T.C.

DÜZCE ÜNİVERSİTESİ

SOSYAL BİLİMLER ENSTİTÜSÜ

YÖNETİM BİLİŞİM SİSTEMLERİ ANABİLİM DALI

ÇAPRAZ PLATFORM UYGULAMA ÇERÇEVELERİNİN MOBİL CİHAZLAR ÜZERİNDEKİ PERFORMANS ETKİLERİ

YÜKSEK LİSANS TEZİ

Hakkı Gürkan Tüçel

Düzce

Haziran, 2021

T.C.

DÜZCE ÜNİVERSİTESİ

SOSYAL BİLİMLER ENSTİTÜSÜ

YÖNETİM BİLİŞİM SİSTEMLERİ ANABİLİM DALI

ÇAPRAZ PLATFORM UYGULAMA ÇERÇEVELERİNİN MOBİL CİHAZLAR ÜZERİNDEKİ PERFORMANS ETKİLERİ

YÜKSEK LİSANS TEZİ

Hakkı Gürkan Tüçel

Danışman: Dr. Öğr. Üyesi Remzi BAŞAR

Düzce

Haziran, 2021

Hakkı Gürkan Tüçel
Düzce Üniversitesi, SBE
Yüksek Lisans Tezi
Haziran, 2021

ÇAPRAZ PLATFORM UYGULAMA ÇERÇEVELERİNİN MOBİL CİHAZLAR ÜZERİNDEKİ PERFORMANS ETKİLERİ

Sosyal Bilimler Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından ...Yönetim Bilişim Sistemleri....Anabilim Dalında oy birliği / ~~oy çokluğu~~ ile YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Başkan Prof. Dr. Alper ERTÜRK	……
Üye Dr. Öğr. Üy. Fatih KAYAALP	……
Üye Dr. Öğr. Üy. Remzi BAŞAR	……

Onay

Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

.../../20..

(İmza Yeri)

Akademik Unvanı, Adı-Soyadı

Enstitü Müdürü

ÖNSÖZ

Teknolojinin gelişmesiyle artan akıllı cihazlar ve onların kullanım oranları yazılım geliştiricilerinin önemini de gereksinimlerini de artırmıştır. Değişen ve sürekli gelişen teknolojideki alanlara göre programlama dillerinin çeşitliliği ve popülerliliği değişiklik göstermektedir. Hızlı ürün geliştirme, dilin özelliklerini etkin kullanma ve kolay algılama kütüphane ve çerçeveleri ortaya çıkarmıştır.

Kütüphane ve çerçevelerinin fazlalığı tercih konusunda yazılım geliştiricinin veya ekibinin kararsızlığına neden olabilir. Topluluk desteği, performans ve tüketim değerleri gibi avantaj ve dezavantajlar üzerinden tercih yapılmaya çalışılır.

Bu tez çalışmasında yazılım geliştirme ve ekibine mobil alanında çapraz platform çerçevelerinin tüketim değerlerini ve farklarını karşılaştırarak göstermek ve tercih edilmelerine bir fikir sunmak temel amaç edinilmiştir.

Yüksek lisans eğitimim boyunca bana rehberlik eden değerli tez danışmanım Sayın Dr. Öğr. Üyesi Remzi BAŞAR'a teşekkürlerimi sunarım. Yüksek lisans yapmam için beni teşvik eden ve desteğini esirgemeyen canım annem ve babama minnettarım. Ayrıca yüksek lisans programındaki tüm hocalarıma ayrı ayrı teşekkür ederim.

Hakkı Gürkan TÜÇEL

ÖZET

ÇAPRAZ PLATFORM UYGULAMA ÇERÇEVELERİNİN MOBİL CİHAZLAR ÜZERİNDEKİ PERFORMANS ETKİLERİ

TÜÇEL, Hakkı Gürkan

Yüksek Lisans, Yönetim Bilişim Sistemleri Anabilim Dalı

Tez Danışmanı : Dr. Öğr. Üyesi Remzi BAŞAR

Haziran 2021, 81 Sayfa

Yazılım dünyasında her bir ekosistemin kendine ait bir işletim sistemi vardır ve bu işletim sistemine uygulama geliştirebilmeniz için onun derleyicisine (Windows: C#, Android: Kotlin, iOS: Swift) uygun programlama dili ile yazmanız gerekir. Bu gereklilik zaman, iş gücü ve maliyet gibi zorluklar ortaya çıkardığı için çapraz platform uygulama geliştirme araçları ortaya çıkmıştır. Hem bu zorluklar, hem de mobil kullanımın artması çapraz platform çerçevelerinin de çeşitliğinin artmasını sağlamıştır.

Çapraz platform uygulama geliştiriciliği ile bir programlama dili ile birden çok ekosisteme (mobil, pc, saat, tv) uygulama geliştirilebilir. Uygulamaların geliştirilmesine yardımcı olan çerçevelerin uygulama boyutunun ve geçici bellek kullanımının azlığı gibi birbirinden üstün olduğu yönleri vardır.

Bu çalışmada çapraz platform mobil uygulama geliştirme için yayınlanan frameworkler (Flutter, React Native, Xamarin) ile Todo List (Yapılacaklar Listesi) mobil uygulaması geliştirildi ve CPU, RAM, enerji, veri (internet) kullanımı üzerinden ara yüz (UI) testine tabii tutuldu. Frameworklerin testlerden aldıkları sonuçlar tablolar ile yansıtılıp, karşılaştırılmıştır. Elde edilecek veriler ile işletmelerin

yazılım ekiplerinin veya serbest mobil uygulama geliştiricilerinin uygulama geliştirirken hangi çerçeveyi tercih etmesi gerektiğine katkı sağlaması hedeflenmiştir.

Anahtar Kelimeler: çapraz platform, mobil geliştirme, mobil uygulama

ABSTRACT

PERFORMANCE EFFECTS OF CROSS PLATFORM APPLICATION FRAMES ON MOBILE DEVICES

TÜÇEL, Hakkı Gürkan

Master Degree, Department of Management Information Systems

Thesis Advisor : Dr. Öğr. Üyesi Remzi BAŞAR

June 2021, 81 Page

In the software world, each ecosystem has its own operating system, and in order to develop applications for this operating system, you need to write it with the appropriate programming language for its compiler (Windows: C #, Android: Kotlin, iOS: Swift). As this requirement creates difficulties such as time, labor and cost, cross-platform application development tools have emerged. Both these challenges and the increase in mobile usage have increased the variety of cross-platform frameworks. With cross-platform application development, applications can be developed for multiple ecosystems (mobile, pc, watch, tv) with a programming language. Frameworks that help develop applications have aspects that are superior to each other, such as the size of the application and the lack of volatile memory.

In this study, the To-Do List mobile application was developed with frameworks (Flutter, React Native, Xamarin) published for cross-platform mobile application development and subjected to interface (UI) testing over CPU, RAM, energy, data (internet) usage. The results obtained from the tests of the frameworks are reflected and compared with tables. With the data to be obtained, it is aimed to contribute to which framework software teams or freelance mobile application developers should prefer while developing applications.

Keywords: cross platform, mobile development, mobile application

İÇİNDEKİLER

ÖNSÖZ ..... **İİ**

ÖZET ..... **İİİ**

ABSTRACT ..... **V**

İÇİNDEKİLER ..... **Vİ**

TABLOLAR LİSTESİ ..... **Vİİİ**

ŞEKİLLER LİSTESİ ..... **İX**

KISALTMALAR ..... **Xİİ**

BİRİNCİ BÖLÜM **1**

1. GİRİŞ 1

1.1. Problem 1

1.2. Araştırmanın Amacı 2

1.3. Araştırmanın Önemi 3

1.4. Araştırmanın Sayıltıları 3

1.5. Araştırmanın Sınırlılıkları 4

1.6. Tanımlar 4

İKİNCİ BÖLÜM **5**

2. LİTERATÜR 5

2.1. Çapraz Platform Mobil Uygulama Geliştirme 5

2.1.1. Flutter 6

2.1.2. React Native 7

2.1.3. Xamarin 9

2.1.4. Cordova 10

2.1.5. Ionic Framework 11

2.2. Test Uygulamasında Kullanılan Teknolojiler 11

2.2.1. Rest Api 11

2.2.2. Paket ve Eklentiler 12

2.2.2.2. Shared Preferences 13
2.3. Uygulama Tasarımı 14
2.4. Arka Uç Tasarımı 16

ÜÇÜNCÜ BÖLÜM 18

3. YÖNTEM 18
3.1. Araştırmanın Modeli 18
3.2. Araştırma Grubu 18
3.3. Veri Toplama Araçları 20
3.3.1. Android Studio Profiler 20
3.3.2. Perfetto UI 20
3.3.3. Instruments 20
3.4. Verilerin Toplanması 21
3.5. Verilerin Analizi 22

DÖRDÜNCÜ BÖLÜM 23

4. BULGULAR VE YORUM 23
4.1. Android İşletim Sistemli Cihazların Tüketim Bulguları 23
4.1.1. Cpu Kullanımı 23
4.1.2. Bellek Kullanımı 28
4.1.3. Enerji Kullanımı 33
4.1.4. Ağ Kullanımı 38
4.1.5. Uygulama Boyutu 43
4.2. İos İşletim Sistemli Cihazların Tüketim Bulguları 44
4.2.1. CPU Kullanımı 44
4.2.2. Bellek Kullanımı 52
4.2.3. Uygulama Boyutu 56

BEŞİNCİ BÖLÜM 57

5. SONUÇLAR VE ÖNERİLER 57
5.1. Sonuçlar 57
5.2. Öneriler 60

KAYNAKÇA 62

TABLOLAR LİSTESİ

Tablo 3.1. Android İşletim Sistemli Test Cihazları 19
Tablo 3.2. iOS İşletim Sistemi Test Cihazları..... 19
Tablo 3.3. Uygulama Test Aşamaları 21
Tablo 4.1. Frameworkler İle Geliştirilen Android Uygulamaların Boyutları 43
Tablo 4.2. Frameworkler İle Geliştirilen İos Uygulamaların Boyutları..... 57

ŞEKİLLER LİSTESİ

Şekil 2.1. Flutter Çalışma Mimarisi 7
Şekil 2.2. React Native Çalışma Mimarisi 8
Şekil 2.3. Xamarin Çalışma Mimarisi 9
Şekil 2.4. Cordova Çalışma Mantığı 10
Şekil 2.5. JSON Çıktısı 12
Şekil 2.6. İos Todo List App Ekran Görüntüleri 14
Şekil 2.7. Android Todo List App Ekran Görüntüleri 15
Şekil 2.8. Uygulama Mimarisi 16
Şekil 2.9. Get Sorgu Sonucu Çıktısı Json Formatı 17
Şekil 2.10. Post Sorgu Sonucu Çıktısı Json Formatı 17
Şekil 4.1. Flutter Çerçevesi Samsung Galaxy A50 Cpu Kullanım Oranları 24
Şekil 4.2. Flutter Çerçevesi Xiaomi Mi A3 Cpu Kullanım Oranları 24
Şekil 4.3. Flutter İçin Samsung Galaxy A51 Cpu Kullanım Oranları 25
Şekil 4.4. React Native İçin Samsung Galaxy A50 Cpu Kullanım Oranları 25
Şekil 4.5. React Native İçin Xiaomi Mi A3 Cpu Kullanım Oranları 26
Şekil 4.0.6. React Native İçin Samsung Galaxy A51 Cpu Kullanım Oranları 26
Şekil 4.7. Xamarin İçin Samsung Galaxy A50 Cpu Kullanım Oranları 27
Şekil 4.8. Xamarin İçin Xiaomi Mi A3 Cpu Kullanım Oranları 27
Şekil 4.9. Xamarin İçin Samsung Galaxy A51 Cpu Kullanım Oranları 28
Şekil 4.10. Flutter Çerçevesi Samsung Galaxy A50 Bellek Kullanım Oranları 29
Şekil 4.11. Flutter Çerçevesi Xiaomi Mi A3 Bellek Kullanım Oranları 29
Şekil 4.12. Flutter Çerçevesi Samsung Galaxy A51 Bellek Kullanım Oranları 29
Şekil 4.13. React Native Samsung Galaxy A50 Bellek Kullanım Oranları 30
Şekil 4.14. React Native Xiaomi Mi A3 Bellek Kullanım Oranları 31
Şekil 4.15. React Native Samsung Galaxy A51 Bellek Kullanım Oranları 31
Şekil 4.16. Xamarin Samsung Galaxy A50 Bellek Kullanım Oranları 31

Şekil 4.17. Xamarin Xiaomi Mi A3 Bellek Kullanım Oranları 32
Şekil 4.18. Xamarin Samsung Galaxy A51 Bellek Kullanım Oranları 32
Şekil 4.19. Flutter Çerçevesi Samsung Galaxy A50 Enerji Kullanım Oranları..... 34
Şekil 4.20. Flutter Çerçevesi Xiaomi Mi A3 Enerji Kullanım Oranları 35
Şekil 4.21. Flutter Çerçevesi Samsung Galaxy A51 Enerji Kullanım Oranları..... 35
Şekil 4.22. React Native Samsung Galaxy A50 Enerji Kullanım Oranları 36
Şekil 4.23. React Native Xiaomi Mi A3 Enerji Kullanım Oranları..... 36
Şekil 4.24. React Native Samsung Galaxy A51 Enerji Kullanım Oranları 37
Şekil 4.25. Xamarin Samsung Galaxy A50 Enerji Kullanım Oranları 37
Şekil 4.26. Xamarin Xiaomi Mi A3 Enerji Kullanım Oranları..... 38
Şekil 4.27. Xamarin Samsung Galaxy A51 Enerji Kullanım Oranları 38
Şekil 4.28. Flutter Samsung Galaxy A50 Ağ Kullanım Oranları 39
Şekil 4.29. Flutter Xiaomi Mi A3 Ağ Kullanım Oranları..... 39
Şekil 4.30. Flutter Samsung Galaxy A51 Ağ Kullanım Oranları 40
Şekil 4.31. React Native Samsung Galaxy A50 Ağ Kullanım Oranları 40
Şekil 4.32. React Native Xiaomi Mi A3 Ağ Kullanım Oranları..... 41
Şekil 4.33. React Native Samsung Galaxy A51 Ağ Kullanım Oranları 41
Şekil 4.34. Xamarin Samsung Galaxy A50 Ağ Kullanım Oranları..... 41
Şekil 4.35. Xamarin Xiaomi Mi A3 Ağ Kullanım Oranları 42
Şekil 4.36. Xamarin Samsung Galaxy A51 Ağ Kullanım Oranları..... 43
Şekil 4.37. Flutter iPhone 8 Cpu Kullanım Oranları 44
Şekil 4.38. Flutter iPhone 8 Cpu Kullanım Oranları Main Thread..... 45
Şekil 4.39. Flutter iPhone SE Cpu Kullanım Oranları..... 45
Şekil 4.40. Flutter iPhone SE Cpu Kullanım Oranları Main Thread 46
Şekil 4.41. Flutter iPhone 8 Plus Cpu Kullanım Oranları..... 46
Şekil 4.42. Flutter iPhone 8 Plus Cpu Kullanım Oranları Main Thread..... 47
Şekil 4.43. React Native iPhone 8 Cpu Kullanım Oranları 47
Şekil 4.44. React Native iPhone 8 Cpu Kullanım Oranları Main Thread..... 47
Şekil 4.45. React Native iPhone SE Cpu Kullanım Oranları..... 48
Şekil 4.46. React Native iPhone SE Cpu Kullanım Oranları Main Thread..... 48
Şekil 4.47. React Native iPhone 8 Plus Cpu Kullanım Oranları Main Thread..... 49
Şekil 4.48. React Native iPhone 8 Plus Cpu Kullanım Oranları 49

Şekil 4.49. Xamarin iPhone 8 Cpu Kullanım Oranları Main Thread 49
Şekil 4.50. Xamarin iPhone 8 Kullanım Oranları..... 50
Şekil 4.51. Xamarin iPhone SE Cpu Kullanım Oranları Main Thread..... 50
Şekil 4.52. Xamarin iPhone SE Kullanım Oranları 50
Şekil 4.53. Xamarin iPhone 8 Plus Cpu Kullanım Oranları Main Thread 51
Şekil 4.54. Xamarin iPhone 8 Plus Kullanım Oranları..... 51
Şekil 4.55. Flutter iPhone 8 Bellek Kullanım Oranları..... 52
Şekil 4.56. iPhone SE Bellek Kullanım Oranları..... 53
Şekil 4.57. Flutter iPhone 8 Plus Bellek Kullanım Oranları..... 53
Şekil 4.58. React Native iPhone 8 Bellek Kullanım Oranları..... 54
Şekil 4.59. React Native iPhone SE Bellek Kullanım Oranları..... 54
Şekil 4.60. React Native iPhone 8 Plus Bellek Kullanım Oranları..... 55
Şekil 4.61. Xamarin iPhone 8 Bellek Kullanım Oranları 55
Şekil 4.62. Xamarin iPhone SE Bellek Kullanım Oranları..... 55
Şekil 4.63. Xamarin iPhone 8 Plus Bellek Kullanım Oranları 56

KISALTMALAR

API: Application Programming Interface

CPU: Central Process Unit

DOM: Document Object Model

JSON: JavaScript Object Notation

NPM: Node Package Manager

SDK: Software Development Kit

UI: User Interface

BİRİNCİ BÖLÜM

1. GİRİŞ

Bu bölümde; problem, araştırmanın amacı ve önemi, sınırlılıklar ve tanımlar yer almaktadır.

1.1.Problem

Yazılım alanındaki dillerin ve geliştiricilerin kullanım kolaylığı sağlaması için çıkarılan kütüphane ve çerçevelerin sayısı hergün artmaktadır. Teknolojinin erişilmesi ve kullanıcı sayısının artmasıyla birlikte akıllı cihazların yaygınlaşması ve bu cihazların birbirleriyle haberleşmesi için geliştirilen programlama dillerin çeşitliliği artması sonucu geliştirici tarafından bilinmesi istenilenin gereksinimlerde bu orantıyla birlikte artmıştır.

Çapraz platform çerçeveleri, akıllı cihazların sayılarının artması ve bu cihazların çalışması için gerekli işletim sistemlerine uygulama geliştirilmesi için ortaya çıkmıştır. Çerçevelerin stabil versiyonlarının eskiye dayanmaması, yeni olması sonucunda ortaya çıkarılan ürün sayısının azlığı ve zaten varolan şekilde native olarak geliştirme yapılan ortamlara göre tüketim değerlerinin fazla olması uygun teknolojinin tercih edilmesi konusunda zorluklar çıkarmaktadır.

Bu bölümde bu konu ile ilgili daha önceki çalışmalara yer verilmiştir.

Isıtan ve Koklu (2020) yaptığı çalışmada React Native, Flutter, Nativescript ve Xamarin çapraz platform uygulama geliştirme çerçevelerini karşılaştırmış, yazılan uygulamalar Android emulatörü üzerinden test edilmiştir. Çalışmada Flutter ve React Native'in daha başarılı sonuçlar verdiği görülmüştür.

Inupakutika vd. (2020) sağlık uygulamaları mHealth üzerinden yerel Android ve çapraz platformlar arası çerçevelerin geliştirilmesi gösterilmiş, uygulama değerlendirmesi yapılmıştır.

Dorfer vd. (2020) yerel Android ile React Native işlemci, bellek ve enerji üzerinden gerçek Android cihazlar üzerinden karşılaştırılmıştır. Çalışmada çapraz platform çervecesi React Native'in yerel Android uygulamasına göre daha fazla işlem ve enerji tükketiği görülmüştür.

Huber ve Demetz (2019) yerel Android ile React Native ve Ionic Cordova çapraz platform çerçeveleri kaydırılabilir liste görünümü üzerinden CPU ve bellek tüketimi ölçülmüştür. Bu iki çapraz platform çerçevelerinin yerel uygulamaya kıyasla CPU kullanımında yaklaşık iki kat, bellek tüketiminde de daha yüksek olduğu sonucuna ulaşılmıştır.

Ciman ve Gaggi (2016) çapraz platform çerçevelerinin enerji tüketiminden nasıl etkilendilerini incelemiş ve gerçek mobil cihazlar üzerinden karşılaştırmıştır. Çalışmada enerji tüketiminde her zaman bir artış olduğunu ve en çok enerji tüketiminin cihazın sensörleri kullandığında yaşandığı gözlenmiştir.

Öberg (2016) Xamarin, Ionic Cordova çapraz platform çerçeveleri ve Native Android ile geliştirilen uygulama CPU ve RAM kullanımları üzerinden geliştirici perspektifleri üzerinden değerlendirilmiştir. Xamarin ile Ionic Cordova kullanıcı perspektifleri üzerinden kullanıcı arayüzü, başlatma zamanı ve geliştirme hızı kriterlerine göre karşılaştırmalar yapılmıştır. Çalışmada çapraz platform çerçevelerinden Xamarin'in Ionic Cordovaya göre daha yüksek puana ulaştığı gözlenmiştir.

1.2.Araştırmanın Amacı

Bu araştırmanın amacı; Çapraz platform çerçevelerinin işlemci, bellek, ağ ve güç kullanımları kriterlerine göre en uygun çerçeveyi seçmeye yardımcı olmaktır. Bu amacı gerçekleştirmek için aşağıdaki sorulara yanıt aranacaktır:

1. Popüler çapraz platform çerçevelerinin tüketim değerleri nasıldır?
2. Çapraz platform çerçeveleri arasında performans farkları var mıdır?

1.3.Araştırmanın Önemi

Mobil cihazların akıllı hale gelmesiyle yaygınlaşan mobil işletim sistemleri günümüzde masaüstü işletim sistemleri kadar kullanışlı hale gelmiştir. Android ve iOS mobil işletim sistemlerinin hakim olduğu pazar payında her platformda olduğu gibi uygulamaların farklı geliştirilme ortamları vardır (Stat Counter, 2020). Bu ortam geliştiriciye her platform için farklı programlama dillerini ve arayüz tasarlama desenlerini öğrenme ve kullanma gerekliliği getirir. Bunun yanında zaman ve maliyet gibi etkenlerde yer alır.

Çapraz platform mobil uygulama geliştirme çerçeveleri (framework) ile bir programlama dili ile birden çok ekosisteme (mobil, desktop, watch, tv) uygulama geliştirilebilir. Zaman, iş gücü, maliyet gibi faydalar sağlar. Bu faydaların yanında CPU, pil gücü, bellek kullanımı ve uygulama boyutunun fazlalığı gibi zararları da yanında getirir.

Bu çalışmada mobil uygulama geliştirmek için Cross Platformlar için yayınlanan frameworkler (Flutter, React Native, Xamarin) ile ayrı ayrı Todo List (Yapılacaklar Listesi) amobil uygulaması geliştirildi ve CPU, RAM, Enerji, Veri (İnternet) kullanımı üzerinden ara yüz (UI) testine tabii tutuldu. Frameworklerin testlerden aldıkları sonuçlar tablolar ile yansıtılıp, karşılaştırılmıştır.

Her çerçevenin (framework) uygulama boyutunun ve geçici bellek (RAM) kullanımının azlığı gibi birbirinden üstün olduğu yönleri vardır. Bu çalışma üzerinden elde edilecek veriler sayesinde işletmelerin yazılım ekiplerinin veya serbest mobil uygulama geliştiricilerinin uygulama geliştirirken hangi çerçeveyi tercih etmesi gerektiğine katkı sağlaması amaçlanmaktadır.

1.4.Araştırmanın Sayıltıları

- Çalışmada kullanılan test programlarının güvenilir olduğu kabul edilmektedir.
- Araştırmaya katkı sağlayan test araçlarının sonuçları doğru gösterdiği varsayılmaktadır.

1.5.Araştırmanın Sınırlılıkları

Bu araştırma, 3 adet Android, 3 adet iOS test cihazları ve Mart 2021 yılı itibariyle çapraz platform çerçevelerinden popüler olan ilk üç çerçeve Flutter, React Native ve Xamarin ile sınırlıdır.

1.6.Tanımlar

Çerçeve: Geliştiricinin metotlar ve arayüzler ile işlerini kolaylaştırmak için programlama dillerinin işlevselliğini sağlayan evrensel, yeniden kullanılabilir bir iskelet görevi gören yazılım ortamıdır (Riehle, 2000 : 9).

API: Yazılımın başka yazımlarla tanımlanmış işlevlerini kullanabilmesi için fonksiyon ve sınıf kümesi görevi gören uygulama programlama arayüzüdür (Ardış ve Göktürk, 2009: 91).

İKİNCİ BÖLÜM

2. LİTERATÜR

2.1.Çapraz Platform Mobil Uygulama Geliştirme

Yerel uygulamalar, platform geliştirileri tarafından sağlanan yazılım geliştirme kitleri (SDK) ile platformlarına özgü olarak geliştirilirler. Yazılım geliştirme kitlerini geliştirmek için tercih ettikleri programlama dilleri vardır. Örneğin, Android işletim sistemi için Java veya Kotlin, iOS için Objective-C veya Swift programlama dilleridir. Geliştirme, her platform için ayrı ayrı geliştirilir.

Çapraz platform uygulama geliştirmenin amacı tek bir programlama dili ile birden çok platforma geliştirme yapabilmektir. Birden fazla platforma uygulama geliştirmenin çeşitli yolları vardır.

Aşamalı Web Uygulamaları: İşaretleme dilleri HTML, CSS ve web tarayıcılarında kullanılan dinamik programlama dili Javascript ile geliştirilen bir web uygulamasıdır. Akıllı telefon ve tablet gibi akıllı cihazların ekran çözünürlüklerine göre duyarlı web tasarımı (responsive) yaklaşımına göre optimize edilir. Platformun uygulama web tarayıcısı (browser) üzerinden çalışır. Cihazın ve işletim sisteminin özelliklerine erişim kısıtlıdır. Konum ve veri depolama gibi özelliklere erişmekle sınırlıdır. Web tarayıcısı üzerinden çalıştığı için internet bağlantısı olmadan çevrimdışı kullanılamaz.

