T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇAPRAZ PLATFORM MOBİL UYGULAMA GELİŞTİRME ARAÇLARININ KARŞILAŞTIRILMASI VE DEĞERLENDİRİLMESİ

Mehmet IŞITAN

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalını

Aralık-2020
KONYA
Her Hakkı Saklıdır

TEZ KABUL VE ONAYI

Mehmet IŞITAN tarafından hazırlanan "Çapraz Platform Mobil Uygulama Geliştirme Araçlarının Karşılaştırılması ve Değerlendirilmesi" adlı tez çalışması 31/12/2020 tarihinde aşağıdaki jüri üyeleri tarafından oy birliği ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Doç. Dr. Humar KAHRAMANLI ÖRNEK	……..
Danışman Dr. Öğr. Üyesi Murat KÖKLÜ	……..
Üye Dr. Öğr. Üyesi Levent CİVCİK	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Sait GEZGİN
FBE Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Mehmet IŞITAN

Tarih: 31.12.2020

ÖZET

YÜKSEK LİSANS TEZİ

ÇAPRAZ PLATFORM MOBİL UYGULAMA GELİŞTİRME ARAÇLARININ KARŞILAŞTIRILMASI VE DEĞERLENDİRİLMESİ

Mehmet IŞITAN

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Murat KÖKLÜ

2020, 64 Sayfa

Jüri
Doç. Dr. Humar KAHRAMANLI ÖRNEK
Dr. Öğr. Üyesi Levent CİVCİK
Dr. Öğr. Üyesi Murat KÖKLÜ

Günümüzde mobil cihazlara ait pek çok farklı işletim sistemi platformu bulunmaktadır. Piyasada kullanılmış ve kullanılmaya devam eden Blackberry, Ubuntu, Symbian, BADA, Palm, Maemo, Meego, Verdict, Open WebOS gibi işletim sistemleri olsa da en çok kullanılanları Android, IOS ve Windows Phone'dur ve bu üçünün de yazım dili ve platformları birbirinden tamamen bağımsızdır. Dolayısıyla her işletim sisteminde çalışacak program da o sisteme uygun şekilde geliştirilmelidir. Bu zorunluluk, mobil uygulama geliştiricilerini oldukça zorlu, vakit alıcı ve maliyetli bir sürece itmektedir. Bu sorunun çözümü için tek seferde yazılan kodla daha hızlı, daha kolay ve daha az maliyetle ihtiyaç duyulan platformlara uygulama çıktısı verecek frameworkler geliştirilmiştir.

Çapraz platform mobil uygulama geliştirme araçları olarak adlandırılan bu sistemlerin son zamanlarda çeşitlenmesi ile hangisinin tercih edilmesi gerektiği, geliştiriciler açısından merak konusu olmuştur. Her birinin kendi aralarında artı ve eksi yönleri bulunmaktadır ve geliştirilecek programın içeriğine göre bile biri diğerine üstünlük sağlayabilmektedir. Ayrıca bu geliştirme ortamlarında kullanılabilecek oldukça fazla 3. parti yazılım bulunmaktadır. Bu yazılımların çeşitliliği, düzgün, hızlı ve hatasız çalışması da seçimde etkili olan parametrelerden biri olmaktadır.

Bu çalışmada, son dönemlerde çıkan çapraz platform mobil uygulama geliştirme araçları da dahil olmak üzere her birinin artıları ve eksileri bir geliştiricinin bakış açısı baz alınarak ayrı ayrı değerlendirilip ölçümleri yapılacak ve işlemci, bellek, pil ve ağ kullanımı, kod yapısı, popülarite, üçüncü parti yazılım desteği, açılma(render) süreleri, hız-performans gibi konularda karşılaştırmalarının yapılarak geliştiricilere kendi ihtiyaçlarına hangi frameworkün daha uygun olduğunu bulmasına yardımcı olunması da hedeflenmiştir.

Anahtar Kelimeler: Çapraz platform, hybrid geliştirme, mobil uygulama, tek kod

ABSTRACT

MS THESIS

COMPARISON AND EVALUATION OF CROSS PLATFORM MOBILE APPLICATION DEVELOPMENT TOOLS

Mehmet IŞITAN

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER ENGINEERING

Advisor: Asst. Prof. Dr. Murat KÖKLÜ

2020, 64 Pages

Jury
Assoc. Prof. Humar KAHRAMANLI ÖRNEK
Asst. Prof. Dr. Levent CİVCİK
Asst. Prof. Dr. Murat KÖKLÜ

Today, there are many different operating system platforms for mobile devices. Although there are operating systems such as Blackberry, Ubuntu, Symbian, BADA, Palm, Maemo, Meego, Verdict, Open WebOS that have been used and continue to be used in the market, the most used ones are Android, IOS and Windows Phone and the writing language and platforms of all three are completely independent from each other. Therefore, the program to run on each operating system should be developed in accordance with that system. This obligation forces mobile application developers to have a very difficult, time-consuming and costly process. In order to solve this problem, frameworks have been developed to give application output to the platforms needed faster, easier and with less cost with the code written at once.

With the recent diversification of these systems, which are called cross platform mobile application development tools, which one should be preferred has been a matter of curiosity for developers. Each has its own pros and cons, and even depending on the content of the program to be developed, one can be superior to the other. There is also quite a lot of 3rd party software that can be used in these development environments. The variety, fast and error-free operation of these software is also one of the effective parameters in the selection.

In this study, the pros and cons of each, including the recent cross-platform mobile application development tools, will be evaluated and measured individually based on a developer's perspective, and the processor, memory, battery and network usage, code structure, popularity, third-party software support, rendering times, speed-performance, etc., helping developers to find out which framework is more suitable for their needs.

Keywords: Cross platform, hybrid development, mobile application, single code

ÖNSÖZ

Tez çalışmamda bana yol gösteren, sürekli teşvik eden, her konuda desteğini esirgemeyen ve her türlü bilimsel katkıyı sağlayan değerli tez danışmanım Sayın Dr. Öğr. Üyesi Murat KÖKLÜ'ye, teşekkür eder şükranlarımı sunarım. Ayrıca, bu tez çalışması süresince bana destek olan tüm aileme (ve biricik oğlum Anıl Efe'me) de minnettarlığımı ve sevgilerimi sunarım.

Mehmet IŞITAN
Konya-2020

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Kısaltmalar

IDE	: Integrated Development Environment
API	: Application Programming Interface
GUI	: Graphical User Interface
SDK	: Software Development Kit
JVM	: Java Virtual Machine
GUI	: Graphical User Interface
CLI	: Command Line Interface
IDC	: International Data Corporation
APK	: Android Application Package
USB	: Universal Serial Bus

1. GİRİŞ

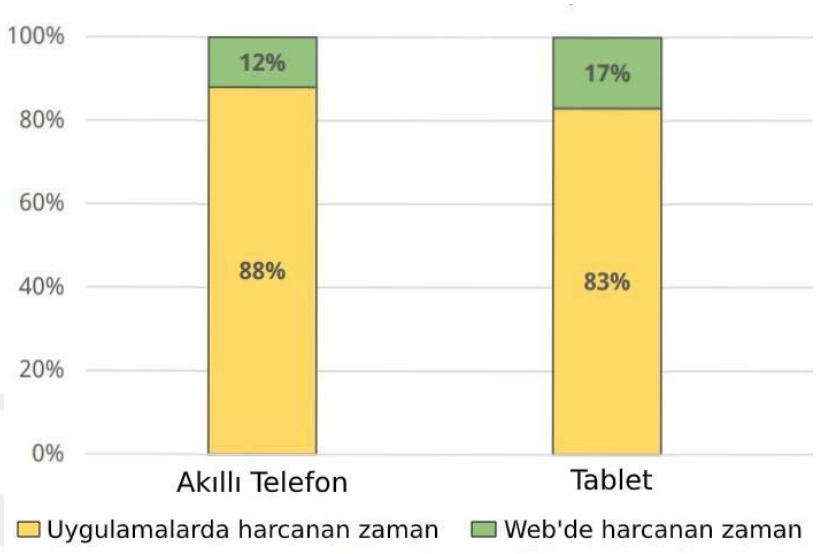
Bilindiği üzere, piyasada kullanılmış ve kullanılmaya devam eden Blackberry, Ubuntu, Symbian, BADA, Palm, Maemo, Meego, Verdict, Open WebOS gibi birçok mobil işletim sistemleri olsa da en çok kullanılanları Android, IOS ve Windows Phone'dur. Bu işletim sistemlerinin üzerinde çalışacak programların programlanma süreçlerine bakıldığında, üçünün de yazım dili ve platformlarının birbirinden tamamen bağımsız olduğu gözlenmektedir. Bu yüzden her işletim sisteminde çalışacak program da o sistemin belirlediği dile ve şablona uygun şekilde geliştirilmelidir. Bu zorunluluk, mobil uygulama geliştiricilerini oldukça zorlu, vakit alıcı ve maliyetli bir sürece itmektedir.

Son yıllarda gelişen teknolojinin de etkisiyle mobil cihazlara uygulama geliştirme konusunda devrim niteliğinde kolaylık sağlayacak bir çözüm yolu geliştirilmiştir. Çapraz platform mobil uygulama geliştirme araçları olarak adlandırılan bu araçlar sayesinde yalnızca bir platform üzerinde yazılım geliştirilerek tüm bu işletim sistemlerinin belirlediği yazılım geliştirme dillerine ve şablonlarına uygun şekilde çıktı alınabilmektedir. Bu araçları kullanırken yalnızca belirlenen aracın kendi belirlediği yazılım dili ve formatına uyarak yazılım geliştirmek yeterli olacaktır. Bu sayede seçilen platformun sağladığı ortamı kullanarak yazılan kodla daha hızlı, daha kolay ve daha az maliyetle ihtiyaç duyulan platformlara uygulama çıktısı alınabilmektedir.

Çapraz platform ve hybrid mobil uygulama geliştirme araçlarının son zamanlarda çeşitlenmesi ile hangisinin tercih edilmesi gerektiği geliştiriciler açısından merak konusu olmuştur. Her birinin kendi aralarında artı ve eksi yönleri bulunmaktadır ve geliştirilecek programın içeriğine göre bile biri diğerine üstünlük sağlayabilmektedir. Ayrıca bu geliştirme ortamlarında oldukça fazla 3. parti yazılım bulunmaktadır. Bu yazılımların çeşitliliği, düzgün, hızlı ve hatasız çalışması da seçimde etkili olan parametrelerden biri olmaktadır.

Bu çalışmada, son dönemlerde çıkan çapraz platform mobil uygulama geliştirme frameworkleri de dahil olmak üzere her birinin artıları ve eksileri bir geliştiricinin bakış açısı baz alınarak ayrı ayrı değerlendirilip ölçümleri yapılacak ve işlemci, bellek, pil ve ağ kullanımı, kod yapısı, açılma(render) süreleri, popülarite, üçüncü parti yazılım desteği, hız-performans gibi konularda karşılaştırmalarının yapılarak geliştiricilere kendi ihtiyaçlarına hangi frameworkün daha uygun olduğunu bulmalarına yardımcı olunması da hedeflenmiştir.

2017 yılında yapılan bir çalışmaya göre kullanıcılar artık yazılımsal uygulamaları daha çok web siteler üzerinden değil, mobil uygulamalar üzerinden kullanmaktadır. Çalışmaya ait rakamsal bilgiler Şekil 1.1. üzerindeki grafikte verilmiştir.



Şekil 1.1. Mobil uygulamalar ve web siteleri üzerinde harcanan zaman payları (Mehlhorn, 2017)

1.1. Çalışmanın Amacı ve Önemi

Mobil uygulama geliştiricileri, çapraz platform mobil uygulama geliştirme araçları çıkmadan önce yazmak istedikleri bir mobil uygulamayı ortaya çıkarabilmek için piyasada en çok kullanılan üç işletim sisteminin gerektirdiği yazılım geliştirme dili ve şablonuna uygun olacak şekilde üç ayrı uygulama geliştirmekteydiler. Çünkü bu üç platformun yazılım geliştirme süreçleri birbirlerinden tamamen bağımsızdır.

Android işletim sistemine uygun uygulama geliştirmek için Java programlama dili ve geliştirmeye uygun Android Studio gibi bir geliştirme aracı, iOS işletim sistemine uygun uygulama geliştirmek için C programlama diline benzeyen yapısıyla Objective-C veya Swift ve geliştirmeye uygun Xcode gibi bir geliştirme aracı, Windows Phone işletim sistemine uygun uygulama geliştirmek için C# dili ve geliştirmeye uygun Visual Studio gibi bir geliştirme aracı gerekmektedir.

Uygulama çıktısı alınmak istenen tüm işletim sistemlerinin gerektirdikleri kurallara uygun ayrı ayrı uygulamalar yazmak oldukça yorucu, vakit alıcı ve en önemlisi maliyetli bir süreçtir. Gelişen teknolojinin etkisiyle, bu sorunlu süreci ortadan kaldırabilen mobil uygulama geliştirme araçları ortaya çıkmıştır. Bu araçlar da son zamanlarda fazlasıyla artarak birbirleri ile rekabet etme noktasına ulaşmıştır. Çünkü bu

geliştirme araçlarının birbirlerine göre birçok eksi ve artı yönleri bulunmaktadır. Bu yüzden geliştirilmek istenen uygulamaya en iyi performansı ve en basit çözümü getirecek geliştirme aracı da ihtiyaca göre bu eksi veya artı yönlere göre geliştirme sürecini oldukça etkileyebilecektir.

Uygulama geliştiricileri mobil uygulama geliştirme araçlarından kendilerine en uygun olanının hangisi olduğunu belirleyebilmek için oldukça uzun vakitler harcamaktadırlar. Çünkü geliştiriciler bu araçlardan birini seçtikten sonra, öncelikle öğrenme ve profesyonelleşme sürecine, akabinde ise uygulamayı geliştirme sürecine girmektedirler.

Geliştirilecek olan uygulamada ihtiyaç duyulan özellikler de dikkate alınarak önce çapraz platform bir mobil uygulama geliştirme aracı ile bu uygulamanın yazılıp yazılamayacağı kontrol edilmelidir. Aynı şekilde geliştirilecek uygulama için ihtiyaç duyulan bu özellikler göz önüne alınarak hangi geliştirme aracında daha iyi performans alınacağı da detaylıca analiz edilmelidir.

Mobil uygulama geliştirme aracının seçiminde bir diğer önemli faktör ise aracın verdiği uygulama çıktılarının ilgili platformda sorunsuz çalışıp çalışamadığıdır. Bu araçları kullanan geliştiricilerin daha önce yaşadığı sorunlara göz atıldığında bu faktör de önemli bir yer tutmaktadır.

Geliştirilecek uygulama içerisinde geçecek modüller iyice belirlenmeli ve bu modüllerin her birinin seçilen uygulamada desteğinin olup olmadığı, yapılabilirliğinin detaylıca araştırılması gerekmektedir. Bu araçlar için yazılmış çok sayıda 3. parti yazılımlar bulunmaktadır. Bu yazılım parçacıkları bazı belirli işleri yapabilmekte ve geliştiricilere ciddi bir iş yükü ve vakit avantajı sağlamaktadırlar. Örneğin mobil cihazlar üzerinden imza atılabilmesini sağlayan yazılım parçacığı bu araçlar için 3. parti uygulama olarak daha önceden yazılmıştır.

Geliştirilecek olan uygulama içerisinde böyle bir modül olması halinde eğer geliştirici imza atma modülünü sıfırdan kendisi yazmak istemiyorsa, öncelikle bu modülün seçecek olduğu uygulama geliştirme aracında var olup olmadığını ve sonrasında performanslı çalışıp çalışmadığını, daha önce kaç kişinin kullandığını ve kullananların memnun kalıp kalmadığını detaylıca araştırmalıdır.

Bu çalışmadaki temel amaç geliştiricilere tüm bu hız, performans, ihtiyaç karşılama ve basit kodlama gibi kriterler göz önünde bulundurulduğunda çapraz platformda bir mobil uygulama geliştirmek için en uygun aracı seçmeye yardımcı olmaktır.

Son dönemlerde çıkan çapraz platform mobil uygulama geliştirme araçları da dahil olmak üzere her birinin artıları ve eksileri bir geliştiricinin bakış açısı baz alınarak ayrı ayrı değerlendirilecektir. Her bir platformun en temel uygulaması üzerinde bir uygulama geliştirilerek ölçümleri yapılacak ve kullanım kolaylığı, hız-performans, işlemci, bellek, pil ve ağ kullanımı, kod yapısı, her platformda sorunsuz çalışabilme gücü, popülarite, üçüncü parti yazılım desteği gibi konularda karşılaştırmalarının yapılarak geliştiricilere kendi ihtiyaçlarına hangi mobil uygulama geliştirme aracının daha uygun olduğunun bulunmasına yardımcı olunması da hedeflenmiştir.

1.2. Kaynak Araştırması

Allen ve arkadaşları (2010) tarafından yapılan çalışmada; platformlar arası geliştirmenin arkasındaki teori ve bunları kodla uygulamaya koyma ele alınmıştır. iPhone, BlackBerry, Windows Mobile ve Android için geliştirme ve dağıtım tekniklerinden, bu platformların her biriyle çalışmak için native kısmına değinilmiştir. PhoneGap ve Rhomobile gibi frameworklerin oluşturulma süreçleri değerlendirilmiştir. Ayrıca, Research In Motion (BlackBerry), Apple ve Microsoft dahil olmak üzere büyük uygulama mağazalarında uygulamaların yayınlanmasına da yer verilmiştir (Allen ve ark., 2010a).

Palmieri ve arkadaşları (2012) tarafından yapılan çalışmada; Rodos, PhoneGap, DragonRad ve MoSync olmak üzere dört popüler araç arasında pratik bir karşılaştırma ele alınmıştır. Karşılaştırmanın ana odak noktaları, uygulama arayüzleri, programlama dilleri, desteklenen mobil işletim sistemleri, lisanslar ve tümleşik geliştirme ortamları olmuştur. Ayrıca, araçlardaki genişletilebilirlik faktörü ve pazar payına getirebilecekleri gibi konulara da değinilmiştir. Karşılaştırma, geliştiricilerin ihtiyaçları doğrultusunda doğru seçimi yapmalarını desteklemeyi amaçlamıştır (Palmieri ve ark., 2012).

Heitkötter ve arkadaşları (2013) tarafından yapılan çalışmada, web uygulamalar, PhoneGap veya Titanium Mobile ile geliştirilen uygulamalar ve karşılaştırmak için native olarak geliştirilen uygulamalar değerlendirilmiştir. Bu değerlendirme ile, PhoneGap ürününün native arayüzler kullanan uygulamalarda daha uygun olduğu sonucuna varılmıştır (Heitkötter ve ark., 2012).

Dalmasso ve arkadaşları (2013), PhoneGap, Titanium ve Sencha Touch için bir anket düzenlemişlerdir. CPU, bellek kullanımı ve güç tüketimi açısından karşılaştırmalar yapılmıştır. Bu araçlar kullanılarak Android işletim sistemi üzerinde test uygulamaları

geliştirilmiştir. PhoneGap'ın, özel arayüz bileşenleri içermediğinden daha az bellek, CPU ve güç harcadığı tespit edilmiştir (Dalmasso ve ark., 2013a).

Amatya ve arkadaşları (2013) ise yaptığı çalışma ile birlikte; çapraz platform frameworklerinin tamamen olgunlaşmamasına rağmen, büyük potansiyel gösterdiklerini savunmuşlardır. Çalışmanın sonucunda, web tabanlı yaklaşımın platformlar arası mobil uygulama geliştirme için en iyisini sunduğu sonucuna varılmıştır (Amatya ve Kurti, 2013).

Gültürk Karlı (2014 tarafından yapılan çalışmada, çapraz platform mobil uygulama araçlarını kullanan geliştiricilere yardımcı olmak için geliştirilen yeni bir yazılım framework'ü önerilmiştir. Önerilen yazılım framework'ü, ortaya çıkan uygulamanın verimliliğini ve kalitesini artırmaya yönelik çeşitli özellikler sağlamıştır. Geliştirilen yazılım framework'ü veri madenciliği uygulamaları üzerinde deneysel olarak uygulanmıştır (Karlı, 2014).

Charkaoui ve arkadaşları (2014), yaptıkları çalışma ile çapraz platform mobil uygulama geliştirme frameworkünün seçiminin şu iki unsura bağlı olduğunu belirtmiştir. Bunlar; ne tür bir mobil uygulamaya ihtiyaç duyulduğu ve hedeflenen platformların gerekliliklerdir. Mevcut araçların yüksek kapasiteli uygulamalar için yetersiz olduğu sonucuna varılmıştır (Charkaoui ve Adraoui, 2014).

Dhillon ve arkadaşları (2014); PhoneGap, Appcelerator Titanium, Adobe Air ve MoSync araçlarını karşılaştırmışlardır. Çalışma sonucunda Adobe Air ve Appcelerator Titanium'un en iyi, PhoneGap'ın ise en kötü sonuçları verdiğini belirtmişlerdir (Dhillon ve Mahmoud, 2015).

Tunali ve arkadaşları (2015) yaptıkları çalışma ile PhoneGap, Xamarin, Appcelerator Titanium ve Smartface App Studio araçlarını teorik olarak bir tablo üzerinde birbirleri ile karşılaştırmışlardır (Tunali ve ark., 2015).

Boushehrinejadmoradi ve arkadaşları (2015) tarafından yapılan çalışmada; X-Checker adlı bir test aracı geliştirilmiştir. Windows Phone uygulamalarının native Android ve iOS uygulamalarında çapraz derlenmesini sağlayan popüler bir framework olan Xamarin test edilmiştir (Boushehrinejadmoradi ve ark., 2015).

Jiang (2016); Xamarin ve Cordova araçlarını kolay kullanım, kurulum, üretkenlik, bellek ve güç tüketimi, güvenlik, proje boyutu gibi parametreler ile puanlayarak değerlendirmiş ve birbirlerine yakın total puanlar elde etmiştir (Jiang, 2016).

Latif ve arkadaşları (2016); çapraz platform mobil uygulama geliştirme araçlarının temel gereksinimlerini araştırmak amacıyla bir anket üzerinde çalışmıştır.

MDA (Model Driven Architecture) yaklaşımının daha üstün olduğu sonucuna varılmıştır (Latif ve ark., 2016).

Öberg (2016) tarafından yapılan çalışmada Xamarin ve Cordova araçları Teknisk Förvaltning ismini verdikleri bir uygulama üzerinde değerlendirilmiştir. CPU ve RAM kullanımı, geliştirme hızı, uygulamanın başlama hızı ve DatePicker, AlertDialog gibi araçların görünüm karşılaştırmaları yapılmıştır (Öberg, 2016).

