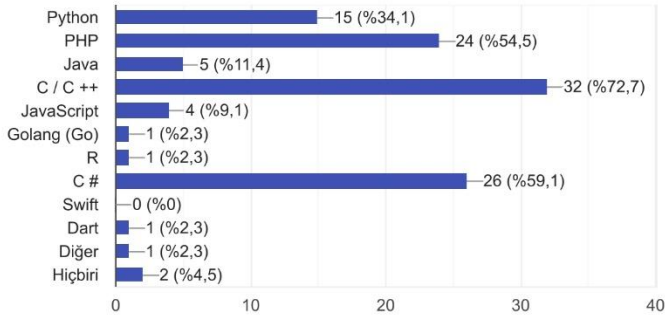
Bu güne kadar herhangi bir yazılım dili ile kodlama yaptınız mı?

44 yanıt



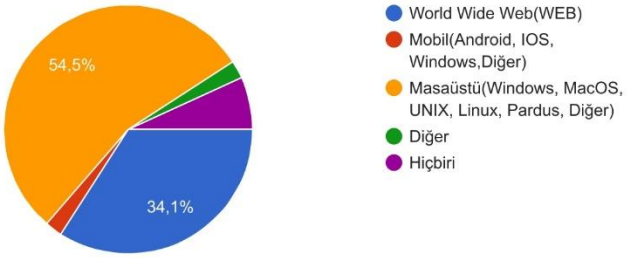
Hakim olduğunuzu düşündüğünüz yazılım dillerini seçer misiniz?(Bu alanda 2020 yılında popüler olan yazılım dillerinden bazıları sıralanmıştır. Birden fazla seçim yapabilirsiniz?)

44 yanıt



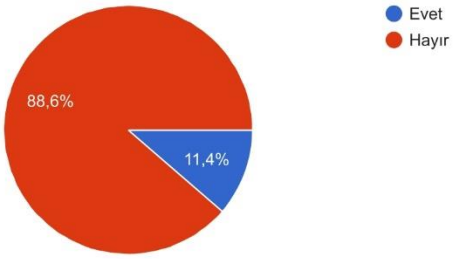
Yaptığınız yazılımlar daha çok hangi platformlarda kullanıldı?

44 yanıt



Aynı anda, birden fazla platforma yazılım yazmak zorunda kaldığınız durumlar oldu mu?

44 yanıt



Birden fazla yazılım yazmak zorunda kaldığınız durumlarda hangi yazılım dillerini kullandınız ve nedenini kısaca açıklarmısınız? (Birden fazla yazılım yazmayan kişiler "Kullanmadım" yazmaları yeterlidir.)

44 yanıt

Kullanmadım

kullanmadım

Kullanmadım

Kullanmadim

Merak ettiğimden öğrenmek istedim

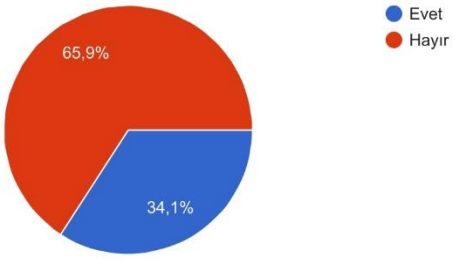
Esneklik ve kullanılabilirlik

hocam Ne kadar doğru olur bilmiyorum ama,
lisede web sitesi oluşturuken javascript ile web sitelerini efektif ve daha pratik bir hale getirmeye çalışıyorduk ama web sitemi access veritabanına bağlamak istediğim noktada asp kullanımına başvurmuştum.

KULLANMADIM

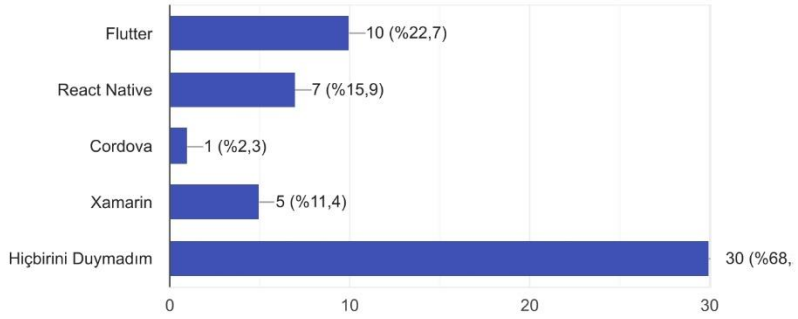
Bu çalışmadan önce, Cross - Platform Software, Platform Bağımsız Yazılım veya Çapraz Platform Yazımlar gibi isimlerini daha önce duydunuz mu?