Hibrit Uygulamalar: Web geliştirme teknolojilerinin mobil ve web yaklaşımlarının bir arada kullanılarak uygulama geliştirilmesidir. HTML,CSS ve Javascript kullanılarak geliştirilen uygulama WebView kullanılarak yeniden işlenerek yansıtılır. Javascript API'ları yardımıyla ilgili alan tarayıcı motoru üzerinden Bluetooth, GPS ve ağ bağlantıları aracılığıyla çağrılarak kullanılabilir. WebView bileşeni sayesinde yansıtılan ve uygulama görüntüsüne bürünen uygulama mağazadan indirilirek

telefona yüklenir ve çevrimdışı kullanılabilir. İonic, Cordova, PhoneGap bu yaklaşımları kullanan ve kullanıcılarına geliştirme olanağı veren çerçevelerdir (Hansen vd., 2020 : 3001).

Çapraz Uygulamalar: Uygulama geliştirilmesi hedeflenen tüm platformların temel işlevlerine erişebilmek için ortak bir API kullanılır. Çalışma katmanında aplikasyon geliştirme kodu çerçevenin tercih ettiği Javascript dili ile React Native ve Nativescript, C# ile Xamarin, Dart ile Flutter gibi programlama dilleri ile yazılır. Yerel olarak işlenmiş kullanıcı arabirimlerinin birleşimlerini kullanılırlar (Hansen vd., 2020 : 3002).

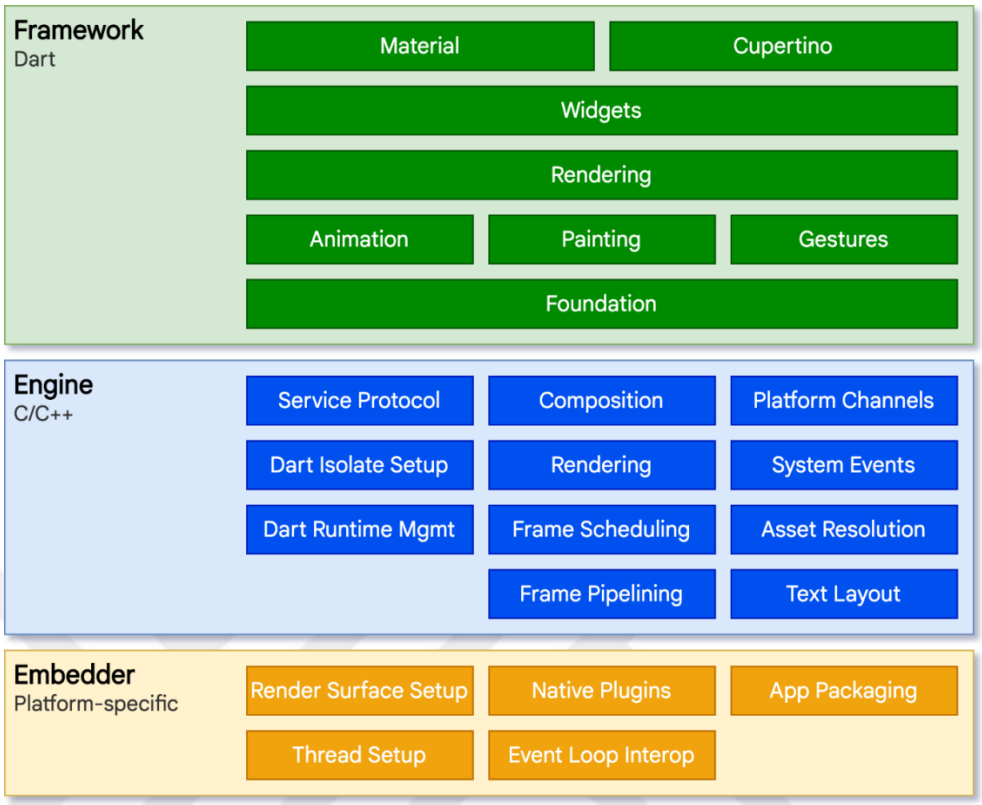
2.1.1. Flutter

Flutter, Google tarafından geliştirilen tek bir kod çıktısından derlenmiş uygulamalar oluşturan ücretsiz ve açık kaynak kodlu bir arayüz araç setidir. 2018 tarihinde ilk sürüm yayınlanmış olup Dart programlama dilini kullanır. Mobil, web ve masaüstü için uygulamalar oluşturmayı destekler (Flutter, 2020).

2.1.1.1.Flutter Çalışma Mimarisi

Flutter, Dart programlama dili ile yazılan bir frameworktur. Framework katmanında Material (Android) ve Cupertino (iOS) tasarım dilleri, Buton, resim, menüler gibi widgetlar, animasyonlar bulunur. Merkezinde C ++ ile yazılmış Flutter motoru (engine) yer alır. Motor katmanı, frameworkteki yapılan değişikleri çizip sonucu göstermekle sorumludur. Dart kodu derleme çıktısı ve çalışması, grafikler, metin düzeni, dosya, ağ, eklentilerin uygulanmasını sağlar. Embedder katmanı ise Flutter kodunu modül olarak mevcut uygulamaya entegre ederek uygulamanın tüm içeriğini oluşturur. Bu katman platformların destekleri dillere göre İos için Objective-C, Android için Java ile yazılmıştır (Flutter, 2020). Şekil 2.1.'de görüldüğü gibi katmanlar yer alır.

Şekil 2.1. Flutter Çalışma Mimarisi



(Kaynak: Flutter, 2020)

2.1.2. React Native

React Native, Android ve iOS uygulamaları oluşturmak için Facebook tarafından geliştirilen açık kaynak kodlu bir çerçevedir. 2015 yılında ilk sürümü yayınlandı. Javascript programlama dilini kullanarak geliştirme yapılır. Mobil, web ve masaüstü için uygulamalar oluşturmayı destekler (React Native, 2020).

2.1.2.1.React Native Çalışma Mimarisi

React Native, Javascript kütüphanesi ReactJS ile geliştirilen bir frameworktur. İşletim sistemi platformlarının kaynaklarına ulaşarak Javascript ile arayüz güncelleştirmelerini ve kullanıcı davranışlarını tanımlar (React Native, 2021).

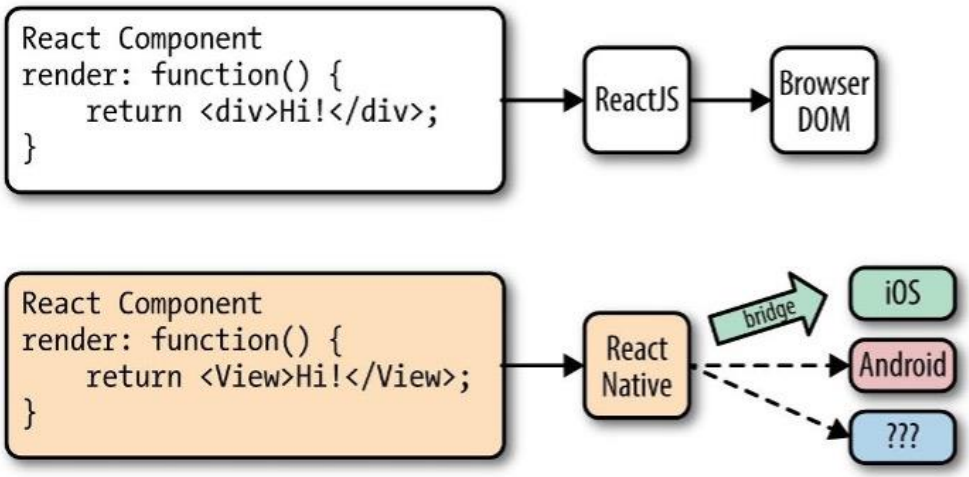
Node.js, Javascript'in sadece istemci tarafında çalışan bir dil olmaktan çıkararak sunucu tarafında da çalışan bir dil haline getiren bir ortamdır. İş mantığının eş zamanlı yürütülmesini sağlamak için Google'ın Chrome V8 Engine adını verdiği

ortam ile Linux, Windows, macOS işletim sistemlerinde çalışır. Javascript kaynak kodunu derler ve yürütür. Node modül sistemiyle uyumlu istemci taraflı Javascript için birçok kitaplık ve çerçeve bulunur. Düğüm paketi yönetici npm ile kitaplıklar, eklentiler ve bağımlılıklar yüklenir (Tilkov ve Vinoski, 2010: 81).

2020 yılı için npm paket yüklenme sayısı 642,683,350'dir (NpmStat, 2021). Test için seçilen kütüphanelerden bir tanesi olan React Native ve yardımcı paketleri npm düğüm paketi yöneticisi ile kurulmuştur.

Node uyumlu modül sistemi ile çalışan Reactjs, Web geliştirme yaparken React kütüphanesindeki geliştiricinin görünümdeki nesnelerin görünmesi için yaptığı ayar, yazdığı kod ile uygulamadaki sayfada oluşması ile geçen iş arasındaki katman görevini Sanal DOM gerçekleştirir. Bu işlemi Google Chrome, Mozilla Firefox gibi kullanıcının tarayıcısından gerçekleştirerek kullanır (Eisenman, 2016: 27).

Şekil 2.2. React Native Çalışma Mimarisi



(Kaynak: Eisenman, 2016: 27)

React Native ise tarayıcının DOM'unda oluşup çağırmak yerine iOS için Objective-C veya Swift API'larını, Androd için Java veya Kotlin API'larını oluşturur. React ana platformunun arayüz öğeleriyle etkileşimi ile işaretlemeyi döndürür ve platform üzerinde görünümü oluşturur. Görünümü oluştururken kendi arayüz bileşenlerini Android ve iOS görünümlerine çevirir. <Text> arayüz bileşeni, Android için <TextView>, iOS için <UITextView>, Web için <p> etiketlerine denk

gelir. Resim, kullanıcı metin girişleri gibi tüm arayüz bileşenleri için görünümler mevcuttur (Eisenman B., 2016: 28-30).

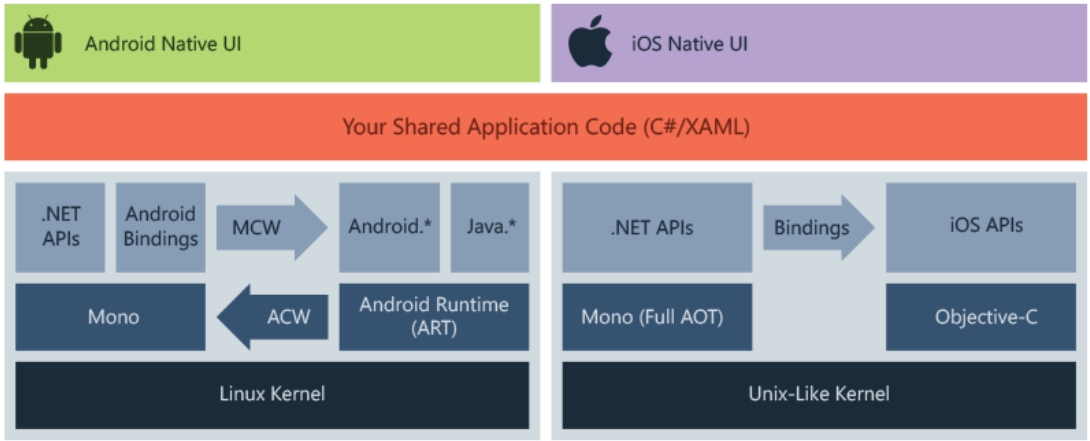
2.1.3. Xamarin

Xamarin, C# ve .NET kütüphanesi ile Microsoft tarafından geliştirilen açık kaynak kodlu uygulama platformudur. 2011 yılında ilk sürümü yayınlandı. Mobil, web, masaüstü ve tvOS gibi diğer akıllı cihazlar için uygulama geliştirilebilir (Microsoft .NET, 2021).

2.1.3.1.Xamarin Çalışma Mimarisi

Xamarin.Android ve Xamarin.iOS uygulamaları mono yürütme ortamında çalışır. Android uygulamaları için yürütme ortamı Android Runtime sanal makinesiyle birlikte Linux çekirdeğinin üzerinde çalışır. Bu alanlara erişi .NET sınıf kitaplıkları sayesinde erişim sağlar. Şekil 3.'de yer aldığı gibi Android sisteminin çalışması için gerekli Android.* ve Java.* alanları Java API'leri aracılığı sunulur (Microsoft Docs, 2021).

Şekil 2.3. Xamarin Çalışma Mimarisi



(Kaynak: Microsoft Docs, 2021)

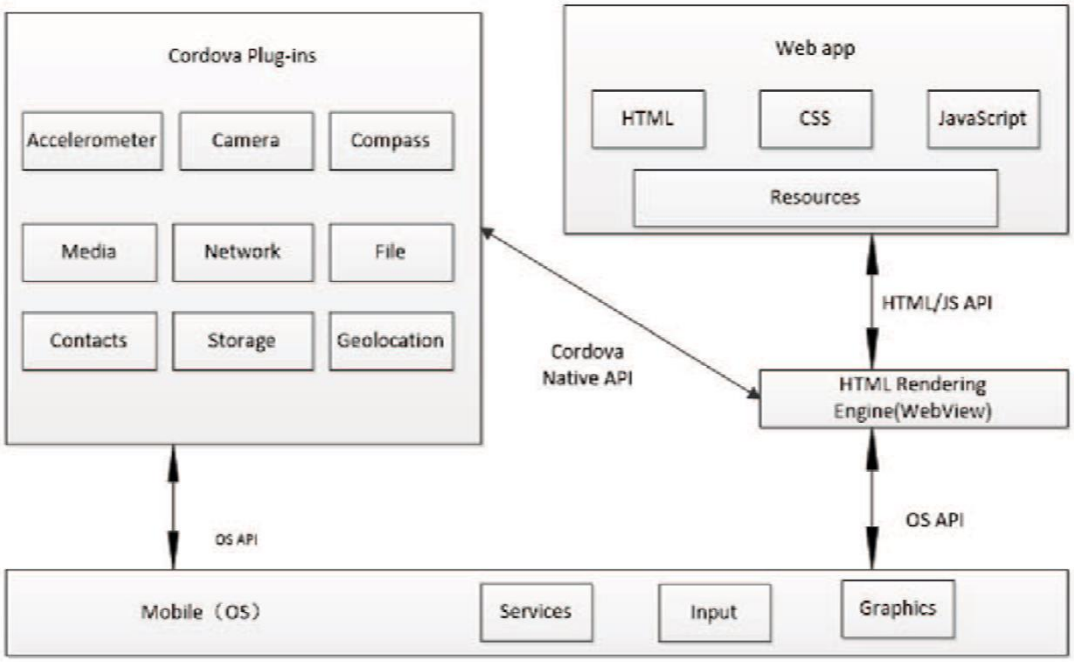
İos uygulamaları için ise yine uygulamalar mono yürütme ortamında çalışır ve C# kodunu ARM derleme diline derlemek için AOT derlemeyi kullanılır. Derlenen kod, Objective-C dili ile birlikte UNIX çekirdeği üzerinde çalışır. C# ve Objective-C, Bindings olarak adlandırılan seçiciler ile iletişim kurar (Microsoft Docs, 2021).

Arayüz, C# kod arkasında bulunan XAML kullanıcı arabirimi dili ile her iki platformda da aynı arayüz görünümünü (UI) veren UI çerçevesi Xamarin.Forms ile geliştirilir. Dosya sistemi, cihaz bilgileri gibi yerel cihaz özelliklerine de API'lar ile erişen Xamarin.Essentials kitaplığı vardır (Microsoft, 2021).

2.1.4. Cordova

Cordova, HTML, CSS ve Javascript web teknolojilerini kullanarak mobil görünümlü uygulamalar tasarlamayı amaçlar. Yerleşik tarayıcı üzerinde çalışan web sayfası görünümü, webview adı verilen Cordova çekirdeği sayesinde mobil işletim sistemli telefonlarda mobil uygulama görünümü olarak yansıtılır. Webview bileşeni HTML sayfasındaki görünümü yansıtır. Kendi kütüphanesinde yer alan Cordova Plug-ins adı verilen eklentiler ile işletim sisteminin API'ları ile iletişime geçerek Şekil 2.4.'deki gibi mobil cihazın kamera, dosyalar, galeri gibi depolama alanlarına ulaşır (Qing vd., 2015 : 451-453).

Şekil 2.4. Cordova Çalışma Mantığı



(Kaynak: Qing vd., 2015 : 451)

2.1.5. Ionic Framework

Ionic Framework, HTML, CSS ve Javascript web teknolojilerini kullanarak mobil uygulama oluşturan bir arayüz araç setidir. Popüler javascript çerçeveleri olan Angular, React ve Vue ile etkileşime geçerek kullanıcı deniyimli ve kullanıcı arayüzlü mobil uygulamalar tasarlamaya odaklanır. Tercih edilen javascript çerçevesi sayesinde mobil işletim sisteminin API'ları ile iletişime geçerek cihazın alanlarına ulaşır (Ionic, 2021).

2.2.Test Uygulamasında Kullanılan Teknolojiler

2.2.1. Rest Api

REST, çeşitli istemciler tarafından kaynak durumlarını HTTP isteklerine ve yanıtlarına göre web hizmetlerini tasarlayabilen bir dizi mimari ilkelerdir. Bu ilkelere göre sunucudan bir kaynağı almak için GET, kaynak oluşturmak için POST, kaynağı değiştirmek için PUT ve kaynağı silmek için DELETE kullanılır.

İstemci kaynakları URI'lar ile ele alır. Kaynak gösterimi istemci uygulamasının talep ettiği kaynağın durumunu yansıtır. Yansıyan durumun detaylarına ulaşabilmek için URI düzeni genellikle hiyerarşik ağaç düğümleri şeklindedir. https://todoprojectdemo.com/todos/ {year}/{month}/

Örneğin 2021 yılı ocak ayındaki yapılacak listesindeki içeriklere ulaşabilmek için geçerli URI: https://todoprojectdemo.com/todos/2021/Ocak/

Kaynak gösterimdeki çıktılar veri modelindeki tanımlara göre HTTP gövdesinde (body) anahtar, değer öznitelikleri biçimiyle görünür. Görülen çıktılar XML, JSON vb. formatındadır (Rodriguez, 2008: 1-11). Şekil 2.5.'te JSON çıktı örneği gösterilmiştir.

Şekil 2.5. JSON Çıktısı

2.2.1.1.Firebase

Firebase gerçek zamanlı veritabanı, bulutta verileri JSON olarak depolayarak her istemciye gerçek zamanlı verileri gönderen bir NoSQL veritabanıdır. Android, iOS, Web, C++, Unity ve REST API için kullanılabilir. Yapılacak Listesi (TodoApp) uygulamasında Firebase bulut tabanlı veritabanı olarak kullanılmaktadır.

Firebase Authentication, parolalar, telefon numaraları, sosyal medya hesapları ile giriş için kimlik doğrulamayı destekler (Google Firebase, 2021). Firebase Auth arka ucunu yeni kullanıcılar oluşturma, oturum açma, silme gibi işlemler için REST API aracılığıyla sorgular yapılmıştır.

2.2.2. Paket ve Eklentiler

2.2.2.1.Http / Fetch

Yapılan ağ isteklerinin okuma, yazma ve yanıtların sonuçlarına göre ayrıştırılmasını sağlamak için http istekleri yapılır. Kullanıcı arabirimi üzerinden istekler ağ gönderme iş parçacağı üzerinden çalışır. Kuyruğa eklenen istek, önbellek tarafından alınır ve önceliklendirilir. İstek önbellekteki başarılı veya başarısız duruma göre iş parçacağında kod çözümlenir ve gelen yanıt ana iş parçacağına teslim edilir. İlk ağ isteği kuyruktan alınarak http işlemleri gerçekleştirilir (Android, 2020).

Test uygulamasında http isteklerini gerçekleştirebilmek için Flutter çerçevesinde http paketi ayriyeten yüklenmiş, React Native ve Xamarinde çerçevelerinde ise kendi içlerinde gelen arayüzler kullanılmıştır. React Native için Javascript geliştirme API içerisinde yer alan fetch(), Xamarin için NET kütüphanesinde yer alan HttpClient arayüzü kullanılmıştır.

2.2.2.2.Shared Preferences

Uygulamalar, kullanıcının varsayılan veritabanındaki değerlere parametre göndererek kullanıcı tercihlerini deporlar. Basit değerleri depolamak ve almak için kullanılır. Parametreler anahtar-değer çiftleri içeren verileri içerir. Bilgiler önbellekte yer alır. Eşzamanlı ve eşzamansız olarak kalıcı depolama ve diğer işlemlerle değiştirilir. Uygulamanın davranışını kullanıcnın tercihlerine göre özelleştirmesine izin veren kullanıcı veritabanına Android işletim sisteminde SharedPreferences, iOS işletim sisteminde UserDefaults adı verilir. (Android, 2020; Apple, 2020).

Test edilecek uygulamada üye yetkinlendirme işlemleri için gerekli olacak token verisi ‘tokenId’, ‘userInfo’ parametreleri şeklinde gönderilip geçiçi veritabanında saklandı. Saklanan token verisi üyenin ekleyeceği ve sileceği verilerin sahiplik bilgisini arka uç’a gönderek doğrulama, yetkinlendirme işlemlerini gerçekleştirecektir. Platformlara özgü depolamayı sunan SharedPreferences ve UserDefaults için çapraz platform çerçevelerinin geliştirdikleri eklentiler mevcuttur. Test uygulamasında Flutter için shared_preferences, React Native için async-storage eklentileri kullanılmıştır. Xamarin çerçevesi için kendi içinde gelen ISharedPreferences arayüzü kullanılmıştır.

2.2.2.3.Sayfalama

Kulllanıcının sayfalar arası hiyerarşik gezinme deneyimi için çerçevelerin Navigation arayüzleri kullanılır. Bir sayfadan başka bir sayfaya geçmek için etkin sayfa push ile yeni bir sayfa gönderir. Önceki sayfaya geri dönmek için ise pop ile geçerli sayfakla eklenir ve en üstteki sayfa etkin olur (Microsoft, 2020). React Native çerçevesi için React Navigation paketi ayriyeten yüklenmiş, Flutter ve Xamarin için ise kendi içlerinde gelen Navigation arayüzleri kullanılmıştır.

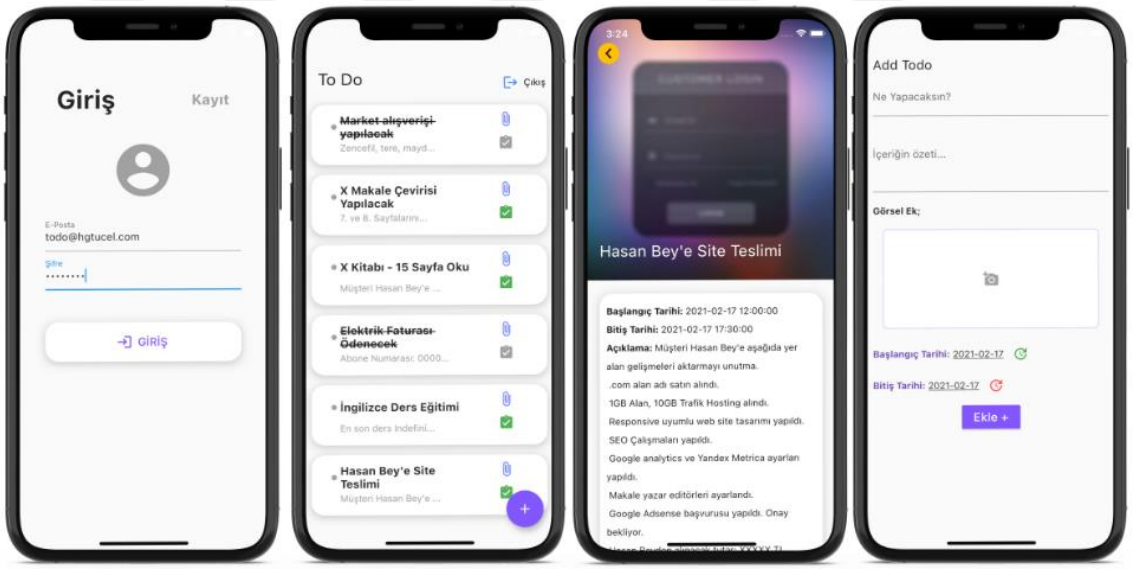
Kullanıcı notlar eklemek ve görüntülemek gibi işlemleri gerçekleştirebilmek için üye girişi veya üye kaydı yapmak zorundadır. Üye yetkinlendirme işlemleri başarılı olduğunda push yaparak yeni bir sayfaya geçer ve eklediği notlar sayfasına ulaşır. Ekranın sağ alt kısmında bulunan artı butonu yardımıyla push işlemi yaparak yeni notlar ekleme sayfasına geçiş yapar. İstenen giriş alanlarını doldurduktan sonra ekle

butonu ile yeni bir not ekleyebilir. İşlem başarılı olursa push işlemini gerçekleştirerek notların listelenme sayfasına geçiş yapar. İstenilirse üst kısmda bulunan geri ok ikonu yardımıyla pop işlemi yaparak önceki sayfaya geri dönüş sağlayabilir. Sayfanın en üst sağ kısmında yer alan çıkış ikonu butonu yardımıyla kullanıcı oturumu kapatabilir. İşlem başarılı olursa push işlemi gerçekleştirerek üye girişi ve kaydı ekranına ulaşır. Ağ istekleri sonucu yapılan hatalar o ekran üzerinden snackbar özelliği üzerinden gösterilir. Ayrı bir sayfa üzerinden push yapılmaz.