Ferreira ve arkadaşları (2018) ise PhoneGap, Sencha Touch ve Titanium frameworkleri ile native uygulamalar arasında karşılaştırma içeren bir çalışma yapmıştır. Araştırma ile mevcut olgunluk durumlarını ölçmeyi amaçlamışlardır. Fotoğraf çeken ve multimedia kaynaklarına erişen bir uygulama ile bellek kullanımları ve gösterdikleri performanslar ölçülmüştür. Native uygulamaların her anlamda daha iyi performans gösterdiklerini ve Titanium framework'ünün diğerlerine göre daha yavaş kaldığı sonucuna varılmıştır (Ferreira ve ark., 2018).

Shah ve arkadaşlarının (2019) yaptıkları çalışmada ise native uygulamalar ve çapraz platform framework'leri ile yazılan uygulamalar teorik olarak karşılaştırılmıştır. Neticede native uygulamalar ile istenilen türden uygulamalar geliştirilebilecekken çapraz platform frameworkleri ile Asphalt oyunu gibi büyük ölçekli uygulamaların geliştirilemeyeceği sonucuna varılmıştır (Shah ve ark., 2019).

1.3. Tezin Organizasyonu

Bu tez çalışmasının ana hatları aşağıdaki gibidir:

Birinci bölümde, tez çalışması hakkında genel bir bakış açısı verilmeye çalışılmıştır. Çalışmanın amacı ve çerçevesi ele alınmış, mevcut çalışmalara göre değerlendirilmiştir.

İkinci bölümde, çapraz platform mobil uygulama geliştirme araçlarının genel olarak uygulama çıktısı verebildiği işletim sistemleri üzerinde durulmuştur.

Üçüncü bölümde çapraz platform mobil uygulama geliştirme araçlarının çalışma mantığına ve ilgili kavramlara yer verilmiştir.

Dördüncü bölümde çapraz platform mobil uygulama geliştirme araçları hakkında bilgilere yer verilmiştir.

Beşinci bölümde son yıllarda piyasada en çok kullanılan çapraz platform mobil uygulama geliştirme araçlarının hangilerinin olduğu tespit edilmiş ve bunlar arasında karşılaştırma yapılabilmesi için kullanılacak yöntemden ve geliştirilen uygulamalardan bahsedilmiştir.

Altıncı bölümde ise bahsedilen uygulamalar ayrı ayrı aynı emülatör üzerinde debug modda çalıştırılmış ve uygulama boyutları, render (oluşturma) süreleri, işlemci, bellek, enerji ve ağ kaynaklarının kullanım oranlarına yer verilmiştir.

Yedinci bölümde ise tüm bu yapılan çalışmalar sonucunda varılan sonuçlara ve bazı önerilere yer verilmiştir.

2. UYGULAMA ÇIKTISI ALINACAK İŞLETİM SİSTEMLERİ

İşletim sistemi, kullanıcı ve donanım arasındaki etkileşimi sağlayan, karmaşık süreçleri emirleriyle yöneten, merkezi işlem birimi, ana bellek ve giriş-çıkış birimleri gibi bilgisayar kaynaklarını yapılmakta olan işler ve kullanıcılar arasında paylaşan, veri girişini ve çıkışını kontrol eden, gelecekteki planlar da dahil olmak üzere kaynakların kullanımını izleyen, uygulamaların işlemleri için uygun ortamlar yaratan ve çalıştıran, bilgisayar sisteminde depolanan verileri organize eden bir program veya yazılım grubudur. İşletim sistemi olmadan herhangi bir bilgisayarı çalıştırmak, işlemleri gerçekleştirmek ve hatta depolanan verilere ulaşmak mümkün değildir. (Karakoç ve Varol, 2016)

Bir işletim sistemi olmadan, her uygulamanın her donanım platformu için özel olarak geliştirilmesi gerekir. Bir işletim sisteminin gerçekleştirdiği ortak görevlerden bazıları dosya yönetimi, bellek yönetimi, ağ bağlantısı, süreç / iş parçacığı yönetimi, giriş / çıkış birimleri, ekranlar ve yazıcılar gibi çevre aygıtlarını kontrol etmektir. (Prendergast, 2017)

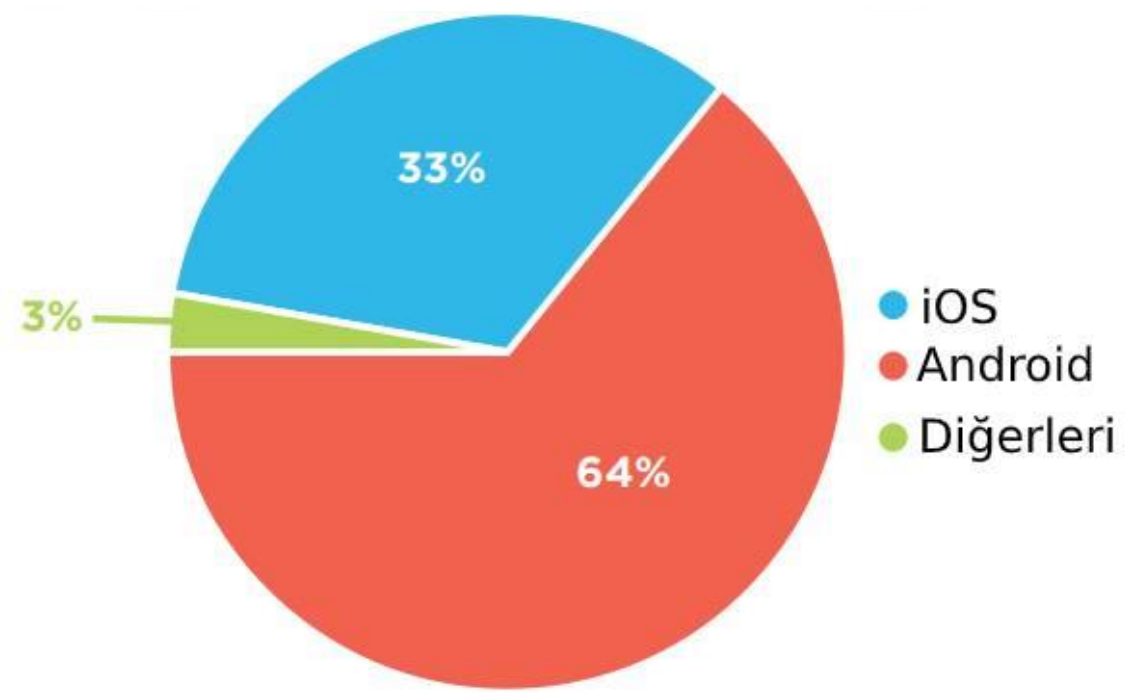
İşletim sistemleri sadece bilgisayar, cep telefonları, oyun konsolları ve web sunucularında değil; akıllı kartlarda, arabalarda, beyaz eşyalarda, el terminallerinde, akıllı kol saatlerinde kısacası içerisinde yazılım bulunan her yerde yüklü olabilir. İşletim sistemleri genel olarak, donanımları bir hedef için programlayıp yönlendirmeye yarayan yazılımlar olarak da değerlendirilebilir.

İşletim sistemleri, kullanım alanlarına dokuza ayrılmaktadır. Bunlar, Ana Bilgisayar (Mainframe) İşletim Sistemleri, Sunucu (Server) İşletim Sistemleri, Çok İşlemcili (Multiprocessor) İşletim Sistemleri, Kişisel Bilgisayar (Personal Computer) İşletim Sistemleri, El Bilgisayarı (Handheld) İşletim Sistemleri, Gömülü (Embedded) İşletim Sistemleri, Sensör-Düğüm (Sensor-Node) İşletim Sistemleri, Akıllı Kart (Smart Card) İşletim Sistemleri ve Gerçek Zamanlı (Real-Time) İşletim Sistemleri'dir.

Mobil işletim sistemi, mobil bir cihazda çalışan bir dizi veri ve programdır. Donanımı yönetir ve akıllı telefonlar, tabletler ve giyilebilir donanımların uygulamaları çalıştırmasını mümkün kılar. Bir mobil işletim sistemi ayrıca mobil multimedya fonksiyonlarını, mobil ve internet bağlantısını, dokunmatik ekranı, bluetooth bağlantısını, GPS navigasyonunu, kameraları, konuşma tanıma özelliğini ve bir mobil cihazda daha fazlasını yönetir. Çoğu işletim sistemi cihazlar arasında değiştirilemez. Bir Apple cihazında yüklü olan iOS işletim sistemi üzerine Android işletim sistemi ve aynı şekilde

bir Anroid işletim sistemi yüklü olan cihaza da iOS işletim sistemi yüklenemez. (Viswanathan, 2019)

Akıllı telefonlar için ilk mobil işletim sistemi olan Symbian, 2000 yılında piyasaya sürülmüştür. Bundan sonraki yirmi yılda, en çok kullanılan işletim sistemleri; Blackberry, Google Android, Apple iOS ve Microsoft Windows Phone olmuştur. Bu 20 yıl boyunca, pazar payı bazı bariz kaybedenler ve kazananlarla çarpıcı bir şekilde değişmiştir. Tüketici pazarı için, Google Android ve Apple iOS son yılların pazar liderleriyken, diğer tüm platformlar birkaç yüzde puanını paylaşmaktadır (Prendergast, 2017). Şekil 2.1'de 2017 yılında Prendergast tarafından yapılan çalışmaya göre mobil işletim sistemlerinin global pazar payına ait bir grafik bulunmaktadır.



Şekil 2.1. 2017 yılındaki mobil işletim sistemleri global pazar payı (Prendergast, 2017)

Google Play'de ve App Store'da mobil uygulamaların sayısı her geçen gün artmaktadır. Araştırma sonuçları, 2018'in sonunda Apple Store'da iki milyondan fazla uygulamanın ve Google Play'de 2.1 milyon uygulamanın bulunduğunu göstermektedir (Jawad, 2019).

Bu tezde incelenen çapraz platform mobil uygulama geliştirme araçları genel olarak yapılan araştırmalar sonucu elde edilen Şekil 2.1'deki grafiğin de etkisi ile pazar payı yaklaşık olarak %99'luk oranla en büyük paya sahip olan ve mobil cihazlarda en çok

kullanılan Android, iOS ve Windows Phone olmak üzere üç ayrı işletim sistemine uygulama çıktısı verebilmektedir.

2.1. Android

Google, ilk olarak 2008'de Android'i piyasaya sürmüş ve her yıl bir çeşit tatlı nesne isimleriyle yeni sürümler yayınlamıştır. Android, Linux çekirdeği üzerine kuruludur ve akıllı telefonlar ve tabletler gibi mobil cihazlar için optimize edilmiştir. 2 milyardan fazla aktif cihazla dünyanın en popüler mobil işletim sistemi haline gelmiştir. Bir tüketici ürünü olarak yayınlanan Google, mobil işletim sistemini daha kullanıcı dostu hale getirmek için sürekli olarak yeni güvenlik ve yönetim geliştirmeleri sunmuştur (Prendergast, 2017). Android, 3.0 ile birlikte büyümekte olan pazarda kendine pay bulabilmek adına birçok yeni özellik ekleyerek bugünkü halini almıştır (Boyer ve Mew, 2016).

International Data Corporation (IDC) tarafından tanıtılan Android, 2016'dan itibaren dünyanın en ünlü akıllı telefon işletim sistemidir (Bennis ve Amali, 2019).

Google'ın bir işletim sistemi olan Android, ara katman yazılımı ve akıllı telefonlar da dahil olmak üzere mobil cihazlarda kullanım için temel uygulamaları içeren açık ve ücretsiz bir yazılım yığınıdır. Açık kaynak kodlu Android mobil işletim sistemi için güncellemeler, yeni geliştirmeler ve iyileştirmelerle alfabetik olarak gelen her yeni sürümle birlikte "ilhamlı tatlı" sürüm adları (Cupcake, Donut, Eclair, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo, Pie, Q) altında geliştirilmiştir (son sürüm olan Q hariç) (Beal, 2018).

Tablo 2.1'de Android versiyonlarının yüzde bazında cihazlar üzerindeki dağılımı verilmiştir (Android, 2019).

Tablo 2.1. Android versiyonlarının yüzde bazında cihazlar üzerindeki dağılımları

Versiyon	Versiyon Adı	API	Yüzde
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Android'de çalışan bir mobil yazılım uygulaması bir Android uygulamasıdır. Uygulamalar, yükleyici dosya uzantısı olarak .apk uzantısını kullanır. Belirli bir Android sürümü için geliştirilen bir Android uygulaması, Android'in önceki sürümlerinde çalışmamaktadır. Örneğin, Android 2.2 (API seviye 8) için geliştirilen bir uygulama, Android 2.1 (API seviye 7) ve altında çalışmaz. Ancak, uygulama Android 2.2 ve üstü için uyumludur (Shah ve Abd Rahman, 2013). Android uygulamaları geliştirmek için Windows, Mac ve Linux dahil olmak üzere birden fazla platformda bulunan Android SDK'sı (uygulama geliştirme kiti) gerekir (Boyer ve Mew, 2016).

Google Play (eski adıyla Android Market), Android işletim sistemi için şarkı, film, kitap, uygulama ve oyun satan çevrimiçi mağazaların hizmetidir. Ocak 2015'ten bu yana bu mağazada 1,43 milyondan fazla uygulama bulunmaktadır. Android, uygulama indirmeleri açısından ilk sırada yer almaktadır. 2013 yazında 50 milyar indirme yapılmıştır. Başlangıçta sadece ücretsiz uygulamalar kabul edilirken 2009'dan beri ücretli uygulamalar da kabul edilmektedir (Novac ve ark., 2017).

Android uygulamaları Java ile yazılmıştır. Kotlin ve C/C+ + gibi diğer dillerde de yazılabilirler, ancak Java bu alanda en temel dildir. Java ve Android API'ları (koleksiyonlar gibi yaygın kütüphaneler vb.) birbirlerine benzemektedirler. Bununla birlikte, Java ve Android'in grafik kullanıcı arayüzü (GUI) bileşenleri tamamen farklıdır. Çünkü Android, GUI programlama için kendi bileşenlerini sunar (Cheon, 2019).

2017'de Google, Kotlin'i Android platformunun resmî bir programlama dili olarak duyurduğunda, geliştiriciler uygulama yazmak için başka bir programlama dili seçeneği kazanmıştır. Kotlin, Java Sanal Makinesi'ni (JVM) ve Android'i hedefleyen açık kaynaklı, statik olarak yazılmış bir programlama dilidir (Mateus ve Martinez, 2019).

Android işletim sistemi açık kaynaklı bir yazılımdır, yani herhangi bir kullanıcı işletim sisteminde iyileştirmeler yapabilir. Bu nedenle yalnızca Google geliştiricilerinin uzmanlığından değil, aynı zamanda üçüncü taraf geliştiricilerin uzmanlığından da yararlanabilir. Google, kaynak kodunun tamamını (kitaplıkların ve API'larin bazı bölümleri hariç) geliştiriciler için açmıştır. Böylece üreticiler açık kaynak topluluğunun kullanımına sunmadan uzantılar ekleyebilmektedir. (Novac ve ark., 2017).

Android geliştirme için resmi IDE(geliştirme ortamı) olan Android Studio, Java tabanlı kodun Kotlin'e dönüştürülmesine yardımcı olan yerleşik bir araç da dahil olmak üzere Kotlin için birinci sınıf destek sağlamaktadır (Cheon, 2019).

Google tarafından yapılan duyurunun Kotlin'in popülerliği üzerinde önemli bir etkisi olmuştur. Resmî bir Android dili olduktan sonra Stackoverflow yıllık geliştirici anketi, Kotlin'i 2018 ve 2019'da en çok istediği ve en çok sevdiği teknolojiler arasına almıştır. Android tarafından yapılan açıklamalara göre, birçok yeni API ve özelliğin önce Kotlin'de sunulacağı belirtilmiştir. Sonuç olarak, yeni bir projeye başlayan geliştiricilere bunu Kotlin dili ile yazmaları tavsiye edilmiştir (Cheon, 2019).

2.2. iOS

iOS (iPhone Operating System), Apple tarafından geliştirilen ve yalnızca Apple donanımı için dağıtılan bir mobil işletim sistemidir. 9 Ocak 2007'de Macworld Conference & Expo'daki iPhone ile tanıtılmış ve o yılın Haziran ayında piyasaya sürülmüştür (Allen ve ark., 2010a).

Apple, mobil işletim sistemini yıllık olarak yenilemiş, iPad ve iPod Touch'ı destekleyecek şekilde genişletmiştir. iOS, dünya çapında en popüler ikinci mobil işletim sistemi haline gelmiştir. Zarif, estetik ve basit bir kullanıcı ara yüzüne sahiptir. Yapılan araştırmalara göre, Apple cihazlarının özellikle bilgi çalışanları için şirket içi kullanımlarda popüler olduğu kanıtlanmıştır. Android'den daha az cihaz seçeneği vardır ve şarj için mikro USB'ye karşı tescilli bir Lightning konektörü gibi bazı sınırlamaları vardır. (Prendergast, 2017)

iOS işletim sistemlerinin temelinde Core OS Layer, Core Services Layer, Media Layer ve Cocoa Touch Layer olmak üzere 4 soyutlama katmanı vardır. Core OS katmanı,

iOS sisteminin alt katmanıdır ve doğrudan donanımla bağlantılıdır. Bu katman, harici aksesuarlara düşük düzeyli ağ erişimi, bellek yönetimi ve dosya sistemi gibi yaygın işletim sistemi hizmetlerini içeren bir dizi hizmet sunar. Core Services katmanı, daha önce oluşturulmuş birçok önceki katmanın inşa edildiği temeli sağlar. Media katmanı, cihazın grafik, ses, video ve araçlarını yönetmeyi içerir. Cocoa Touch katmanı, iOS uygulamaları oluşturmak için temel framework sağlar. Bu katman, temel uygulama altyapısını ve çoklu görev, dokunmaya dayalı giriş, bildirimler ve birçok üst düzey sistem hizmeti gibi temel destekleri sağlar. Uygulamanın bir kısmı serbestçe indirilebilir. iOS uygulaması diğer uygulamalarla doğrudan iletişim kuramaz. Bunun nedeni tüm kullanıcı kimlik bilgilerinin saldırıya uğramamasını sağlamak, hassas verilerin dışarıdakilere maruz kalmadığından emin olmak ve virüsleri anlamak, önlemek, düzeltmek ve kaldırmaktır (Ahmad ve ark., 2013).

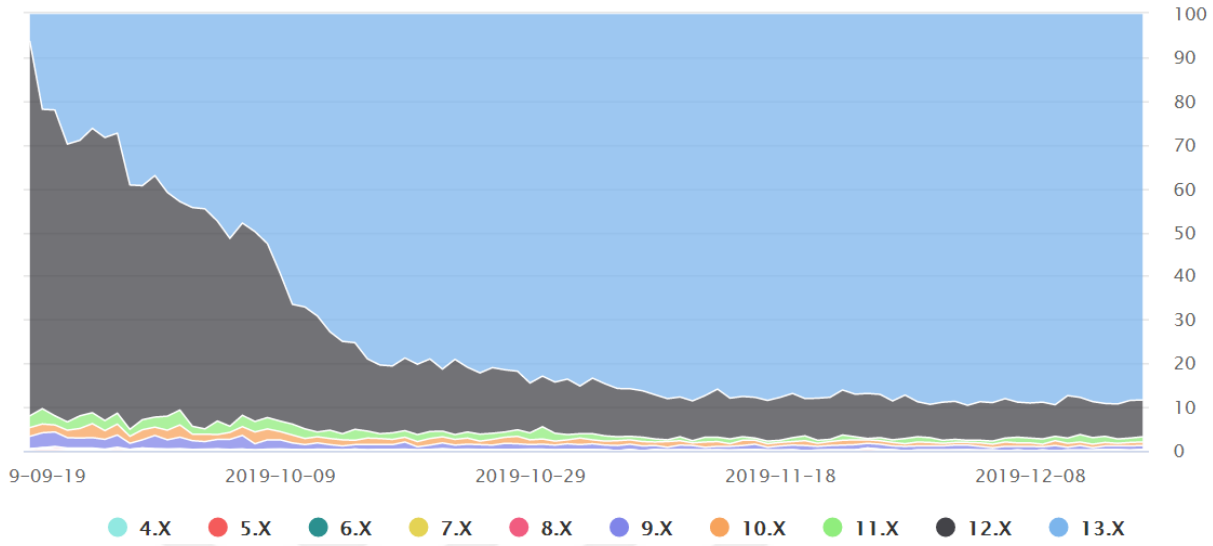
Kapalı ekosistemi nedeniyle, Apple iOS doğal olarak güvenlidir. Ayrıca, işletim sistemi, aygıt ve veri güvenliğini garanti etmek için tüm araçları sağlar. Güvenli önyükleme zinciri, çok faktörlü kimlik doğrulama, AES-256 şifreleme ve diğer birçok güvenlik özelliği ile güçlendirilmiştir. (Prendergast, 2017)

iOS yazılım geliştirme kiti (SDK), iOS işletim sistemi yüklü olan cihazlarda çalışacak olan native uygulamaları geliştirmek, yüklemek, çalıştırmak ve test etmek için gereken araçları ve arayüzleri içermektedir. Bu geliştirme kiti ile geliştirilen uygulamalar Objective-C dili kullanılarak oluşturulmakta ve doğrudan iOS üzerinde çalışmaktadır. Bu uygulamalar web uygulamalarının aksine, bir cihaza fiziksel olarak yüklenmekte ve bu nedenle ağ alışverişine ihtiyaç duymadığı taktirde cihaz uçak modundayken bile her zaman kullanıcı tarafından kullanılabilmektedir.

iOS işletim sistemine uygulama geliştirmek için, Mac OS çalıştıran bir bilgisayar gereklidir. Genellikle iOS uygulamalarını yazmak için kullanılan uygulama Xcode'dur. Güçlü bir düzenleyicinin yanı sıra bir analiz aracı, iOS simülatörü ve SDK içerir (Grønli ve ark., 2014).