44 yanıt



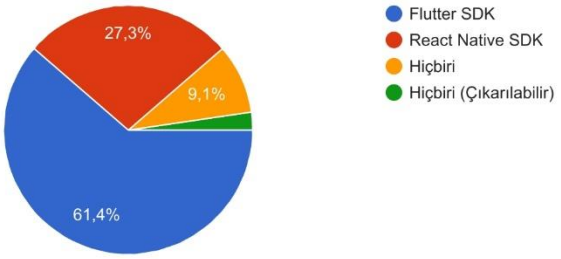
Bu çalışmadan önce, Aşağıda belirtilen Yazılım Geliştirme Kitlerinden (SDK) herhangi biri/birilerini daha önce duydunuz mu?(Bu alanda birden fazla seçim yapabilirsiniz?)

44 yanıt



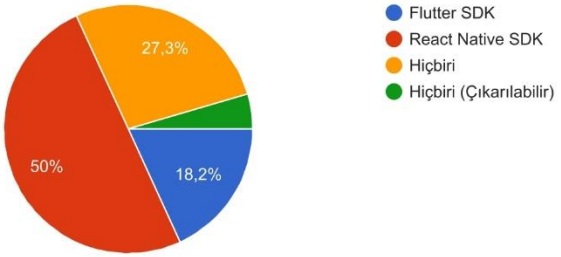
Anket kapsamında sizeden inceleminiz istenilen Flutter ve React Native SDK'larından hangisinin arayüzünü diğerinden daha kullanışlı buldunuz?

44 yanıt



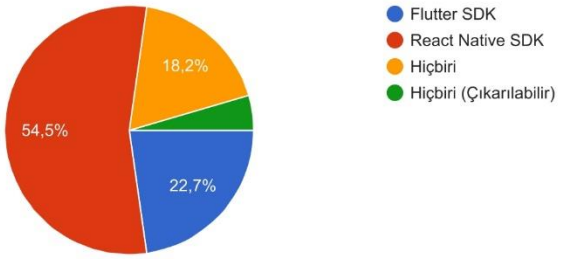
Anket kapsamında sizden istenilen ilk uygulama "Hello Flutter" ve "Hello React Native" yazılımlarından hangisini oluştururken daha fazla zorlandınız?

44 yanıt



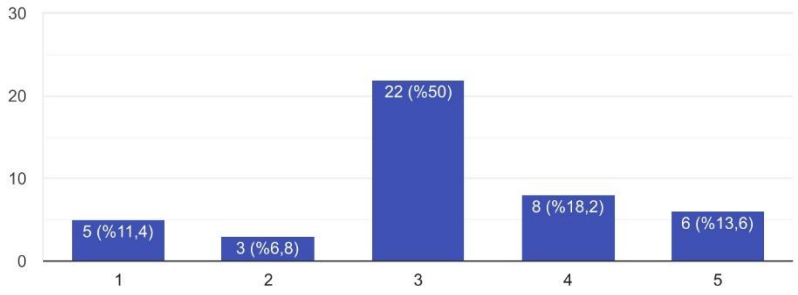
Anket kapsamında sizden istenilen ikinci uygulama "Flutter Counter" ve "React - Native Counter" yazılımlarından hangisini oluştururken daha fazla zorlandınız?