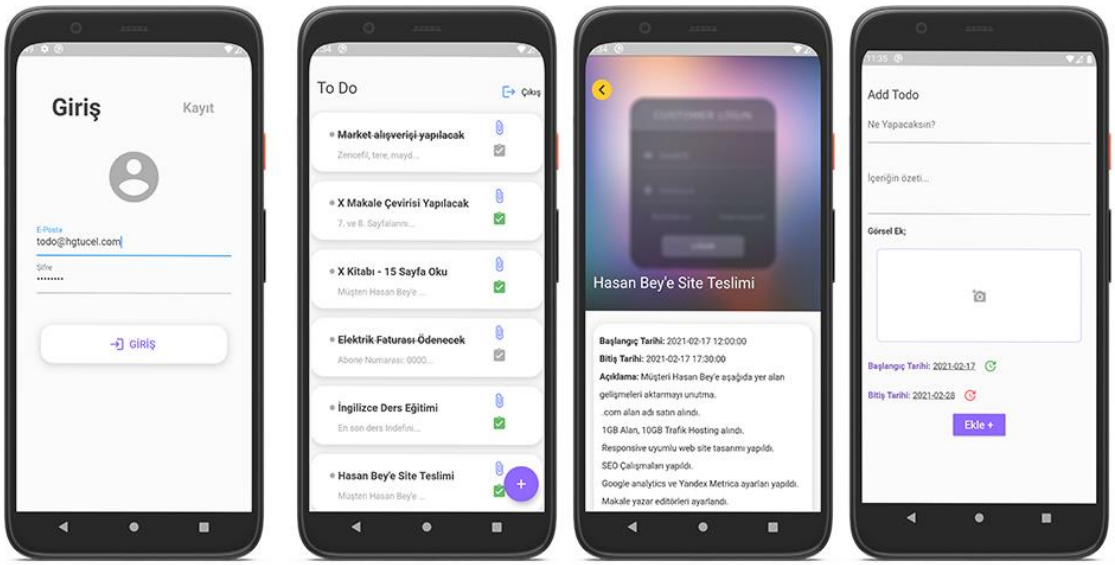
2.3.Uygulama Tasarımı

Çapraz platform çerçeveleri arasındaki farkları karşılaştırmak için uygulama üç kez her çerçeve için oluşturulmuştur. Değerlendirme için tasarlanan uygulama dört ekrandan oluşmaktadır. Şekil 2.6. ve Şekil 2.7.'deki gibi birinci ekran üye girişi veya kaydının yapıldığı, ikinci ekran üyenin yapılacak listesinin yer aldığı, üçüncü ekran yapılacaklar listesi ekleme ve dördüncü son ekran yapılacak listesinin detay ekranı.

Şekil 2.6. İos Todo List App Ekran Görüntüleri



Şekil 2.7. Android Todo List App Ekran Görüntüleri



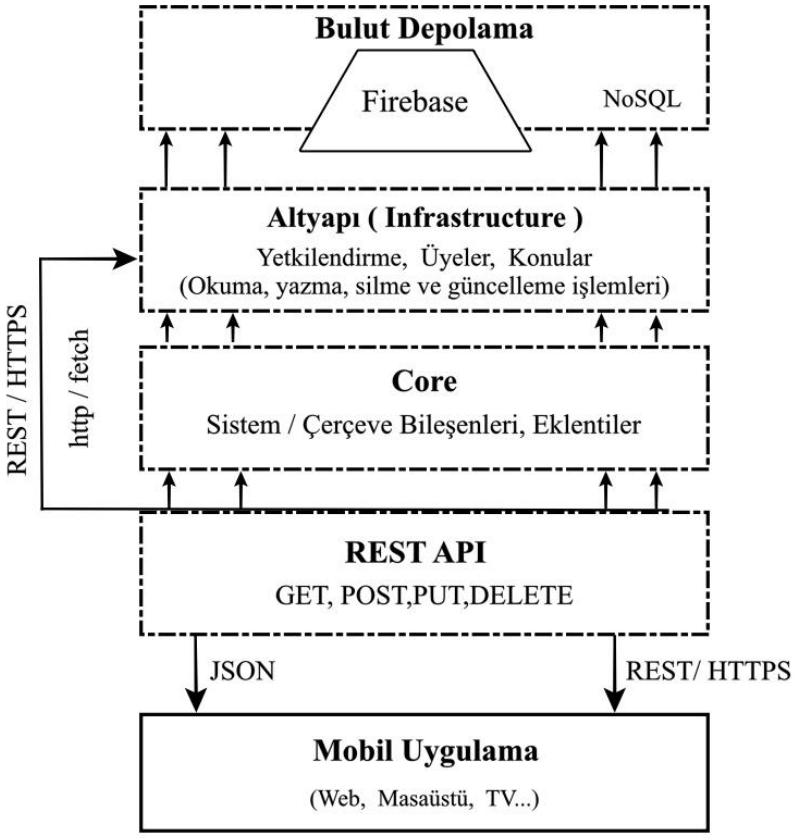
Uygulamalarda kullanılan ortak teknolojiler: Todo List uygulamasının arka uç (backend) kısmında Firebase kullanıldı. Firebase Auth özelliği ile üye girişi, üye kaydı, Firebase Realtime Database ile de kullanıcıların verileri NOSQL bir veritabanında kayıt altına alındı. Yapılacaklar listesi eklerken kullanıcı dosya eki eklendiğinde dosyalar Firebase Storage'a kayıt edildi.

Firebase'in kullanılabilmesi için çerçevelerin paketleri mevcut ancak her üç uygulama için de en iyi sonuçları elde etmek için Firebase REST API kullanıldı. GET, POST, PUT, DELETE sorguları kullanılmış oldu. API'ın yanıtları JSON formatında biçimlenip geriye döndürülmüştür.

Frontend kısmında ise REST API istek yapabilmek için (GET,POST,PUST,DELETE) http, üyelik bilgilerini (token) geçici hafızada saklamak için shared preferences eklentileri kullanıldı.

2.4.Arka Uç Tasarımı

Şekil 2.8. Uygulama Mimarisi

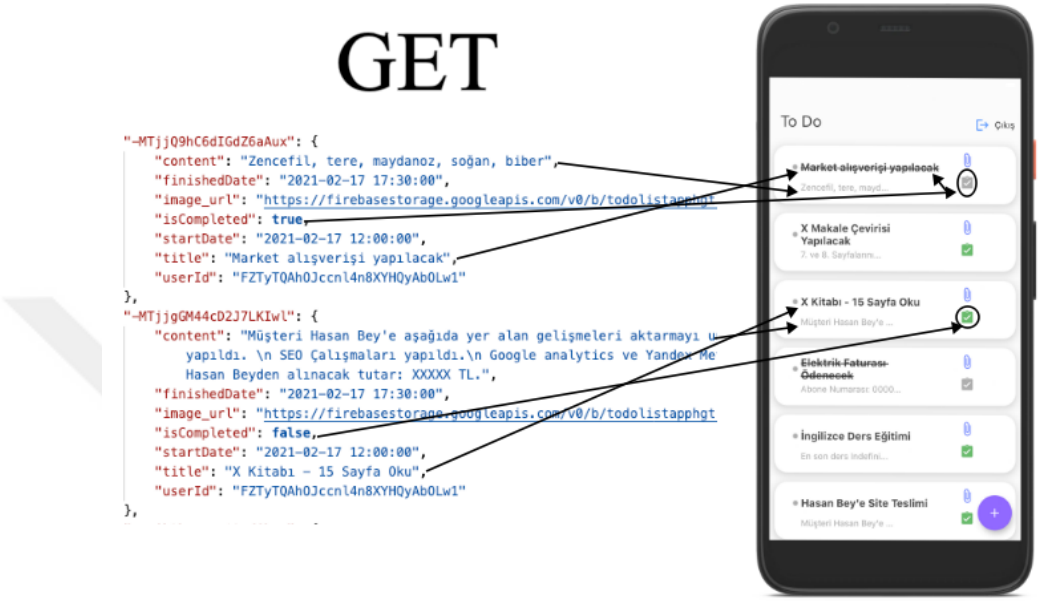


Uygulama, Firebase'in gerçek zamanlı veri tabanı adını verdiği NoSQL veri tabanı bulut depolamayı kullanır. Üyeler ve kaydettikleri konular veri tabanına kayıt edilir. Altyapı katmanı uygulamada yer alan özelliklerin mantığının yansıtılmasını sağlar. Ortak olan veri modelleri, yetkilendirme işlemleri, konuların listelenmesi, eklenmesi, silinmesi ve güncellenmesi için veri ayrımını yapar. Temel çekirdek Core katmanı ile kullanıcının davranışlarına göre iletişim temsili bir REST API üzerinden JSON kullanılarak gerçekleşir. Firebase, veri tabanı görevi yanında arka uç görevi de sağlar.

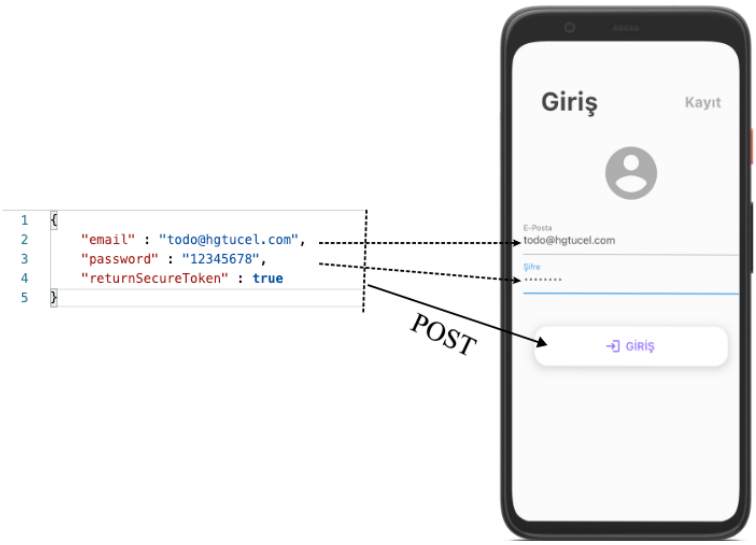
Son katman REST API katmanında, kullanıcı uygulama üzerinden üye girişi ve kaydı, konu ekleme ve silme gibi işlemlerini kullanıcı arabirimi (UI) ile üzerinden yapar. Kullanıcı arabirimi yani istemci burada mobil uygulamadır. Kullanıcı

arabirimi geliştirici tarafından tercih istenilirse web, masaüstü, televizyon gibi cihazlar için de gerçekleştirilebilir.

Şekil 2.9. Get Sorgu Sonucu Çıktısı Json Formatı



Şekil 2.10. Post Sorgu Sonucu Çıktısı Json Formatı



ÜÇÜNCÜ BÖLÜM

3. YÖNTEM

Bu bölümde; araştırmanın modeli, çalışma grubu, veri toplama araçları, verilerin toplanması ve verilerin analizine ait bilgilere yer verilmiştir.

3.1.Araştırmanın Modeli

Bu araştırma çapraz platform mobil uygulama geliştirme çerçevelerinin performans farklarını incelemeye yönelik bir çalışmadır. İstatistiksel verileri sunan Statista ve geliştiriciler tarafından kullanılan Stackoverflow internet sitelerindeki kullanım oranlarına ve proje geliştirme sayılarına bakılarak en yüksek ilk üç çerçeve Flutter, React Native ve Xamarin'in karşılaştırma yöntemini kullanılarak karşılaştırılmasına karar verilmiştir (Statista, 2020; Stackoverflow, 2020).

Seçilen üç çerçeve arasında performans testleri yapılacaktır. Performans karşılaştırma testleri ikinci bölümde anlatılan uygulama her üç platform içinde geliştirilecektir. Yazılan uygulamalar üç Android cihazı ve üç iOS cihazında ölçülecektir.

3.2.Araştırma Grubu

Araştırma örneklemini çapraz platform geliştirme çerçevelerinden Flutter, React Native ve Xamarin oluşturmaktadır. Araştırma grubunu 3 Android ve 3 iOS cihazı olmak üzere toplam 6 cihaz oluşturmaktadır. Performans testlerindeki farkların görülmesine katkı sağlayacak cihazlar farklı donanımlara sahip cihazlardır. Kullanılan Android cihazların özellikleri Tablo 3.1.'de, iOS cihazların özellikleri ise Tablo 3.2.'de gösterilmiştir.

Tablo 3.1. Android İşletim Sistemli Test Cihazları

Cihaz Adı	Cihaz Özellikleri
Samsung Galaxy A50	Chipset: Exynos 9610 (10nm) CPU: Octa-core (4x2.3 GHz Cortex-A73 & 4x1.7 GHz Cortex-A53) RAM: 6GB OS: Android 10 Batarya: Li-Po 4000 mAh
Xiaomi Mi A3	Chipset: Qualcomm SDM665 Snapdragon 665 (11 nm) CPU: Octa-core (4x2.0 GHz Kryo 260 Gold & 4x1.8 GHz Kryo 260 Silver) RAM: 4GB OS: Android 11 Batarya: Li-Po 4030 mAh
Samsung Galaxy A51	Chipset: Exynos 9611 (10 nm) CPU: Octa-core (4x2.3 GHz Cortex-A73 & 4x1.7 GHz Cortex-A53) OS: Android 11 Batarya: Li-Ion 4000 mAh

Tablo 3.2. iOS İşletim Sistemi Test Cihazları

Cihaz Adı	Cihaz Özellikleri
Apple iPhone 8	Chipset: Apple A11 Bionic (10 nm) CPU: Hexa-core (2x Monsoon + 4x Mistral) RAM: 2GB OS: iOS 14.4 Batarya: Li-Ion 1821 mAh
Apple iPhone SE (2020)	Chipset: Apple A13 Bionic (7 nm+) CPU: Hexa-core (2x2.65 GHz Lightning + 4x1.8 GHz Thunder) RAM: 3GB OS: iOS 14.4 Batarya: Li-Ion 1821 mAh
Apple iPhone 8 Plus	Chipset: Apple A11 Bionic (10 nm) CPU: Hexa-core (2x Monsoon + 4x Mistral) RAM: 3GB OS: iOS 14.4 Batarya: Li-Ion 2691 mAh

3.3.Veri Toplama Araçları

Araştırma verilerinin toplanmasında işlemci, bellek, enerji ve ağ tüketim değerlerini kayıt altında tutabilmek amacıyla test cihazları üzerinde çalıştırılan uygulamalar, çerçeveler arası karşılaştırma yapabilmek için Android işletim sistemli cihazlarda Android Studio Profiler ve Perfetto UI, iOS işletim sistemli cihazlarda ise Insturments programları kullanılmıştır.

3.3.1. Android Studio Profiler

Çalışmada kullanılan Android Studio Profiler programı Google tarafından Android geliştiricileri için geliştirilen uygulamanın performans profillerini anlık olarak göstermek için tasarlanan bir araçtır (Android, 2020). Bu araç ile işlemci, bellek, batarya ve ağ kullanım oranları toplanmıştır.

3.3.2. Perfetto UI

Perfetto, Google tarafından geliştirilen üretim düzeyinde performans izleme analizi sonuçlarını görselleştirmek için kullanılan açık kaynak kodlu web tabanlı bir kullanıcı arabirimidir. Linux ve Android işletim sistemlerindeki sistem ara yüzlerinden performans verilerini toplayarak veri kaynağı haline getirir (Perfetto, 2021). Tek çekirdek üzerindeki performansı ve batarya üzerindeki etkisi için veriler toplanmıştır.

3.3.3. Instruments

Insturments, Apple tarafından geliştirilen iOS, watchOS, tvOS ve macOS işletim sistemlerinin performans analizini gerçekleştiren test aracıdır. Veriler profiller halinde toplanır ve sonuçlar ayrıntılı olarak sunulur (Apple, 2019). Veriler, iOS işletim sistemli iPhone cihazların Apple Macbook Pro bilgisayarına bağlanarak Xcode araç seti üzerinden kayıt edilmiştir. Bu araç ile işlemci ve bellek kullanım oranları toplanmıştır.

3.4.Verilerin Toplanması

Araştırmanın verileri yapılacaklar listesi (todolist) uygulamasının Flutter, React Native ve Xamarin çerçeveleri için ayrı ayrı geliştirilikten sonra test cihazlarına yüklenerek elde edilmiştir. Toplam 6 cihaz üzerinden veriler toplanmıştır. Ölçüm verileri 2 dakika 10 saniye boyunca kaydedildi. Uygulama 2 dakika boyunca çalıştırıldı. Geriye kalan 10 saniye bekleme süreleri ve uygulamanın açılması ve kapanması sürelerinde harcandı. Cihazlarda çalışma boyunca arkaplanda başka bir uygulama çalışmamıştır. Aşamalar Tablo 3.3'de gösterildiği gibidir.

Tablo 3.3. Uygulama Test Aşamaları

Aşamalar	**Test**
Aşama 1	Sistem izlemenin başlatılması
Aşama 2	Uygulamanın açılması ve yüklenmenin beklenmesi
Aşama 3	Ana ekrana dönülmesi (Arka Planda Çalışma)
Aşama 4	Arka plandan tekrar uygulamanın çalıştırılması ve üye girişi yapılması
Aşama 5	Listenin görülmesi ve scrool view kullanılması. (Listenin aşağı yukarı yapılması)
Aşama 6	Ana ekrana dönülmesi (Arka planda çalışma)
Aşama 7	Arka plandan tekrar uygulamanın çalıştırılması ve ekleme sayfasına geçilmesi
Aşama 8	Yapılacaklar listesi ekleme alanlarının doldurulması (başlık, açıklama, dosya ekleme)
Aşama 9	Listenin görülmesi
Aşama 10	Uygulamadan tamamen çıkış. (Arka plandan kapatma)
Aşama 11	Uygulamanın açılması ve yüklenmenin beklenmesi
Aşama 12	Uygulamadan üye çıkışı yapılması
Aşama 13	Sistem izlemenin durulması

3.5.Verilerin Analizi

Test cihazları üzerinden çalıştırılan uygulamalar, çerçeveler arası işlemci, bellek, enerji ve ağ karşılaştırmaları yapabilmek amacıyla Android işletim sistemli cihazlar için Android Studio Profiler ve Perfetto UI, iOS işletim sistemli cihazlarda ise Insturments programları kullanılarak analiz edilmiştir.

Android ve iOS mobil işletim sistemlerinin altyapıları farklı olduğu için test araçlarıda farklıdır. Test araçlarında toplanan android işletim sistemli cihazlarda çalışan uygulamaların işlemci kullanım oranları makisumum düzeyde yüzde cinsinden ne kadar kullandığı üzerindendir. Analiz aracı tarafından işlemci üzerinde oluşturulan iş parçacağı (thread) üzerinde bulunan uygulama değeri alınmıştır. Bellek üzerindeki kullanımı uygulamanın maksimum tüketim sağladığı bilgisayar ölçü birimi megabayt üzerindendir. Batarya, enerji tüketimi oranı düşük, orta, ve yüksek olarak kullanım değerini belirtir. Ağ kullanım oranı ise yüklenen ve indirilen dosyaların boyutları cinsi üzerindendir.

Ana iş parçacağı üzerinde yüksek miktarda kullanım, çalışan uygulamanın yavaş çalışmasına neden olur. Mümkün olduğunca iş, ana iş parçacığından uzaklaştırılmalıdır (Apple, 2019). İos işletim sistemli cihazlarda çalışan uygulamaların test araçlarında toplanan işlemci kullanım oranı ana iş parçacağı üzerindeki ağırlığı yüzde olarak ne kadar kullandığı üzerinedir. Bellek üzerindeki kullanımı ise uygulamanın tüm yığın üzerindeki tüketim değeri megabayt üzerinden belirtilmiştir.

DÖRDÜNCÜ BÖLÜM

4. BULGULAR VE YORUM

Araştırmanın bu bölümünde çerçeveler için ayrı ayrı geliştirilmiş uygulamalardan elde edilmiş performans ve tüketim değerlerine ilişkin bulgular ve yorumlara yer verilmiştir.

4.1.Android İşletim Sistemli Cihazların Tüketim Bulguları

4.1.1. Cpu Kullanımı

İlk olarak Android işletim sistemli cihazlar ile çerçeveler ayrı ayrı çalıştırılmıştır. Android Profiler ile sonuçların verildiği grafiklerde dikey eksende CPU kaynağının kullanım oranı yüzde cinsinden, yatay eksende ise saniye cinsinden zaman belirtilmektedir.

Flutter çerçevesi Android işletim sistemi için birinci test cihazı Samsung Galaxy A50 ile test edilmiştir. Şekil 4.1.'deki gibi işlemciyi en fazla %39 oranında 7. Saniyede kullanmıştır. Tek çekirdek işlemci üzerinde çalışır durumdaki işlem süresi 4,82sn, seçilen dilim sayısı ise 4653'dir. 8 çekirdeğin hepsini kullandığında çalışma süresi 58 sn, seçilen dilim sayısı ise 49086'dir.

Şekil 4.1. Flutter Çerçevesi Samsung Galaxy A50 Cpu Kullanım Oranları

İkinci test cihazı Xiaomi Mi A3 ile uygulama Flutter çerçevesi için test edilmiştir. Şekil 4.2.'deki gibi işlemciyi en fazla %34 oranında 4. Saniyede kullanmıştır. 8 çekirdekli işlemci üzerinde, tek çekirdekte çalışır durumdaki işlem süresi 2,85 sn, seçilen dilim sayısı 2425'dir. İşlemcideki tüm çekirdekleri kullandığında ise işlem süresi 61,24 sn, seçilen dilim sayısı 39813'dir.

Şekil 4.2. Flutter Çerçevesi Xiaomi Mi A3 Cpu Kullanım Oranları

Üçüncü test cihazı Samsung Galaxy A51 ile uygulama Flutter çerçevesi için test edilmiştir. Şekil 4.3.'deki gibi işlemciyi en fazla %37 oranında 1. Dakika 39. Saniyede kullanmıştır. 8 çekirdekli işlemci üzerinde, tek çekirdekte çalışır durumdaki işlem süresi 4,24 sn, seçilen dilim sayısı 5842'dir. İşlemcideki tüm çekirdekleri kullandığında ise işlem süresi 55,64 sn, seçilen dilim sayısı 14290'dir.

Şekil 4.3. Flutter İçin Samsung Galaxy A51 Cpu Kullanım Oranları

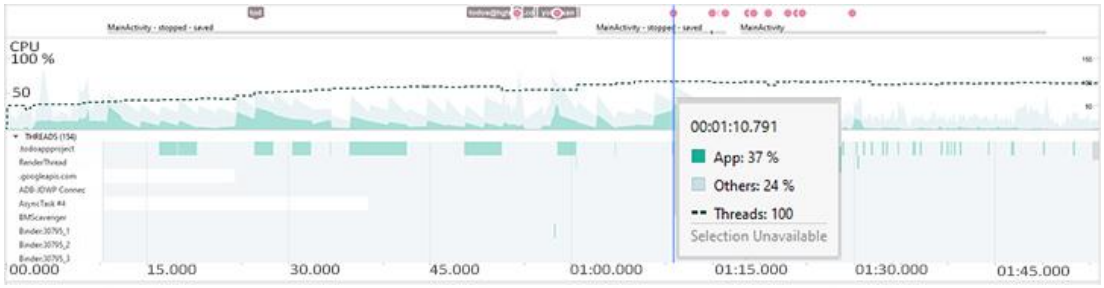
React Native Android işletim sistemi için birinci test cihazı Samsung Galaxy A50 ile test edilmiştir. Şekil 4.4.'deki gibi işlemciyi en fazla %44 oranında 21. Saniyede kullanmıştır. Tek çekirdek işlemcideki çalışır durumdaki işlem süresi 2.42 sn, seçilen dilim sayısı 1797'dir. İşlemcideki 8 çekirdek üzerindeki işlem süresi 71,94sn, seçilen dilim sayısı ise 12282'dir.

Şekil 4.4. React Native İçin Samsung Galaxy A50 Cpu Kullanım Oranları

İkinci test cihazı ile uygulama React Native için test edildiğinde ise Şekil 4.5.'deki gibi işlemciyi en fazla %37 oranında 1. Dakika 10. Saniyede kullanmıştır. 8

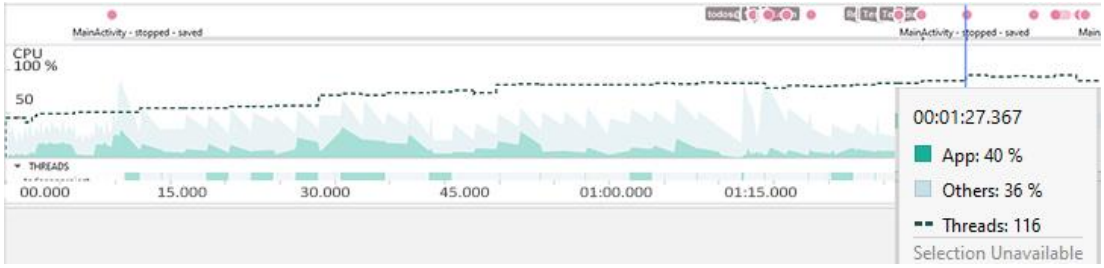
çekirdekli işlemci üzerinde, tek çekirdekte çalışır durumdaki işlem süresi 5,42 sn, seçilen dilim sayısı 3873'dir. İşlemcideki tüm çekirdekleri kullandığında ise işlem süresi 74,29 sn, seçilen dilim sayısı 59478'dir.

Şekil 4.5. React Native İçin Xiaomi Mi A3 Cpu Kullanım Oranları



Üçüncü test cihazı ile uygulama React Native için test edildiğinde ise Şekil 4.6.'deki gibi işlemciyi en fazla %40 oranında 1. Dakika 27. Saniyede kullanmıştır. 8 çekirdekli işlemci üzerinde, tek çekirdekte çalışır durumdaki işlem süresi 5,66 sn, seçilen dilim sayısı 9030'dir. İşlemcideki tüm çekirdekleri kullandığında ise işlem süresi 55,95 sn, seçilen dilim sayısı 12576'dir.