App Store, iOS OS için Apple tarafından geliştirilen ve sürdürülen mobil uygulamalar için bir dijital dağıtım platformudur. App Store ile kullanıcılar Apple iOS işletim sisteminin SDK'sı ile geliştirilen uygulamaları edinebilmektedirler. Uygulamalar doğrudan iOS işletim sistemli bir cihaza veya kişisel bir bilgisayara indirilebilmektedir. 1 Şubat 2015'te App Store 1,4 milyondan fazla uygulamaya ve 75 milyardan fazla indirmeye ulaşmıştır (Novac ve ark., 2017).

iOS versiyonları 1.0'dan başlamış ve son olarak 17 Aralık 2019'da iOS 13.3.1 sürümü yayınlanmıştır. Şekil 2.2'de iOS sürümlerinin iOS kullanıcıları tarafından ne derece beğenildiği ile ilgili bir grafik bulunmaktadır. (Smith, 2019)



Şekil 2.2. iOS sürümlerinin iOS kullanıcıları tarafından beğenilme oranları

2.3. Windows Phone

Windows Phone işletim sistemi, Microsoft tarafından geliştirilmiştir. Windows Phone OS, Pocket PC 2000 işletim sisteminden başlayarak Windows CE çekirdeğine dayanan Windows Mobile platformunun halefidir. Windows Phone 7 işletim sistemi 15 Şubat 2010'da duyurulup 8 Kasım 2010'da piyasaya sürülmüştür. Bu işletim sisteminin öncesinde var olanı ise Windows Mobile 6.x sürümüdür. Kronolojik sırayla, Windows Phone sürümleri: Windows Phone 7, Windows Phone 7.5, Windows Phone 7.8, Windows Phone 8 (GDR1, CDR2, GDR3), Windows Phone 8.1 (GDR1, CDR2) ve Windows 10 (mobil)'dir (Novac ve ark., 2017).

Windows Phone OS, yeni Microsoft stratejisi nedeniyle 2015 yılında geri çekilmiştir. 2015 yılında, Windows 10'un PC sürümü hakkında evrensel bir deneyim sağlamak için tasarlanan Windows 10 OS (mobil) piyasaya sürülmüştür (Novac ve ark., 2017). Microsoft için tamamen yeni bir felsefe ortaya koyan "Bir Windows Platformu" stratejisi, tüm cihazlarda tek ve tutarlı bir işletim sistemi olma amacını taşımaktadır. Bu strateji, geliştiricilerin tüm platformlarda ve cihaz sınıflarında çalışan evrensel uygulamalar oluşturmasını sağlayan Evrensel Windows Platformu'nu (UWP) içermektedir. (Prendergast, 2017)

Apple (iOS) ve Google (Android) tarafından 2007 yılında yapılan değişikliklerden sonra Microsoft yeni bir yön almaya karar verip Windows Mobile yerine Windows Phone'u geliştirmiştir. iOS ve Android gibi diğer alternatiflere benzer şekilde, Windows Phone akıllı telefonlar için bir işletim sistemidir. Genellikle dokunmatik ekranlı cihazlarda kullanılır ve ağ oluşturma, sensörler ve kamera entegrasyonu gibi işlevler sunmaktadır (Grønli ve ark., 2014).

Android işletim sisteminde olduğu gibi, Windows Phone işletim sistemi de birden çok donanım platformu ve yonga seti için tasarlanmıştır. Böylece uygulama geliştiricileri uygulamalarını gerçek zamanlı olarak derleyebilmektedir. Microsoft, C# dilini ana geliştirme dili olarak kullanmayı seçmiştir. Uygulama, Android işletim sisteminin Dalvik sanal makinesine benzer şekilde Microsoft'un kendi sanal makinesi olan CLR'de derlenmiştir (Novac ve ark., 2017).

Windows Phone için geliştirme Visual Studio'da yapılmıştır. Windows Phone işletim sistemine program geliştirmek için kullanılabilecek iki dil vardır. Bunlar Visual Basic .NET ve C#'dır. Windows Phone için oluşturulan programlar, Silverlight uygulama paketi olan XAP dosyalarına paketlenir. Windows Phone, Silverlight ve XNA olmak üzere iki popüler programlama platformunu desteklemektedir. Geliştiricilere gelişmiş kullanıcı arabirimleri oluşturma olanağı sağlamaktadır (Grønli ve ark., 2014).

Windows Phone işletim sistemine yapılan ana eleştiri, iOS ve Android işletim sistemlerine kıyasla mağazasında uygulamaların çok daha az olmasıdır. Windows Phone Store, Microsoft tarafından Windows Phone platformu için geliştirilmiş bir dijital dağıtım platformudur. Microsoft, Windows Mağazası'nda 200.000'den, Windows Phone Mağazası'nda ise 385.000'den fazla uygulamanın olduğunu doğrulamıştır. Buna göre akıllı telefon uygulama platformları arasında, Windows Mağazası platformu, Android OS ve iOS OS platformlarının arkasında üçüncü sırada yer almaktadır (Novac ve ark., 2017). Statcounter.com üzerinde yapılan araştırmalar da bu sıralamayı desteklemektedir. Son zamanlarda mobil cihaz kullanımları, yaklaşık her bireyin kullanacağı kadar artmasına rağmen, Windows Phone platformu hala % 0.13 pazar payına sahiptir. Android, % 74.17 ile zirvede iken onu % 24.75 ile iOS takip etmektedir (Statcounter, 2020).

2.4. Bölüm Değerlendirmesi

Bu bölümde, son dönemde en çok kullanılan çapraz platform mobil uygulama geliştirme araçlarının uygulama çıktısı verebildiği işletim sistemleri hakkında bilgilere yer verilmiştir. Bu bilgiler doğrultusunda Android işletim sisteminin her anlamda iOS ve

Windows Phone işletim sistemlerinden önde olduğu görülmüştür. Android'in yakın takipçisi olan iOS'un ardından Windows Phone işletim sistemi gelmektedir. Son dönemde çıkan frameworklerin çoğu, Windows Phone'un rakiplerine oranla geride kalmasından dolayı uygulama çıktısı desteği vermemektedir.

3. ÇAPRAZ PLATFORM MANTIĞI

Bir mobil uygulama geliştirebilmek için geliştirilecek uygulamanın öncelikle hangi platformda çalışacağını belirlemek ve o platformun geliştirme kurallarına harfiyen uyulması gerekmektedir. Örneğin Android işletim sistemi üzerinde çalışacak bir uygulama geliştirebilmek için Java veya Kotlin gibi platformun belirlemiş olduğu dil ve yine Android Studio gibi bir IDE kullanılarak geliştirme işleminin gerçekleştirilmesi gerekmektedir.

Eğer geliştirilecek olan uygulamanın yalnızca bir işletim sisteminde çalışması isteniyorsa, böyle bir durumda yalnızca o işletim sisteminin geliştirme kurallarına hâkim olunması yeterli olacaktır. Fakat hem iOS hem de Android gibi birden fazla platform üzerinde geliştirme yapılacaksa bu durumda her iki platformun da geliştirme kurallarına hâkim olmak gerekmektedir.

Çapraz platform mobil uygulama geliştirme araçları kullanılarak uygulama çıktısı alınmak istenen tüm platformların geliştirme kurallarına hâkim olunması ve her biri için tek tek emek ve zaman harcanmasına gerek yoktur. Yalnızca seçilecek bir aracın ortamında yazılımı geliştirmek yeterli olacaktır.

Bu araçlar, yoğun grafik geçişleri olmayan küçük uygulamalarda iyi bir performans gösterse de, Facebook uygulaması gibi daha karmaşık uygulamalar için aynı kaliteye sahip değildir (Martinez, 2019).

Bir yazılımın mobil uygulama üzerinde çalışması için üç ayrı geliştirme yöntemi vardır. Bunlar Web uygulamaları, Native uygulama ve Hybrid uygulamadır. Bunların birbirlerine göre artıları ve eksileri aşağıdaki Tablo 3.1'de verilmiştir.

Tablo 3.1. Mobil uygulama geliştirme yöntemlerinin karşılaştırılması (El-Kassas ve ark., 2016)

Yöntem Adı	Artıları	Eksileri
Web Uygulamaları	1. Web teknolojilerini kullanarak öğrenmesi ve geliştirmesi kolaydır. 2. Bir kez geliştirilir ve mobil tarayıcılar kullanılarak farklı platformlarda erişilebilir.	1. Uygulama mağazasından indirilemez. 2. Mobil cihaz özelliklerine erişilemez.
Native Uygulamalar	1. Mobil cihaz özelliklerine tam erişimlidir. 2. Yüksek performanslıdır. 3. Native görünüm ve kullanıma sahiptir.	1. Cihaza indirip yüklemeden çalışmaz. 2. Öğrenmek ve geliştirmek daha zordur. 3. Farklı platformlar için ayrı ayrı geliştirilmelidir.
Hybrid Uygulamalar	1. Mobil cihaz özelliklerine erişebilir. 2. Web teknolojileri kullanılarak geliştirilen ve native bir uygulamada oluşturulan web sayfaları farklı platformlarda yeniden kullanılabilir.	1. Cihaza indirip yüklemeden çalışmaz. 2. Native uygulamalardan daha az performanslıdır.

Mobil uygulamaların geliştirilmesi, cihazların ve işletim sistemlerinin özelliklerinden yararlanmak için platformların ve araçların ortaya çıkmasına neden olmuştur. Öte yandan, mobil cihazlardaki işletim sistemlerinin çeşitliliği, bu sistemlerin uyumsuzluğu, daha uzun geliştirme süresi, daha yüksek maliyetlerin dezavantajı nedeniyle, birden fazla programlama dilini öğrenme ve çeşitli sistemler ve ara yüzler için aynı uygulamayı geliştirme ihtiyacını doğurmuştur. Bu sorunu kolaylaştırmak için yeni bir yöntem daha geliştirilmiştir. Böylece mobil uygulama geliştirme yöntemleri; Native uygulamalar ve Hybrid uygulamalar olmak üzere iki farklı kategoriye ayrılmıştır (Pinto ve Coutinho, 2018).

3.1. Native

Native uygulama geliştirme, görsel görünüm ve kullanıcı etkileşimlerinin ilgili platform yönergelerine tam olarak uygulanabilmesini sağlar. Aynı zamanda, uygulama cihazın direkt olarak sistemine hitap ettiğinden, geliştiriciler mobil cihazlarda en iyi performansı elde edebilmektedirler (Rieger ve Majchrzak, 2019).

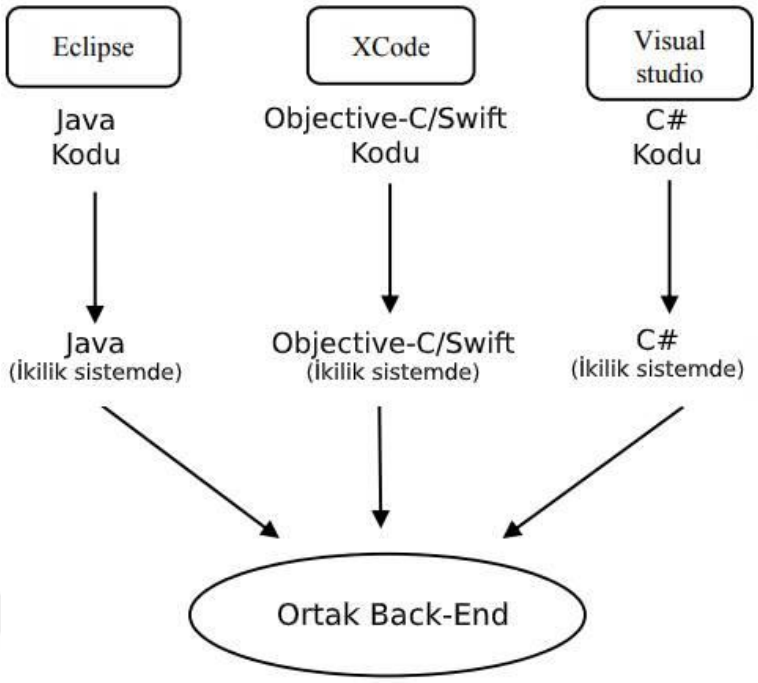
Native uygulamalar, uygulandıkları belirli işletim sistemi programlama dilleri kullanılarak geliştirilir. Farklı işletim sistemlerine sahip çeşitli cihazlar için mobil uygulamaların geliştirilmesi, ek bir maliyet ve bunların geliştirilmesinde daha fazla

zorluk anlamına gelmektedir. Çünkü farklı programlama dillerinde becerilerin artırılmasına ihtiyaç duyulmaktadır (Pinto ve Coutinho, 2018).

Native uygulamalar, belirli bir platformun tüm olası özelliklerine erişebilmektedir. Yine, bu esneklik ilgili platformun ara yüzleri üzerinde de sağlanmıştır ve geliştiriciler bu özellikleri istedikleri gibi kullanabilmektedirler. Ancak geliştiricilerin bu yeteneklerden faydalanırken ve uygulamanın harici içerik değişikliklerini yaparken azami bir dikkat göstermesi gerekmektedir (Rieger ve Majchrzak, 2019).

Native kod genellikle derlenmektedir ve bu da HTML ve JavaScript gibi yorumlanan dillerden daha hızlı olmasını sağlamaktadır. Daha düşük kaynak tüketimi (örn. CPU, GPU, pil) ile dengelenerek daha iyi bir son kullanıcı deneyimi sağlamaktadır. Native SDK'ları kullanarak, programcı eklentileri ve üçüncü taraf bağımlılıkları olmadan mobil cihazın tüm özelliklerine erişilebilmektedir. Ancak platformlar arası geçiş yapamamaktadır. Native mobil uygulamalar, SDK ve mobil platforma özgü programlama dili kullanılarak geliştirilen uygulamalardır. Bu mobil uygulamaların temel sınırlaması, uygulamayı sıfırdan yazmadan uygulamaları başka bir platforma aktaramamaktır (Pinto ve Coutinho, 2018).

Native mobil uygulama geliştirme yaklaşımı Şekil 3.1‘de ifade edilmiştir (Latif ve ark., 2016).



Şekil 3.1. Native mobil uygulama geliştirme yaklaşımı

3.2. Hybrid

Çapraz platform uygulamaları geliştirmek için, native olmayan bir dilde yazılmış uygulamadan native kod üreten çapraz derleyici mobil geliştirme araçları kullanılmaktadır. Çapraz derleyici araçlarını kullanarak, geliştiriciler farklı platform uygulamaları arasında kod paylaşarak hem geliştirme hem de bakım maliyetlerini azaltabilmekte ve aynı zamanda native mobil uygulamalar elde edebilmektedirler (Martinez, 2019). Böyle bir mobil uygulama geliştirme aracını kullanarak native kod elde etme işlemine hybrid mobil uygulama geliştirme adı verilmektedir.

Hybrid uygulamalar, web uygulamaları ile native uygulamanın bir kombinasyonudur. Bu türle birlikte, aynı anda native işlevler sağlayan native bir uygulamadaki kapsayıcı denetiminde görüntülenmektedir. Bu tür, cihaz özelliklerini aynı anda kullanmanın yanı sıra native bir uygulamada bulunan web sayfalarını da desteklemektedir (El-Kassas ve ark., 2016). HTML özelliklerini, cihaza özel özelliklere erişim sağlayan API'ler aracılığıyla genişletmektedir (Dalmasso ve ark., 2013b).

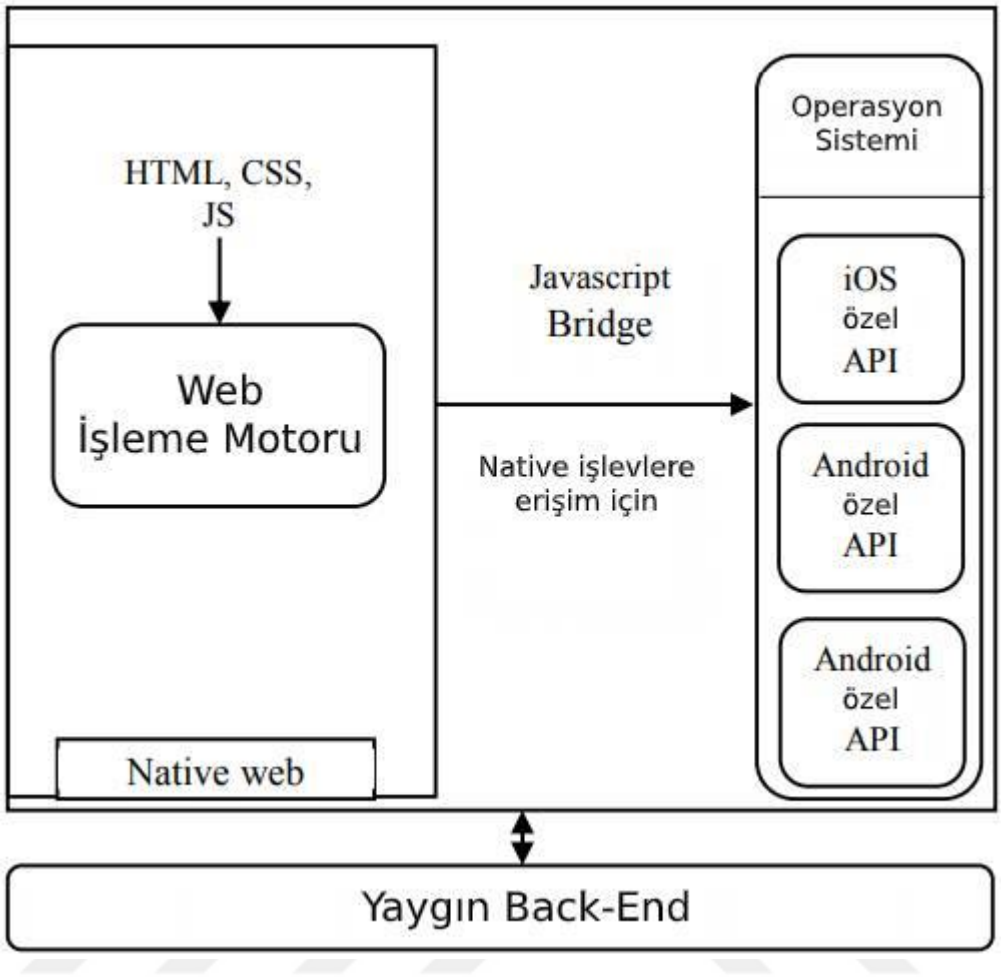
Bu uygulamalar HTML / JS / CSS gibi web teknolojileri kullanılarak geliştirilmekte ve WebView adı verilen bir platform için native sistem tarayıcısında paketlenmektedir. Bu yaklaşım, web ve cihaza özel uygulamanın bir arada kullanılması mantığına dayanmaktadır. Kapsayıcı native uygulama, tarayıcı ve cihaz API'leri arasında

köprü sağlayan bir HTML oluşturma motoru oluşturmak için işletim sistemi platformunun API'lerini kullanmaktadır (Leeladevi ve ark., 2015). Programcılar tüm kodu Javascript ile yazabilmektedir. Bu sayede şirketlerin farklı bir platforma uygulama gerekiyorsa programcıları yeniden eğitmek veya yeni programcılar kiralamak zorunda olmayacaktır (Ionescu, 2016).

Tek bir dile sahip tüm mobil cihazlar için mobil uygulamalar geliştirmek, bu araçlar sayesinde mümkün olabilmektedir. Bu sayede iOS çalıştıran cihazlar için Objective-C ve Swift, Android çalıştıran cihazlar için Java ve Kotlin, Windows çalıştıran cihazlar için C# gibi çeşitli programlama dillerini öğrenme zorunluluğu ortadan kalkmıştır. Bu geliştirme dilleri ise, native dilden yararlanan uygulamalar için tasarlanmıştır (Pinto ve Coutinho, 2018).

Web bölümü native oluşturma motorunu kullanmakta ve native kısım cihaz özelliklerinden yararlanmaktadır. Kullanıcı arayüzü web teknolojileri ile oluşturulduğundan, web bölümünde tekrar kullanılabilmektedir. Hybrid bir uygulama, uygulamanın native özelliklere ihtiyacı varsa ve uygulama mağazası aracılığıyla dağıtılması gerekiyorsa uygun bir yaklaşım olarak görülmektedir (Leeladevi ve ark., 2015).

Hybrid arabirimlerde, yürütme işlemi tarayıcı motorunda gerçekleştiğinden, native arabirimlere kıyasla performans daha düşüktür. Arayüzün native görünüm ve izlenime erişimi yoktur. Bunu elde etmek için belirli geliştirme kütüphanelerini kullanmak gerekli hale gelmektedir. Hybrid mobil uygulama geliştirme yaklaşımı Şekil 3.2‘de ifade edilmiştir (Latif ve ark., 2016).



Şekil 3.2. Hybrid mobil uygulama geliştirme yaklaşımı

3.3. Bölüm Değerlendirmesi

Çapraz platform mobil uygulama geliştirme alanında Native ve Hybrid kavramları birbirlerine oldukça benzer olduklarından dolayı aralarındaki fark sürekli karıştırılmaktadır. Bu bölümde bu iki kavramdan ve aralarındaki farklardan, birbirlerine göre artı ve eksi yönlerinden bahsedilmiştir.

4. ÇAPRAZ PLATFORM MOBİL UYGULAMA GELİŞTİRME ARAÇLARI

Çapraz platform mantığı ile mobil uygulama geliştirmek amacı ile tasarlanmış birçok araç vardır. Her birinin diğerine göre eksi ve artı yönleri bulunmaktadır. Geliştirme yöntemleri, dilleri, kullandıkları IDE'ler birbirlerine göre farklılık gösterebilmektedir. Bu bölümde bu bahsedilen mobil uygulama geliştirme araçları ele alınacaktır.

4.1. React Native

2009'da Facebook mühendis ekibi tarafından iOS ve Android gibi native platformlar için web tabanlı hybrid bir uygulama yaklaşımı kullanılarak oluşturulmuştur. Facebook, bu geliştirme stratejisi ile birlikte iOS, Android ve mobil web için aynı kodun çoğundan yararlanılmasına izin vermektedir. Ayrıca, yaklaşım, uygulamaların yeni sürümlerini yayınlamak zorunda kalmadan test ederek hızlı bir şekilde uygulanabilmesini sağlamıştır. (Martinez ve Lecomte, 2017).

React Native, 2013'te bir hackathon projesi olarak başlatılan, bir kez öğren, her yerde çalıştır mantığıyla çalışan bir uygulama geliştirme aracıdır. Özetle React Native, geliştiricilerin iOS ve Android platformları için native bileşenlere dönüştürülen JavaScript dilinde bileşenler yazmalarına izin vermektedir. Günümüzde; Facebook, Instagram ve Airbnb React Native kullanılarak geliştirilen platformlar arası mobil uygulamalara örnektir (Martinez ve Lecomte, 2017).