44 yanıt



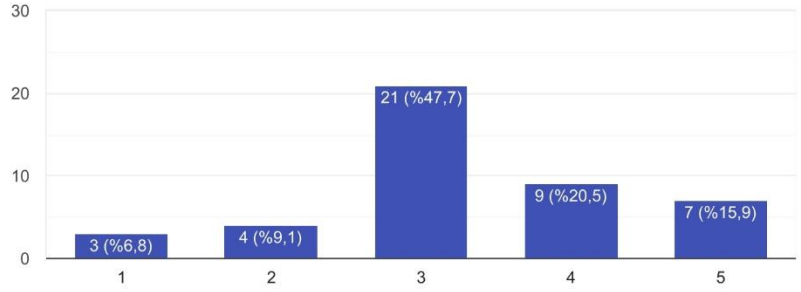
Flutter SDK yazılımını ve Dart dilini kullanmaya çalışırken yaşadığınız zorluğu puanlar mısınız, lütfen?

44 yanıt



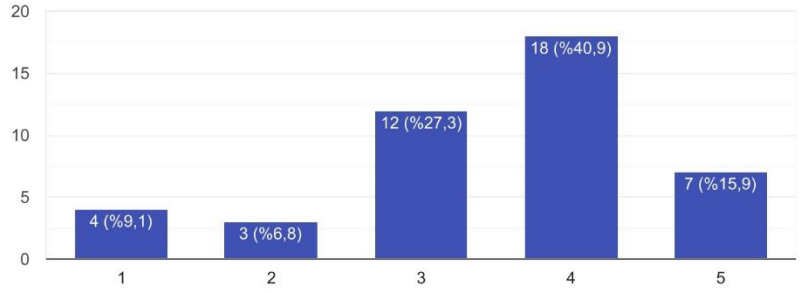
Dart dili ile yazılım oluştururken yaşadığınız zorluğu puanlar mısınız, lütfen?

44 yanıt



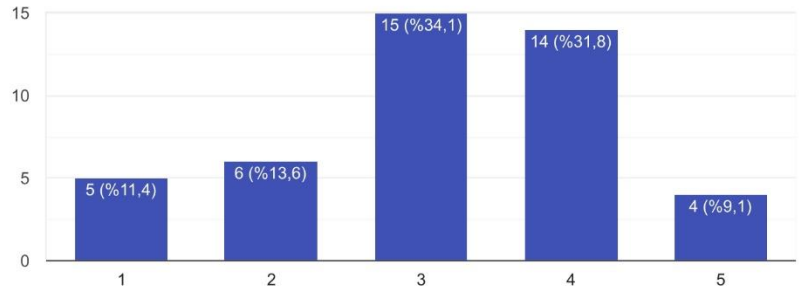
React Native SDK yazılımını kullanmaya çalışırken yaşadığınız zorluğu puanlar mısınız, lütfen?

44 yanıt



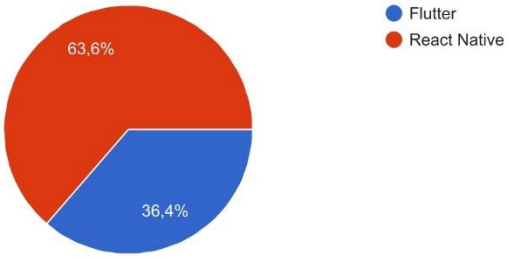
JavaScript dili ile yazılım oluştururken yaşadığınız zorluğu puanlar mısınız, lütfen?

44 yanıt



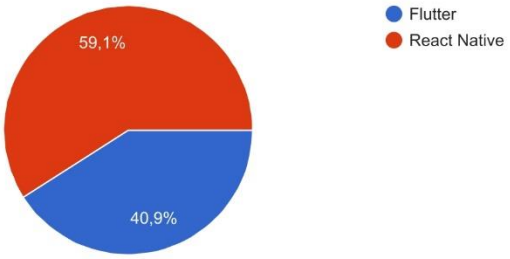
Aşağıda belirtilen SDK yazılımlarını test ederken bilgisayarınızın cihaz özelliklerini göz önünde bulundurarak, bilgisayarınızda daha fazla İşlemci Gücünü(GHz) meşgul ettiğini gözlemlediniz?