Şekil 4.0.6. React Native İçin Samsung Galaxy A51 Cpu Kullanım Oranları

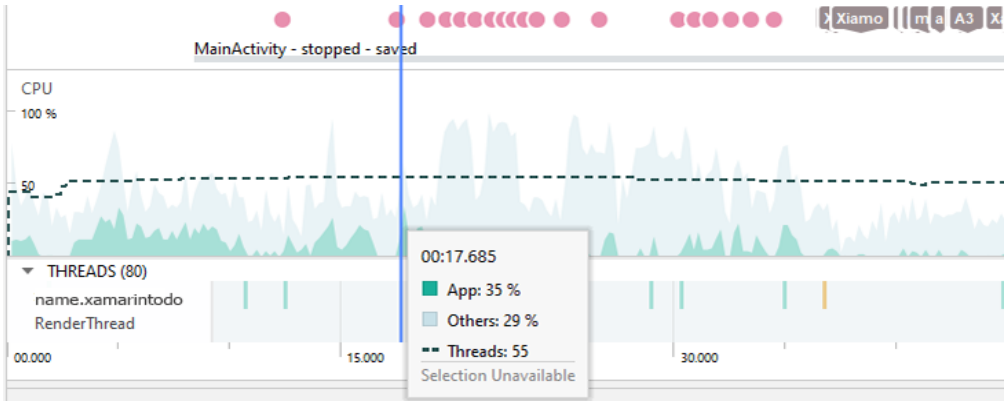


Üçüncü çerçeve olan Xamarin, Android işletim sistemi için uygulama birinci test cihazı Samsung Galaxy A50 ile test edilmiştir. Şekil 4.7.'deki gibi işlemciyi en fazla %35 oranında 7. Saniyede kullanmıştır. Tek çekirdek işlemcideki çalışır durumdaki işlem süresi 3.42 sn, seçilen dilim sayısı 3178'dir. İşlemcideki 8 çekirdek üzerindeki işlem süresi 29,22 sn, seçilen dilim sayısı ise 34544'dir.

Şekil 4.7. Xamarin İçin Samsung Galaxy A50 Cpu Kullanım Oranları

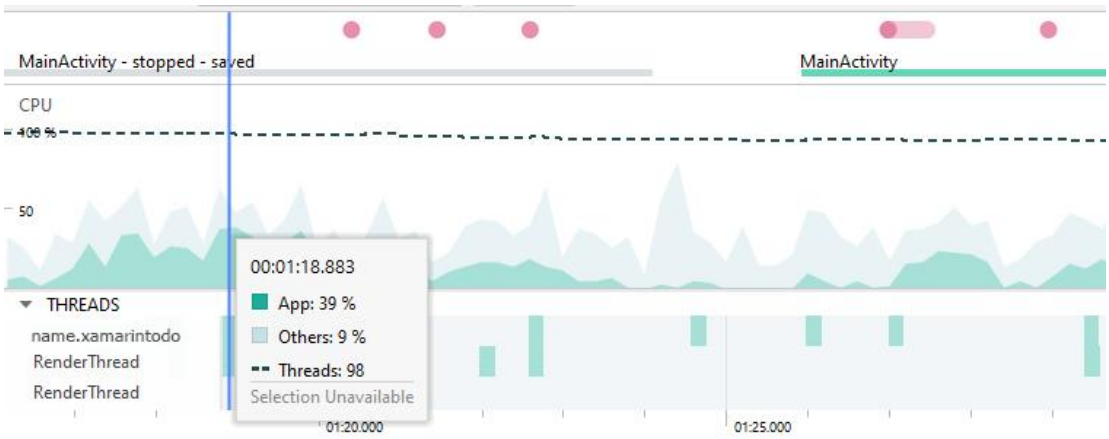
İkinci test cihazı Xiaomi Mi A3 ile uygulama Flutter çerçevesi için test edilmiştir. Şekil 4.8.'deki gibi işlemciyi en fazla %34 oranında 4. Saniyede kullanmıştır. 8 çekirdekli işlemci üzerinde, tek çekirdekte çalışır durumdaki işlem süresi 2,85 sn, seçilen dilim sayısı 2425'dir. İşlemcideki tüm çekirdekleri kullandığında ise işlem süresi 44,18 sn, seçilen dilim sayısı 35542'dir.

Şekil 4.8. Xamarin İçin Xiaomi Mi A3 Cpu Kullanım Oranları



Üçüncü test cihazı ile uygulama Xamarin için test edildiğinde ise Şekil 4.9.'deki gibi işlemciyi en fazla %39 oranında 1. Dakika 18. Saniyede kullanmıştır. 8 çekirdekli işlemci üzerinde, tek çekirdekte çalışır durumdaki işlem süresi 2,75 sn, seçilen dilim sayısı 2392'dir. İşlemcideki tüm çekirdekleri kullandığında ise işlem süresi 74,29 sn, seçilen dilim sayısı 60478'dir.

Şekil 4.9. Xamarin İçin Samsung Galaxy A51 Cpu Kullanım Oranları

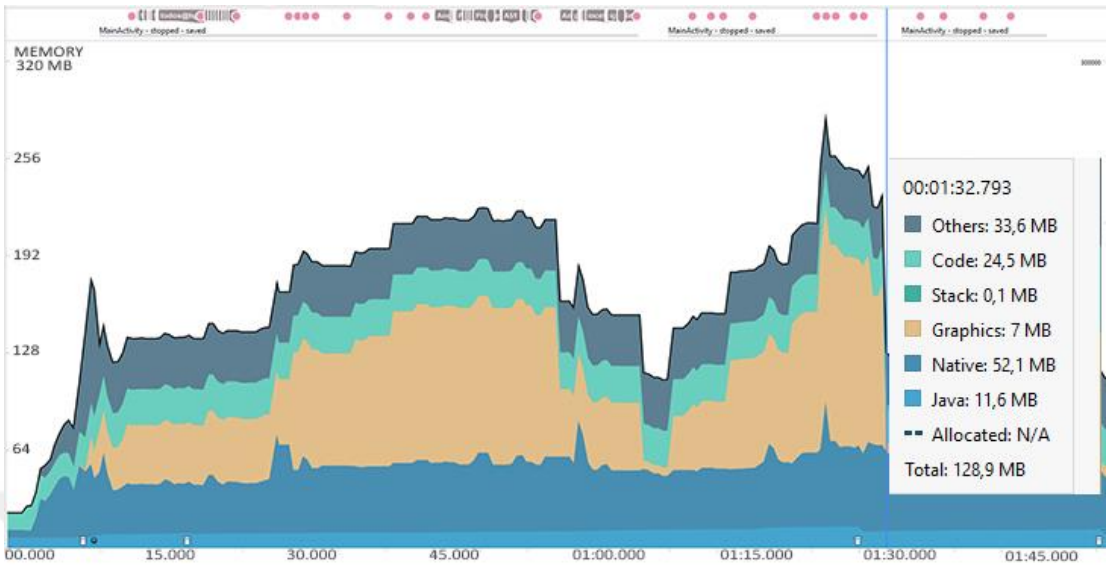


4.1.2. Bellek Kullanımı

Bellek kullanımları test cihazlarında Tablo 1.'de yer aldığı aşamalar uygulanarak ölçülmüştür. Grafiklerde yer alan dikey eksen kullanılan bellek miktarını MB cinsinden, yatay eksen ise saniye cinsinden zamanı belirtmektedir. Grafiklerdeki yeşil alan, çerçeveler ile yazılan uygulamaların kullandığı bellek miktarını kod etiketi ile göstermektedir.

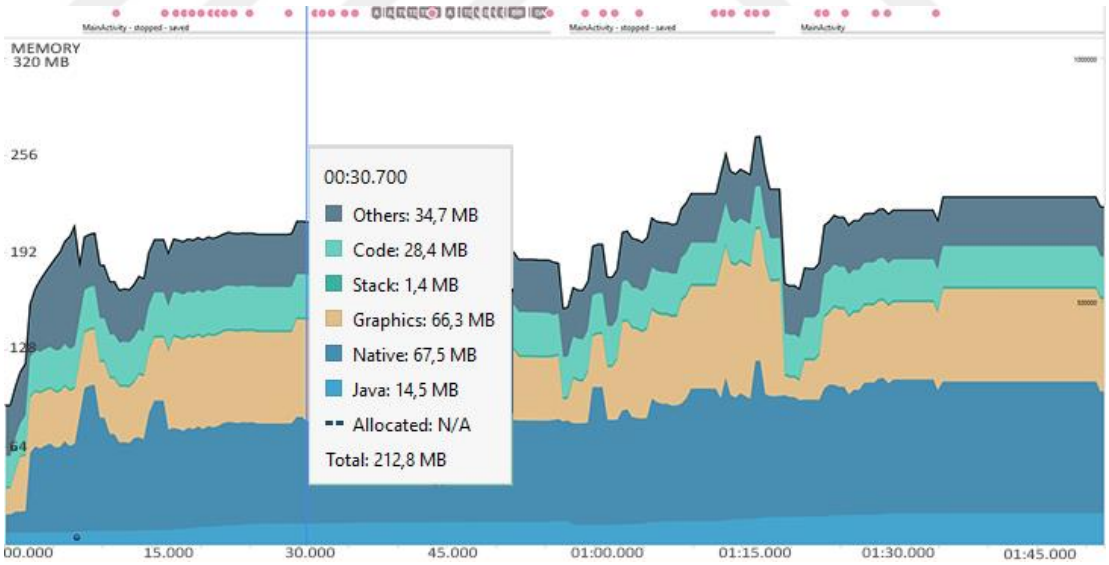
Flutter çerçevesi ile yazılan uygulama birinci test cihazı ile en fazla 24,5 MB bellek kullanım kaynağına ihtiyaç duymuştur. Bu ihtiyacı Grafik Şekil 4.10.'da görüldüğü gibi 1. Dakika 32. Saniyede kullanmıştır.

Şekil 4.10. Flutter Çerçevesi Samsung Galaxy A50 Bellek Kullanım Oranları



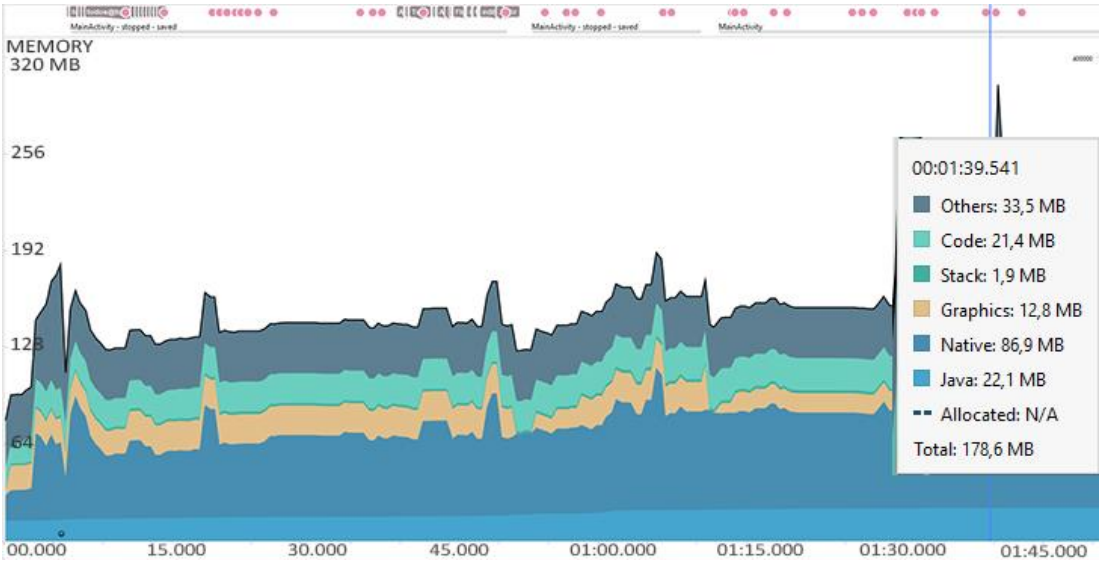
İkinci test cihazı ile Flutter çerçevesi ile yazılan uygulama Şekil 4.11.'de görüldüğü gibi 30. Saniyede en fazla 28,4 MB bellek kullanmıştır.

Şekil 4.11. Flutter Çerçevesi Xiaomi Mi A3 Bellek Kullanım Oranları



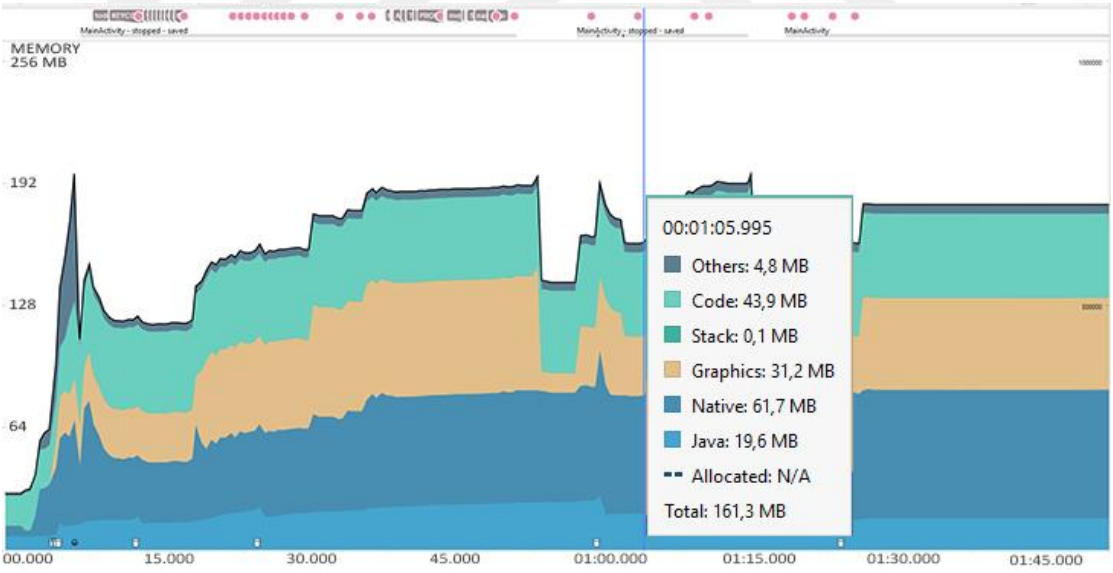
Üçüncü test cihazı ile ölçülen Flutter çerçevesi ile yazılan uygulama ise Şekil 4.12.'de görüldüğü gibi 1. Dakika 39. Saniyede en fazla 21,4 MB bellek kullanmıştır.

Şekil 4.12. Flutter Çerçevesi Samsung Galaxy A51 Bellek Kullanım Oranları



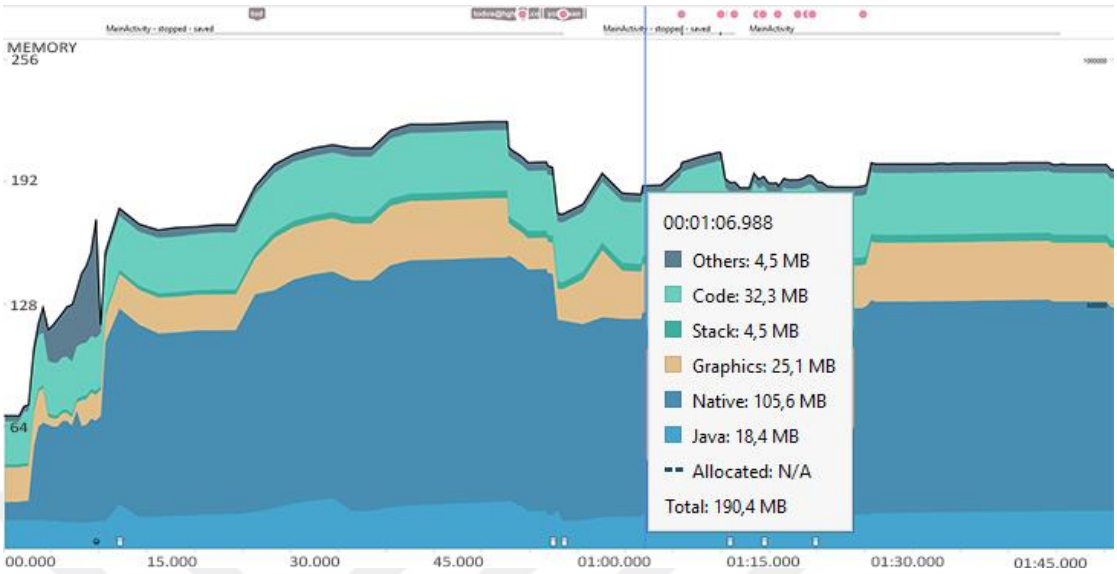
React Native ile yazılan uygulama birinci test cihazı ile en fazla 43,9 MB bellek kullanım kaynağına ihtiyaç duymuştur. Bu ihtiyacı Grafik Şekil 4.13.'de görüldüğü gibi 1. Dakika 5. Saniyede kullanmıştır.

Şekil 4.13. React Native Samsung Galaxy A50 Bellek Kullanım Oranları



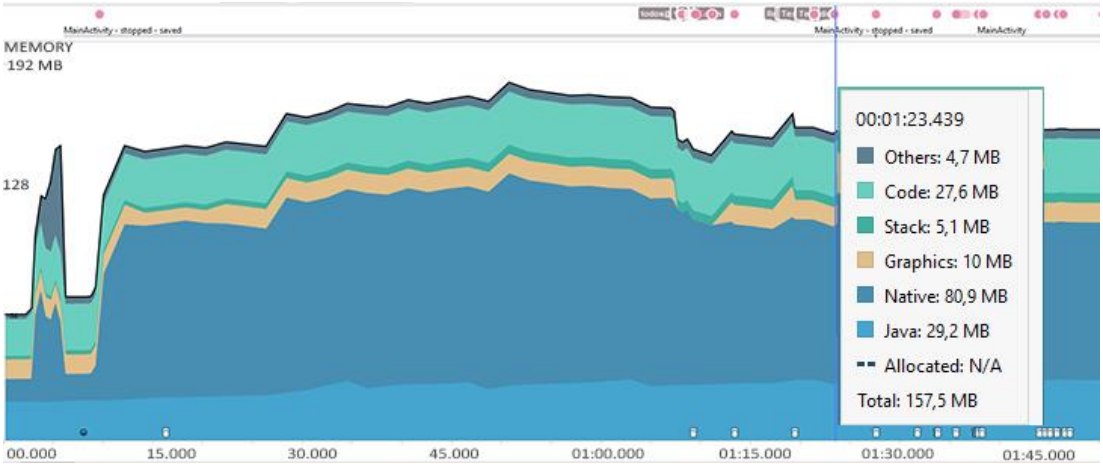
İkinci test cihazı ile test edilen React Native uygulaması Şekil 4.14.'de görüldüğü gibi en fazla 32,3 MB belleğe ihtiyaç duymuş ve bu ihtiyacı 1. Dakika 6. Saniyede kullanmıştır.

Şekil 4.14. React Native Xiaomi Mi A3 Bellek Kullanım Oranları



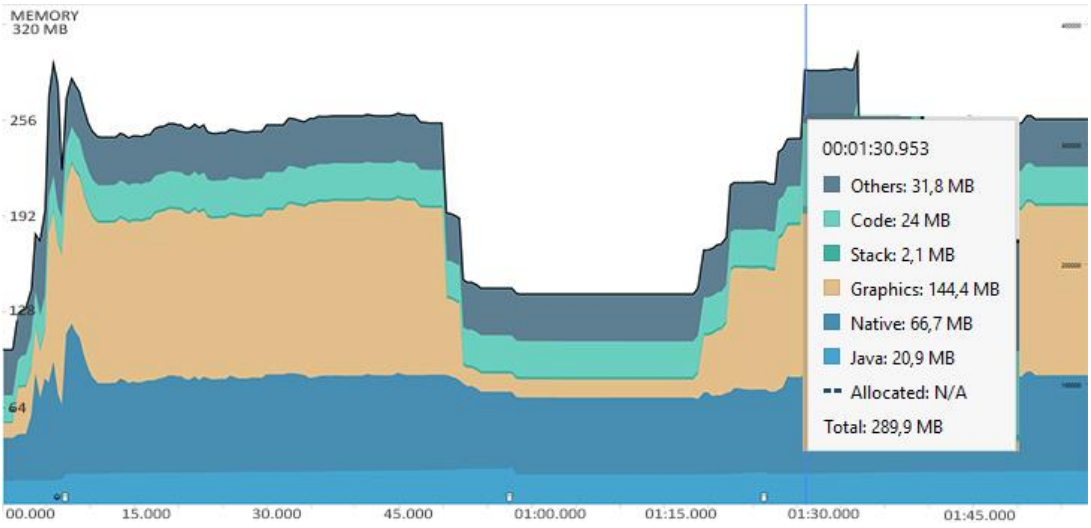
Üçüncü test cihazı ile test edilen React Native uygulaması ise Şekil 4.15.'de görüldüğü gibi en fazla 27,6 MB belleğe ihtiyaç duymuş ve bu ihtiyacı 1. Dakika 23. Saniyede kullanmıştır.

Şekil 4.15. React Native Samsung Galaxy A51 Bellek Kullanım Oranları



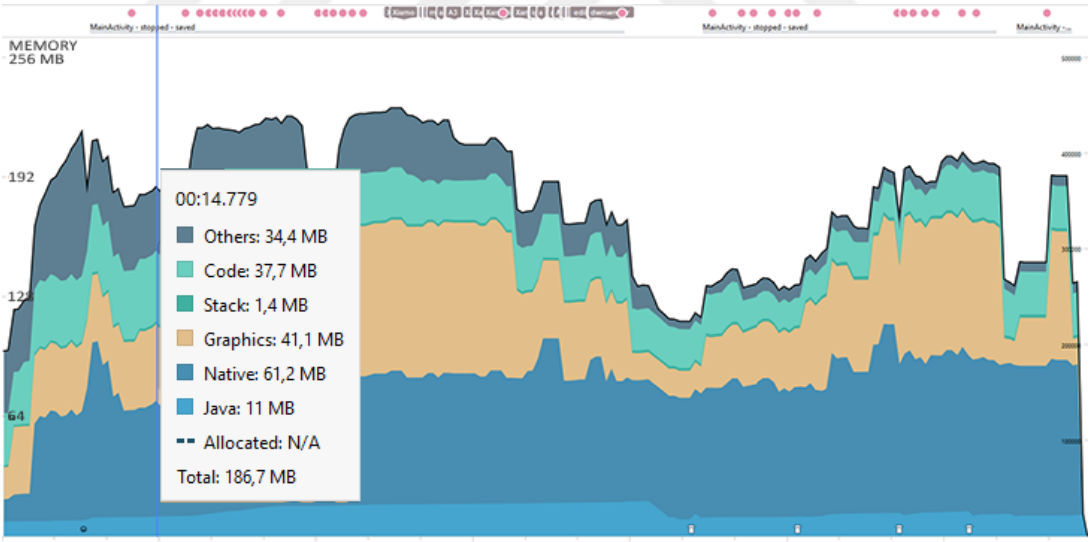
Xamarin ile yazılan uygulama birinci test cihazı ile en fazla 24 MB bellek kullanım kaynağına ihtiyaç duymuştur. Bu ihtiyacı Grafik Şekil 4.15.'da görüldüğü gibi 1. Dakika 30. Saniyede kullanmıştır.

Şekil 4.16. Xamarin Samsung Galaxy A50 Bellek Kullanım Oranları



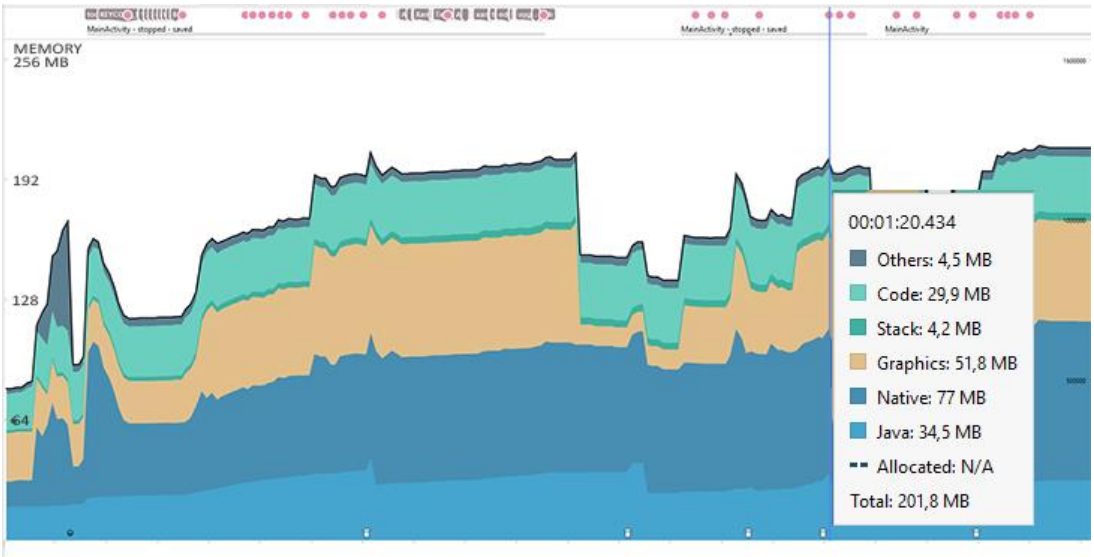
İkinci test cihazı ile test edilen Xamarin uygulaması Şekil 4.17.'de görüldüğü gibi en fazla 37,7 MB belleğe ihtiyaç duymuş ve bu ihtiyacı 14. Saniyede kullanmıştır.