4.2. Xamarin

Geliştiriciler, Windows Phone için uygulamaları C# dilini kullanarak oluşturmaktadır. Bu kod, Visual Studio araç zinciri kullanılarak doğrudan Windows Phone uygulamalarına derlenebilmektedir. Xamarin, geliştiricilerin native Android veya iOS uygulamaları oluşturmak için aynı kaynak kodunu kullanmalarına izin vermektedir. Xamarin, koddaki Windows SDK API çağrılarını, temel Android ve iOS SDK'ların ilgili API çağrılarına çeviren uyumluluk kitaplıkları sağlamaktadır. Xamarin, bir uygulamayı platformlar arasında taşımak için gereken platforma özgü kod miktarını en aza indirgemek için geliştiricilere destek sağlamayı amaçlamaktadır. Bu amaca ulaşmak için, Xamarin tabanlı bir uygulamanın çekirdeğinin ana bileşenlerinden biri, taşınabilir sınıf kütüphaneleri olarak da adlandırılan platformlar arası uyumluluk kütüphaneleri (Xamarin'deki PCL)'dir. PCL, geliştiricilerin kodlarını iOS, Android ve Windows Phone

gibi birden çok platformda paylaşılabilen kütüphaneler üretmesine olanak tanıyan özel bir proje türüdür (Boushehrinejadmoradi ve ark., 2015).

Xamarin, doğrudan Windows Phone desteği sağlamasa da C# ve .NET uygulama kodu Windows Phone'da da çalışabilmektedir. Xamarin, Microsoft .NET uygulamalarının diğer platformlarda çalışmasına izin veren ücretsiz ve açık kaynaklı Mono .NET framework'üne dayanmaktadır. Xamarin.iOS (eski adıyla MonoTouch) ve Xamarin.Android (eski adıyla MonoDroid) olmak üzere iki mobil geliştirme ürünü sunmaktadır. Sınıf kütüphaneleri, sanal makine ve bir C# derleyicisinden oluşmaktadır. Uygulamalar Visual Studio veya Xamarin'in kendi geliştirme ortamı olan Xamarin Studio'da oluşturulmaktadır. Xamarin ile geliştirilen uygulamalarda, tüm kodlar C# ve .NET ile yazılmakta ve bu kod tüm platformlar arasında paylaşılmaktadır. Yerel işletim sisteminin görünümünü sağlamak için UI katmanı her platform için farklıdır ve yerel API ile geliştirilmektedir (Pazirandeh ve Vorobyeva, 2015).

4.3. Flutter

Flutter, mobil uygulama geliştiricilerine Android ve iOS platformları için uygulama geliştirme imkânı sunan çapraz platform bir mobil uygulama geliştirme aracıdır. Flutter, Google tarafından geliştirilmiş, açık kaynak kodlu, tüm uygulama geliştiricileri tarafından ücretsiz olarak kullanılabilen bir geliştirme aracıdır. Flutter, hafif katmanlı yapısıyla C++ ile yazılmış kod yapısına sahip olup, nesne yönelimli geliştirmeyi destekleyen ve Javascript'in rakibi olarak kabul edilen Dart programlama dili kullanılarak geliştirilebilmektedir (WMAracı). Ayrıca Flutter ile uygulama geliştirme işlemi Windows, Mac veya Linux ortamında yapılabilmektedir.

Doğrudan mobil cihazlardaki native işlemci mimarisi koduyla derlenmekte ve Google'a göre tüm iOS veya Android platformunun API ve hizmetlerine erişebilmektedirler (Vijayan, 2018). Bu sayede Flutter ile oluşturulan bir uygulama, her iki platformda da native yeteneklere sahip olabilmektedir. Aynı zamanda Flutter kütüphanesi içerisinde hazır olarak gelen iki adet tema bulunmaktadır. Uygulamanın görsel tasarımı ile vakit kaybetmek istemeyen geliştiriciler Android için Material Design ve iOS için ise Cupernito Style temaları ile başarılı tasarımlar ortaya çıkarabilmektedir. Ayrıca geliştiriciler hazır olarak sunulan bu temelara istekleri doğrultusunda yeni özellikler de ekleyebilmektedirler.

Flutter'i diğer mobil uygulama geliştirme araçlarından ayıran en büyük özelliği, arada bir Javascript köprüsü veya bir Webview gibi diğer araçlarda kullanılan bir etken

olmayışıdır. Bunların yerine Flutter kendi yüksek performanslı render(yorumlama) motorunu kullanarak uygulama ekranlarının daha hızlı yüklenmesini sağlamaktadır.

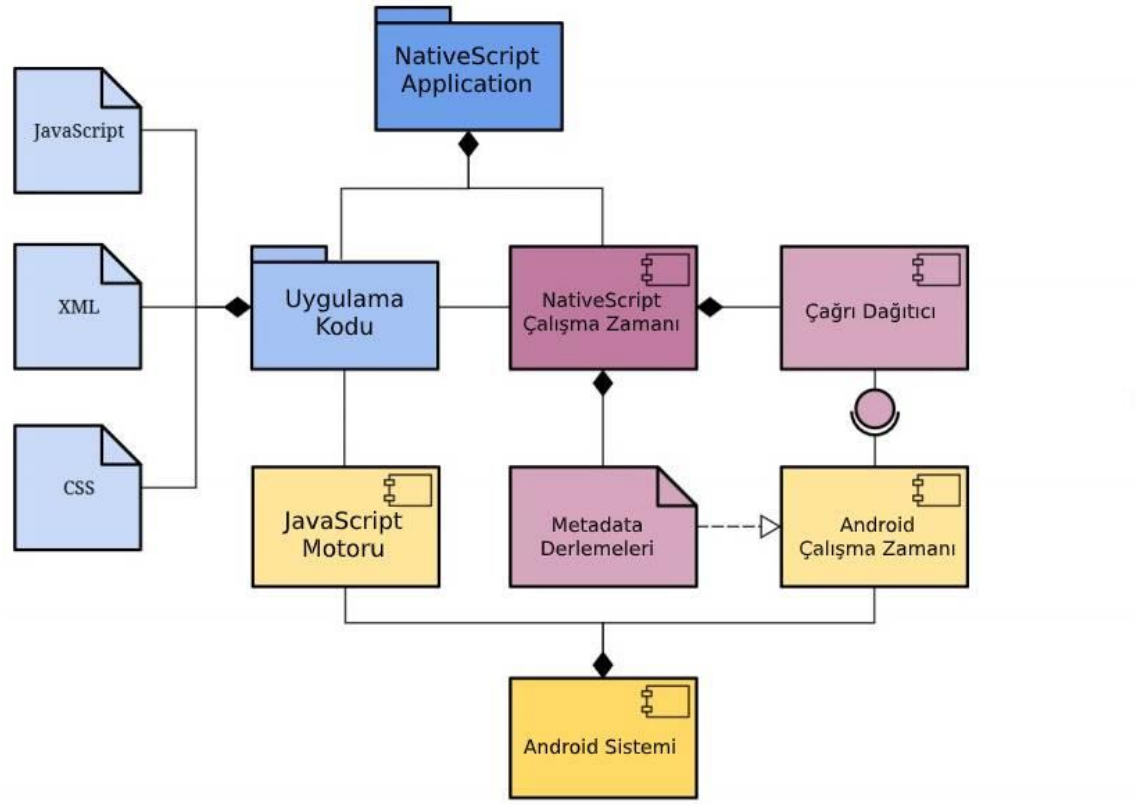
Google'a göre Flutter, hem uygulama geliştirmeye yeni başlayan hem de deneyimli mobil uygulama geliştiricileri tarafından kullanılmak üzere tasarlanmıştır. Popüler geliştirme araçlarıyla entegre olabilmektedir, böylece geliştiriciler her zaman kullandıkları düzenleyiciyi veya entegre geliştirme ortamını kullanmaya devam edebilirler. Flutter'ın en önemli özelliklerinden biri hızlı gelişimi desteklemesidir. Flutter, geliştiricilerin kodlarını denemelerine, yeni özellikler eklemelerine veya hataları düzeltmelerine ve bu değişikliklerin uygulama üzerindeki etkisini hemen görmelerine izin veren durum bilgisi olan Hot Reload özelliğini desteklemektedir. Flutter, geliştiricilerin native iOS ve Android mobil uygulamalarını daha hızlı bir şekilde oluşturmak için kullanabileceği bir dizi özelleştirilebilir kütüphane, entegre bir araç seti ve animasyon kitaplıkları da içermektedir (Vijayan, 2018).

Flutter'in ilgi çekici yanlarından biri ise Hummingbird özelliğidir. Bu özellik sayesinde mobil uygulama için yazılan proje üzerinde ekstra bir işlem yapmaya gerek duymadan uygulamanın web ortamında localhost adresi ve 8080 portu aracılığı ile görüntülenmesi sağlanabilmektedir.

Android ortamında Jelly Bean 16. versiyon 4.1.x ve daha üst SDK sürümlerine ve iOS tarafında ise iOS 8 ve daha üst SDK sürümlerine uygulama geliştirmesi yapılabilmektedir. Android ortamında donanım olarak ARM işlemcisini, iOS ortamında ise 64-bit çalışan işlemcileri desteklemesi ile iPhone'un 5s ve üzerindeki cihazlarına uygun uygulamalar geliştirme olanağı sunmaktadır (Öner, 2019).

4.4. Nativescript

Telerik tarafından geliştirilen NativeScript, neredeyse tamamen JavaScript ile platformlar arası mobil uygulamalar oluşturmak için kullanılan açık kaynak geliştirme sistemidir. Arayüz tasarımlarını geliştirmeyi basitleştirmek için isteğe bağlı CSS ve XML kullanımını desteklemektedir. Apple'ın iOS ve Google'ın Android işletim sistemleri üzerinde uyumludur. NativeScript, Android'de V8 motorunu (Google Chrome ve Node.js tarafından kullanılan) ve iOS cihazlarında ise Apple'ın JavaScriptCore motorunu kullanmaktadır. NativeScript için yazılan kodlar bir web görüntüleyicisi ile değil, JavaScript'in doğrudan cihaz platform API'lerine erişmesi vasıtasıyla çalışmaktadır (Anderson, 2016).



Şekil 4.1. NativeScript uygulamalarının mimarisini ve Android sistemiyle entegrasyonunu gösteren UML bileşen diyagramı (Mehlhorn, 2017)

Şekil 4.1'de belirtildiği üzere uygulama kodu, sistemin JavaScript motorunun içinde yürütülmektedir. Uygulamadan yapılan tüm çağrılar, ilişkili native tamamlayıcıya yönlendirilmektedir. Karşılığında sistemden kaynaklanan tüm değişiklikler JavaScript nesnelerine yansıtılmaktadır. Bu mekanizma ile Nativescript uygulamaları, native bir uygulamanın oluşturulma biçimine yakın bir mantıkla hazırlanmış olmaktadır (Mehlhorn, 2017).

Nativescript ile yazılan tüm kodların son çıktısının JavaScript olması gerektiğinden, uygulama veya modüllerden herhangi birisi TypeScript, CoffeeScript veya JavaScript'e aktarılabilen başka bir dilde geliştirilebilmektedir (Anderson, 2016).

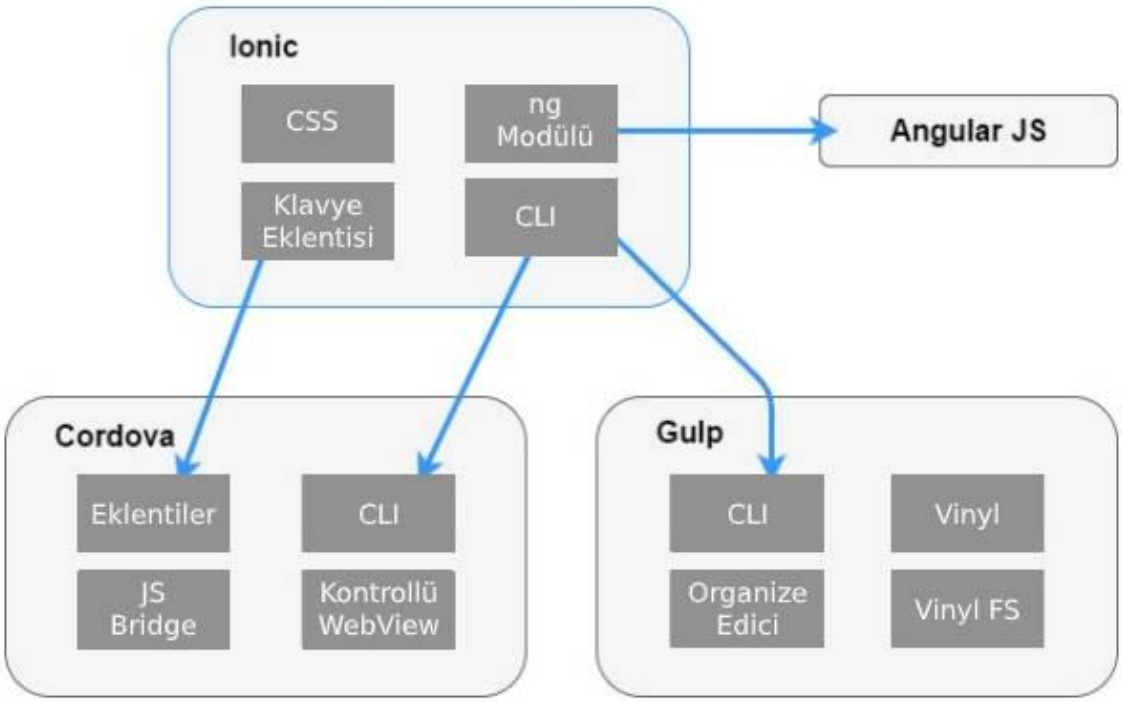
4.5. Ionic Framework

Ionic Framework ilk olarak Kasım 2013'te piyasaya sürülmüştür. Ionic Framework'ün temel özelliği, web tabanlı uygulama geliştirme için kullanıcı arabirimi bileşenlerini sağlamaktır. Örneğin, sekme çubuğu birçok mobil uygulamada bulunan yaygın bir kullanıcı arayüzü bileşenidir. Ancak bu bileşen native bir HTML öğesi olarak mevcut değildir. Ionic Framework, HTML'i bir kitaplık oluşturacak şekilde genişletir. Bu

bileşenler, HTML, CSS ve JavaScript'in kombinasyonuyla oluşturulmuştur ve her biri yeniden oluşturduğu native denetimlere benzer çalışmaktadır. Ionic Framework'ün ana odağı, kullanıcı arayüzü katmanında native bileşenler sağlamak için Angular ve Cordova ile entegrasyonu sağlamaktır (Griffith, 2017).

Ionic Creator, sürükle ve bırak yöntemini kullanarak Ionic mobil uygulamaları oluşturmak için geliştirilmiş çevrimiçi bir arayüz oluşturucudur. Ionic Creator kullanılarak tasarlanan görünümlerin kodu bir ZIP dosyası olarak dışa aktarılabilir ve indirilebilir. Bu dosya, mevcut Ionic uygulamasıyla da entegre edilebilen HTML, CSS ve JavaScript öğelerini içermektedir (Khanna ve ark., 2017).

Ionic ayrıca Syntactically Awesome Style Sheets (SASS) CSS uzantısını da desteklemektedir. Ionic, jQuery Mobile'ın gelişen teknolojinin hızına yetişememesinden dolayı, jqLite'ı kullanmaktadır. Böylece tüm jQuery'yi yüklemeden, daha az bellek kullanmaktadır (Sánchez Blanco, 2016). Ionic bir HTML5 uygulaması olduğundan, native bir uygulama olarak çalışmak için Cordova veya PhoneGap gibi native bir sarmalayıcıya ihtiyaç duymaktadır (Ionic, 2020).



Şekil 4.2. Ionic mimarisi (Sánchez Blanco, 2016)

Şekil 4.2'de Ionic mimarisi verilmiştir. Hybrid mobil uygulamalar oluşturmak için AngularJS (güçlü bir Javascript kütüphanesi), Cordova ve Javascript çalıştırıcısını

birbirine bağlamaktadır. Şekilde verilen Gulp, JavaScript kodunu küçültme ve bazı geliştirme araçlarını otomatikleştirme işini yapan sistemdir.

4.6. Unity 3D

Unity3D, en ünlü sanal gerçeklik araçlarından biri, aynı zamanda platformlar arası oyun geliştirme yazılımıdır. Mac OSX'in yanı sıra Unity3D, WindowsXP, Vista ve 7'yi tamamen destekleyebilmektedir. Unity3D üç kodlama dilini desteklemektedir. Bunlar; JavaScript, C# ve Python Boo'dur. Üçü de aynı derecede hızlı ve birlikte çalışabilmekte ve veri tabanlarını, düzenli ifadeleri, XML, dosya erişimini ve ağları destekleyen temel .NET kitaplıklarını kullanabilmektedir. Unity3D'de komut dosyaları native koda derlenmekte ve neredeyse C++ kadar hızlı çalışmaktadır. Komut dosyası sayesinde hızlı yineleme süreleri ve kullanım kolaylığı elde edilebilmektedir. Oyun geliştirme projelerindeki bu üç dil de karışık olarak kullanılabilmektedir. Unity3d ayrıca Windows platformu üzerinde C#, VB.net, VB6, Delphi ve diğer programlama dillerini de desteklemektedir (Wang ve ark., 2010).

Unity3D'nin bir oyunu kolayca programlamada özel avantajları vardır. Örneğin, platformla ilgili işlemler dahili olarak kapsüllenir. Karmaşık oyun nesnesi ilişkileri farklı görsel görünümlerle yönetilir ve bir oyunu programlamak için JavaScript, C# veya Boo komut dosyası dilleri uygulanır. Bir komut dosyası programı otomatik olarak bir .NET DLL dosyasına derlenir. Bu nedenle üç komut dosyası dili özünde aynı performansa sahiptir. Yürütme hızları geleneksel JavaScript'ten 20 kat daha hızlıdır. Bu script dilleri de iyi çapraz platform yeteneğine sahiptir. Bu, geliştiricilerin oyunları Windows, Mac, Xbox 360, PlayStation 3, Wii, iPad, iPhone ve Android gibi farklı platformlara dağıtabilecekleri anlamına gelmektedir. Ayrıca, oyunlar bir eklenti yükleyerek Web üzerinde çalışabilir.

4.7. Cocos2D

Cocos2D-x, iOS için Cocos2D motorunun C++ çapraz platform bağlantı noktasıdır. MIT lisansı altında açık kaynaklı bir oyun motorudur. Bununla birlikte, geliştiriciler sadece oyunlar değil, uygulamalar ve animasyonlu arka planlar gibi diğer çapraz platform arayüz etkileşimli programları da geliştirebilmektedirler. Cocos2D-x, geliştiricilerin prototipleme işlemlerinden yüksek performanslı oyunlara / uygulamalara kadar C++, Lua ve Javascript dillerini kullanarak zaman ve emek tasarrufu sağlayan platformlar arası oyunlar / uygulamalar geliştirmelerini sağlamaktadır. Cocos2D-X,

Batı'da Unity3D kadar popüler veya iyi bilinmemekle birlikte, Asya'da çok popülerdir ve Cocos2D'ye dayanmaktadır (Hussain ve ark., 2014).

Cocos2d-x, platformlar arası bir oyun motorudur. Bir oyun motoru, tüm oyunların ihtiyaç duyduğu ortak işlevleri sağlayan bir yazılım parçasıdır. Bu oyun motoru, birlikte kullanıldığında geliştirme süresini hızlandıracak ve genellikle native motorlardan daha iyi performans gösterecek birçok bileşen içermektedir (Cocos, 2020).

4.8. Titanium

Titanium, web teknolojilerini kullanarak platformlar arası native mobil uygulamalar geliştirmek için ticari olarak desteklenen, açık kaynaklı bir platformdur. Kaynak kodu Apache 2 lisansı altında yayınlanmıştır. Kuruluş yeri Kaliforniya olan Appcelerator Inc, Aralık 2008'de platformu tanıtmıştır. Appcelerator, BlackBerry için de çalışan bir Titanium Mobile sürümü geliştirmiştir (Allen ve ark., 2010b).

Native kod modüllerine yönelik Javascript tabanlı bir arayüz,,iPhone veya Android'in yerel dosya sistemini kullanarak kullanıcı tercihlerini depolayabilmekte, veri dosyalarını kaydedebilmekte veya bir çerezin mobil sürümünü uygulayabilmektedir (Smutný, 2012).

Titanium, hedef platforma temel yapı oluşturmak için gerekli araçları, derleyicileri ve API'leri sağlayan bir SDK ve Titanium Developer adı verilen Titanium tabanlı projeleri yönetmek için görsel bir ortamdan oluşmaktadır. Titanium, Mac, Linux ve Windows için kullanılabilir. iPhone (veya iPad)'e uygulama geliştirmek için iPhone SDK'yı kullanarak Mac üzerinde çalıştırmak gerekmektedir. Android'e uygulama geliştirmek için Android SDK gerekmekte ve Mac, Windows veya Linux kullanılabilmektedir. Titanium API, gezinme çubukları, menüler, iletişim kutuları ve bildirimler dahil native arayüz bileşenlerine ve dosya sistemi, ses, ağ ve native veri tabanı dahil native aygıt işlevlerine erişim sağlamak için platformdan bağımsız bir API sunmaktadır (Allen ve ark., 2010b).

Titanium, kullanıcı arabirimini oluşturmak için HTML kullanmaz. Bunun yerine, kullanıcı arayüzü tamamen programlı olarak uygulanır. Geliştiriciler, Titanium API'sını kullanarak arayüzü oluşturmak ve mantık ve verileri uygulamak için JavaScript kullanmaktadır. Kod daha sonra Titanium'un motoruyla paketlenir. Çalışma zamanında, bu motor JavaScript kodunu yorumlar ve kullanıcı arayüzünü oluşturur. Bununla birlikte, görünüm öğeleri, tipik platform görünümüne benzemektedir. Bu sayede ise kullanıcı arayüzü native öğelerden oluşmaktadır (El-Kassas ve ark., 2017).

4.9. Phonegap

PhoneGap, iPhone, Android, BlackBerry, Palm webOS ve Symbian WRT (Nokia) için HTML, CSS ve Javascript kullanarak native mobil uygulamalar oluşturmak için açık kaynaklı bir araçtır. PhoneGap, bir mobil web uygulamasını native bir uygulamaya dönüştürmek için geliştirilmiştir. Web geliştiricileri için kullanımı kolaydır. iPhone ve Android gibi, özellikle HTML5'in gelişmiş JavaScript ve CSS'sine sahip WebKit tarayıcısını içeren platformlarda daha iyi çalışmaktadır. PhoneGap, merkezi Vancouver BC'de bulunan bir yazılım danışmanlığının projesidir. Framework, 2008 yılında başlamıştır. Bir MIT lisansı altında ücretsiz olarak kullanılabilmektedir (Allen ve ark., 2010b).