44 yanıt



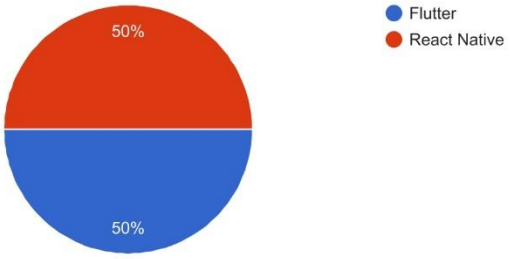
Aşağıda belirtilen SDK yazılımlarını test ederken bilgisayarınızın cihaz özelliklerini göz önünde bulundurarak, bilgisayarınızda daha fazla Ekran Kartı bellek miktarını meşgul ettiğini gözlemlediniz?

44 yanıt



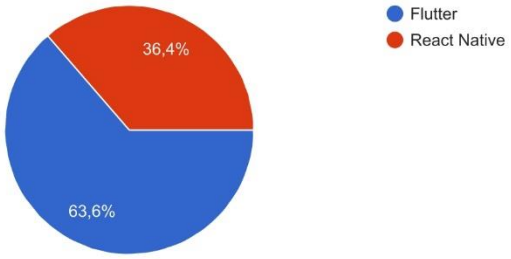
Aşağıda belirtilen SDK yazılımlarını test ederken bilgisayarınızın cihaz özelliklerini göz önünde bulundurarak, bilgisayarınızda daha fazla Ram Bellek miktarını meşgul ettiğini gözlemlediniz?

44 yanıt



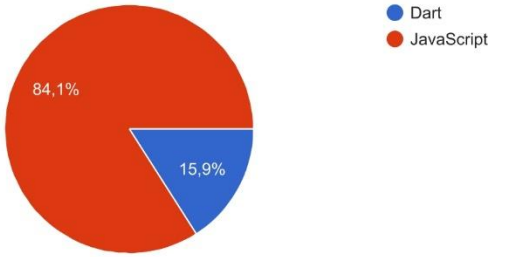
Aşağıda belirtilen SDK yazılımları hakkında yaptığınız araştırmalarda hangisi ile ilgili daha hızlı ve daha fazla örnek içeriklere ulaşabildiniz?

44 yanıt

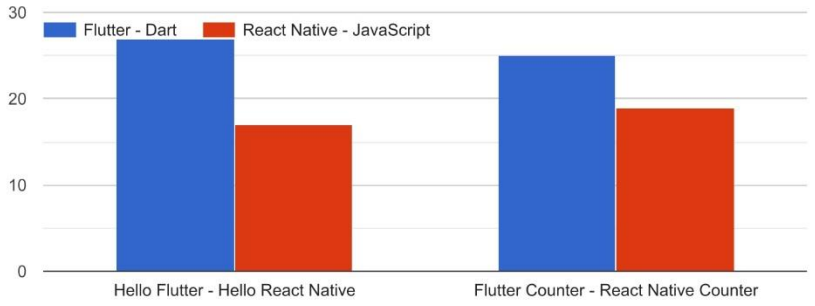


Aşağıda belirtilen Yazılım Dilleri hakkında yaptığınız araştırmalarda hangisi ile ilgili daha hızlı ve daha fazla fazla örnek içeriklere ulaşabildiniz?

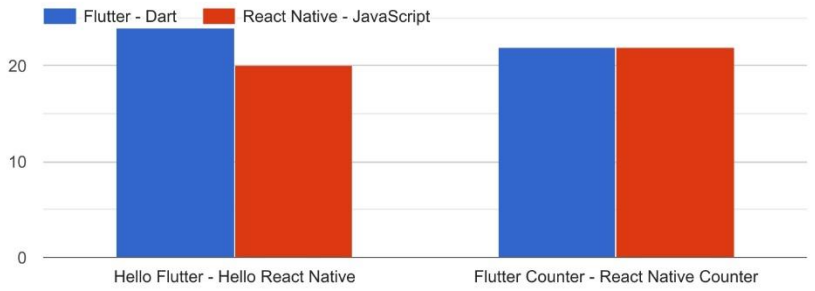
44 yanıt



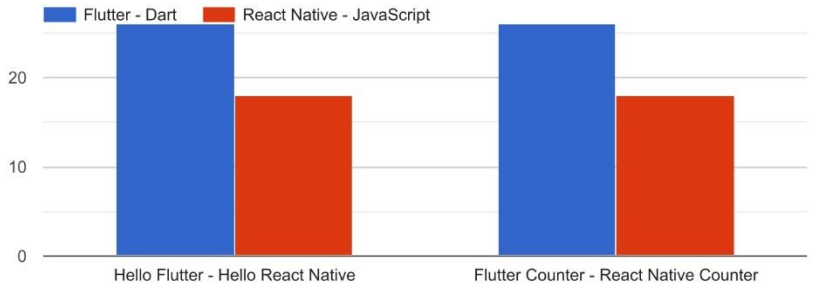
Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın daha fazla FPS(Frame Per Second/Saniyelik Görüntü Sayısı) daha yüksek FPS verdiğini gözlemlediniz?