Şekil 4.17. Xamarin Xiaomi Mi A3 Bellek Kullanım Oranları



Üçüncü test cihazı ile test edilen Xamarin uygulaması Şekil 4.18.'de görüldüğü gibi en fazla 29,9 MB belleğe ihtiyaç duymuş ve bu ihtiyacı 1. Dakika 20. Saniyede kullanmıştır.

Şekil 4.18. Xamarin Samsung Galaxy A51 Bellek Kullanım Oranları

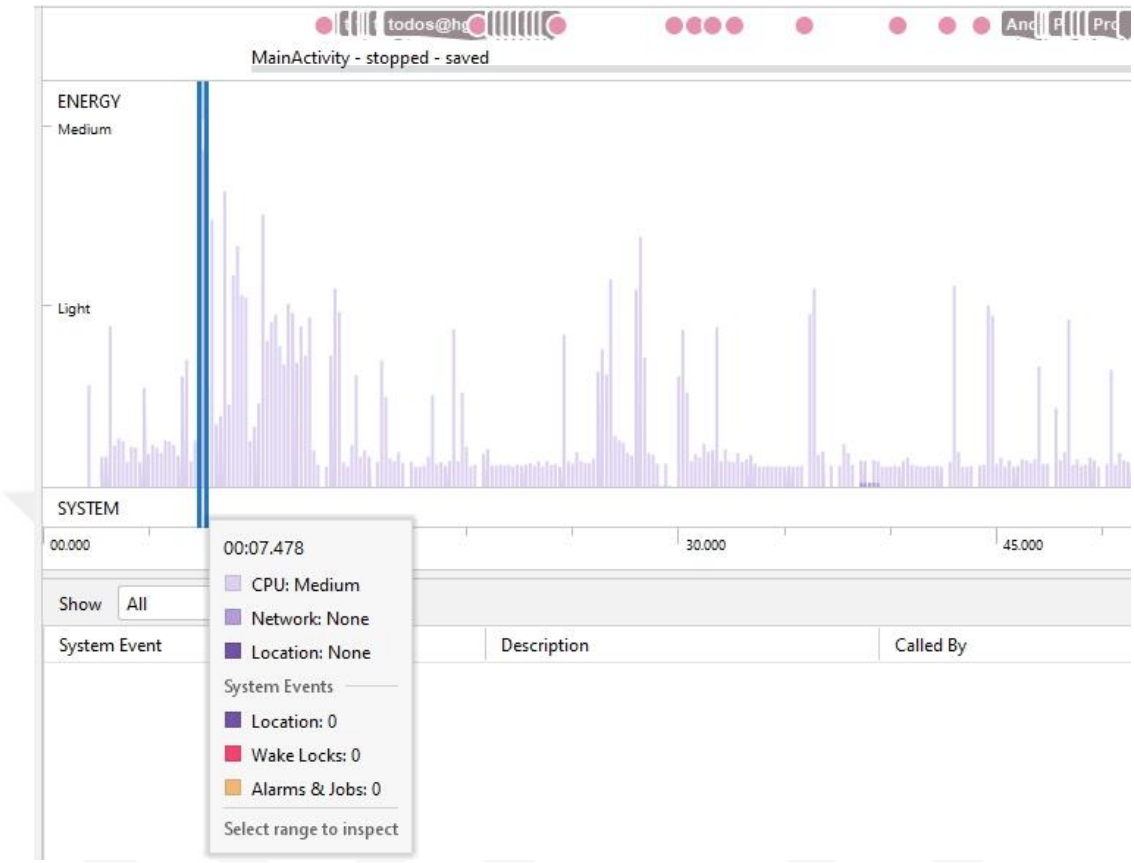


4.1.3. Enerji Kullanımı

Enerji kullanımları test cihazlarında Tablo 3.3.'de yer aldığı aşamalar uygulanarak ölçülmüştür. Grafiklerde yer alan dikey eksen bataryanın kullanım oranını düşük, orta ve yüksek olarak, yatay ekseni ise zamanı cinsinden saniye olarak belirtmektedir.

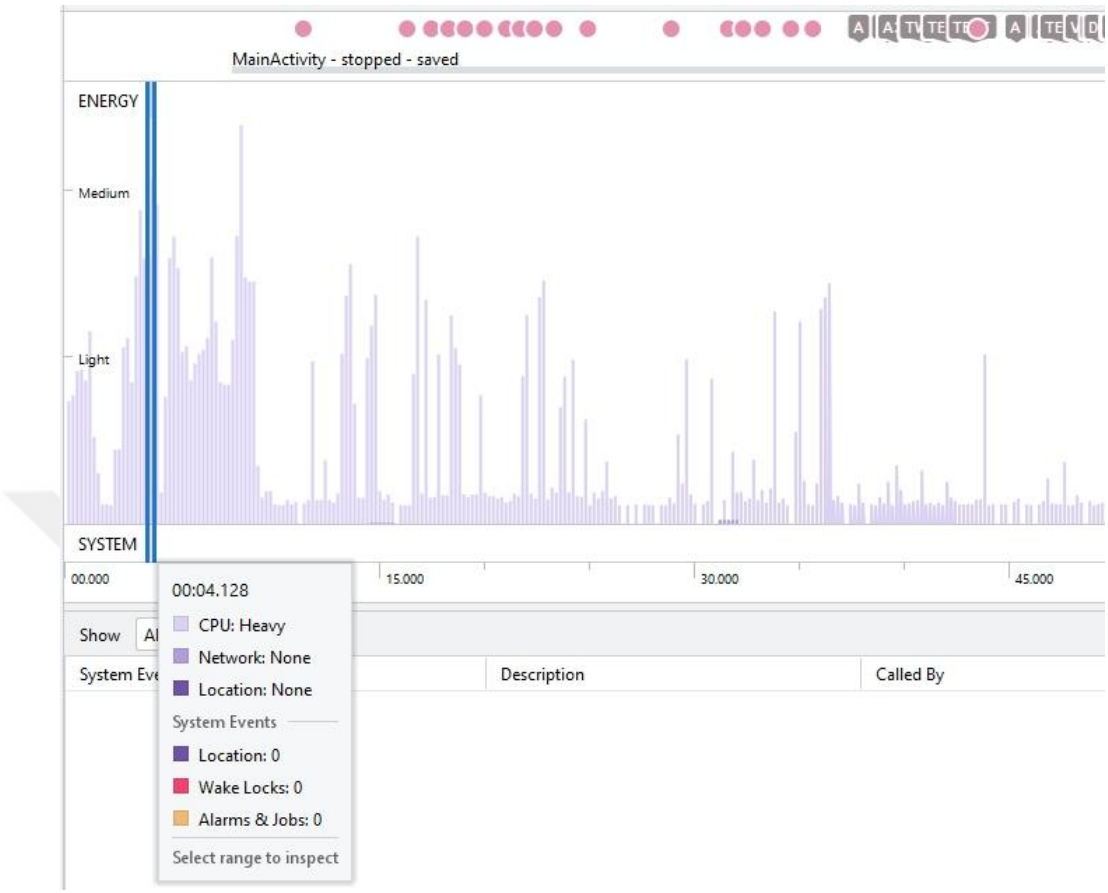
Flutter çerçevesi ile yazılan uygulamanın Samsung Galaxy A50 test cihazında enerji kullanım oranları Şekil 4.19.'deki gibidir. En fazla kullanım değeri 7. Saniyede orta olarak değerlenmiştir.

Şekil 4.19. Flutter Çerçevesi Samsung Galaxy A50 Enerji Kullanım Oranları



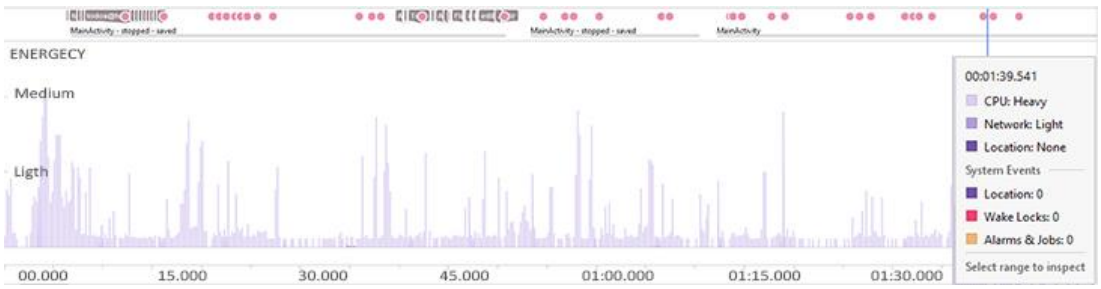
İkinci test cihazı Xiaomi Mi A3 ile yapılan enerji test sonucu Flutter çerçevesi için Şekil 4.20.'deki gibidir. En fazla kullanım değeri 4. Saniyede yüksek olarak değerlenmiştir.

Şekil 4.20. Flutter Çerçevesi Xiaomi Mi A3 Enerji Kullanım Oranları



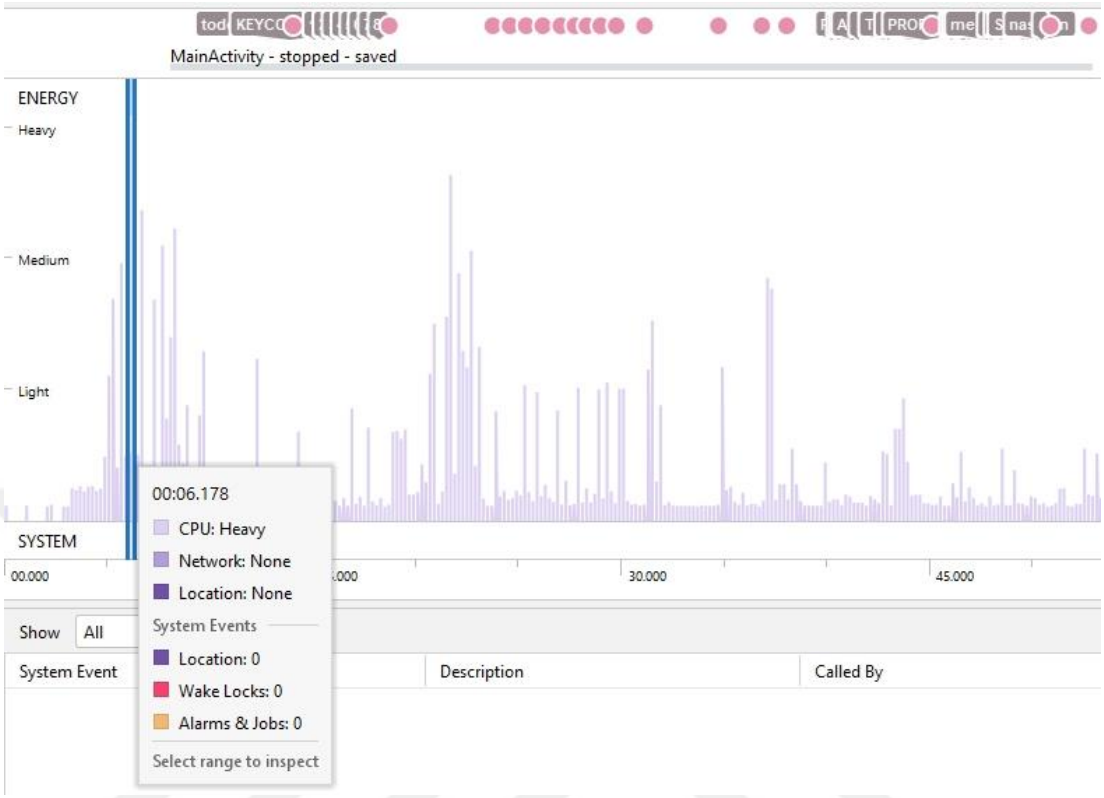
Üçüncü test cihazı Samsung Galaxy A51 ile yapılan enerji test sonucu Flutter çerçevesi için Şekil 4.21.'deki gibidir. En fazla kullanım değeri 1.Dakika 39. Saniyede yüksek olarak değerlenmiştir.

Şekil 4.21. Flutter Çerçevesi Samsung Galaxy A51 Enerji Kullanım Oranları



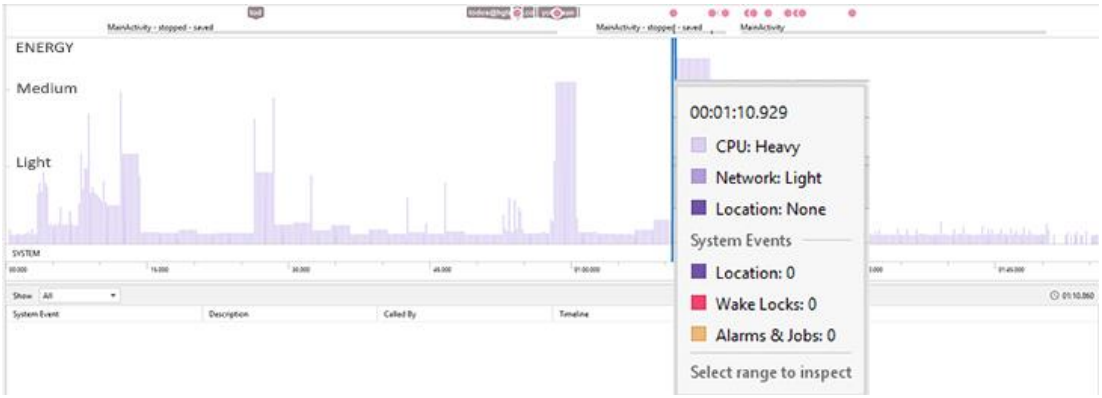
React Native ile yazılan uygulamanın Samsung Galaxy A50 test cihazında enerji kullanım oranları Şekil 4.22.'deki gibidir. En fazla kullanım değeri 6. Saniyede yüksek olarak değerlenmiştir.

Şekil 4.22. React Native Samsung Galaxy A50 Enerji Kullanım Oranları



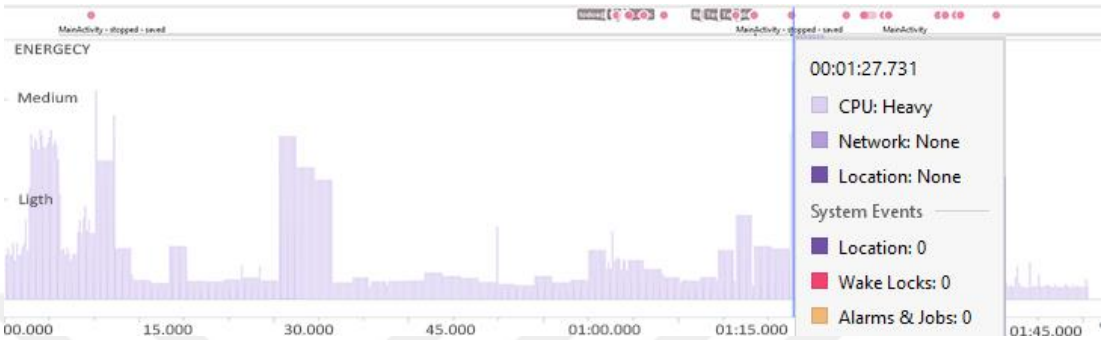
React Native ile yazılan uygulama, ikinci test cihazı olan Xiaomi Mi A3 ile incelendiğinde enerji kullanım oranları Şekil 4.23.'deki gibidir. En fazla kullanım değeri 1. Dakika 10. Saniyede yüksek olarak değerlenmiştir.

Şekil 4.23. React Native Xiaomi Mi A3 Enerji Kullanım Oranları



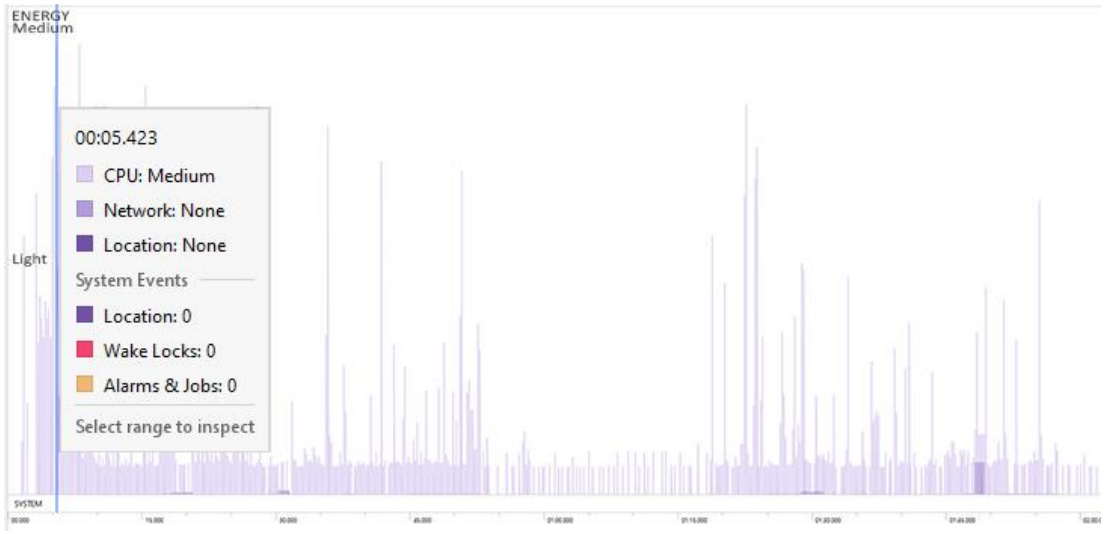
React Native ile yazılan uygulama, üçüncü test cihazı ile incelendiğinde enerji kullanım oranları Şekil 4.24.'deki gibidir. En fazla kullanım değeri 1. Dakika 27. Saniyede yüksek olarak değerlenmiştir.

Şekil 4.24. React Native Samsung Galaxy A51 Enerji Kullanım Oranları



Xamarin ile yazılan uygulama, birinci test cihazı ile incelendiğinde enerji kullanım oranları Şekil 4.25.'deki gibidir. En fazla kullanım değeri 5. Saniyede orta olarak değerlenmiştir.

Şekil 4.25. Xamarin Samsung Galaxy A50 Enerji Kullanım Oranları



Xamarin ile yazılan uygulama, ikinci test cihazı olan Xiaomi Mi A3 ile incelendiğinde enerji kullanım oranları Şekil 4.26.'deki gibidir. En fazla kullanım değeri 17. Saniyede yüksek olarak değerlenmiştir.

Şekil 4.26. Xamarin Xiaomi Mi A3 Enerji Kullanım Oranları

Üçüncü test cihazı ile çalıştırılan Xamarin uygulaması, incelendiğinde enerji kullanım Şekil 4.27.'de görüldüğü gibi 1. Dakika 18. Saniyede yoğun olarak değerlenmiştir.

Şekil 4.27. Xamarin Samsung Galaxy A51 Enerji Kullanım Oranları

4.1.4. Ağ Kullanımı

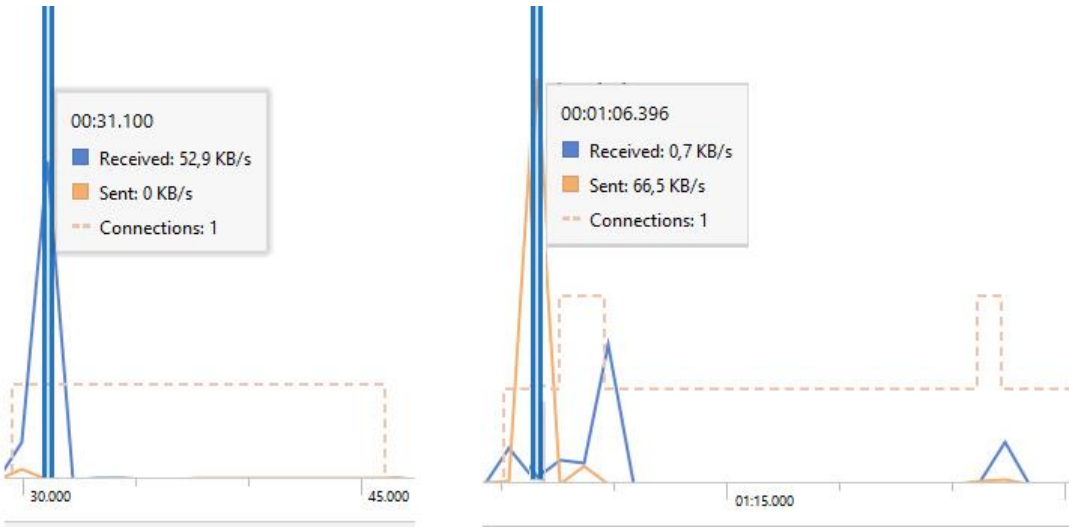
Bu bölümde test cihazlarında Tablo 3.3.'de yer aldığı aşamalar uygulanarak ağ kullanımlarının değerleri incelenecektir. Test aşamasında üye girişinde, yapılacaklar listenin getirilmesinde, yeni konu oluşturulmasında ve dosya ekinin yüklenmesinde ağ istekleri yapılmıştır. Yüklenen dosya biçimi JPG formatındadır. Tüm test cihazlarında ve çerçevelerinde aynı resim yüklenmiştir.

Flutter ile yazılan uygulamadaki birinci test cihazı ağ istekleri sonuç grafiği Şekil 4.28.'de verilmiştir. En fazla indirme işlemi 38. Saniyede 55,1 KB/s, yükleme işlemi ise 1.dakika 21. Saniyede 30,6 KB/s' dir.

Şekil 4.28. Flutter Samsung Galaxy A50 Ağ Kullanım Oranları

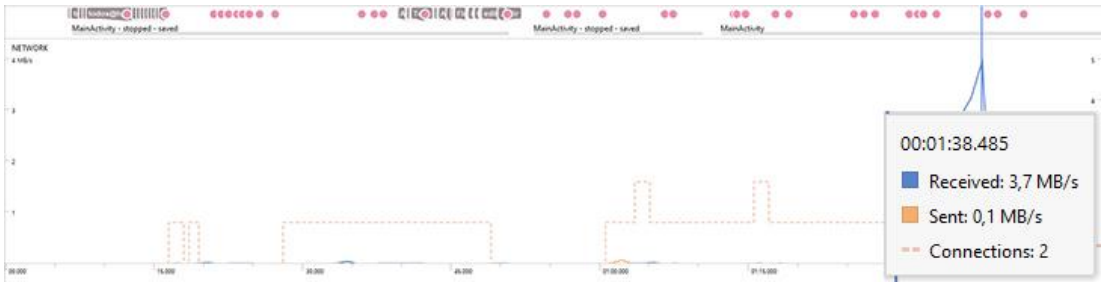
İkinci test cihazı ile Flutter uygulaması için ağ istekleri sonuç grafiği Şekil 4.29.'de verilmiştir. En fazla indirme işlemi 31. Saniyede 52,9 KB/s, yükleme işlemi ise 1.dakika 6. Saniyede 66,5 KB/s' dir. Gönderme ve yükleme istek bağlantı sayıları 1 adettir.

Şekil 4.29. Flutter Xiaomi Mi A3 Ağ Kullanım Oranları



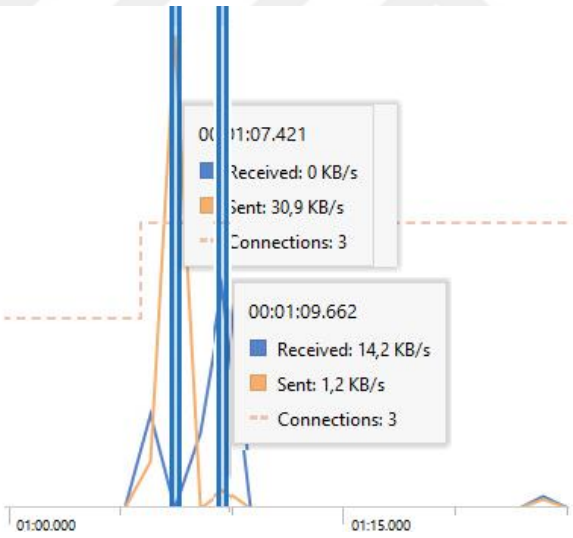
Üçüncü test cihazı ile Flutter uygulaması için ağ istekleri sonuç grafiği Şekil 4.30.'de verilmiştir. En fazla indirme ve yükleme işlemi 1. Dakika 38. Saniyede gerçekleşmiştir. 3,7 MB/s yükleme, 0,1 MB/s ise indirmedir. Bağlantı sayıları 2 adettir.

Şekil 4.30. Flutter Samsung Galaxy A51 Ağ Kullanım Oranları



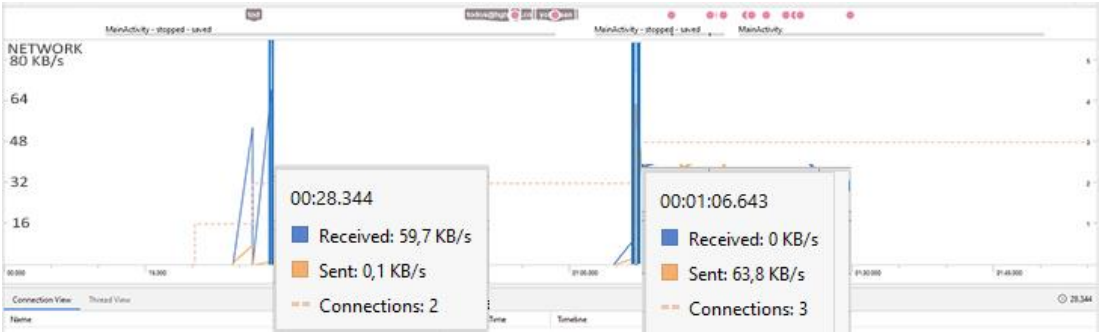
React Native ile yazılan uygulamadaki birinci test cihazı ağ istekleri sonuç grafiği Şekil 4.31.'de verilmiştir. En fazla indirme işlemi 1.Dakika 9. Saniyede 14,2 KB/s, yükleme işlemi ise 1.dakika 7. Saniyede 30,9 KB/s' dir. Her iki indirme ve yükleme ağ isteklerinde bağlantı sayıları 3 adettir.