PhoneGap açık kaynak framework'ü, yalnızca HTML, JavaScript ve CSS kullanarak native mobil uygulamalar oluşturmak için iyi bir araç kutusu sağlar. Esas olarak esnekliği, basit mimarisi ve kullanım kolaylığı nedeniyle kullanıcılar arasında oldukça popülerdir (Dalmasso ve ark., 2013b).

PhoneGap, web uygulamalarını native uygulamalara yüklemeye ve cihaz işlevlerine erişmelerine olanak tanıyan bir sarmalayıcıdır. PhoneGap, tüm mobil platformlarda uygulama geliştirmek için bir geliştirme aracı şartı aramaz. Alternatif olarak, geliştiricilerin uygulamalarını bulutta derlemelerini sağlayan PhoneGapBuild adlı bir hizmet sunar. Bu hizmetle native uygulamalar oluşturmak için mobil platform SDK'larının kurulmasına gerek yoktur. PhoneGap; Android, iOS, Blackberry ve Windows Phone mobil platformları desteklemektedir. En eksiksiz desteği Android ve iOS için sağlamaktadır (El-Kassas ve ark., 2017).

Native kullanıcı arayüzü bileşenleri, tasarım modelleri ve geliştirme araçları için destek eksikliği vardır. Ancak, geliştiriciler, uygulamalar üzerinde daha iyi bir kullanıcı arayüzü oluşturmak için PhoneGap'i JQuery veya Sencha gibi başka bir araçla birleştirerek daha iyi performans alabilmektedirler (Dalmasso ve ark., 2013b).

Native uygulamayı web teknolojileri ile oluşturduğundan native öğelere ve uygulama mağazalarına erişim sağlayan bir HTML5 uygulama platformuna sahiptir. Uygulamayı oluşturma işlemi, Objective-C veya Corona uygulamaları yerine web teknolojileri üzerinden yapılmaktadır (Smutný, 2012).

PhoneGap, mobil web uygulamasıyla yapılabilecek her şey için uygundur. Kullanıcı arayüzü için tarayıcıdan yararlanan tüm platformlar arası framework'ler gibi, yoğun matematik hesaplamaları veya 3 boyutlu animasyonlar gerektiren uygulamalar için uygun değildir. Native verileri kullanarak senkronize bir şekilde çevrimdışı çalışması

gereken veri odaklı uygulamalar için de uygun değildir. PhoneGap belirli bir veritabanı desteği sağlamamaktadır (Allen ve ark., 2010b).

PhoneGap, mobil cihazın coğrafi konum, kişiler, titreşim, ivmeölçer, kamera, ses çalma, cihaz bilgileri, arama gibi donanımsal özelliklerine erişim sağlayabilmekte ve yönetebilmektedir.

4.10. Sencha Touch

HTML5, CSS3 ve Javascript'ten yerel depolama, video ve ses bileşenleri gibi en yüksek güç ve esneklik düzeyinden yararlanmak için özel olarak oluşturulmuş açık kaynaklı bir geliştirme aracıdır. Ayrıca, araç Ajax, Json ve Yql gibi çeşitli kaynaklardan veri entegrasyonu da sunmaktadır (Smutný, 2012).

Sencha Touch, JavaScript ile yazılmış açık kaynaklı bir platformlar arası mobil uygulama geliştirme aracıdır. Android, iOS, BlackBerry, Kindle, Windows Phone ve Tizen gibi tüm platformlarda geliştirmeyi destekler. Mobil web uygulamaları için arayüzleri geliştirmek, böylece uygulamaları mobil cihazlarda native uygulamalara benzetmek için kullanılır. Sencha Touch, popüler JavaScript kütüphanesi Ext JS, jQTouch ve Raphael birleştirildikten sonra oluşturulan bir Sencha ürünüdür. MVC tarzı bir mimari, Sencha native paketleyicisi ve ayrıca PhoneGap ile kullanılabilecek çeşitli arayüz temaları sunar (Nayyar, 2016).

Sencha Touch, etkileşimli web uygulamaları oluşturmak için Sencha tarafından geliştirilen bir JavaScript kütüphanesi olan Ext JS'yi kullanır. Model-View-Controller tasarım desenine dayanır ve aynı zamanda native bileşen görünümlü ve yüksek performansa odaklanmış bir frameworktür. iOS, Android, Windows Phone, Tizen ve Blackberry platformlarını desteklemektedir (Sánchez Blanco, 2016).

Web uygulamalarının hem iPhone'da hem de Android'de tutarlı bir görünüme ve izlenime sahip olmasını sağlar. Herhangi bir işletim sistemine benzemeyen bir arayüz kütüphanesine sahiptir. Sencha Touch'ın iyi bir topluluk desteği vardır. Farklı boyutlardaki cihazlarla (telefonlar ve tabletler) uyumlu farklı uygulamalar geliştirme imkânı sunmaktadır (Pazirandeh ve Vorobyeva, 2015).

4.11. Appcelerator Titanium

2006 yılında piyasaya sürülmüş, ancak Android ve iOS desteği 2009 yılında eklenmiştir. Titanium uygulamaları HTML, JavaScript ve CSS kullanılarak PHP, Ruby ve Python desteği ile yazılmıştır. Titanium geliştiricinin native özelliklere, native arayüz

modüllerine ve bazı isteğe bağlı modüllere erişmesine olanak tanır. Titanium native kütüphaneleri, JavaScript kodunu bayt kodunda derler. Derleyici daha sonra hedef platform için paket oluşturur. Geliştirilen kodun %70-80'i diğer platformlara doğrudan uyarlanabilmektedir. Çıktı uygulaması çoğunlukla native kod içerir ve oluşturma işlemi native olarak yürütülür (Pazirandeh ve Vorobyeva, 2015).

Appcelerator Titaniom, iOS, Android ve BlackBerry işletim sistemlerinde native uygulamalar oluşturmak için açık ve genişletilebilir bir geliştirme ortamıdır. Aynı zamanda 5000'den fazla cihazda uygulama çalıştırabilen açık kaynaklı bir framework'tür. Appcelerator Studio adında Eclipse tabanlı geliştirme ortamına sahiptir. Bir MVC framework'ü ve kullanıma hazır bir mobil Cloud Services içerir (Titanium).

Menüler, gezinme çubukları veya iletişim kutuları ve native aygıt özellikleri gibi bileşenlere erişmek için native kullanıcı arabirimi ve platform desteği sağlamaktadır. Titanium, JavaScript kodlarını native kodlara bağlar, bayt koduna derlemektedir. Daha sonra platform SDK'sı, istenen hedef platform için paketi native simülatörde veya test için cihazda veya dağıtım için çalıştırılabilir uygulama olarak oluşturmaktadır. Çıktı uygulaması çoğunlukla native kod içermekte ve oluşturma işlemi native olarak yürütülmektedir. Ayrıca, çıktı uygulaması bir JavaScript RunTime yorumlayıcı ve bir Webkit oluşturma motoru içermektedir (Ribeiro ve da Silva, 2012).

4.12. Apache Cordova

Cordova projelerinde iki geliştirme yolu bulunmaktadır. Bunlar; çapraz platform iş akışı ve platform merkezli iş akışıdır. Birincisi, en popüler yaklaşımdır ve saf hybrid uygulamaları kapsamaktadır. İkinci yaklaşım, karma uygulamalar için gerekli iş akışını açıklar. Bu tür karma bir hybrid uygulamanın tipik yapısı, özel native bileşenlerin ve bir veya daha fazla karıştırılmış WebView'lerin kombinasyonunu içerir. Native bileşenler, uygulamanın araç çubukları, gezinme çubukları ve sayfa gibi daha az güncelleme gerektiren statik parçalarıdır. Diğer yandan, sık değişen içerik WebView'lerde gösterilmektedir. Karışık karma uygulamaya Apple'ın Apple Store uygulaması örnek olarak gösterilebilmektedir (Smeets ve Aerts, 2016).

4.13. Rhodes

Rhodes, Rhomobile tarafından Cupertino, Kaliforniya'da bir girişim topluluğu tarafından geliştirilen çapraz platform bir akıllı mobil uygulama geliştirme framework'üdür. Aralık 2008'de piyasaya sürülmüştür. Rhodes, iPhone, BlackBerry,

Android, Windows Mobile ve Symbian dahil olmak üzere çoğu akıllı telefon için kullanılabilir. Rhodes, geliştiricilerin HTML, CSS, JavaScript ve Ruby programlama dillerini kullanarak platformlar arası akıllı mobil uygulamaları oluşturmalarına olanak tanımaktadır. Native mobil uygulamalar geliştirmek için web uygulama geliştirme yöntemlerinden yararlanır. Rhodes, web geliştirmek için arka planı olan ve her bir mobil cihaz platformunda platform SDK'larını ve native dilleri öğrenmek zorunda kalmadan mobil uygulamalar oluşturmak isteyen geliştiricilere yöneliktir. Rhomobile araçları Mac, Windows ve Linux'ta kullanılabilir; ancak, belirli aygıtlarda derleme yapabilmek için aygıtın SDK'sının yüklenmesi gerekmektedir. BlackBerry ve Windows Mobile cihazları için Windows işletim sistemi, iPhone cihazları için Mac, Android ve Symbian cihazları Java üzerinde çalışır ve platformlar arasıdır (Allen ve ark., 2010b).

Uygulamalar çoğunlukla Ruby dilinde bir MVC mimarisi kullanılarak geliştirilmektedir. Oluşturulan uygulamalar Ruby 1.9 bayt kodunda derlenmekte ve bu daha sonra Ruby Sanal Makinesi tarafından hedef platform için yorumlanmaktadır. Mobil cihazın native API'sine erişim Rhodes API tarafından sağlanır. JQTouch, dokunmatik ekran arayüzlerini desteklemektedir (Pazirandeh ve Vorobyeva, 2015).

Rhodes öncelikli olarak kurumsal uygulamalar geliştirmeyi hedeflemektedir. Framework, haritalama gibi yaygın telefon kullanıcı arabirimleri de dahil olmak üzere standart arayüz bileşenleri içeren uygulamalar oluşturmayı kolaylaştırır. Hızlı etkileşimli oyunlar ve zengin etkileşimli grafik arabirimleri veya platforma özgü native arayüz denetimleri talep eden diğer tüketici uygulamaları için uygun değildir. Rhodes, MIT Lisansı kapsamında lisanslanmış, ticari olarak desteklenen açık kaynaklı bir üründür. Ticari düzeyde desteğe ihtiyaç duyan şirketler Rhomobile'den kurumsal lisans satın alabilmektedirler. Rhodes açık kaynak olduğundan, kodların her kısmına istenildiği gibi müdahale edilebilir. Gerekirse genişletilebilir, iyileştirmelere ve düzeltmelere katkıda bulunulabilir veya kişiye özel Rhodes sürümü geliştirilebilir (Allen ve ark., 2010b).

4.14. Onsen UI

OnsenUI, Ionic gibi AngularJS yazımına dayanmaktadır ve platformlardan bağımsız olarak bir dizi arayüz bileşenini desteklemektedir. Native bir uygulama ile benzer düzeyde arayüzler sunar. OnsenUI, resmi sayfasında temaya göre şablonlar ve renkler oluşturarak kaynak kodlarının indirilmesine yardımcı olmaktadır. Temaya göre değişse de seçilen temanın kaynak kodu, framwork'ün toplam kapasitesinin yaklaşık 270 kb'sini almaktadır. Bu nedenle sayfa yükleme süresi azalmaktadır. Buna ek olarak iOS

7+ ve Android 4.0.2+ sürümünü desteklemektedir. Böylece diğer araçlarla karşılaştırıldığında nispeten yüksek bir sınırlamaya sahiptir (Sohn ve ark., 2015). Kullanımı kolay mobil uygulamaların oluşturulması için kullanılır (Deshmukh ve Rajput, 2016).

Onsen UI, Angular JS ile çalışmakta ve performans ve kullanım kolaylığına odaklanan bir PhoneGap / Cordova uygulaması oluşturmak için jQuery'i desteklemektedir. Android, iOS, Firefox OS, Windows Phone 8.1 ve masaüstü tarayıcılarını destekler. Monaca adlı bir HTML5 geliştirme aracı ile geliştirilmektedir. Topcoat CSS kütüphanesi etrafında inşa edilmiştir. Android ve iOS için native görünümlü temaları desteklemektedir (Sánchez Blanco, 2016).

4.15. Framework 7

Ionic ile aynı kategorideki bir başka ücretsiz ve açık kaynaklı geliştirme aracıdır. Hızlı prototipleme için de kullanılır. iOS ve Android uygulamalarını yalnızca HTML, CSS ve Javascript kullanarak oluşturmayı sağlar (Shtika, 2017).

Framework 7, iOS ve Android'e odaklanmış ücretsiz ve açık kaynaklı bir geliştirme aracıdır. iOS veya Android (Google Material design ile) uygulamaları oluşturmak için arayüz bileşenleri kümesi sağlar (Sánchez Blanco, 2016).

4.16. Kony

Kony, tek bir kod tabanından uygulama oluşturmaya yardımcı olmak için çeşitli araçlar sunmaktadır. Studio platformu, JavaScript vasıtasıyla sürükleyip bırakma gibi yeniden kullanılabilir widgetları ve mevcut veya harici kütüphaneleri içe aktarma yeteneği gibi basit özellikleri uygulamaktadır (Yosef, 2017).

Kony Visualizer, mobil uygulamaların hızlı geliştirilmesi ve devreye alınması için entegre bir geliştirme ortamıdır. Kony Visualizer IDE, mobil uygulamaları tasarlamak ve geliştirmek için öğrenmesi ve kullanması kolay bir görsel geliştirme ortamı sunmaktadır. Kony Visualizer ayrıca, cihazın kamerasına erişmek, ortak şifreleme görevleri gerçekleştirebilmek ve uygulamanın birden çok yerel ayarını kontrol edebilmek gibi görevleri yerine getirmek için kullanılabilecek bir grup API sağlamaktadır (Kony, 2020).

4.17. Jasonette

Jasonette, native uygulamalar geliştirmenin farklı bir yoludur. Cihazda çalışacak bir uygulama programlamak yerine, bir sunucuda barındırılan bir JSON dosyası yazmak prensibine dayalıdır. Jasonette uygulamaları, uygulama her açıldığında kendilerini istek üzerine oluşturmak için kullanılmaktadır (Jasonette, 2020).

Jasonette, yalnızca Android ve iOS işletim sistemleri için uygulama çıktısı alınabilmesini sağlamaktadır. Buna istinaden, framework, bir isim, açıklama veya ikon tanımlamak için meta verilerin saklanabileceği uygulama sağlar. Meta veriler ayrıca uygulamanın JSON kodunun depolandığı dosyayı gösteren bir URL içerir. Jasonette'in yapılarının, içeriklerinin ve işlevlerinin uygulanması için sadece JSON tanımlama diline ihtiyaç vardır. Bu sayede yeni bir programlama dili öğrenmeye gerek kalmamaktadır (Schmitt ve ark., 2018).

4.18. iFactr

iFactr, hem eski hem de modern Windows, Windows CF, iOS ve Android'i destekleyen mobil platformu sağlamaktadır. iFactr, genellikle kurumsal uygulamalarda tercih edilmekte ve barkod tarama, coğrafi konum, fotoğraflar ve imza yakalama gibi özellikleri desteklemektedir (iFactr, 2020).

iFactr hızlı bir şekilde mobil uygulamalar geliştirmek için planlanmıştır. iFactr platformu, native uygulamalarını iOS ve Android'de derlemek için Xamarin'i kullanmaktadır (Yosef, 2017). Altyapı olarak .Net'i kullanan iFactr, bir veya iki günde öğrenilecek kolaylıkta bir geliştirme dili sunmaktadır.

4.19. FeedHenry

FeedHenry, temel teknoloji olarak HTML5 kullanan, platformlar arası uygulama geliştirme hizmeti sunan ilk şirketlerden biridir. 2007 yılında kurulan FeedHenry, bir web uygulaması frameworkü ile başlamış ve mobil kullanımlar için de uyarlanmıştır. İstemcideki bu HTML5 stratejisine ek olarak, FeedHenry en başından itibaren sunucu tarafı JavaScript iş mantığı geliştirmesini sağlayan bir bulut barındırma platformu sunmuştur. Başlangıçta bu sistem Mozilla'nın Java tabanlı Rhino motoru kullanılarak oluşturulmuş, ancak sonrasında Node kullanmaktadır. FeedHenry sistemi, yalnızca uygulama geliştirilmesine değil, sistem yönetimi hakkında da destekler sunmaktadır. Bu, sayede uygulamalar daha hızlı bir şekilde oluşturulup dağıtılabilmektedir (Rodger, 2011).

4.20. Qt

Qt, 1990'ların başından bu yana az gelişmiş olan, iyi belgelenmiş çapraz platform uygulama framework'üdür. Qt, MacOSX, Linux ve Windows dahil olmak üzere çoğu ticari yazılım platformunda çalışan zengin bir araç setini destekler. Buna ek olarak, Qt cep telefonları, PDA'lar, GPS alıcıları ve avuç içi medya oynatıcılar dahil olmak üzere çeşitli yerleşik cihazlarda ve uygulamalarda kullanılmaktadır (Mikkonen ve ark., 2009).

QT SDK Framework, mobil cihazlar için Symbian, Maemo ve Meego platformlarını geliştirmek üzere geliştirilen bir platformlar arası framework'tür (Akinkuolie ve ark., 2011).

4.21. Corona

Corona SDK, platformlar arası oyunlar, uygulamalar ve e-kitaplar oluşturmaya olanak tanıyan bir mobil uygulama geliştirme platformudur. Bu platform, öğrenmesi kolay bir programlama dili olan Lua adlı bir komut dosyası dilini kullanmaktadır. iOS, Android, Kindle ve Nook'ta uygulama çıktıları alıp yayınlama olanağı sunmaktadır (Williams, 2013).

Gerçek zamanlı cihaz testi ile geliştirilmekte olan mobil uygulama yalnızca bir kez oluşturup dağıtıldıktan sonra yerel ağa bağlı cihazlar üzerinde anında güncellemesi görülebilmektedir. Geliştirme işleminde kullanılan makineye kurulan Corona Simulator ile, test için derlemek veya dağıtmak zorunda kalmadan uygulama önizlenebilir. Temel framework'e ek olarak, Corona belirli işlevler ekleyen ve uygulamanın gelişimini hızlandırmaya yardımcı olan eklentiler de içermektedir (Corona, 2020).

4.22. Bölüm Değerlendirmesi

Bu bölümde, bugüne kadar geliştirilmiş ve kullanılmış olan bazı çapraz platform mobil uygulama geliştirme araçları hakkında bilgiler verilmiştir. Bu bilgiler doğrultusunda birbirleri arasında teorik bir ön karşılaştırma temeli oluşturulması hedeflenmiştir.

5. MATERYAL VE YÖNTEM

Bu bölümde, önce son yıllarda en çok kullanılan çapraz platform mobil uygulama geliştirme araçları tespit edilecek ardından bu araçların karşılaştırılması için her framework üzerinde ayrı ayrı geliştirilen uygulamalar hakkında bilgi verilecektir.

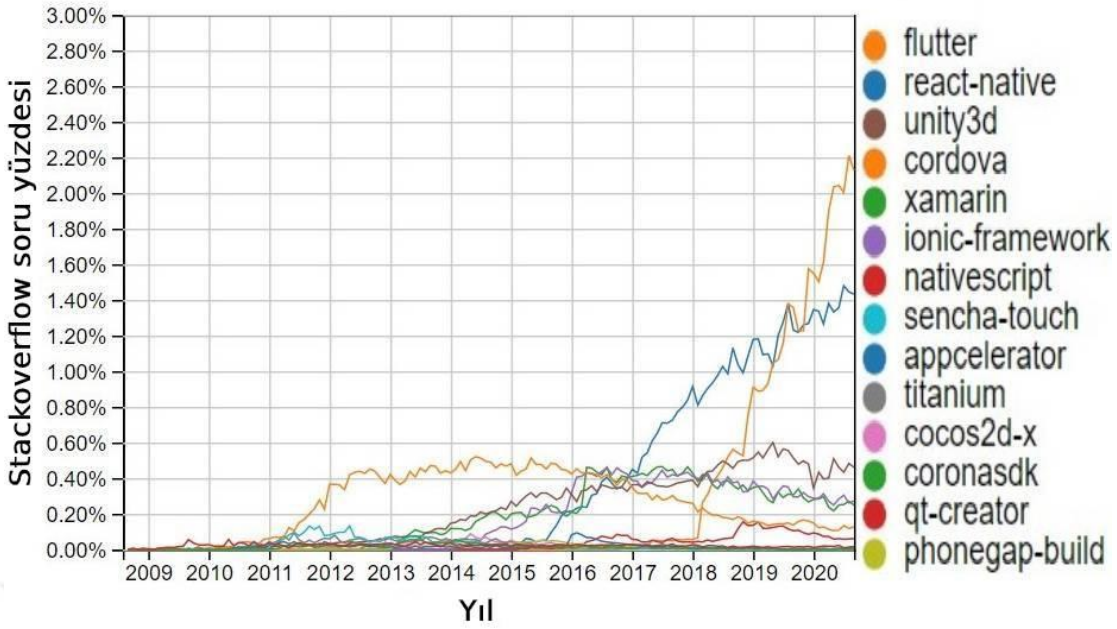
5.1. Son Yıllarda Trend Olan Frameworkler

Son yıllarda en çok kullanılan çapraz platform mobil uygulama geliştirme araçlarının tespiti için programcıların en çok kullandıkları üç sistemden yardım alınacaktır. Bunlar; Google'da arama sayılarına göre karşılaştırma yapmayı sağlayacak Google Trends, programcıların herhangi bir sorun ile karşılaştıklarında yardım alacakları neredeyse ilk kaynak olan Stackoverflow ve programcıların projelerini bulut üzerinde depolayabildikleri, binlerce 3. parti uygulamaya erişebildikleri açık kaynak kodlu içerik sitesi olan GitHub olacaktır.

5.1.1. Stackoverflow

StackOverflow (SO), yazılım geliştirme sorularına yanıtlar elde etmek için en popüler topluluktur ve algoritmalardan dillere ve araçlara kadar değişen konularda hızla büyüyen bir bilgi tabanıdır (Bosu ve ark., 2013).