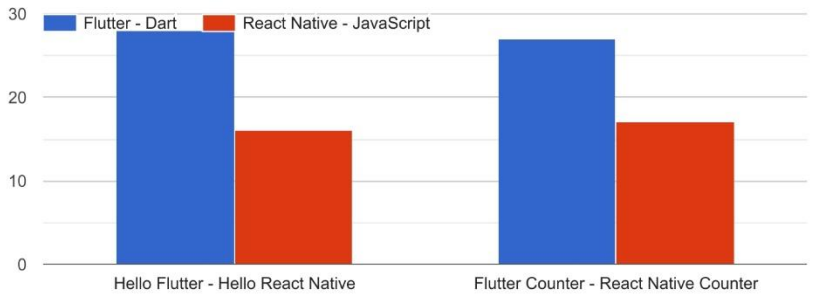
Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın daha az İşlemci Gücü kullandığını gözlemlediniz?



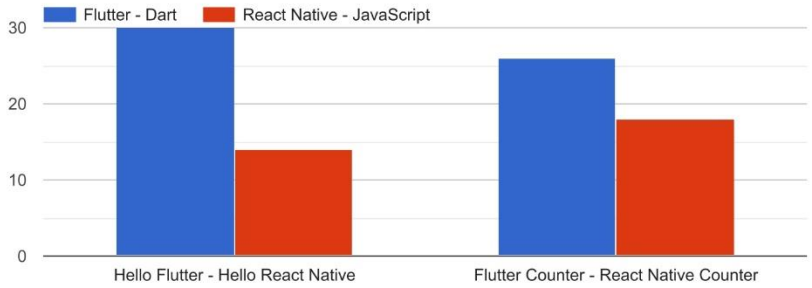
Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın daha az Ram Bellek kullandığını gözlemlediniz?



Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın daha az Batarya kullandığını gözlemlediniz?

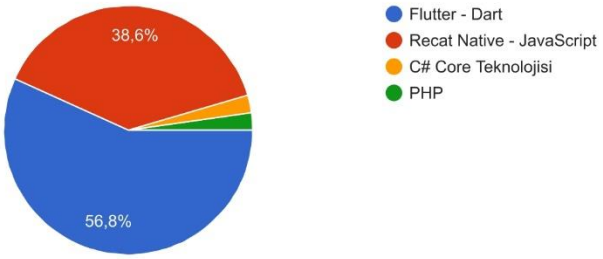


Yaptığınız uygulamalarda kullandığınız Emülatör veya Gerçek Cihaz testlerinde hangi SDK yazılımı ile yapılmış uygulamanın boyutu daha az yer(HDD) tutmaktaydı?



Yeni bir Çapraz Programlama dili öğrenmek istediğinizde aşağıda belirtine hangi SDK ve Programlama dillini öğrenmeyi tercih ederdiniz?

44 yanıt



Bu içerik Google tarafından oluşturulmamış veya onaylanmamıştır. Kötüye Kullanımı Bildirme - Hizmet Şartları - Gizlilik Politikası

Google Formlar

ÖZGEÇMİŞ

1. **Adı Soyadı:** Cumali TEKSÖZ

2. **Öğrenim Durumu**

   Ön Lisans: 2014 - 2016, Anadolu Üniversitesi, Bankacılık ve Sigortacılık Bölümü.

   Lisans: 2011 - 2015, İstanbul Aydın Üniversitesi, Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bölümü.

   Yüksek Lisans: 2019 - , İstanbul Aydın Üniversitesi, Bilgisayar Mühendisliği Bölümü.