Şekil 4.31. React Native Samsung Galaxy A50 Ağ Kullanım Oranları



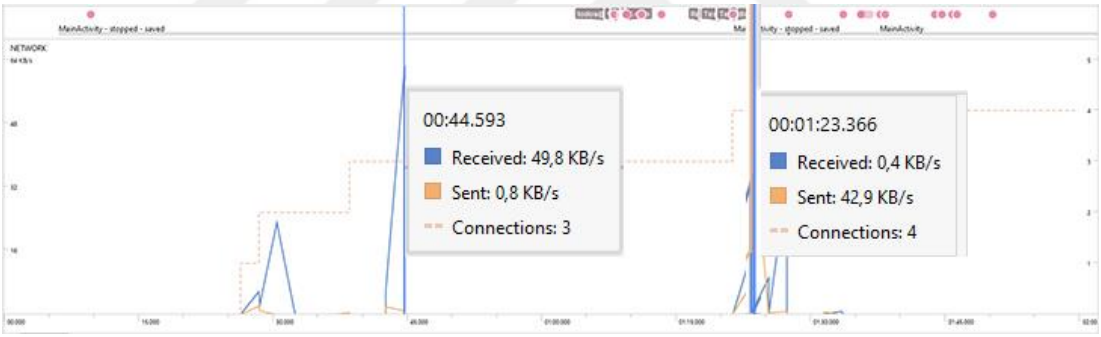
Ağ kullanım oranlarında ikinci test cihazı ile test edilen React Native uygulaması için ağ istekleri sonuç grafiği Şekil 4.32.'de görüldüğü gibi en fazla indirme işlemi 28. Saniyede 59,7 KB/s, yükleme işlemi ise 1. Dakika 6. Saniyede 63,8 KB/s'dir. İndirme işleminde ağ isteğinde bulunduğu bağlantı sayısı 2, yükleme işleminde ise 3 adettir.

Şekil 4.32. React Native Xiaomi Mi A3 Ağ Kullanım Oranları



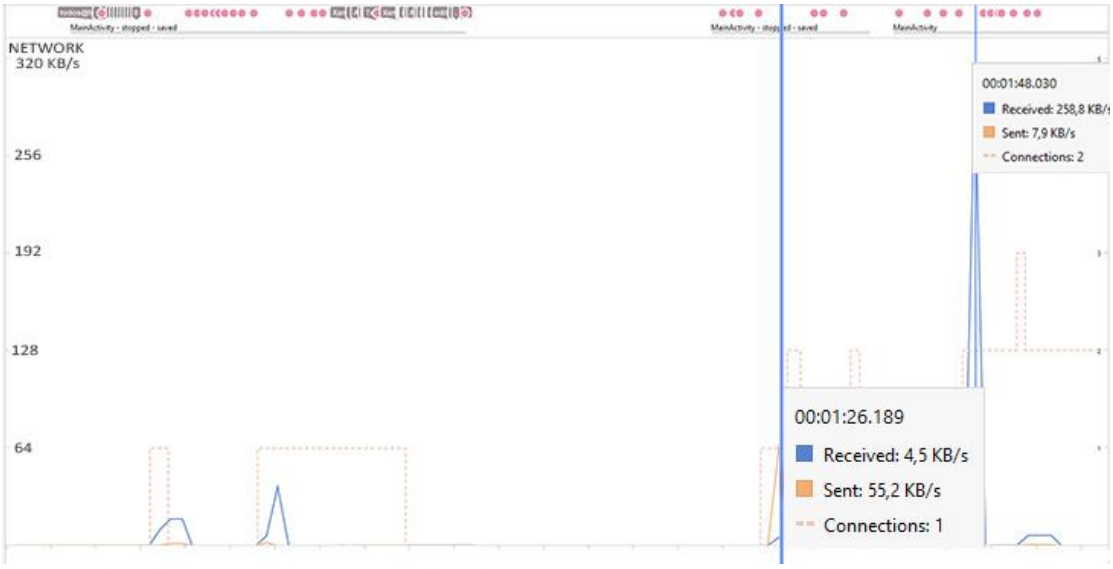
Üçüncü test cihazı ile test edilen React Native uygulaması için ağ istekleri sonuç grafiği ise Şekil 4.33.'de görüldüğü gibi en fazla indirme işlemi 4. Saniyede 49,8 KB/s, yükleme işlemi ise 1. Dakika 23. Saniyede 42,9 KB/s'dir. İndirme işleminde ağ isteğinde bulunduğu bağlantı sayısı 3, yükleme işleminde ise 4 adettir.

Şekil 4.33. React Native Samsung Galaxy A51 Ağ Kullanım Oranları



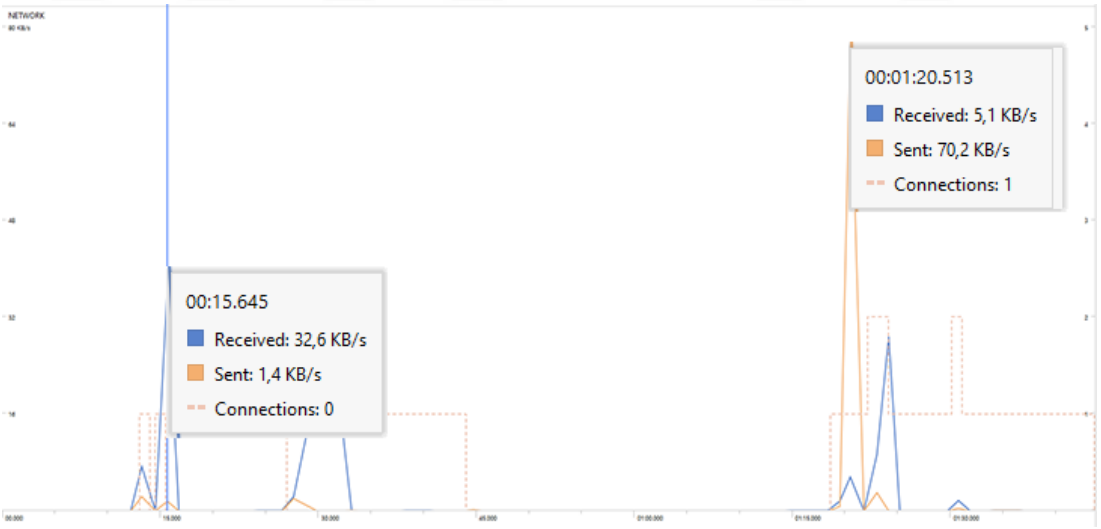
Xamarin ile yazılan uygulamadaki birinci test cihazı ağ istekleri sonuç grafiği Şekil 4.34.'de verilmiştir. En fazla indirme işlemi 1.Dakika 9. Saniyede 55,2 KB/s, yükleme işlemi ise 1.dakika 7. Saniyede 258,89 KB/s' dir. İndirme işleminde ağ isteğinde bulunduğu bağlantı sayısı 2, yükleme işleminde ise 1 adettir.

Şekil 4.34. Xamarin Samsung Galaxy A50 Ağ Kullanım Oranları



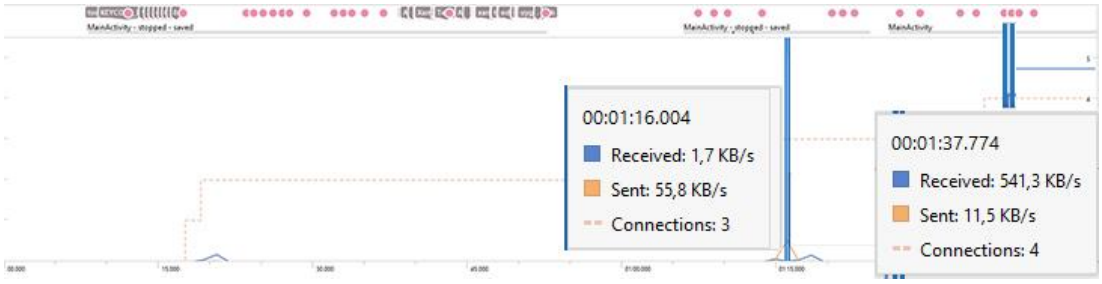
İkinci test cihazı ile test edilen Xamarin uygulaması için ağ istekleri sonuç grafiği ise Şekil 4.35.'de görüldüğü gibi en fazla indirme işlemi 15. Saniyede 32,6 KB/s, yükleme işlemi ise 1. Dakika 20. Saniyede 70,2 KB/s'dir. İndirme işleminde ağ isteğinde bulunduğu bağlantı sayısı 0 yükleme işleminde ise 1 adettir.

Şekil 4.35. Xamarin Xiaomi Mi A3 Ağ Kullanım Oranları



Üçüncü test cihazı ile test edilen Xamarin uygulaması için ağ istekleri sonuç grafiği ise Şekil 4.36.'de görüldüğü gibi en fazla indirme işlemi 1. Dakika 37. Saniyede 541,3 KB/s, yükleme işlemi ise 1. Dakika 16. Saniyede 55,8 KB/s'dir. İndirme işleminde ağ isteğinde bulunduğu bağlantı sayısı 4, yükleme işleminde ise 3 adettir.

Şekil 4.36. Xamarin Samsung Galaxy A51 Ağ Kullanım Oranları



4.1.5. Uygulama Boyutu

Çapraz platform frameworklerinden Flutter için 2.0.1 sürümü, React Native için 0.64.0 sürümü ve Xamarin için 4.8.0 sürümü kullanılarak uygulamalar geliştirilmiştir. Gerekli kütüphaneler ve eklentiler kurulduktan sonra proje dosya boyutları ölçülmüştür. Android geliştirme ortamı için Windows 10 işletim sistemi kullanılmış olup proje dosya boyutu Flutter için 697 MB, React Native için 493 MB, Xamarin için 357 MB'dir. Uygulama boyuları ise sırasıyla 83,95 MB, 71,47 MB ve 78,37 MB'dır. Ölçüm yapılan platformlardaki boyutlar Tablo 4.1.'de verilmiştir.

Tablo 4.1. Frameworkler İle Geliştirilen Android Uygulamaların Boyutları

Framework Adı	Framework Sürümü	Proje Dosya Boyutu (MB)	Uygulama Boyutu (MB)
Flutter	2.0.1	697 MB	83,95 MB
React Native	0.64.0	493 MB	71,47 MB
Xamarin	4.8.0.1269	357 MB	78,37 MB

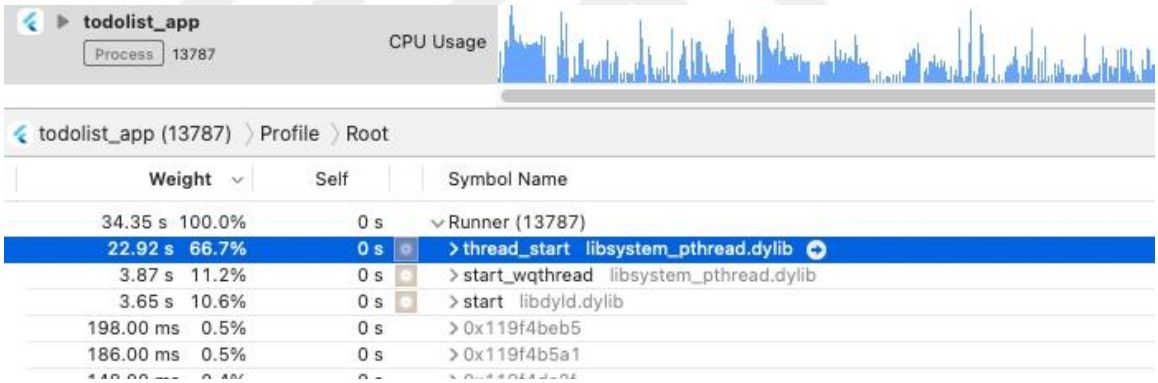
4.2. İos İşletim Sistemli Cihazların Tüketim Bulguları

4.2.1. CPU Kullanımı

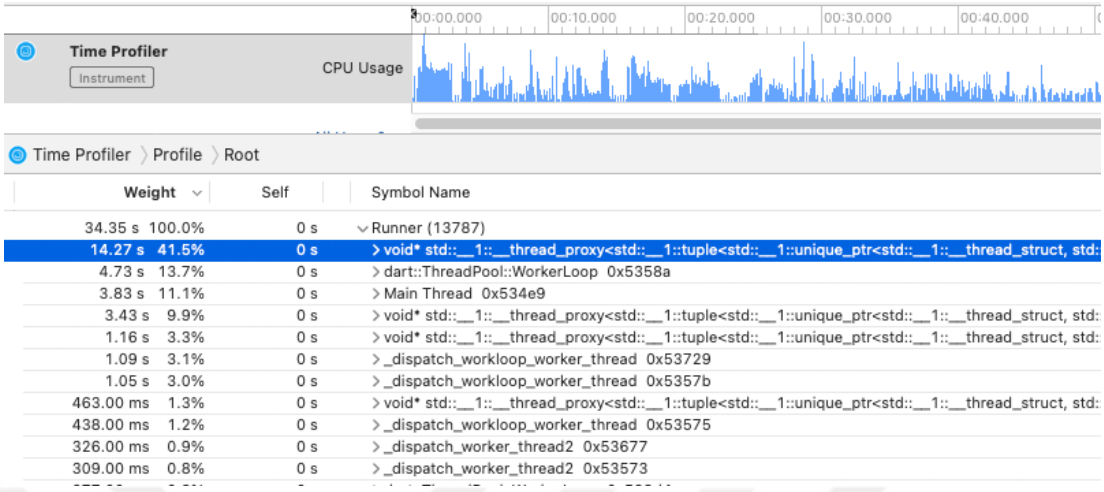
iOS işletim sistemli cihazlar ile çerçeveler ayrı ayrı çalıştırılmıştır. XCode üzerinden Instruments ile sonuçların verildiği grafiklerde dikey eksende CPU kaynağının kullanım oranı yüzde cinsinden, yatay eksende ise saniye cinsinden zaman belirtilmektedir.

Birinci test cihazı iPhone 8 ile çalıştırılan Flutter uygulamasının işlemciye başlangıç yükü olarak kullanım oranı Şekil 4.37.'de gösterildiği şekilde %66.7'dir. İşlemcinin iş parçacıklarına ayrıldığındaki toplam genel kullanım oranı Şekil 4.38.'deki gibi en fazla %41.5, ana iş parçacığı üzerinde kullanım miktarı ise %11.1'dir.

Şekil 4.37. Flutter iPhone 8 Cpu Kullanım Oranları

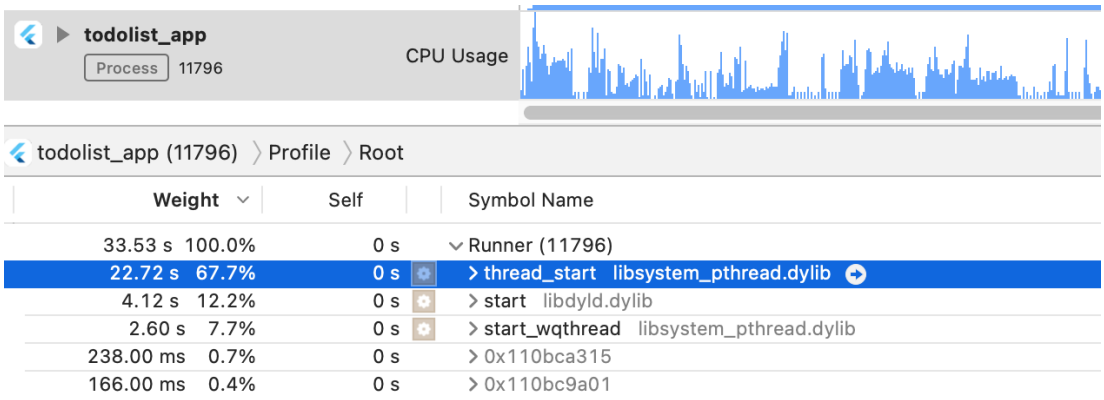


Şekil 4.38. Flutter iPhone 8 Cpu Kullanım Oranları Main Thread

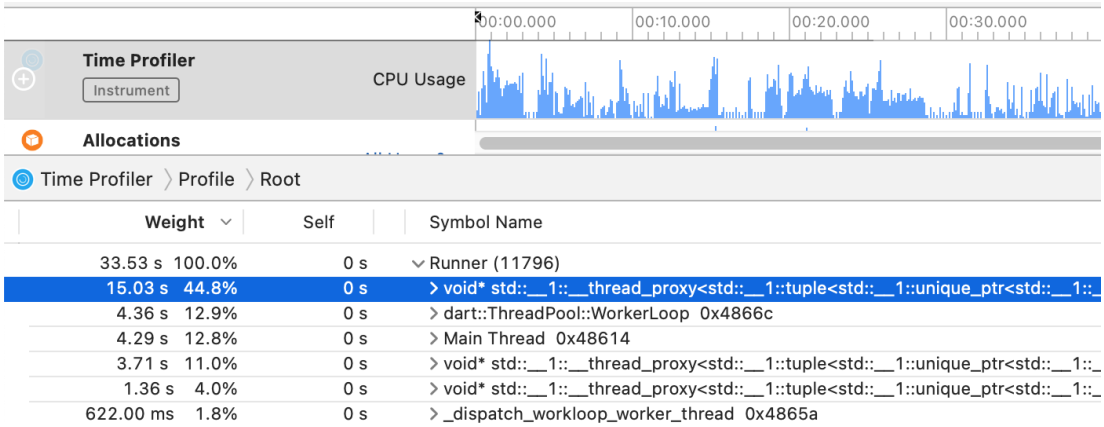


İkinci test cihazı iPhone SE ile çalıştırılan Flutter uygulamasının işlemciye başlangıç yükü %67.7, işlemcinin iş parçacıklarına ayrıldığındaki genel kullanım oranı %44.8, ana iş parçacağı üzerinde kullanım miktarı ise %12.8'dir. Bu oranlar Şekil 4.39. ve Şekil 4.40.'de gösterilmiştir.

Şekil 4.39. Flutter iPhone SE Cpu Kullanım Oranları

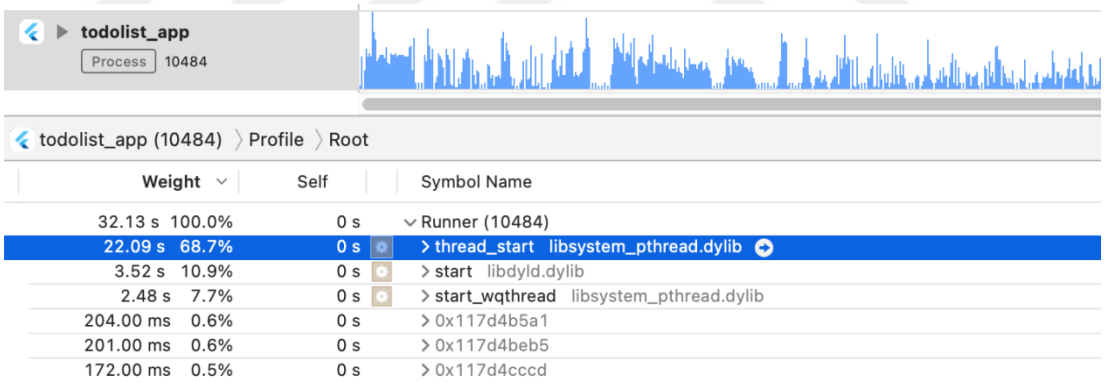


Şekil 4.40. Flutter iPhone SE Cpu Kullanım Oranları Main Thread

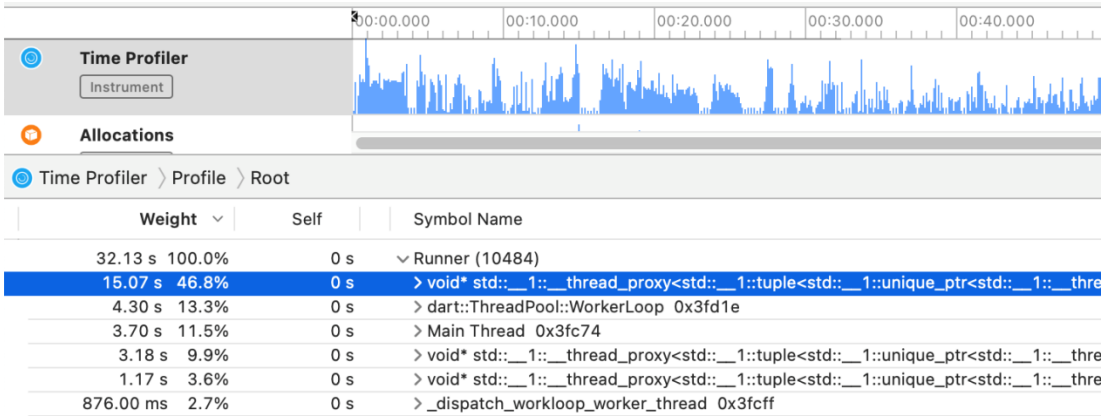


Üçüncü test cihazı iPhone 8 Plus ile çalıştırılan Flutter uygulaması Şekil 4.41.'de gösterildiği gibi işlemcinin iş parçacağı kullanım oranı en fazla %68.7'dir. İşlemcinin iş parçacıklarına ayrıldığındaki toplam genel kullanım oranı en fazla %46.8, ana iş parçacığı üzerinde kullanım miktarı ise %13.3'dir.

Şekil 4.41. Flutter iPhone 8 Plus Cpu Kullanım Oranları

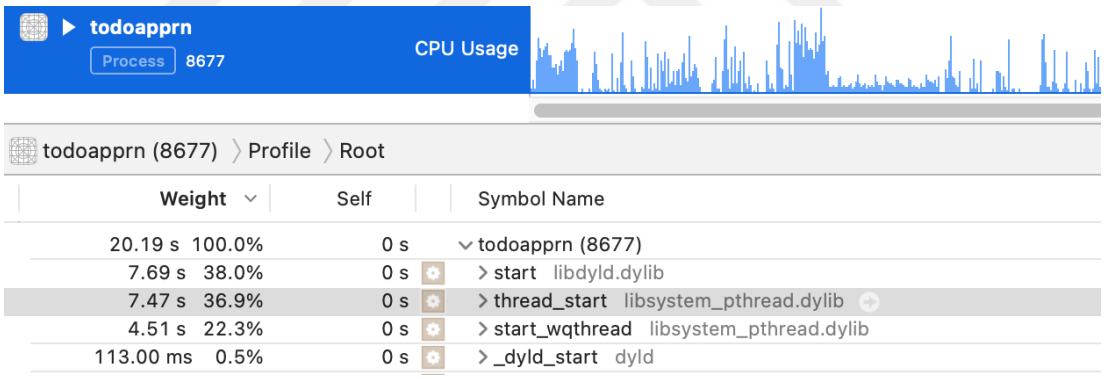


Şekil 4.42. Flutter iPhone 8 Plus Cpu Kullanım Oranları Main Thread

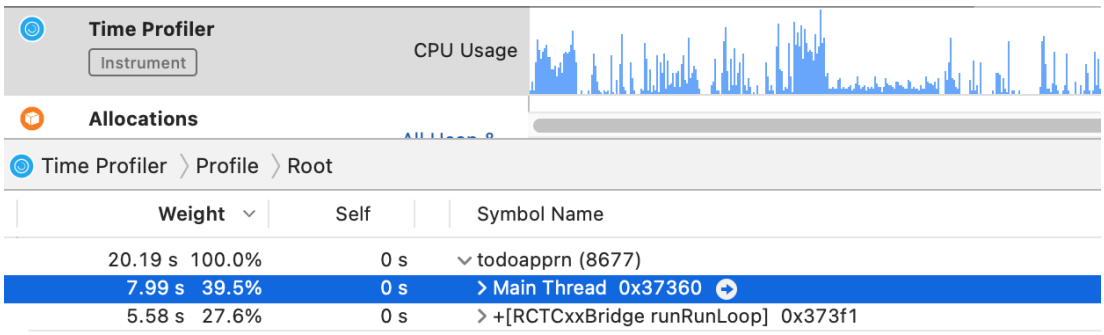


React Native ile geliştirilen uygulama birinci test cihazı ile çalıştırıldığında CPU üzerinde ana iş parçacağındaki ağırlığı %39.5'dur. İş parçacıklarına ayrıldıktan sonra başlama ağırlı ise Şekil 4.43.'deki gibi %36.9'dur.