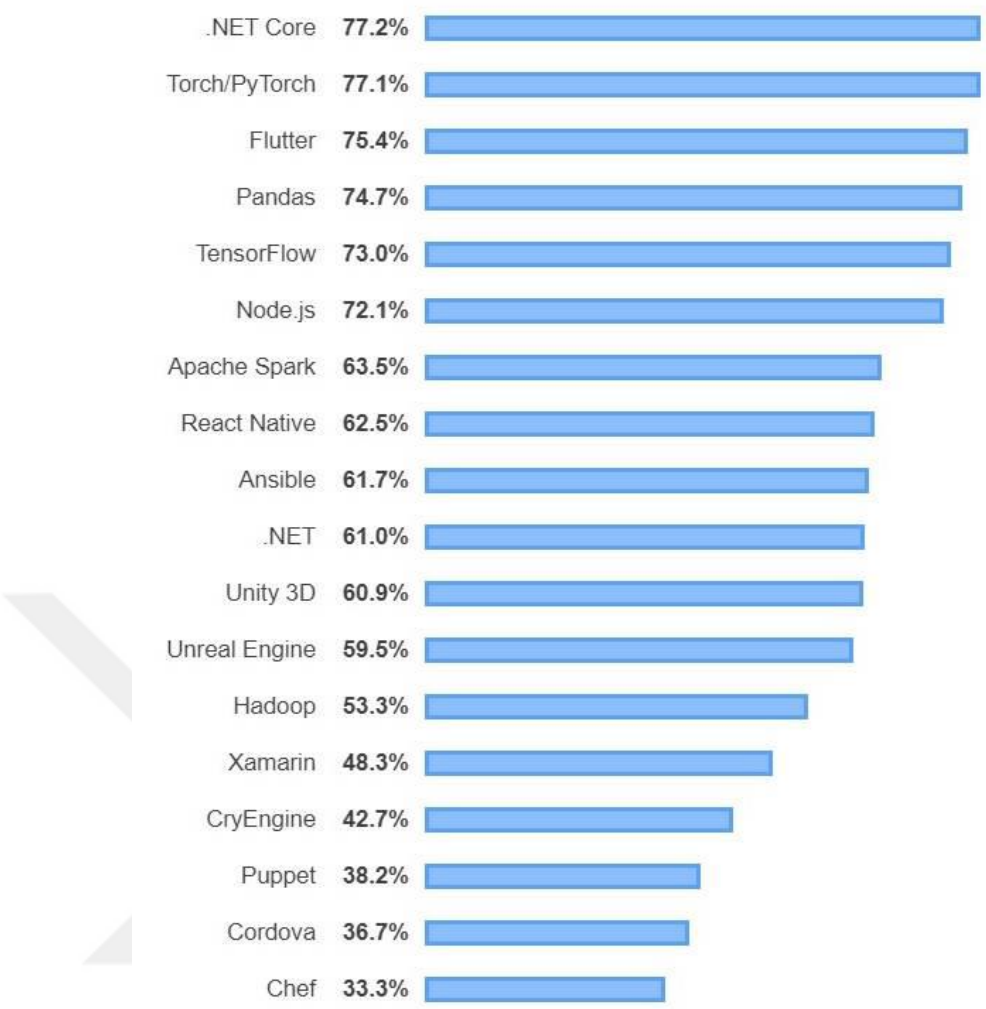
Hedef kitle, uygulama geliştiricileri olduğu için öncelikle Stackoverflow sitesinde son yıllarda en çok aranan ve trend olmuş mobil uygulama geliştirme araçları incelenmelidir. Bu durumu gösteren grafik Şekil 5.1'de verilmiştir.



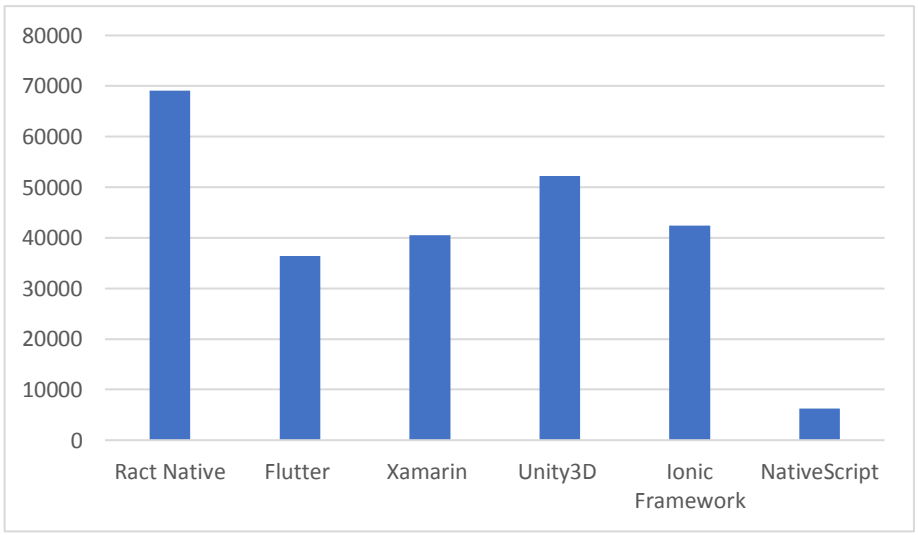
Şekil 5.1. Stackoverflow sitesi üzerinde son yılların trend olan frameworkleri (Stackoverflow, 2020a)

Stackoverflow'ın açıkladığı verilere göre geliştiriciler tarafından en sevilen framework'ler listesinde Flutter 3. React Native ise 8. olmuştur. Listenin tam hali ise Şekil 5.2'de verilmiştir.

Stackoverflow kayıtlarına göre şimdiye kadar Ract Native hakkında 69117, Flutter hakkında 36368, Xamarin hakkında 40475, Unity3D hakkında 52194, Ionic Framework hakkında ise 42397 kayıt açılmıştır. Bu değerlere ait grafik ise Şekil 5.3'te verilmiştir (Stackoverflow, 2020b).



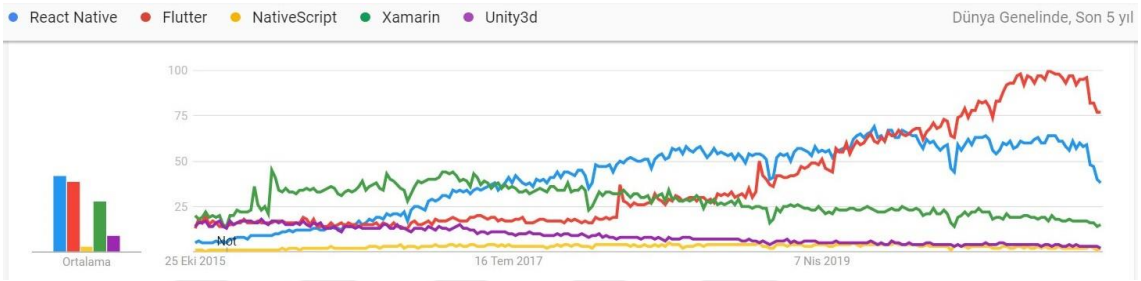
Şekil 5.2. Stackoverflow 2019 verilerine göre en sevilen frameworkler listesi (Stackoverflow, 2020c)



Şekil 5.3. Stackoverflow üzerinde şimdiye kadar belirtilen frameworkler hakkında açılan kayıt sayıları

5.1.2. Google Trends

Google üzerinde yapılan arama istatistiklerini paylaşan servis olan Google Trends'ten alınan Şekil 5.1'deki grafik incelendiğinde, son yıllarda yazılımcıların en çok kullandığı ve en çok destek aradığı geliştirme araçlarının React Native ve Flutter olduğu rahatlıkla görülebilmektedir. Bu ikisini takip eden diğer trend mobil uygulama geliştirme araçları ise Xamarin, Unity3D ve Nativescript olduğu görülmektedir. Bu 5'inin Google Trendlerine göre karşılaştırılmasına bakıldığında ise Şekil 5.4'teki grafik ortaya çıkmaktadır.



Şekil 5.4. Google Trends'e göre son 5 yılın karşılaştırması (Trends, 2020)

Yukarıdaki grafikten de anlaşılacağı üzere, yine React Native ve Flutter diğer trend olan frameworklere göre özellikle 2018 yılı sonrasında en çok tercih edilenler arasına girmiştir.

5.1.3. GitHub

GitHub, yazılım geliştiriciler için en popüler web tabanlı iş birliği platformudur. 2008'den başlayarak, 28 milyondan fazla kullanıcısı ve Git deposuna bir web arayüzü ve iş birliğini yönetmek için araçlar sağlayarak kod paylaşımını kolaylaştıran 57 milyon genel havuzuyla dünyanın en büyük açık kaynak yazılım platformudur. Kodlarda yapılan tüm revizyonların kaynak kodu ve tam geçmişi burada saklanmaktadır. Geliştiriciler, birbirlerinin çalışmalarını takip edebilmekte, derecelendirebilmekte ve kendi projelerini paylaşabilmektedir (Mohan, 2019).

GitHub bu bağlamda çapraz platform mobil uygulama geliştirme araçları için geliştirilen 3. parti uygulamalar için de en büyük paylaşım alanıdır. Örneğin React Native uygulaması üzerinden bir kullanıcının imzasının alınması isteniyorsa, bu durumda baştan bir imza çizim ve kaydetme altyapısı geliştirmeye gerek yoktur. Çünkü bunun için daha önceden geliştirilmiş ve uygulamaya doğrudan entegre edilebilen araçlar GitHub

platformu üzerinde paylaşılmıştır. Bu uygulama parçacıklarının çokluğu, seçilecek platformda hem zaman hem iş gücü hem de uygulama maliyeti açısından çok büyük rol oynamaktadır.

GitHub üzerinde paylaşılan ve anahtar kelime baz alınarak hesaplanan 3. parti uygulama sayıları, Tablo 5.1'de verilmiştir (GitHub, 2020).

Tablo 5.1. GitHub üzerindeki proje sayıları (GitHub, 2020)

Platform adı	**GitHub üzerindeki proje sayısı**
React Native	21,832
Flutter	15,140
Nativescript	687
Xamarin	3,129

Yine GitHub üzerinde kullanıcılar, beğendikleri uygulamaları yıldızlayarak kullanışlı olduklarını diğer kullanıcıların bilgisine sunarlar. Bu bağlamda yıldız sayısı da oldukça büyük öneme sahiptir. Aşağıdaki Tablo 5.2'de ise açık kaynakla paylaşılan bu platformların yıldız sayılarına yer verilmiştir.

Tablo 5.2. GitHub üzerindeki yıldız sayıları

Platform adı		**GitHub üzerindeki yıldız sayısı**
React Native		90.7k
Flutter		105k
Nativescript		19.1k
Xamarin	Xamarin.Forms	4.9k
	Xamarin-macios	1.9k
	Xamarin-android	1.5k

Stackoverflow ve Google Trands verilerine bakıldığında en çok tercih edilen frameworklerden birinin de Unity3D olduğu görülmektedir. Unity3D geliştirme aracı, öne çıkan diğer geliştirme araçlarından farklı olarak daha çok oyun veya yoğun grafikler içeren uygulamaları geliştirmeye uygundur. Bu tarz çapraz platform bir mobil uygulama geliştirmek isteyen bir yazılımcının Unity3D frameworkünü tercih etmesi daha doğru olacaktır. Bu frameworkün haricinde kalan frameworklerin teorik olarak önceki bölümde açıklanan özelliklerine binaen Tablo 5.3 ortaya çıkarılabilmektedir.

Tablo 5.3. Popüler çapraz platform mobil uygulama geliştirme araçlarının bazı özellikleri

	React Native	**Flutter**	**Xamarin**	**Ionic Framework**	**NativeScript**
Açık Kaynak	Evet	Evet	Evet, ücretli	Evet, ücretli	Evet, ücretli
Geliştirici Şirket	Facebook	Google	Microsoft	Drifty.co	Telerik
Geliştirme yapılabilecek diller	React.js, Javascript	Dart	C# (Objective-C, Java ve C++ tabanlı kütüphaleler kullanılabilir.)	Javascript, TypeScript	Javascript, Angular, TypeScript
Uygulama çıktısı verebileceği platformlar	iOS, Android	iOS, Android	iOS, Android, Windows	iOS, Android	iOS, Android
Çıkış Tarihi	2013	2015	2011	2013	2014

5.2. Karşılaştırma Yöntemi

Tüm bu veriler doğrultusunda son dönemlerde en çok tercih edilen çapraz platform mobil uygulama geliştirme araçlarından React Native, Flutter, Xamarin ve Nativescript'in karşılaştırılmasına karar verilmiştir.

Karşılaştırma işleminde; her mobil uygulama geliştirme platformu üzerinde aynı uygulama yazılıp bu uygulamaların aynı cihaz üzerinde yüklenme performansları ölçülecektir. Bu uygulamalar, toplamda 1000 elemandan oluşan ve stillerle özelleştirilmiş bir listeyi içermektedir. Yazılan uygulamalar, Android 10 işletim sistemi yüklü olan Google Pixel 3 cihazının emülatörü üzerinde debug modda test edilecektir. Sonuçlar, bu uygulamadaki nesnelerin tamamının ekranda görüntülenmesi süresince çıkarılacaktır.

5.3. Uygulama Kodları

Aşağıdaki şekillerde sırasıyla React Native, Flutter, Xamarin ve Nativescript platformlarında yazılan kodlar verilmiştir. Xamarin ve Nativescript'te tasarım kodları ile program kodları ayrı yazıldığından aşağıda da dizayn kodları ve fonksiyon kodları şeklinde ayrı ayrı verilmiştir.

```
export default function App(){
  var myloop = [];
    for (let i = 0; i < 1000; i++) {
      myloop.push(
        "Mehmet" + i
      );
    }
  return (
    <View style={styles.container}>
      <FlatList style={{ flex: 0 }} initialNumToRender={myloop.length}
        keyExtractor={(item) => item}
        data={myloop}
        renderItem={ ({item}) => (
          <View>
            <Text style={styles.item}>{item}</Text>
          </View>
        ) }
      />
    </View>
  );
}
```

Şekil 5.5. React Native ile yazılan uygulamanın kodları

```
class BodyLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return _myListView(context);
  }
}
Widget _myListView(BuildContext context) {
  return ListView.separated(
    itemCount: 1000,
    padding: const EdgeInsets.all(8),
    itemBuilder: (context, index) {
      return Container(
        height: 50,
        color: Colors.green,
        child: Center(child:  Text('Entry ${index}',style: TextStyle(color: Colors.white,fontSize: 22))),
      );
    },
    separatorBuilder: (context, index) {
      return Divider();
    },
  );
}
```

Şekil 5.6. Flutter ile yazılan uygulamanın kodları

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="ilkXamarin.MainPage">
    <ListView x:Name="listview" HasUnevenRows="True"
    SeparatorVisibility="None" RowHeight="100" Margin="20"
    BackgroundColor="LightCyan">
        <ListView.ItemTemplate>
            <DataTemplate>
                <ViewCell>
                    <StackLayout VerticalOptions="Center"
                    Orientation="Horizontal" HorizontalOptions="Center"
                    Padding="20">
                        <Image Source="{Binding ImgUrl}"/>
                        <StackLayout Orientation="Horizontal"
                        BackgroundColor="Aqua">
                            <Label Text="{Binding Name}" FontSize="22" />
                            <Label Text="{Binding Money}"/>
                        </StackLayout>
                    </StackLayout>
                </ViewCell>
            </DataTemplate>
        </ListView.ItemTemplate>
    </ListView>
</ContentPage>
```

Şekil 5.7. Xamarin ile yazılan uygulamanın dizayn kodları

```
namespace ilkXamarin
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
            var list = new List<Personal>();
            for(int i=0; i < 1000; i++)
            {
                Personal p = new Personal();
                p.Name = "Mehmet" + i;
                p.Money = i;
                list.Add(p);
            }
            listview.ItemsSource = list;
        }
    }
}
```

Şekil 5.8. Xamarin ile yazılan uygulamanın fonksiyon kodları

```
<Page xmlns="http://schemas.nativescript.org/tns.xsd" navigatingTo="onNavigatingTo">
    <ActionBar title="My App" icon=""></ActionBar>
    <StackLayout class="p-20">
        <ListView items="{{ myTitles }}"
                itemTap="onItemTap"
                loaded="{{ onListViewLoaded }}"
                separatorColor="red" rowHeight="70"
                class="list-group" id="listView" row="2">
            <ListView.itemTemplate>
                <StackLayout class="list-group-item">
                    <Label text="{{ title || 'Downloading...' }}" textWrap="true" class="title" />
                </StackLayout>
            </ListView.itemTemplate>
        </ListView>
    </StackLayout>
</Page>
```

Şekil 5.9. Nativescript ile yazılan uygulamanın dizayn kodları

```
const createViewModel = require("./main-view-model").createViewModel;
function onNavigatingTo(args) {
    const page = args.object;
    page.bindingContext = createViewModel();
}
exports.onNavigatingTo = onNavigatingTo;
const fromObject = require("tns-core-modules/data/observable").fromObject;
function onNavigatingTo(args) {
    const page = args.object;
    const vm = fromObject({
        myTitles: [
            { title: "" },
        ]
    });
    for (let index = 0; index < 999; index++) {
        vm.myTitles.push({ title: "Mehmet " + index });
    }
    page.bindingContext = vm;
}
exports.onNavigatingTo = onNavigatingTo;

function onListViewLoaded(args) {
    const listView = args.object;
}
exports.onListViewLoaded = onListViewLoaded;
```

Şekil 5.10. Nativescript ile yazılan uygulamanın fonksiyon kodları

Aşağıdaki şekillerde ise bu uygulamalara ait build.gradle dosyalarının ekran görüntüleri verilmiştir. Xamarin framewokünün temel uygulaması build.gradle dosyası bulundurmamaktadır. Lakin istenilirse sonradan buradaki temel ayarlar değiştirilebilmektedir. Dolayısıyla belirtilen sürümlerin tamamının temel uygulamasındaki gradle ayarları birbirleri ile aynıdır.

```
buildscript {
    ext {
        buildToolsVersion = "29.0.2"
        minSdkVersion = 16
        compileSdkVersion = 29
        targetSdkVersion = 29
    }
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath("com.android.tools.build:gradle:3.5.3")
    }
}
allprojects {
    repositories {
        mavenLocal()
        maven {
            url("$rootDir/../node_modules/react-native/android")
        }
        maven {
            url("$rootDir/../node_modules/jsc-android/dist")
        }
        google()
        jcenter()
        maven { url 'https://www.jitpack.io' }
    }
}
```

Şekil 5.11. React Native uygulamasının gradle.build dosyası