Şekil 4.43. React Native iPhone 8 Cpu Kullanım Oranları

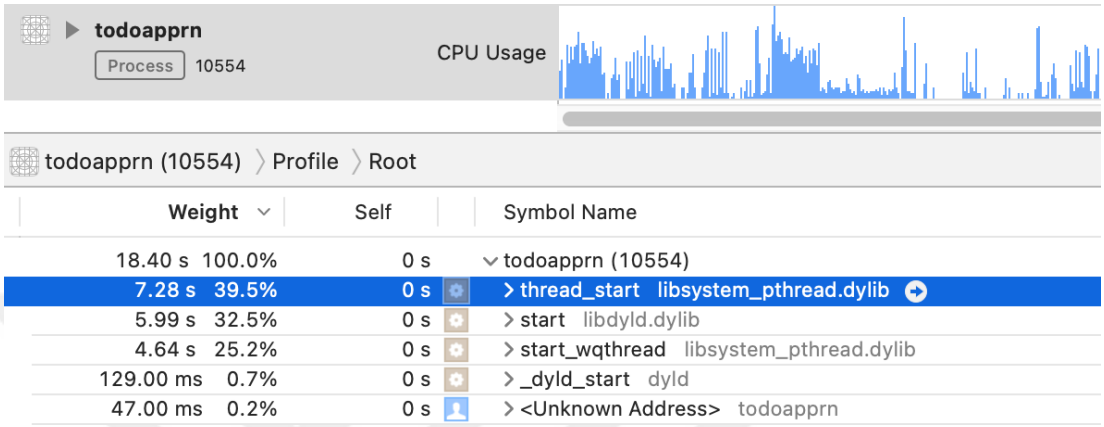


Şekil 4.44. React Native iPhone 8 Cpu Kullanım Oranları Main Thread

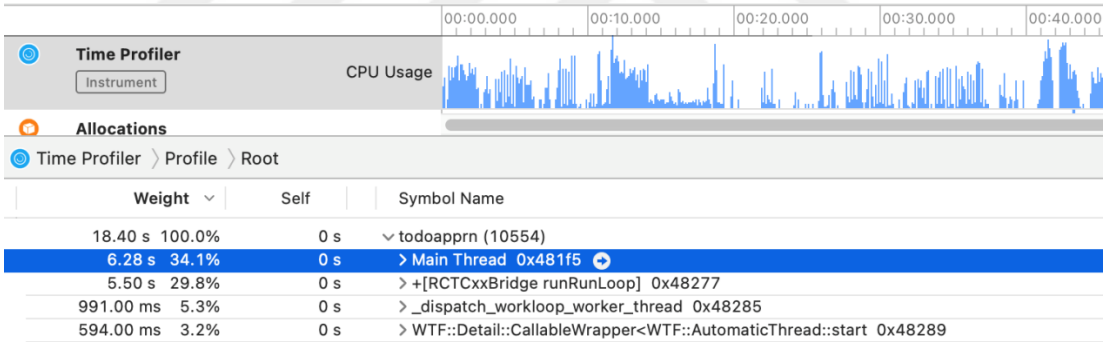


İkinci test cihazı ile çalıştırılan React Native uygulaması CPU üzerinde ana iş parçacağındaki ağırlığı %34.1'dir. İş parçacıklarına ayrıldıktan sonra başlama ağırlı ise Şekil 4.45.'deki gibi %39.5'dur.

Şekil 4.45. React Native iPhone SE Cpu Kullanım Oranları

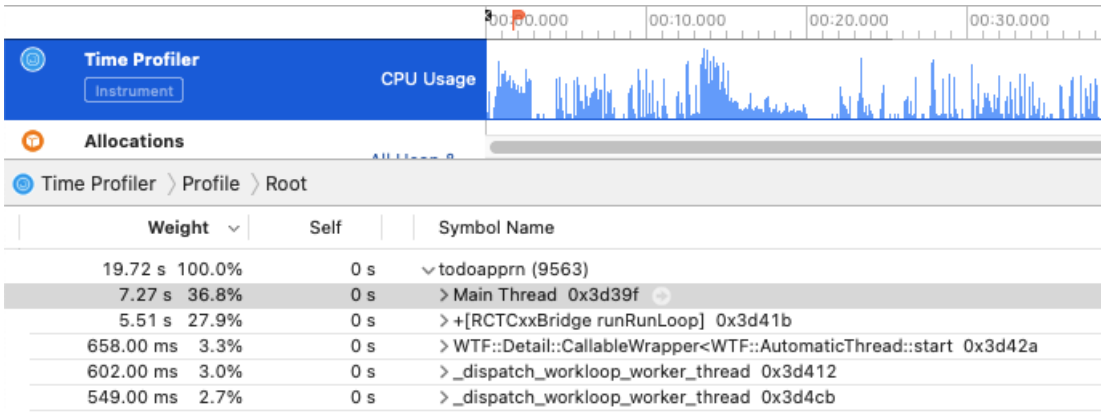


Şekil 4.46. React Native iPhone SE Cpu Kullanım Oranları Main Thread

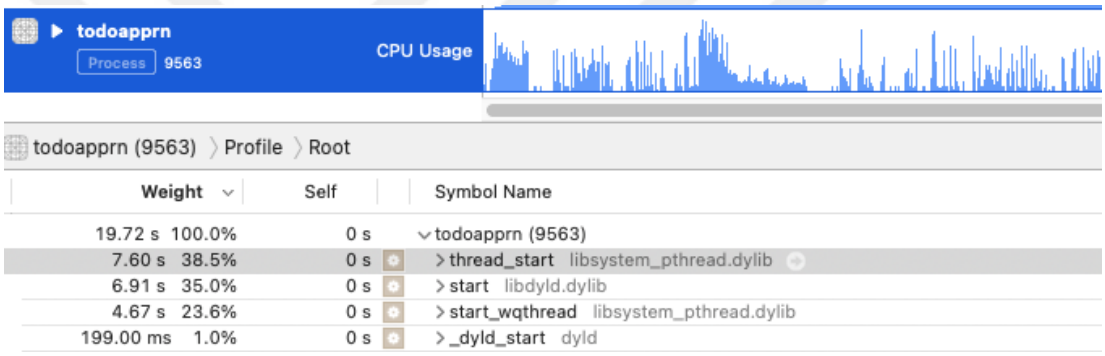


Üçüncü test cihazı ile geliştirilen React Native uygulaması CPU üzerinde ana iş parçacağındaki ağırlığı %36.8'dir. İş parçacıklarına ayrıldıktan sonra başlama ağırlı ise Şekil 4.48.'deki gibi %38.5'dur.

Şekil 4.47. React Native iPhone 8 Plus Cpu Kullanım Oranları Main Thread

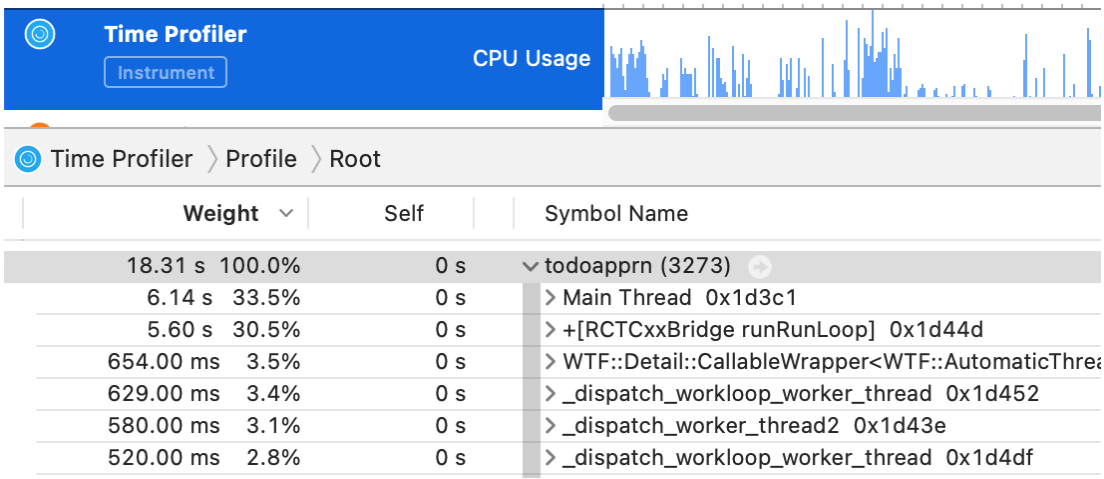


Şekil 4.48. React Native iPhone 8 Plus Cpu Kullanım Oranları

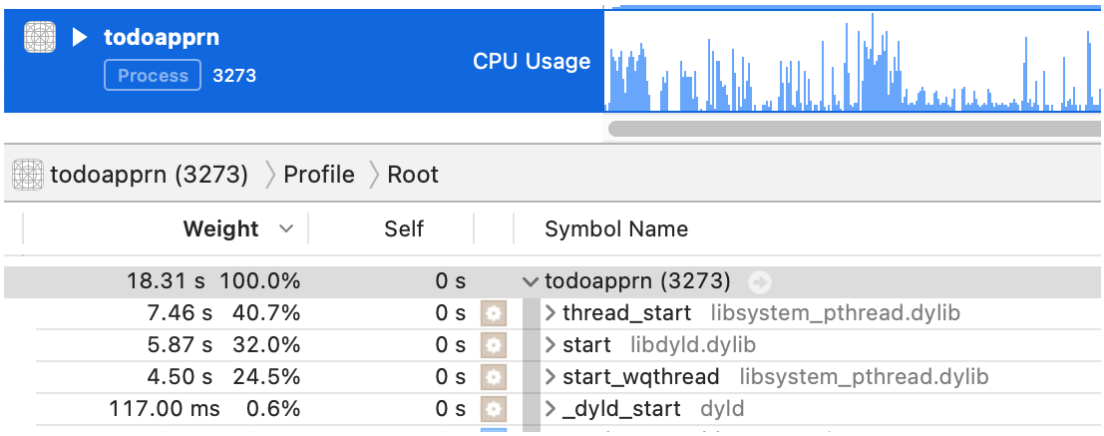


Xamarin ile geliştirilen uygulama birinci test cihazı ile çalıştırıldığında CPU üzerinde ana iş parçacağındaki ağırlığı %33.5'dur. İş parçacıklarına ayrıldıktan sonra başlama ağırlı ise Şekil 4.50.'deki gibi %40.7'dir.

Şekil 4.49. Xamarin iPhone 8 Cpu Kullanım Oranları Main Thread

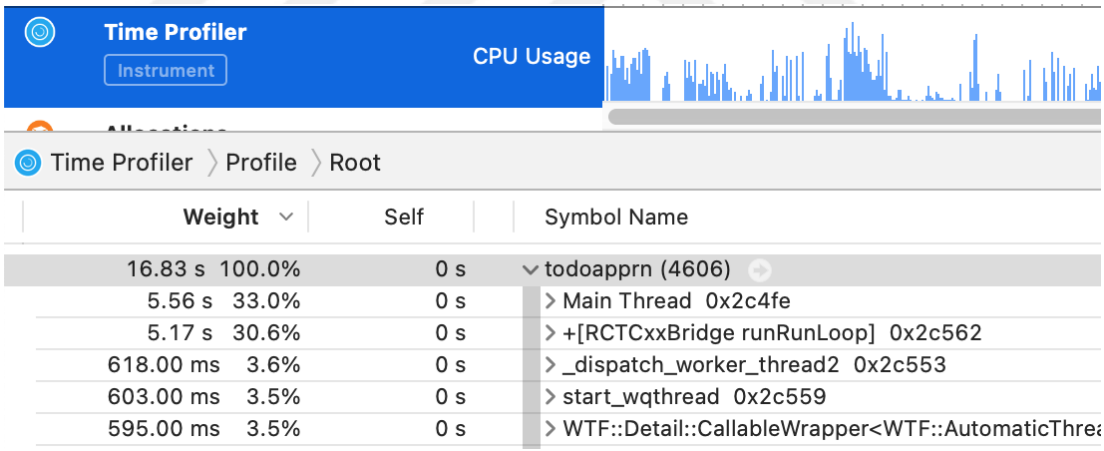


Şekil 4.50. Xamarin iPhone 8 Kullanım Oranları

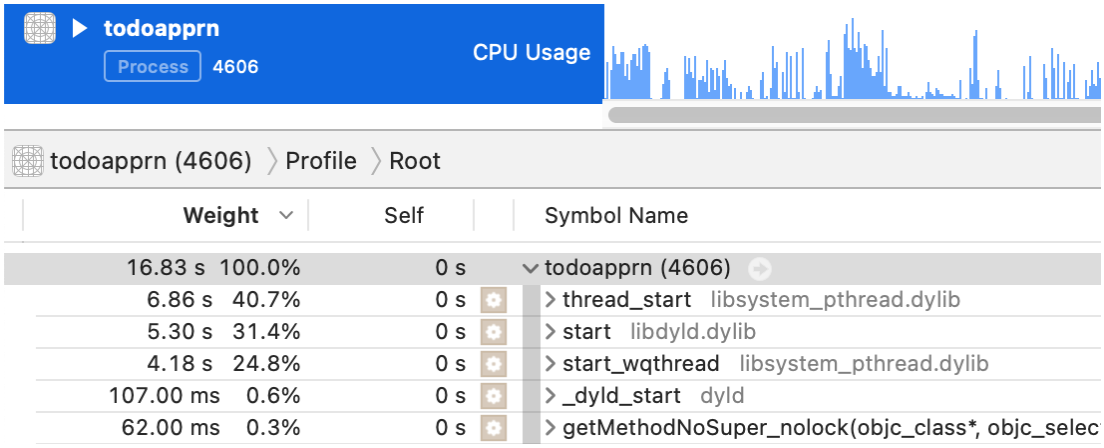


İkinci test cihazı ile çalıştırılan Xamarin uygulaması CPU üzerinde ana iş parçacağındaki ağırlığı %33.0'dir. İş parçacıklarına ayrıldıktan sonra başlama ağırlı ise Şekil 4.52.'deki gibi %40.7'dir.

Şekil 4.51. Xamarin iPhone SE Cpu Kullanım Oranları Main Thread

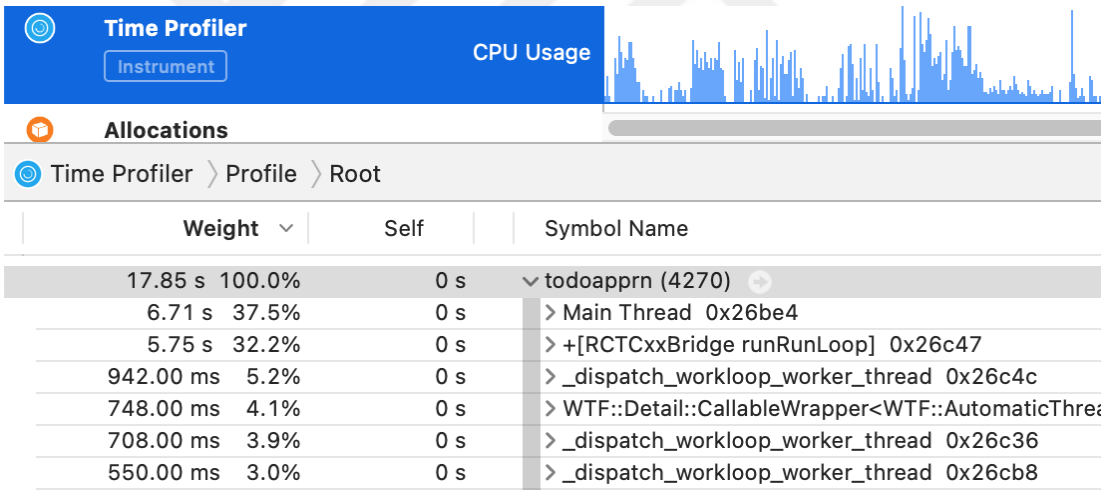


Şekil 4.52. Xamarin iPhone SE Kullanım Oranları

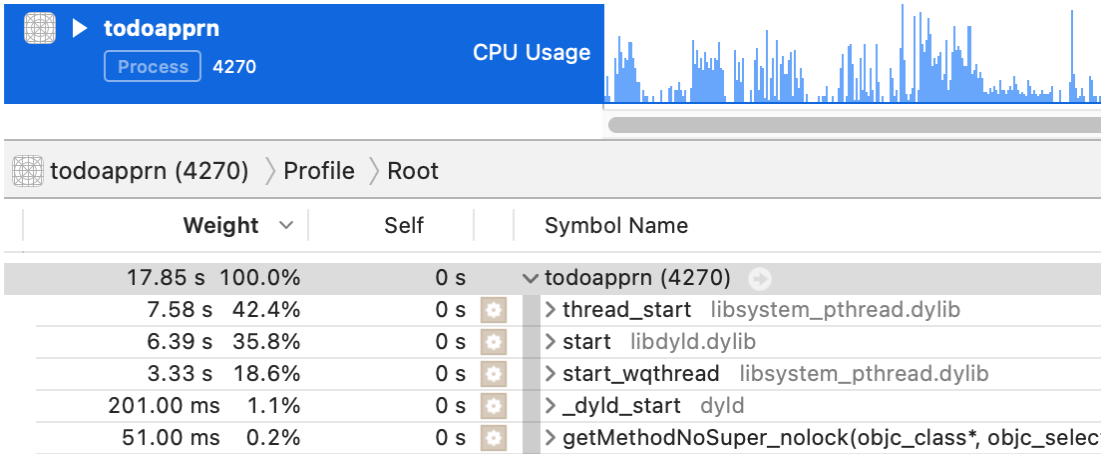


Üçüncü test cihazı ile geliştirilen Xamarin uygulaması CPU üzerinde ana iş parçacağındaki ağırlığı %37.5'dir. İş parçacıklarına ayrıldıktan sonra başlama ağırlı ise Şekil 4.54.'deki gibi %42.4'dir.

Şekil 4.53. Xamarin iPhone 8 Plus Cpu Kullanım Oranları Main Thread



Şekil 4.54. Xamarin iPhone 8 Plus Kullanım Oranları

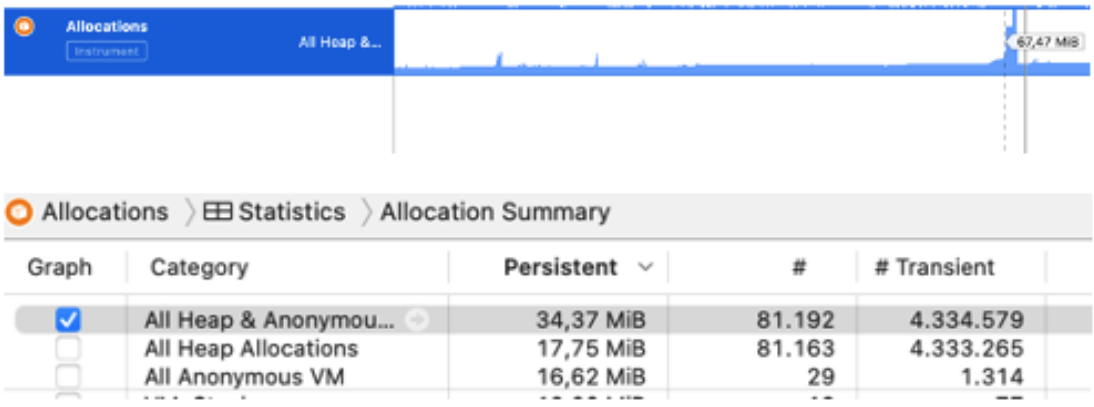


4.2.2. Bellek Kullanımı

İos işletim sistemli cihazlarda bellek kullanımı için Tablo 1.'de yer alan aşamalar tüm çerçeveler için uygulandıktan sonra maksimum kullandığı bellek miktarı MB cinsinden gösterilmiştir.

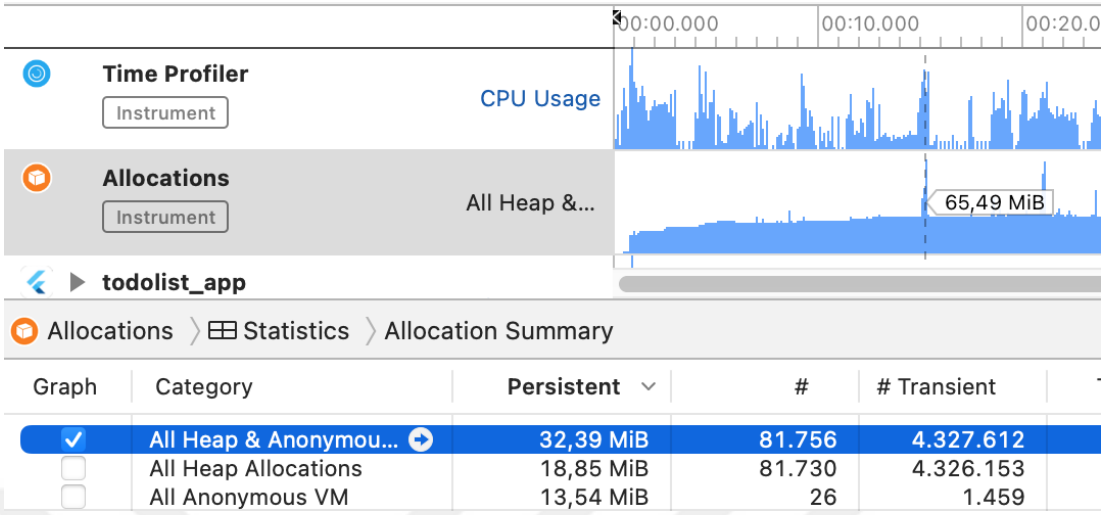
Birinci test cihazı ile Flutter için yapılan bellek kullanımında maksimum 67,47 MB oranında kullanım sağlanmıştır. Sürekli olarak kullandığı bellek miktari ise 34,37 MB'dir. İkinci test cihazına ait bellek kullanım oranı Şekil 4.55.'de verilmiştir.

Şekil 4.55. Flutter iPhone 8 Bellek Kullanım Oranları



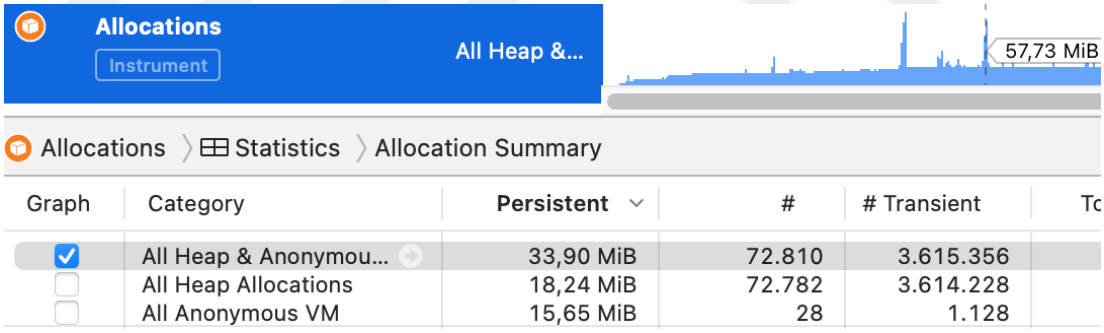
İkinci test cihazı ile Flutter ile yazılan uygulama çalıştırıldığında kullandığı bellek miktarı ise Şekil 4.56.'deki gibi 32,39 MB'dir.

Şekil 4.56. iPhone SE Bellek Kullanım Oranları



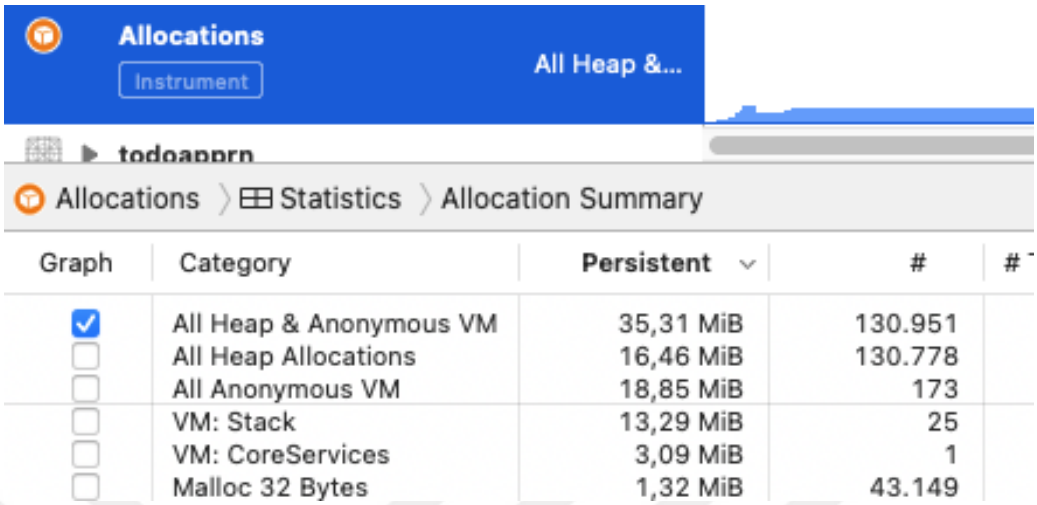
Flutter ile yazılan uygulama için üçüncü test cihazı ile çalıştırılan uygulamanın Şekil 4.57.'deki gibi tüm yığın üzerinde kullandığı bellek miktarı 33,90 MB'dir.

Şekil 4.57. Flutter iPhone 8 Plus Bellek Kullanım Oranları



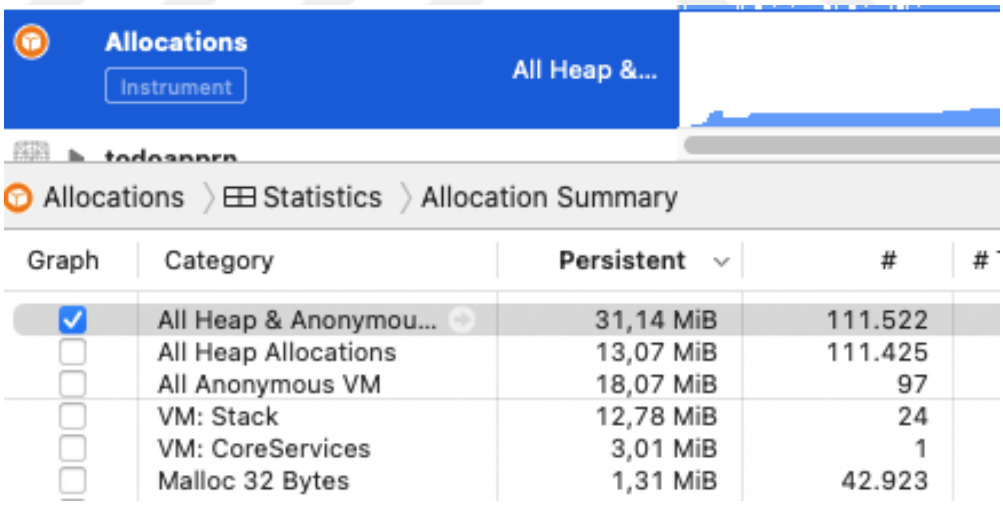
React Native ile geliştirilen uygulama birinci test cihazı ile çalıştırıldığında uygulamanın yığın üzerinde kullandığı bellek miktarı Şekil 4.58.'de gösterildiği gibi 35,31 MB'dır.

Şekil 4.58. React Native iPhone 8 Bellek Kullanım Oranları



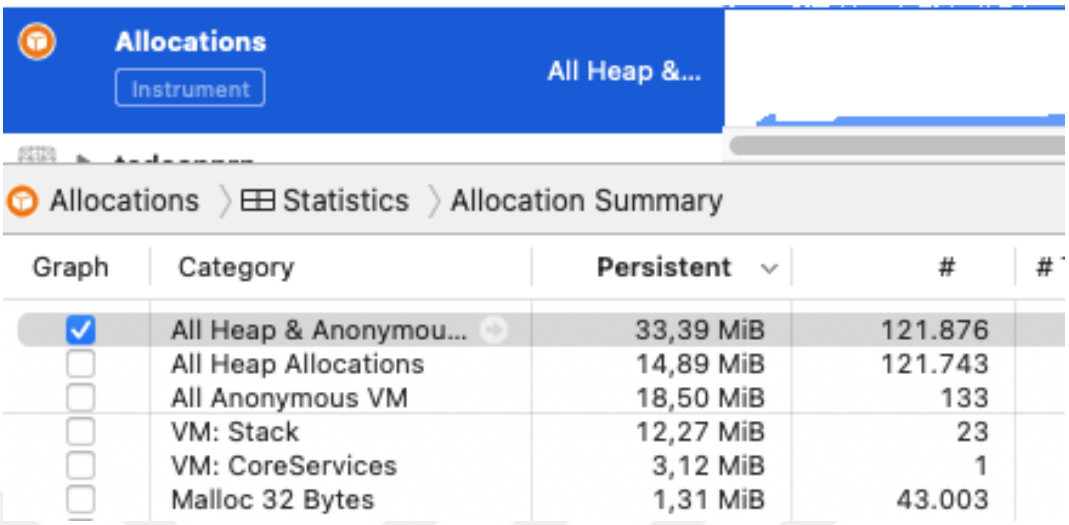
İkinci test cihazı ile React Native için geliştirilen uygulama çalıştırıldığında yığın üzerinde kullandığı bellek miktari ise 31,14 MB'dır. Kullanım miktarı Şekil 4.59.'de gösterilmiştir.

Şekil 4.59. React Native iPhone SE Bellek Kullanım Oranları



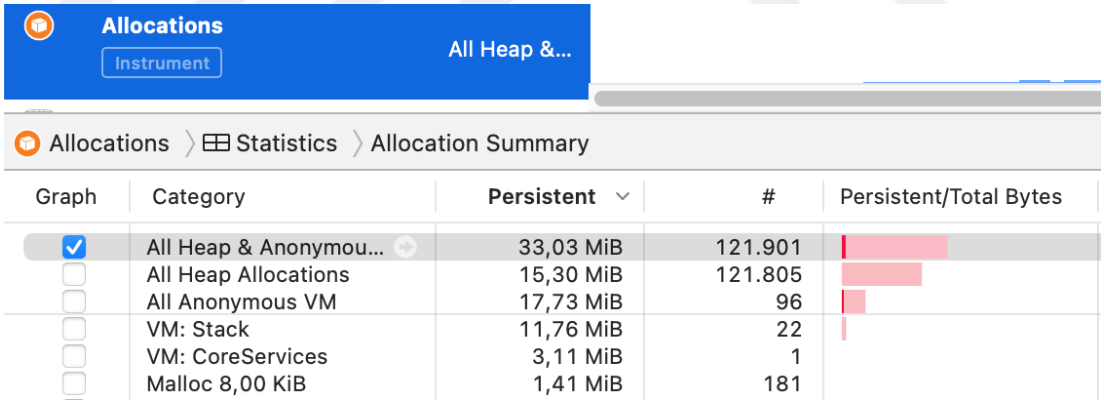
Üçüncü test cihazı ile React Native için geliştirilen uygulama çalıştırıldığında uygulamanın yığın üzerinde kullandığı bellek miktarı ise Şekil 4.60.'de gösterildiği gibi 33,39 MB'dır.

Şekil 4.60. React Native iPhone 8 Plus Bellek Kullanım Oranları



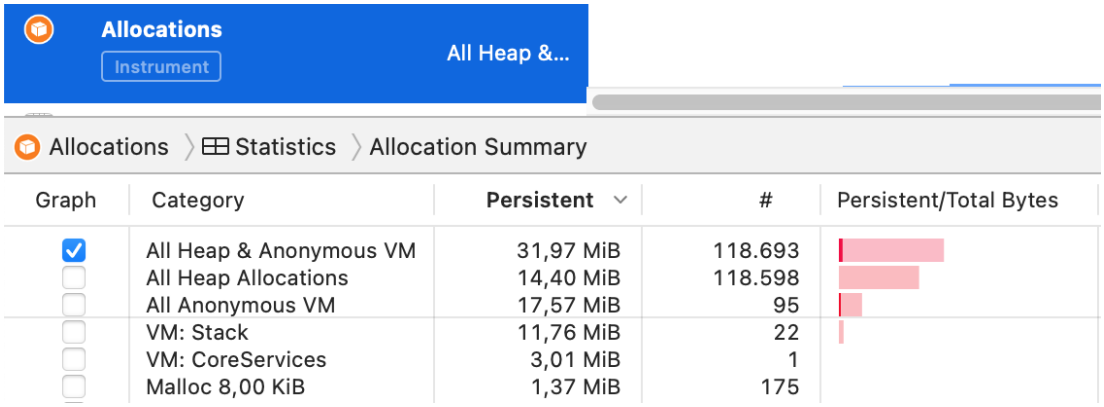
Xamarin ile geliştirilen uygulama birinci test cihazı ile çalıştırıldığında uygulamanın yığın üzerinde kullandığı bellek miktarı ise Şekil 4.61.'da gösterildiği gibi 33,03 MB'dır.

Şekil 4.61. Xamarin iPhone 8 Bellek Kullanım Oranları



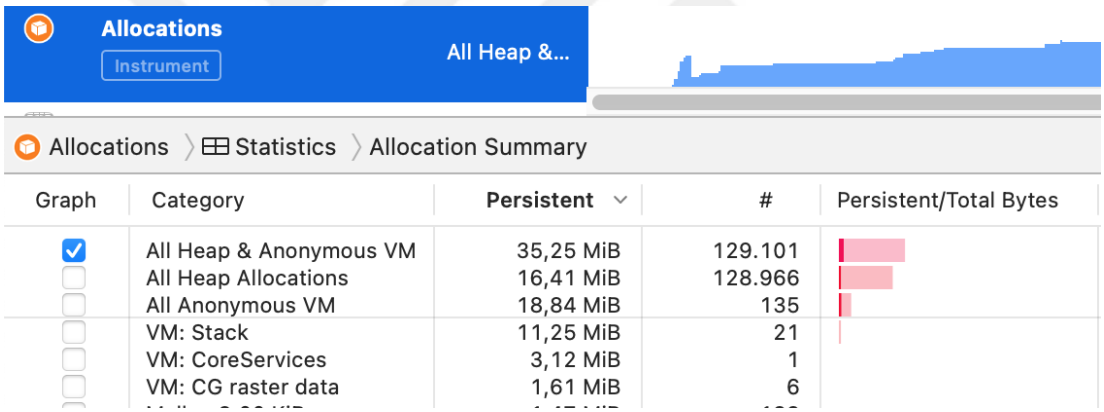
İkinci test cihazı ile Xamarin için geliştirilen uygulama çalıştırıldığında yığın üzerinde kullandığı bellek miktari 31,97 MB'dır. Kullanım miktarı Şekil 4.61.'de gösterilmiştir.

Şekil 4.62. Xamarin iPhone SE Bellek Kullanım Oranları



Üçüncü test cihazı ile Xamarin için geliştirilen uygulama çalıştırıldığında uygulamanın yığın üzerinde kullandığı bellek miktarı ise Şekil 4.63.'de gösterildiği gibi 35,25 MB'dır.

Şekil 4.63. Xamarin iPhone 8 Plus Bellek Kullanım Oranları



4.2.3. Uygulama Boyutu

İos işletim sistemli cihazlar için geliştirilen frameworklerden Flutter için 2.0.1 sürümü, React Native için 0.64.0 sürümü ve Xamarin için 4.8.0 sürümü kullanılarak geliştirilmiştir. Gerekli kütüphaneler ve eklentiler kurulduktan sonra proje dosya boyutları ölçülmüştür. İos geliştirme ortamı için macOS Big Sur işletim sistemi kullanılmış olup proje dosya boyutu Flutter için 1034.24 MB, React Native için 724.4 MB, Xamarin için 834,6 MB'dir. Uygulama boyutları ise sırasıyla 83,95 MB, 71,47 MB ve 76,24 MB'dır. Ölçüm yapılan platformlardaki boyutlar Tablo 4.2'de verilmiştir.

Tablo 4.2. Frameworkler İle Geliştirilen İos Uygulamaların Boyutları

Framework Adı	Framework Sürümü	Proje Dosya Boyutu (MB)	Uygulama Boyutu (MB)
Flutter	2.0.1	1034.24 MB	83,95 MB
React Native	0.64.0	742,4 MB	71,47 MB
Xamarin	4.8.0.1269	834,6 MB	76,24 MB

BEŞİNCİ BÖLÜM

5. SONUÇLAR VE ÖNERİLER

5.1. Sonuçlar

Çapraz platform çerçevelerinin mobil cihazlar üzerindeki performans tüketim değerleri üzerinden incelenen bu araştırmada çapraz platform uygulama geliştirme, platform çerçeveleri, tüketim ve performans kavramlarından bahsedilmiştir. Çapraz platform çerçeveleri ile geliştirilen uygulamaların tüketim değerlerini performans açısından inceleyen çok sayıda çalışma olmasına rağmen test edilen uygulamaların gerçek kullanıcı senaryosuna uygun tasarlanması ve değerlendirilmesi konusunda araştırma boşluğu bulunduğu görülerek incelenmesi kararlaştırılmıştır.

Biørn-Hansen vd. (2019), Huber ve Demetz (2019), Dorfer vd. (2020), Biorn-Hansen vd. (2020) ve Nawrocki P. vd. (2021) çalışmalarında bulunan sonuçlara göre yerel (native) uygulamaların çapraz platform çerçevelerine göre daha performanslı ve daha az tüketim değerlerine sahip olduklarını ve çapraz platform çerçevelerinin tüketim değerlerinde dezavantajlı olduklarını doğrular niteliktedir.

Bu çalışmada Flutter, React Native ve Xamarin çerçeveleri için uygulama aynı tasarım ve özellikler ile ayrı ayrı geliştirildikten sonra mobil işletim sistemleri Android ve iOS'un kendi geliştirdikleri test araçlarıyla tüketim değerleri ölçülmüş, işlemci, bellek, enerji ve ağ kriterlerine göre karşılaştırılmıştır.

Genel olarak araştırma sonuçları incelendiğinde Android ve iOS işletim sistemli cihazlarda Flutter'ın diğer iki çerçeve geliştirme ortamına göre daha az tüketim değerlerine sahip olduğu görülmüştür.

Her iki işletim sistemindeki CPU tüketim değerlerinde Flutter en az, React Native en fazla tüketim değerine, Xamarin çerçevesi ise React Native çerçevesine göre daha az tüketim değerine sahiptir. İos işletim sistemli cihazlarda, işlemci için çalışan uygulamanın yavaş tepki vermesine neden olan ana iş parçacağı üzerindeki yük Flutter çerçevesinde diğerlerine göre daha az tüketim değerlerine sahiptir. Başlangıç için işlemci üzerine binen yükte ise en fazla tüketim Flutter çerçevesinde, React Native ve Xamarin çerçevesinde ise daha az ve birbirlerine yakın düzeydedir.

İlgili literatür incelendiğinde CPU kullanımı üzerine (Nawrocki P. vd., 2021; Biorn-Hansen A. vd., 2020) yapılan çalışmalarda Flutter çerçevesinin diğer çerçevelere göre daha az tüketim değerine sahip olması sonucuna varılması bakımından bu çalışmayla uyumludur. Native, React Native, Xamarin ve Ionic ile mobil uygulama geliştirilip Flutter çerçevesinin yer almadığı, animasyonlar üzerinden test edilen bir başka çalışmada da (Biorn-Hansen A. vd., 2019) CPU kullanımında Xamarin'in React Native'e göre daha az kullanım değerine sahip olduğu sonucuna varılmıştır.

RAM tüketim değerlerinde de Android işletim sistemli cihazlardaki sonuçlarda Flutter çerçevesinin daha düşük tüketim değerine sahip olduğu görülürken, iOS işletim sistemindeki bellek kullanımlarında ise her üç çerçeve için belirli bir fark ortaya çıkmamış aynı düzeylerde tüketim değerlerine sahip oldukları gözlenmiştir. Android işletim sistemli cihazlar üzerinde çalıştırılan uygulamalarda Flutter çerçevesi dışında kalan iki çerçevede tüketim değerlerinde cihazlara göre üstünlükler görüldüğünden net bir sonuç ortaya çıkmamıştır. Birinci test cihazında

React Native tüketim değeri daha fazla iken ikinci test cihazında Xamarin çerçevesi daha fazladır.

Basit (merhaba dünya yazılı) ve detaylı (menüler arası geçişli) olarak nitelendirilen liste gibi belleği zorlamayan, yoğun verilerin yer almadığı iki uygulama üzerinden test edilen başka bir çalışmada (Nawrocki P. vd., 2021) ise RAM kullanımlarında tüketim değerleri konusunda Android işletim sisteminde React Native, iOS işletim sisteminde ise Xamarin çerçevesinin daha az tüketim değerine sahip olduğu sonucuna varılmıştır. İlgili çalışmada her işletim sisteminde bir cihaz üzerinden çerçeveler test edildiği için RAM tüketim değerlerinde bir sonuç elde edilmiştir. Bu çalışma ile ilgili çalışma arasında, işletim sistemlerinde birden fazla cihaz ile test edilip farklı sonuçlar elde edildiği için bellek tüketiminde Flutter çerçevesi dışında diğer iki çerçeve arasında net bir sonuç elde edilmemesi yönünden farklılıklar bulunmaktadır. Ayrıca çerçevelerin çıkarılan her yeni sürümlerinde performans tüketim değerleri üzerine iyileştirilmeler yapılmaktadır. İncelenen çalışmada çerçevelerin sürümlerinin belirtilmemesi ve en son sürümlerinin kullanıp kullanılmadığının bilinmemesi çerçeveler arasındaki tüketim değerleri sırasında değişiklik gösterilmesine neden olabilir. Ama yine de çalışma Native uygulamaların ve diğer çapraz platform uygulamalarının tüketim değerlerini bir arada göstermesi açısından değerlidir. Bir başka Animasyon geçişleri temelli geliştirilen uygulamalar ile test edilen ve Flutter çerçevesinin yer almadığı çalışmada (Biorn-Hansen A. vd., 2019) Android ve iOS işletim sistemli cihazda React Native çerçevesinin Xamarin'e göre daha az bellek kullandığı görülmüştür. İlgili çalışmada kullanılan animasyon temelli farklı üç uygulama ile bu çalışmada yer alan yapılacaklar listesi uygulamasının farklı veriler tutmasından ötürü bellek sonuçlarındaki sıralamalarda farklılıklar bulunmaktadır.

Batarya tüketiminde tüm Android cihazlar yoğun kullanım derecesine sahipken Flutter ve Xamarin çerçevelerinde birinci test cihazı ile ölçüm yaparken orta düzeyde kullanım görülmüştür. React Native ve Xamarin çerçevelerinin ise birbirlerine yakın tüketim değerlerine sahiptirler. Ayrıca Android işletim sistemli cihazlarda CPU tüketiminin yüksek olduğu zamanlarda batarya tüketiminin de yoğun derecesinde olduğu sonucuna varılmıştır. Ağ tüketim değerlerinde ise uygulamada

aynı işlemler ve dosyalar yüklenmesine rağmen React Native çerçevesinde daha fazla ağ isteğinde bulunulduğu gözlenmiştir.

For döngüsü ile 1000 adet yazı listesi ile test edilen Flutter, React Native, Xamarin ve NativeScript çerçeveleri ile incelenen çalışmada (Isıtan ve Koklu, 2020) batarya tüketim değerleri ve ağ istek sayıları tüketim değerleri bakımından ilgili çalışma ile bu kriterler benzerdir. İncelenen çalışmada test için yapılan uygulamanın herhangi bir ağ isteğinde bulunmamasına rağmen React Native çerçevesinin ağ istek sayısındaki fazlalılığı ilgili çalışmada da görülmüştür.

Uygulama boyutları konusunda ise her iki işletim sisteminde de hem proje dosyası boyutunda hem de uygulama boyutunda en az boyuta sahip olan çerçeve React Native çerçevesi olduğu görülmektedir. Çerçevelerin uygulama boyutlarını gösteren bir başka çalışmada (Biorn-Hansen vd., 2020) "merhaba dünya" yazılı ve menüler arası geçiş yapılan iki uygulamada test edilen sonuçlara göre Android işletim sistemli cihazlarda Flutter, iOS işletim sistemli cihazlarda ise React Native çerçevesi daha az uygulama boyutuna sahiptir. İlgili çalışma ile Android işletim sistemli cihazlar arasında bulunan boyut farklılıklarındaki sıralamalar uygulamada kullanılan eklentilerden ve çerçevelerin sürümlerinin eski olmasından kaynaklı olabilir.

Sonuç olarak çapraz platform çerçevelerinin tüketim değerleri arasında farklılıklar görülmüş olup Flutter çerçevesinin daha az tüketim değerine sahip olduğu için daha performanslı olduğu sonucuna varılmıştır.

5.2.Öneriler

Mobil işletim sistemlerinin yeni versiyonlarının çıkarılması sonucunda sistem ve tüketim değerleri üzerindeki performans iyileştirmeleri, material design gibi tasarım geliştirme kültürleri üzerinde yenilikler yer alır. Bu yeniliklerin çerçeveler üzerinden geliştirilen uygulamalara uygulayabilmek ve trendlere ayak uydurabilmek için yeterli geliştirici desteğinin olması gereklidir. Yazılım geliştiricilerinin sıklıkla kullandığı

Github, Gitlab ve Stackoverflow sitelerinde Flutter ve React Native'in aktif olduğu ve büyük geliştirici desteğine sahip olduğu bilinmektedir.

Web geliştirme için kullanılan Javascript ve onun kütüphanesi olan React'ın yaygınlığı, React Native'in popülerliğini arttırmada büyük etkendir. Web geliştirme deneyimi olan kişinin mobil uygulama geliştirme ortamına daha kolay adapte olacağından React Native tercih edilebilir. Sürdürülebilir kod yazmak ve kurumsal mimari ile nesne yönelimli programlama yaklaşımını kullanmak isteyenler ise Dart dili ile Flutter çerçevesini tercih edebilirler. React Native için bu yaklaşımları uygulayabilmek için Javascript'in yanında TypeScript diline de hakim olmaları gerekmektedir.

Flutter, React Native ve Xamarin çerçevelerinin açık kaynak kodlu geliştirme projesine sahip olduğu için bildirilen hatalar, çökme raporları ve performans avantajları her yeni sürümde düzeltilmektedir. Çerçevelerin son sürümlerinin kullanılması tüketim değerleri konusunda bir önceki sürüme göre avantaj sağlayacaktır.

KAYNAKÇA

Android, (2020). Measure app performance with Android Profiler, https://developer.android.com/studio/profile/android-profiler adresinden 10 Mart 2021 tarihinde alınmıştır.

Android, (2020). Save key-value data, https://developer.android.com/training/data-storage/shared-preferences adresinden 3 Mart 2021 tarihinde alınmıştır.

Android, (2020). Send a simple request, https://developer.android.com/training/volley/simple adresinden 5 Mart 2021 tarihinde alınmıştır.

Apple, (2019). Instruments Overview, https://help.apple.com/instruments/mac//#/dev7b09c84f5 adresinden 15 Mart 2021 tarihinde alınmıştır.

Apple, (2019). Track CPU core and thread use, https://help.apple.com/instruments/mac/current/#/dev44b2b437 adresinden 10 Mart 2021 tarihinde alınmıştır.

Apple, (2020). UserDefaults, https://developer.apple.com/documentation/foundation/userdefaults adresinden 3 Mart 2021 tarihinde alınmıştır.

Ardıç B., Göktürk M. (2009), Kullanılabilir Uygulama Programlama Arayüzleri, 4. *Ulusal Yazılım Mühendisliği Sempozyumu*, 91-92.

Biorn-Hansen A., Grønli T. ve Ghinea G. (2019), Animations in Cross-Platform Mobile Applications: An Evaluation of Tools, Metrics and Performance, *Mobile Computing and Ubiquitous Networking*, 19(9), 6-21.

Biorn-Hansen A., Rşeger C., Gronli T., Majchrzak T. ve Ghinea G. (2020), An empirical investigation of performance overhead in cross-platform mobile development frameworks, Empirical Software Engineering volume, 25, 2997–3040.

Ciman M. ve Gaggi O.(2016), An empirical analysis of energy consumption of cross-platform frameworks for mobile development, *Pervasive and Mobile Computing*, 39 (2017), 214–230.

Dorfer T., Demetez L. Ve Huber S. (2020), Impact of mobile cross-platform development on CPU, memory and battery of mobile devices when using common mobile app features, *Procedia Computer Science*, 175 (2020), 189–196.

Eisenman, B. (2016). *Learning React Native: Building Native Mobile Apps with JavaScript* (1.Basım). Sebastopol: O'Reilly Media.

Flutter, (2021). Faq Flutter, https://flutter.dev/docs/resources/faq adresinden 26 Mart 2021 tarihinde alınmıştır.

Flutter, (2021). Flutter architectural overview, https://flutter.dev/docs/resources/architectural-overview adresinden 26 Mart 2021 tarihinde alınmıştır.

Google Firebase, (2021). Firebase Authentication, https://firebase.google.com/docs/auth adresinden 2 Şubat 2021 tarihinde alınmıştır.

Huber S. ve Demetez L. (2019), Performance Analysis of Mobile Cross-platform Development Approaches based on Typical UI Interactions, *14th International Conference on Software Technologies*, 40-48.

Inupakutika D., Kaghyan S., Akopian D., Chalela P. ve Ramirez A. (2020), Facilitating the development of cross-platform mHealth applications for chronic supportive care and a case study, *Journal of Biomedical Informatics*, 105(2020).

Ionic, (2021). Ionic Framework Docs, https://ionicframework.com/docs 6 Mart 2021 tarihinde alınmıştır.

Isıtan M. ve Koklu M., (2020), Comparison and Evaluation of Cross Platform Mobile Application Development Tools, *International Journal of Applied Mathematics, Electronics and Computers*, 8(4), 273-281.

Microsoft .NET, (2021). What is Xamarin? https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin adresinden 30 Ocak 2021 tarihinde alınmıştır.

Microsoft Docs, (2021). Xamarin nedir?, https://docs.microsoft.com/tr-tr/xamarin/get-started/what-is-xamarin#xamarinandroid adresinden 30 Ocak 2021 tarihinde alınmıştır.

Microsoft Docs, (2021). Xamarin. Android Mimari, https://docs.microsoft.com/tr-tr/xamarin/android/internals/architecture adresinden 30 Ocak 2021 tarihinde alınmıştır.

Microsoft Docs, (2021). Xamarin. iOS Mimari, https://docs.microsoft.com/tr-tr/xamarin/ios/internals/architecture adresinden 30 Ocak 2021 tarihinde alınmıştır.

Microsoft, (2020). Hiyerarşik Gezinme, https://docs.microsoft.com/tr-tr/xamarin/xamarin-forms/app-fundamentals/navigation/hierarchical adresinden 5 Mart 2021 tarihinde alınmıştır.

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

NpmStat, (2021). Download statistics for package clone, https://npm-stat.com/charts.html?package=clone&from=2020-01-01&to=2020-12-31 adresinden 8 Mart 2021 tarihinde alınmıştır.

Öberg, L. (2016). Evaluation of Cross-Platform Mobile Development Tools. Bağımsız yüksek lisans tezi, Umeå Üniversitesi, İsveç.

Perfetto Docs, (2021). System profiling, app tracing and trace analysis, https://perfetto.dev/docs/ adresinden 15 Şubat 2021 tarihinde alınmıştır.

Qing, Z., Ying, L., Yuan, P. ve Sheng, L. (2015). Music Player Based on the Cordova Cross-Platform. *3rd International Conference on Applied Computing and Information Technology*, 451-453.

React Native, (2021). Core Components and Native Components, https://reactnative.dev/docs/intro-react-native-components adresinden 30 Ocak 2021 tarihinde alınmıştır.

Riehle, D. (2000). Framework Design: A Role Modeling Approach. Doktora tezi, Hambur Üniversitesi, İsviçre.

Rodriguez, A. (2008). RESTful Web services: The basics, *IBM Developer Works*, 1-11.

Stackoverflow, (2020). Most Loved, Dreaded, and Wanted Other Frameworks, Libraries, and Tools, https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-all-respondents adresinden 10 Ocak 2021 tarihinde alınmıştır.

StatCounter Global Stats, (2021). Mobile & Tablet Operating System Market Share Worldwide,https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/#monthly-200901-202012 adresinden 10 Ocak 2021 tarihinde alınmıştır.

Statista, (2020). Cross-platform mobile frameworks used by software developers worldwide in 2019 and 2020, https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/ adresinden 5 Haziran 2021 tarihinde alınmıştır.

Tilkov S. ve Vinoski S, (2010). Node.js: Using JavaScript to Build High-Performance Network Programs, *IEEE Internet Computing*, 14(6), 80-83.