```
buildscript {
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.0'
    }
}
allprojects {
    repositories {
        google()
        jcenter()
    }
}
rootProject.buildDir = '../build'
subprojects {
    project.buildDir = "${rootProject.buildDir}/${project.name}"
}
subprojects {
    project.evaluationDependsOn(':app')
}
task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Şekil 5.12. Flutter uygulamasının gradle.build dosyası

```
buildscript {
    def initialize = { ->
        def userDir = "${rootProject.projectDir}/../.."
        apply from: "$rootDir/gradle-helpers/user_properties_reader.gradle"
        apply from: "$rootDir/gradle-helpers/paths.gradle"
        rootProject.ext.userDefinedGradleProperties = getUserProperties("${getAppResourcesPath(userDir)}/Android")
    }
    initialize()
    def computeKotlinVersion = { -> project.hasProperty("kotlinVersion") ? kotlinVersion : "1.3.72"
    }
    def kotlinVersion = computeKotlinVersion()
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.6.4'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlinVersion"
    }
}
allprojects {
    repositories {
        google()
        jcenter()
    }
    beforeEvaluate { project ->
        if (rootProject.hasProperty("userDefinedGradleProperties")) {
            rootProject.ext.userDefinedGradleProperties.each { entry ->
                def propertyName = entry.getKey()
                def propertyValue = entry.getValue()
                project.ext.set(propertyName, propertyValue)
            }
        }

    }
}
task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Şekil 5.13. Nativescript uygulamasının gradle.build dosyası

5.4. Bölüm Değerlendirmesi

Bu bölümde öncelikle mobil uygulama geliştiricilerinin aktiviteleri baz alınarak son yıllarda en çok tercih edilen çapraz platform mobil uygulama geliştirme araçları tespit edilmiştir. Bu işlem için Stackoverflow, Google Trends ve GitHub internet sitelerinin sundukları verilerden faydalanılmıştır. Ardından, tespit edilen bu framework'leri birbirleri ile karşılaştırabilmek için her biri üzerinde ayrı ayrı geliştirilen temelde aynı olan uygulama hakkında bilgilere yer verilmiştir.

6. DENEYSEL SONUÇLAR

Bu bölümde, yazılan uygulamalar her platformda ayrı ayrı debug modda çalıştırılıp uygulama boyutu, oluşturma süresi, cihazın kaynaklarının kullanımı gibi parametreler baz alınarak ölçümleri yapılacaktır.

6.1. Uygulama Boyutu

Bu bölümde, son yıllarda trend olan çapraz platform mobil uygulama geliştirme araçları ile en temel uygulamaları çalışır hale getirip boyutları incelenecektir.

6.1.1. Uygulama kaynağının disk üzerindeki boyutu

0.63.3 sürümlü React Native ile birden fazla platformda çalışacak hâlde olan en temel uygulama kurulup çalıştırıldıktan sonra, bilgisayar üzerinde uygulama kaynaklarının diskteki toplam boyutu 467 MB olarak ölçülmüştür. 1.22.1 sürümlü Flutter'da ise bu boyut 401 MB olarak ölçülmüştür.

Visual Studio üzerinde 16.7.000.456 sürümlü Xamarin için gerekli eklentiler kurulup Xamarin.Forms ile yeni bir uygulama oluşturulduğunda, oluşan uygulamanın boyutu 133 MB olarak ölçülmüştür. Bu platformda kaynak boyutunun düşük olmasındaVisual Studio'nun 2019 sürümü ile birlikte gelen paket yönetim sisteminin rolü de büyüktür.

7.0.10 sürümlü Nativescript platformunda da aynı şekilde en temel uygulama kurulduğunda uygulama için gerekli olan kaynak dosyalarının boyutu 434 MB olarak ölçülmüştür. Ölçüm yapılan platformlardaki dosya boyutları Tablo 6.1'de verilmiştir.

Tablo 6.1. Ölçüm yapılan platformlardaki kaynak dosyaların disk üzerindeki boyutları

Platform Adı	**Kaynak Dosyalarının Diskteki Boyutu (MB)**	**Sürüm**
React Native	467	0.63.3
Flutter	401	1.22.1
Nativescript	434	7.0.10
Xamarin	133	16.7.000.456

6.1.2. Uygulamanın kurulum dosyasının boyutu

Flutter üzerinde en temel projenin Android işletim sistemi için olan kurulum dosyasının boyutu 15.2 MB olarak ölçülmüştür. React Native'de ise aynı şekilde ilk

kurulum ile birlikte gelen en temel programın kurulum dosyasının boyutunun 23.4 MB olduğu görülmüştür. Aynı şekilde Nativescript için oluşturulan release moddaki kurulum dosyasının boyutu 23,5 MB olarak, Xamarin platformunda ise aynı işlem sonucu kurulum dosyasının boyutu 11,1 MB olarak ölçülmüştür.

6.2. Oluşturma Süresi

React Native ile yazılan uygulama çalıştırıldığında listedeki elemanlar varsayılan olarak 20'şerli olacak şekilde ekranda oluşturulmaktadır. Bu haliyle uygulamanın açılıp listenin tamamının yüklenmesi toplamda 3 dakika 26 saniye sürmektedir. Listenin tamamının aynı anda yüklenmesini sağlayan ayarlar yapıldığında, uygulamanın başlamasından listenin en altındaki elemanı görene kadar geçen süre 51 saniye olarak ölçülmüştür. Listenin en altındaki öğe görüntülendiği esnada liste hâlâ aşağı yönde boş listeyi göstermeye devam etmekte ve 3 saniye sonra listenin altındaki boş elemanlar otomatik olarak atılıp son listeyi gösterecek şekilde kısıtlanmıştır.

Flutter ile aynı şekilde bir uygulama oluşturulduğunda, uygulamanın yüklenip listenin en altındaki elemanın görüntülenmesi toplamda 28 saniye sürmüştür.

Nativescript aynı işlem 14 saniye, Xamarin'de ise 36 saniye sürmüştür.

6.3. Cihaz Kaynaklarının Kullanımı

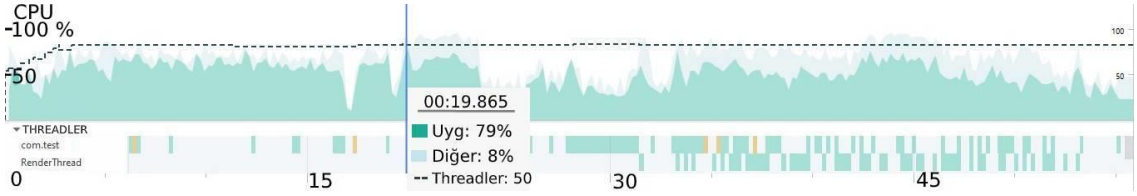
Bu bölümde her platform için ayrı ayrı yazılan mobil uygulamaları cihaz üzerinde çalıştırıp tüm nesnelerin ekranda gösterilmesi esnasındaki cihaz kaynak tüketimleri ölçülecektir. Ölçüm işleminde Android Studio uygulamasının Profiler özelliği kullanılacaktır.

6.3.1. CPU kullanımı

Çapraz platform mobil uygulama geliştirme araçlarının her biri ile yazılan uygulamalar sırasıyla çalıştırıldıktan sonra cihaz üzerinde çalışan threadler listesinde o anki uygulamanın adını içeren bir thread oluşmuştur. Ölçümler yalnızca o anki uygulamaya ait thread'in kullandığı CPU miktarları baz alınarak yapılmıştır. Ölçümler, her bir uygulamadaki listeler içerisindeki tüm objeler ekranda gösterilebilecek şekilde yükleninceye kadar yapılmıştır.

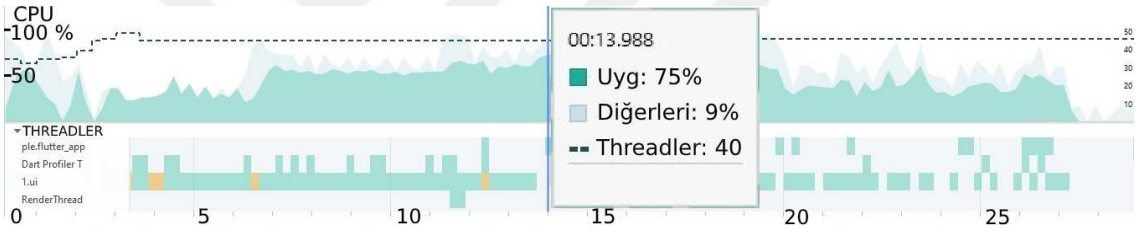
İlk olarak React Native ile yazılan uygulama çalıştırılmış ve thread listesinde test(com.test) adında bir thread oluşmuştur. Bu thread Şekil 6.1'deki gibi işlemciyi 51 saniye boyunca kullanmış ve maksimum %79 oranında kaynağa ihtiyaç duymuştur.

Grafiklerdeki dikey eksen yüzde cinsinden CPU'nun kullanılan kaynağının oranını, yatay eksen ise saniye cinsinden zamanı belirtmektedir.



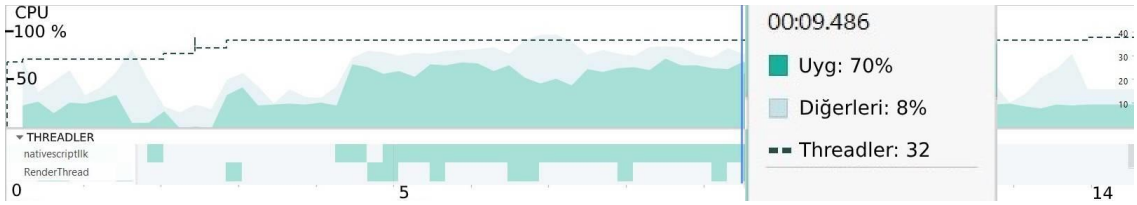
Şekil 6.1. React Native ile yazılan uygulamanın kullandığı CPU oranları

Flutter ile yazılan uygulama çalıştırıldığında thread listesinde flutter_app(ple.flutter_app) isminde bir thread oluşmuştur. Bu thread Şekil 6.2'deki gibi işlemciyi 28 saniye boyunca kullanmış ve maksimum %75 oranında kaynağa ihtiyaç duymuştur.

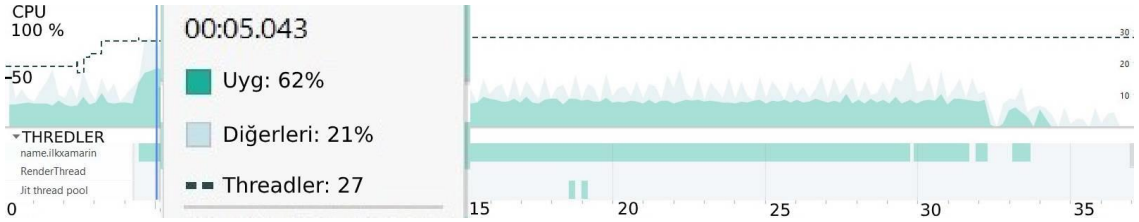


Şekil 6.2. Flutter ile yazılan uygulamanın kullandığı CPU oranları

Nativescript ile yazılan uygulama çalıştırıldığında thread listesinde nativescriptIlk (org.nativescript.nativescriptIlk) isminde bir thread oluşmuştur. Bu thread Şekil 6.3'teki gibi işlemciyi 14 saniye boyunca kullanmış ve maksimum %70 oranında kaynağa ihtiyaç duymuştur. Aynı işlem ile Xamarin'e ait grafik Şekil 6.4'te verilmiştir. Bu işlem 36 saniye sürmüş ve maksimum %62 oranında kaynağa ihtiyaç duymuştur.



Şekil 6.3. Nativescript ile yazılan uygulamanın kullandığı CPU oranları

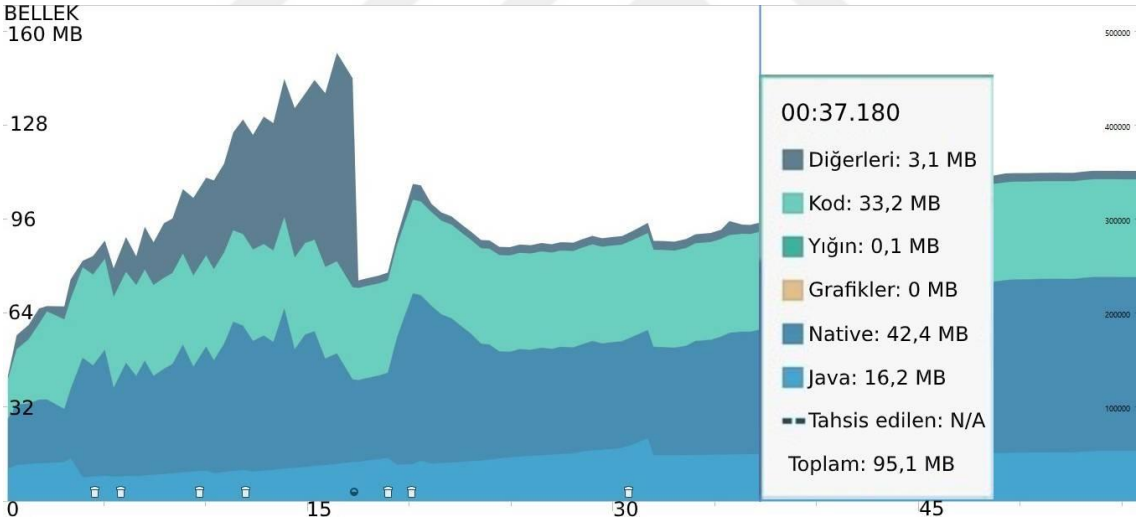


Şekil 6.4. Xamarin ile yazılan uygulamanın kullandığı CPU oranları

6.3.2. Bellek kullanımı

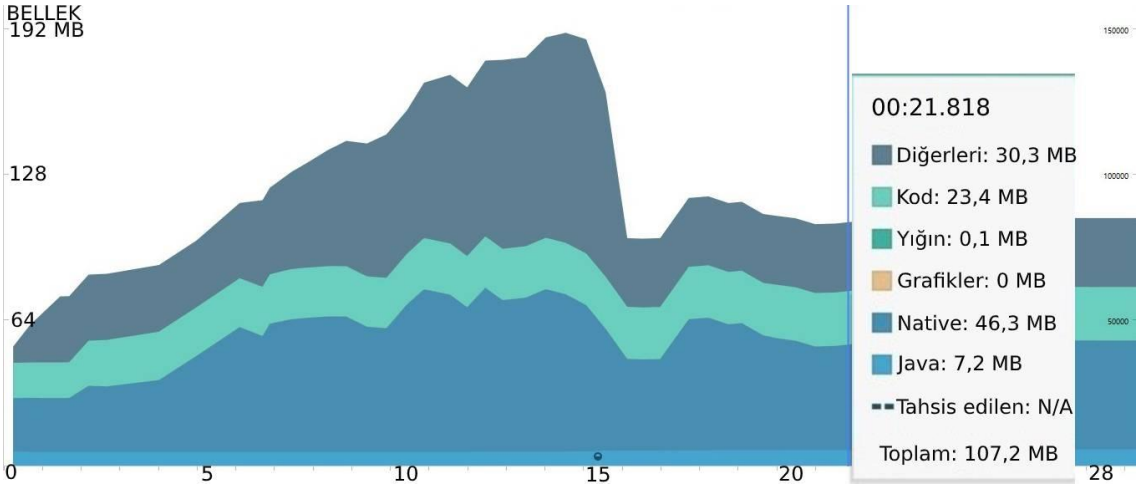
Bellek kullanımını ölçmek için de aynı şekilde her platform için yazılan uygulamalar teker teker çalıştırılmış ve 1000 adet objenin tamamının yüklenmesi esnasındaki bellek kullanım miktarları ölçülmüştür.

React Native ile yazılan uygulamada bu işlem gerçekleştirildiğinde Şekil 6.5'deki grafik ortaya çıkmış ve uygulama maksimum 33.2 MB'lık kaynağa ihtiyaç duymuştur. Grafikteki yeşil kısım uygulamanın kullandığı miktarı göstermektedir. Dikey eksen kullanılan bellek miktarını, yatay eksen ise saniye cinsinden zamanı belirtmektedir.

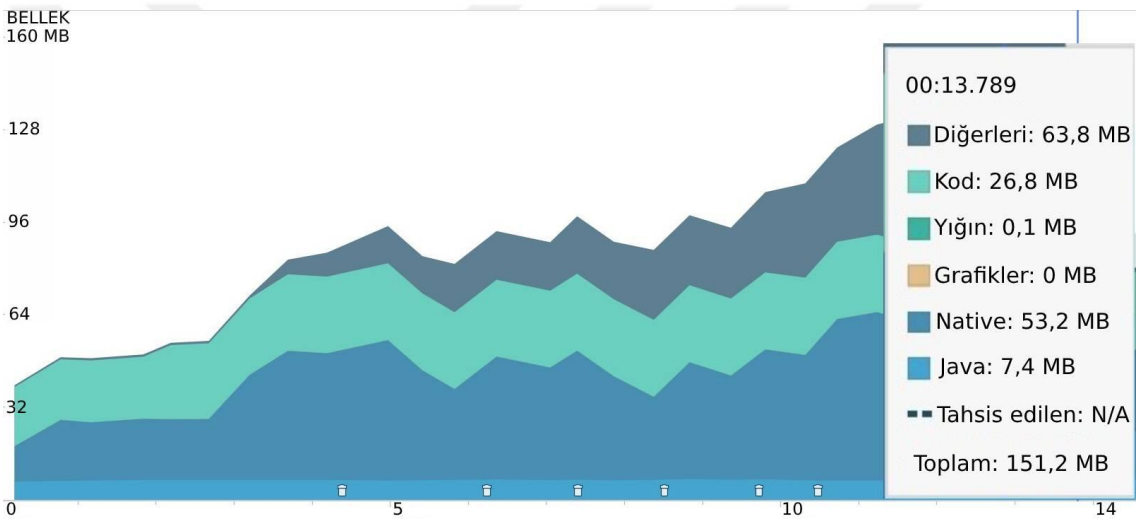


Şekil 6.5. React Native ile yazılan uygulamanın kullandığı bellek oranları

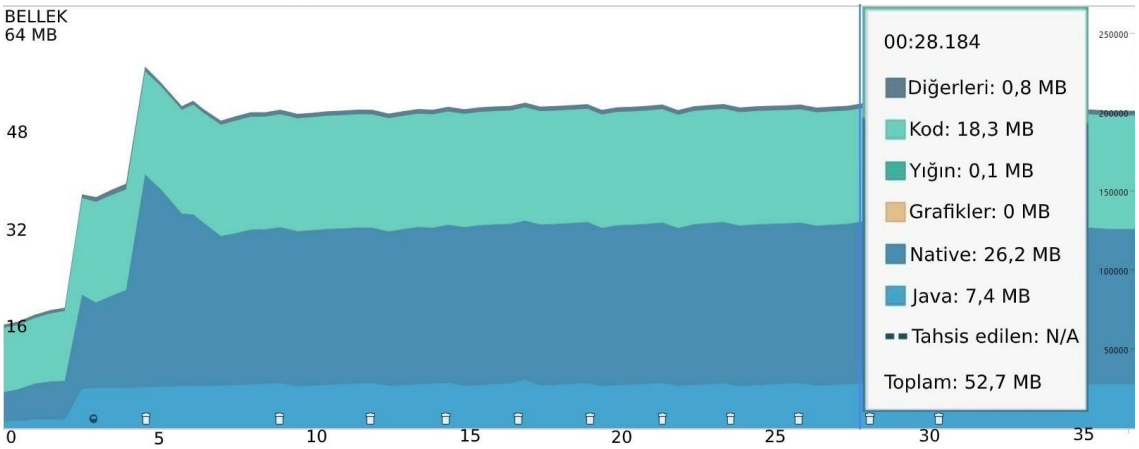
Flutter ile yazılan uygulamada aynı işlem gerçekleştirildiğinde Şekil 6.6'daki grafik ortaya çıkmış ve uygulama maksimum 23.4 MB'lık kaynağa ihtiyaç duymuştur. Nativescript'e ait grafik Şekil 6.7'de verilmiş ve maksimum 26.8 MB'lık kaynağa ve son olarak Xamarin'e ait grafik ise Şekil 6.8'de verilmiş ve maksimum 18.3 MB'lık kaynağa ihtiyaç duymuştur.



Şekil 6.6. Flutter ile yazılan uygulamanın kullandığı bellek oranları



Şekil 6.7. Nativescript ile yazılan uygulamanın kullandığı bellek oranları

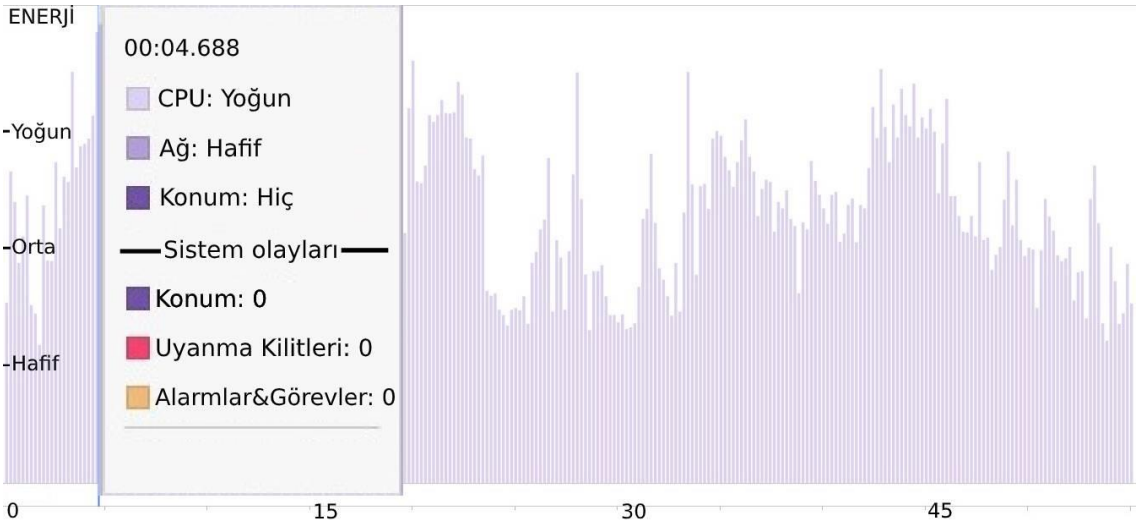


Şekil 6.8. Xamarin ile yazılan uygulamanın kullandığı bellek oranları

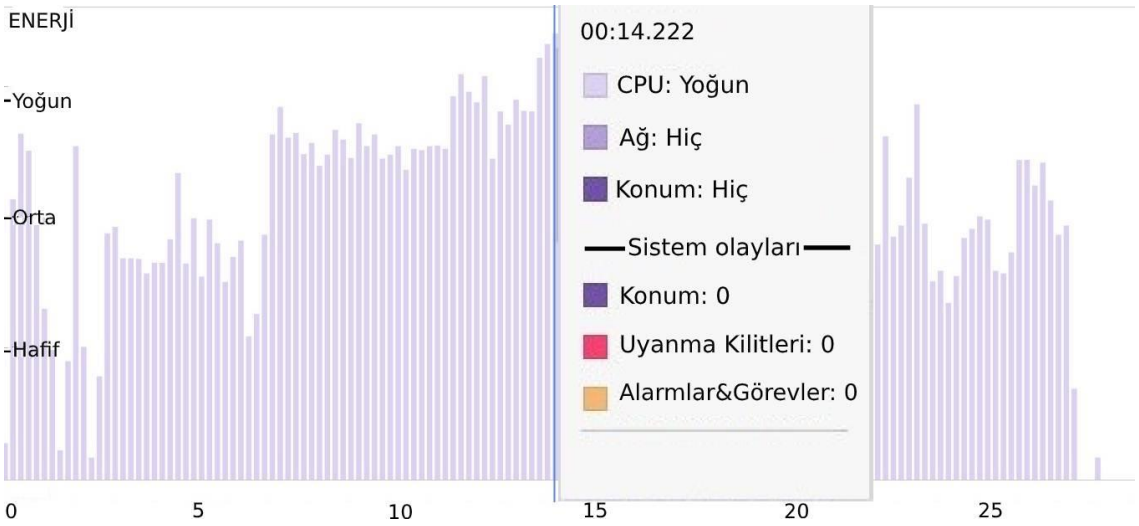
6.3.3. Enerji tüketimi

Enerji tüketimi veya bir başka deyişle batarya kullanımını ölçmek için de aynı şekilde uygulamaların tüm nesneleri yüklemesi esnasındaki durumlarına bakılmıştır.

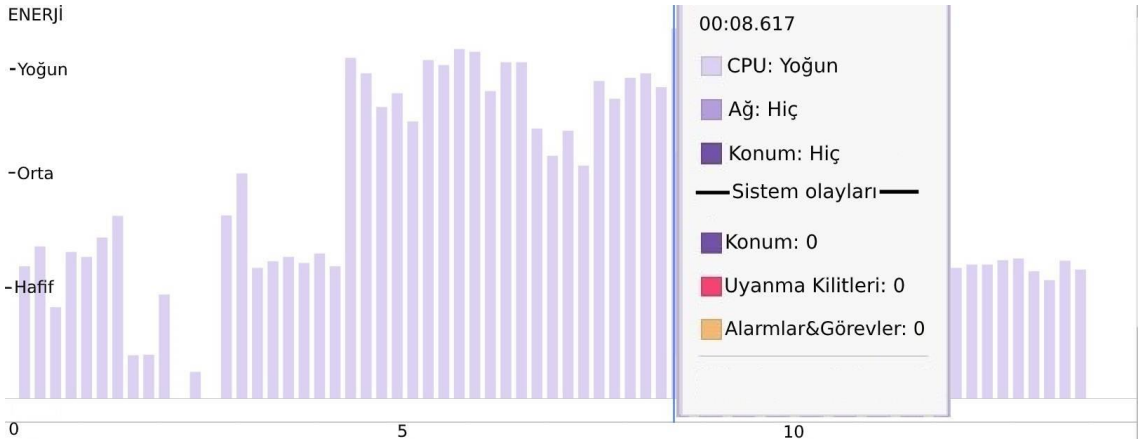
React Native ile yazılan uygulamanın içeriğinin yüklenmesi esnasındaki batarya kullanım verileri Şekil 6.9'da verilmiştir. Bu grafikteki dikey eksen bataryanın kullanım oranını, yatay eksen ise saniye cinsinden zamanı belirtmektedir. Flutter'a ait grafik Şekil 6.10'da, Nativescript grafiği Şekil 6.11'de ve son olarak Xamarin'e ait grafik ise Şekil 6.12'de verilmiştir.



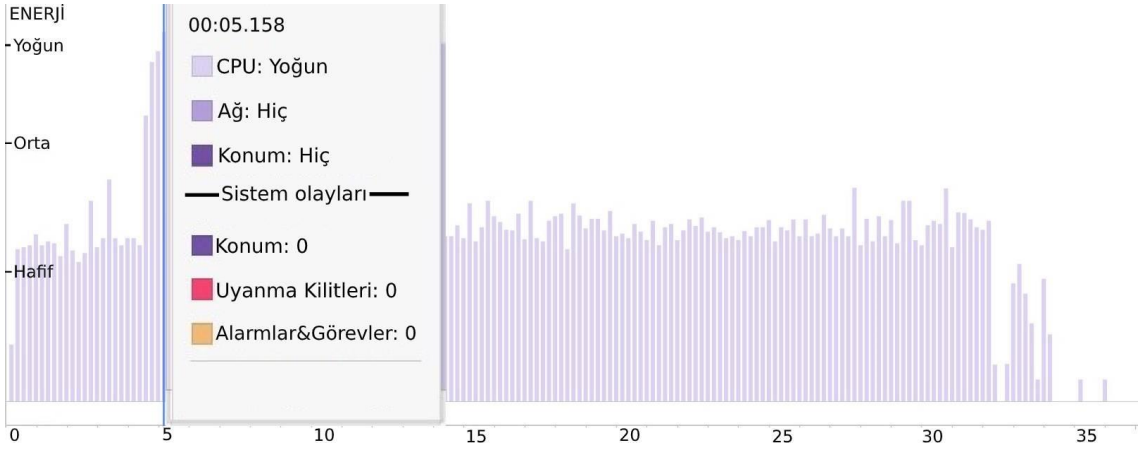
Şekil 6.9. React Native ile yazılan uygulamanın kullandığı enerji oranları



Şekil 6.10. Flutter ile yazılan uygulamanın kullandığı enerji oranları



Şekil 6.11. Nativescript ile yazılan uygulamanın kullandığı enerji oranları

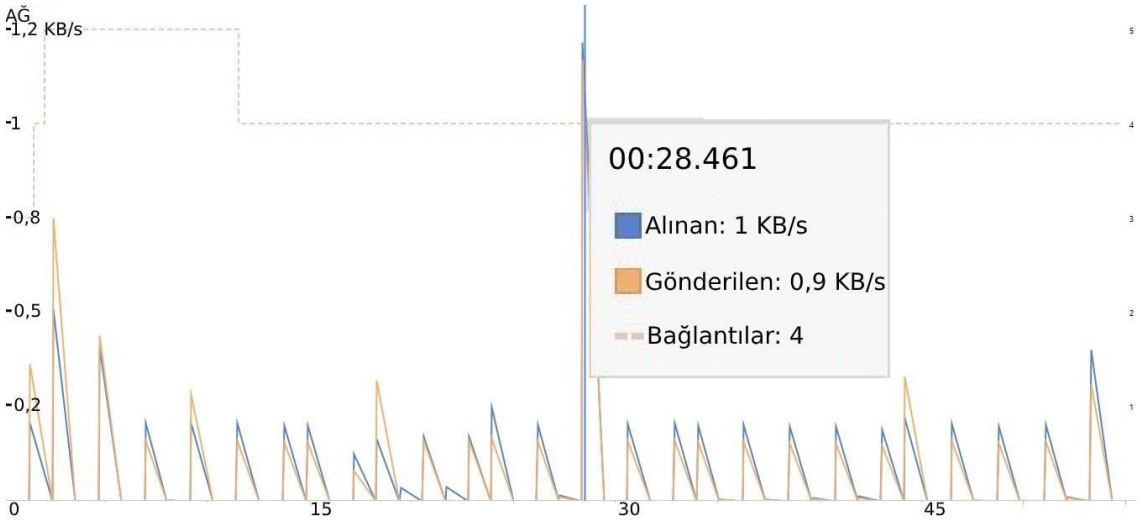


Şekil 6.12. Xamarin ile yazılan uygulamanın kullandığı enerji oranları

6.3.4. Ağ kullanımı

Bu bölümde tüm liste yüklenene kadarki ağ kullanımlarına bakılacaktır. Uygulamaların hiçbirinde ağ isteğine yer verilmemiştir. Bu durumda uygulama kodunun herhangi bir ağ alışverişi yapmaması beklenir.

React Native ile yazılan uygulamada aralıklı olarak ağ istekleri yapılmaktadır. İlgili grafik Şekil 6.13'te verilmiştir. Bu grafikte dikey eksen saniyede gerçekleşen ağ alışveriş miktarını, yatay eksen ise saniye cinsinden zamanı belirtmektedir.

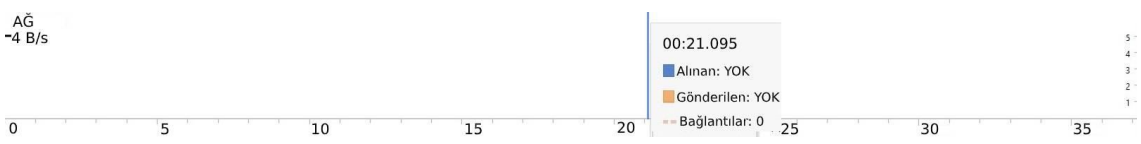


Şekil 6.13. React Native ile yazılan uygulamanın kullandığı ağ oranları

Diğer mobil uygulama geliştirme araçları ile yazılan uygulamalar herhangi bir ağ iletişiminde bulunmamıştır. Flutter'a ait grafik Şekil 6.14'te, Nativescript'e ait grafik Şekil 6.15'de ve Xamarin'a ait grafik ise Şekil 6.16'da verilmiştir.

Şekil 6.14. Flutter ile yazılan uygulamanın kullandığı ağ oranları

Şekil 6.15. Nativescript ile yazılan uygulamanın kullandığı ağ oranları



Şekil 6.16. Xamarin ile yazılan uygulamanın kullandığı ağ oranları

6.4. Bölüm Değerlendirmesi

Bu bölümde, önceki bölümde bahsedilen mobil uygulamalar aynı emülatör üzerinde Android Studio'nun Profiler özelliği kullanılarak debug modda çalıştırılmış ve her birinin bu esnadaki CPU, bellek, enerji ve ağ kullanım değerleri ölçülmüş, bazı boyutları ve oluşturma (render) süreleri karşılaştırılmış, uygulama performansları hakkında bilgilere yer verilmiştir.

7. SONUÇ VE ÖNERİLER

Çapraz platform mobil uygulama geliştirme aracının seçilmesi için en önemli iki kriter, geliştirilen uygulamanın performanslı ve hatasız çalışması ve arkasında yeterli geliştirici desteğinin olmasıdır. Bu bağlamda Stackoverflow, GitHub ve Google Trends verilerine bakıldığında son yıllarda en büyük popülarite ve geliştirici desteğine React Native ve Flutter sahiptir. Bu kritere göre Flutter, son birkaç yıl içerisinde React Native'in önüne geçmiştir. Performans anlamında da Flutter'ın, React Native'den önde olduğu görülmektedir.

Eğer yalnızca bir uygulama geliştirmek için bu araçlar kullanılacaksa, bakılması gereken önemli kriterlerden birisi de 3. parti yazılım desteğidir. Geliştirilmek istenen uygulama modüllere ayrılmalı ve bu modüllerin daha önce seçilecek araç için geliştirilip geliştirilmediğine bakılmalıdır. Eğer hali hazırda web teknolojilerine hakimiyet varsa, hızlı öğrenmek için geliştirme dillerine göz atılmalıdır. Bu anlamda React Native, Javascript ile geliştirme yapılabileceğinden bir adım öndedir. Flutter'da ise kendi geliştirme dili olan Dart öğrenilmelidir.

Tüm bu değerlendirmelerin sonucunda React Native ve Flutter araçlarından herhangi birinin tercih edilmesi önerilmekle birlikte nihai karar yine geliştiricinin kendisine bırakılmıştır.

KAYNAKLAR

Ahmad, M. S., Musa, N. E., Nadarajah, R., Hassan, R. ve Othman, N. E., 2013, Comparison between android and iOS Operating System in terms of security, *2013 8th International Conference on Information Technology in Asia (CITA)*, 1-4.

Akinkuolie, B. B., Lin, C.-F. ve Yuan, S.-M., 2011, A cross-platform mobile learning system using QT SDK framework, *2011 Fifth International Conference on Genetic and Evolutionary Computing*, 163-167.

Allen, S., Graupera, V. ve Lundrigan, L., 2010a, Pro smartphone cross-platform development: iPhone, blackberry, windows mobile and android development and distribution, Apress, p.

Allen, S., Lundrigan, L. ve Graupera, V., 2010b, Pro smartphone cross-platform development: iPhone, BlackBerry, windows mobile, and android development and distribution, Apress Media LLC, p.

Amatya, S. ve Kurti, A., 2013, Cross-platform mobile development: challenges and opportunities, *International Conference on ICT Innovations*, 219-229.

Anderson, N. J., 2016, Getting Started with NativeScript, Packt Publishing Ltd, p.

Android, 2019, Distribution dashboard, https://developer.android.com/about/dashboards: [23/12/2019].

Beal, V., 2018, Mobile Operating Systems (Mobile OS) Explained, https://www.webopedia.com/DidYouKnow/Hardware_Software/mobile-operating-systems-mobile-os-explained.html: [16.12.2019].

Bennis, L. l. g. c. ve Amali, S., 2019, From Learning Game to Adaptive Ubiquitous Game Based Learning, *International Journal of Emerging Technologies in Learning*, 14 (16), 55-65.

Bosu, A., Corley, C. S., Heaton, D., Chatterji, D., Carver, J. C. ve Kraft, N. A., 2013, Building reputation in StackOverflow: An empirical investigation, *2013 10th Working Conference on Mining Software Repositories (MSR)*, 89-92.

Boushehrinejadmoradi, N., Ganapathy, V., Nagarakatte, S. ve Iftode, L., 2015, Testing cross-platform mobile app development frameworks (t), *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 441-451.

Boyer, R. ve Mew, K., 2016, Android Application Development Cookbook - Second Edition, *Birmingham, UK*, Packt Publishing, p.

Charkaoui, S. ve Adraoui, Z., 2014, Cross-platform mobile development approaches, *2014 Third IEEE International Colloquium in Information Science and Technology (CIST)*, 188-191.

Cheon, Y., 2019, Multiplatform Application Development for Android and Java, *2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)*, 1-5.

Cocos, 2020, Basic Cocos2d-x Concepts, https://docs.cocos.com/cocos2d-x/manual/en/basic_concepts/: [27.02.2020].

Corona, 2020, Introduction to Corona, https://docs.coronalabs.com/guide/programming/intro/index.html:

Dalmasso, I., Datta, S. K., Bonnet, C. ve Nikaein, N., 2013a, Survey, Comparison and Evaluation of Cross Platform Mobile Application Development Tools, *2013 9th International Wireless Communications and Mobile Computing Conference (Iwcmc)*, 323-328.

Dalmasso, I., Datta, S. K., Bonnet, C. ve Nikaein, N., 2013b, Survey, comparison and evaluation of cross platform mobile application development tools, *2013 9th International Wireless Communications and Mobile Computing Conference (Iwcmc)*, 323-328.

Deshmukh, M. S. ve Rajput, S. R., 2016, Smartphone Based Citizen Complaint System for Urban Maintenance Using GIS, *International Journal of Scientific & Engineering Research*, 7 (5), 1591-1599.

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

El-Kassas, W. S., Abdullah, B. A., Yousef, A. H. ve Wahba, A. M., 2016, Enhanced Code Conversion Approach for the Integrated Cross-Platform Mobile Development (ICPMD), *IEEE Transactions on Software Engineering*, 42 (11), 1036-1053.

El-Kassas, W. S., Abdullah, B. A., Yousef, A. H. ve Wahba, A. M., 2017, Taxonomy of Cross-Platform Mobile Applications Development Approaches, *Ain Shams Engineering Journal*, 8 (2), 163-190.

Ferreira, C. M. S., Peixoto, M. J. P., Duarte, P. A. S., Torres, A. B. B., Junior, M. L. S., Rocha, L. S. ve Viana, W., 2018, An Evaluation of Cross-Platform Frameworks for Multimedia Mobile Applications Development, *IEEE Latin America Transactions*, 16 (4), 1206-1212.

GitHub, 2020, Topics on GitHub, https://github.com/topics/: [19.10.2020].

Griffith, C., 2017, Mobile App Development with Ionic, Revised Edition: Cross-Platform Apps with Ionic, Angular, and Cordova, " O'Reilly Media, Inc.", p.

Grønli, T., Hansen, J., Ghinea, G. ve Younas, M., 2014, Mobile Application Platform Heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS, *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, 635-641.

Heitkötter, H., Hanschke, S. ve Majchrzak, T. A., 2012, Evaluating cross-platform development approaches for mobile applications, *International Conference on Web Information Systems and Technologies*, 120-138.

Hussain, F., Jones, G. ve Gurung, A., 2014, Cocos2d-x game development essentials, Packt Publishing, p.

iFactr, 2020, iFactr Documentation, https://developer.zebra.com/community/tools/ifactr: [24.02.2020].

Ionescu, V. M., 2016, Using cross platform development libraries. Telerik mobile, *2016 15th RoEduNet Conference: Networking in Education and Research*, 1-6.

Ionic, 2020, https://ionicframework.com/docs/intro: [18.02.2020].

Jasonette, 2020, Getting Started Jasonette, https://docs.jasonette.com/: [20.02.2020].

Jawad, H. M., 2019, Android Mobile App Development as a Motivation towards Computer Programming, *2019 IEEE International Conference on Electro Information Technology (EIT)*, 169-175.

Jiang, S., 2016, Comparison of native, cross-platform and hyper mobile development tools approaches for iOS and Android mobile applications, *Department of Computer Science and Engineering. University of Gothenburg*, 1-15.

Karakoç, M. M. ve Varol, A., 2016, National Distribution Project and Pardus Operating System, *Ulusal Dağıtım Projesi ve Pardus İşletim Sistemi.*, 11 (2), 25-34.

Karlı, G., 2014, Cross-platform mobile development, *Dokuz Eylül Üniversitesi*.

Khanna, R., Yusuf, S. ve Phan, H., 2017, Ionic: Hybrid Mobile App Development, Packt Publishing Ltd, p.

Kony, 2020, Kony Documentation, https://docs.kony.com/konylibrary/visualizer/viz_api_dev_guide/Default.htm#preface.htm:

Latif, M., Lakhrissi, Y., Nfaoui, E. H. ve Es-Sbai, N., 2016, Cross platform approach for mobile application development: A survey, *2016 International Conference on Information Technology for Organizations Development (IT4OD)*, 1-5.

Leeladevi, B., Hazarika, P. ve Nanyam, H. P. K., 2015, Transforming a website from desktop to mobile a cross platform viewpoint, *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 563-566.

Martinez, M. ve Lecomte, S., 2017, Towards the Quality Improvement of Cross-Platform Mobile Applications, *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 184-188.

Martinez, M., 2019, Two Datasets of Questions and Answers for Studying the Development of Cross-Platform Mobile Applications using Xamarin Framework, *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 162-173.

Mateus, B. G. ve Martinez, M., 2019, On the adoption, usage and evolution of Kotlin Features on Android development.

Mehlhorn, N., 2017, Mobile Cross-Platform Development from a Progressive Perspective.

Mikkonen, T., Taivalsaari, A. ve Terho, M., 2009, Lively for Qt: A platform for mobile web applications, *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, 1-8.

Mohan, T. V. V. A., 2019, Trend Prediction of GitHub using Time Series Analysis, *IEEE*.

Nayyar, A., 2016, Tools to Develop Cross-Platform Mobile Apps, *Open Source for You*, 4 (10), 77.

Novac, O. C., Novac, M., Gordan, C., Berczes, T. ve Bujdosó, G., 2017, Comparative study of Google Android, Apple iOS and Microsoft Windows Phone mobile operating systems, *2017 14th International Conference on Engineering of Modern Electric Systems (EMES)*, 154-159.

Öberg, L., 2016, Evaluation of Cross-Platform Mobile Development ToolsDevelopment of an Evaluation Framework.

Öner, M., 2019, Flutter Nedir ve Neden Flutter?, https://www.muratoner.net/flutter/flutter-nedir-ve-neden-flutter: [11.02.2020].

Palmieri, M., Singh, I. ve Cicchetti, A., 2012, Comparison of cross-platform mobile development tools, *2012 16th International Conference on Intelligence in Next Generation Networks*, 179-186.

Pazirandeh, A. ve Vorobyeva, E., 2015, Evaluation of cross-platform tools for mobile development.

Pinto, C. M. ve Coutinho, C., 2018, From Native to Cross-platform Hybrid Development, IEEE**:** 669-676.

Prendergast, C., 2017, Which Mobile OS is in your Future. SOTI. SOTI.

Ribeiro, A. ve da Silva, A. R., 2012, Survey on cross-platforms and languages for mobile apps, *2012 Eighth International Conference on the Quality of Information and Communications Technology*, 255-260.

Rieger, C. ve Majchrzak, T. A., 2019, Towards the definitive evaluation framework for cross-platform app development approaches, *Journal of Systems and Software*, 153, 175-199.

Rodger, R., 2011, Beginning mobile Application development in the Cloud, John Wiley & Sons, p.

Sánchez Blanco, A., 2016, Development of hybrid mobile apps: Using Ionic Framework.

Schmitt, N., Wesemeyer, T., Kissel, M., Horvat, D., Wünschel, M., Sattel, M., Lehmann, Y., Herbstreith, A. ve Gröschel, M., 2018, Evaluation of the Development Framework Jasonette, *Asian Journal of Computer and Information Systems (ISSN: 2321–5658)*, 6 (03).

Shah, K., Sinha, H. ve Mishra, P., 2019, Analysis of Cross-Platform Mobile App Development Tools, *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, 1-7.

Shah, S. ve Abd Rahman, K., 2013, Android Development Tools for Eclipse, *Birmingham*, Packt Publishing, p.

Shtika, L., 2017, Challenges and benefits of developing an open-source & full-stack" conference management" framework.

Smeets, R. ve Aerts, K., 2016, Trends in Web Based Cross Platform Technologies, *International Journal of Computer Science and Mobile Computing*, 5 (6), 190-199.

Smith, D., 2019, iOS Version Stats, https://david-smith.org/iosversionstats/: [04.01.2020].

Smutný, P., 2012, Mobile development tools and cross-platform solutions, *Proceedings of the 13th International Carpathian Control Conference (ICCC)*, 653-656.

Sohn, H.-j., Lee, M.-g., Seong, B.-m. ve Kim, J., 2015, Quality evaluation criteria based on open source mobile HTML5 UI framework for development of cross-platform, *Int J Soft Eng Appl*, 9 (6), 1-12.

Stackoverflow, 2020a, Stackoverflow Trends, https://insights.stackoverflow.com/trends?tags=react-native%2Cflutter%2Cxamarin%2Cnativescript%2Cionic-framework%2Cunity3d%2Ccocos2d-x%2Ctitanium%2Cphonegap-build%2Csencha-touch%2Cappcelerator%2Ccordova%2Ccoronasdk%2Cqt-creator: [03.03.2020].

Stackoverflow, 2020b, Stackoverflow Soru Etiketleri, https://stackoverflow.com/tags: [05.10.2020].

Stackoverflow, 2020c, Most Loved, Dreaded, and Wanted Other Frameworks, Libraries, and Tools, https://insights.stackoverflow.com/survey/2019#technology-_-most-loved-dreaded-and-wanted-other-frameworks-libraries-and-tools: [03.03.2020].

Statcounter, 2020, Mobile Operating System Market Share Worldwide, https://gs.statcounter.com/os-market-share/mobile/worldwide: [05.01.2020].

Titanium, Appcelerator Titanium, https://www.appcelerator.com/Titanium/: [23.02.2020].

Trends, G., 2020, Son 5 yılda en çok aranan çapraz platform mobil uygulama geliştirme araçları, https://trends.google.com.tr/trends/explore?date=today%205-y&q=%2Fg%2F11h03gfxy9,%2Fg%2F11f03_rzbg,%2Fg%2F11c57wr2yt,Xamarin,Unity3D: [02.11.2020].

Tunali, V., Zafer, S. J. C. B. U. F. o. T. D. o. S. E., Maltepe University: Faculty of Engineering ve Engineering, N. S. D. o. S., 2015, Comparison of popular cross-platform mobile application development tools.

Vijayan, J., 2018, Google Releases Beta Version of Flutter Mobile App Development Tool, QuinStreet, Inc.**:** 1-1.

Viswanathan, P., 2019, A mobile OS powers your smartphone, tablet, and smart wearables, https://www.lifewire.com/what-is-a-mobile-operating-system-2373340:

Wang, S., Mao, Z., Zeng, C., Gong, H., Li, S. ve Chen, B., 2010, A new method of virtual reality based on Unity3D, *2010 18th International Conference on Geoinformatics*, 1-5.

Williams, D., 2013, Corona SDK application design, Packt Publishing Ltd, p.

WMAracı, Flutter Nedir?, https://wmaraci.com/nedir/flutter: [11.02.2020].

Yosef, A., 2017, Success factors of mobile application development at selected innovation hub & co-working spaces in Addis Ababa, Ethiopia, *AAU*.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	MEHMET IŞITAN
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Samsun, 1994
Telefon	**:**	+90 541 766 13 71
Faks	**:**	
e-mail	**:**	mehmetisitann@gmail.com

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	**Merkez Teknik Lise, Samsun**	**2012**
Üniversite	:	**Selçuk Üniversitesi, Konya**	**2017**
Yüksek Lisans	:	**Selçuk Üniversitesi, Konya**	**-**
Doktora	:		

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
3 Yıl	Yeşilmavi Yazılım	Yazılım Geliştirme Uzmanı
6 Ay	Dinamik Tütün Mamülleri A.Ş.	Yazılım Departmanı Yöneticisi
1 Yıl 7 Ay	Reform Klima Havalandırma A. Ş.	Yazılım Departmanı Yöneticisi
2 Yıl	Bks Bilgisayar	Yazılım Departmanı Yöneticisi
7 Ay	Mikro Ajans	Web Programcısı

UZMANLIK ALANI

.Net Core, Asp.Net MVC5, SQL, Javascript, React Native, Ionic Framework

YABANCI DİLLER

İngilizce