

T.C.
BEYKENT ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

FLUTTER İLE TO-DO UYGULAMASI GELİŞTİRME
Yüksek Lisans Projesi

Projeyi Hazırlayan:
MUSTAFA GÜNEY

İstanbul, 2021

T.C.
BEYKENT ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

FLUTTER İLE TO-DO UYGULAMASI GELİŞTİRME
Yüksek Lisans Projesi

Projeyi Hazırlayan:
MUSTAFA GÜNEY
1908020036
(0000-0001-5918-6223)

DANIŞMAN:
DR. ÖĞR. ÜYESİ, TURHAN KARAGÜLER

İstanbul, 2021

YEMİN METNİ

Yüksek Lisans Projesi olarak sunduğum “**FLUTTER İLE TO-DO UYGULAMASI GELİŞTİRME**” başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını belirtir ve bunu onurumla doğrularım./...../.....

Mustafa GÜNEY

Adı ve Soyadı : Mustafa GÜNEY
Danışmanı : Dr. Öğr. Üyesi Turhan KARAGÜLER
Türü ve Tarihi : Yüksek Lisans Projesi, 2021
Alanı : Bilgisayar Mühendisliği
Anahtar Kelimeler : Flutter, Cross Platform Mobile Development, To-Do

ÖZ

FLUTTER İLE TO-DO UYGULAMASI GELİŞTİRME

Günümüzde mobil cihazlar üzerinde çalışan mobil uygulamaların önemi her geçen gün daha da artmaktadır. Artık yer mekân önemi olmadan mobil uygulamalar ile her zaman ve her yerden hizmet alınmakta, hızlı geçen zamana ancak mobil uygulamalar ile ayak uydurulmaktadır.

Bu çalışmada esasen Flutter teknolojisi ile bir To-Do uygulamasının geliştirilmesi ve sonuçlarının değerlendirilmesi amaçlanmıştır.

Söz konusu amaca istinaden mevcut mobil uygulama türlerinden ve teknolojilerinden bahsedilmiş, sonrasında Google tarafından geliştirilmiş, bir çoklu platform teknolojisi olan Flutter ‘ ın özellikleri, mimarisi, avantajları, geleceği anlatılmış ve en büyük rakibi olan React Native teknolojisi ile karşılaştırılmaları yapılmıştır. Daha sonra Flutter ile geliştirilecek olan To-Do uygulamasının gereksinim ve teknik analizleri yapılmış, nihayetinde To-Do uygulamasının Flutter ile kodlama safhasına geçilmiştir. Bu safhada kullanılan araçlar, yazılım projesinin yapısı, yapılandırılması ve yazılan kod ayrıntılarıyla anlatılmış, son kısımda ise kullanıcı arayüzleri ve yazılım kodunun tamamı Gitlab’ da paylaşılmıştır.

Yapılan çalışma sonucunda Flutter ile geliştirilen To-Do uygulamasının tek kod tabanında Andorid, IOS ve Web platformlarında başarıyla çalıştığı sonucuna varılmıştır.

Name and Surname : Mustafa GÜNEY
SuperVisor : Asst. Prof. Turhan KARAGÜLER
Degree and Date : Master Project, 2021
Majör : Computer Engineering
Keywords : Flutter, Cross Platform Mobile Development, To-Do

ABSTRACT

DEVELOPING TO-DO APPLICATIONS WITH FLUTTER

Today, the importance of mobile applications running on mobile devices is increasing day by day. Now, regardless of place and place, services are received from anywhere and anytime with mobile applications, and we can keep up with the fast time only with mobile applications.

In this study, it is mainly aimed to develop a To-Do application with Flutter technology and to evaluate the results.

For this purpose, existing mobile application types and technologies were mentioned, and then the features, architecture, advantages and future of Flutter, a multi-platform technology developed by Google, were explained and compared with React Native technology, which is its biggest competitor. Later, the requirements and technical analyzes of the To-Do application, which will be developed with Flutter, were made, and finally, the coding phase of the To-Do application with Flutter was started. The tools used in this phase, the structure of the software project, its configuration and the code written are explained in detail. In the last part, the user interfaces and the entire software code are shared on Gitlab.

As a result of the study, it was concluded that the To-Do application developed with Flutter works successfully on Andorid, IOS and Web platforms on a single code base.

İÇİNDEKİLER

Sayfa No:

ÖZ.....	i
ABSTRACT.....	ii
KISALTMALAR	viii
SÖZLÜK.....	ix
GİRİŞ.....	1
BİRİNCİ BÖLÜM.....	3
ÇOKLU PLATFORM GELİŞTİRME ÜZERİNE LİTERATÜR TARAMASI	3
İKİNCİ BÖLÜM	7
MOBİL UYGULAMA GELİŞTİRME TEKNOLOJİLERİ	7
1. Sayılarla Mobil Uygulama Kullanımları.....	7
2. Mobil Uygulama Geliştirme Türleri	8
2.1. Yerel Uygulamalar (Native App).....	8
2.2. Hibrit Uygulamalar (Hybrit App)	9
2.3. İleri Web Uygulamaları (PWA).....	10
2.4. Çoklu Platform Uygulamaları (Cross-Platform App).....	10
2.4.1 Avantajlar	10
2.4.2 Dezavantajlar	10
3. Çoklu Platform Uygulama Geliştirme Teknolojileri	11
3.1. Flutter	11
3.2. React Native	12
3.3. Xamarin	13
4. Flutter Teknolojisi	14
5. Flutter' ın Mimarisi ve Temelleri	15
5.1. Mimari Yapı	16
5.2. Widgets.....	17

5.3. Kompozisyon (Composition)	18
5.4. Flutter' ın Çalışma Şekli.....	19
5.4.1. Widget Oluşturma (Widget Rendering)	19
5.4.2. Platform Kanalları (Platform Channels).....	19
5.5. Dart' ın Etkisi	20
5.5.1. Çöp Toplayıcı (Garbage Collection)	20
5.6. Flutter ile React Native' in Karşılaştırılması	21
5.6.1. Dilin Dinamik veya Statik Olması	21
5.6.2. Yerleşim Düzeni (Layout)	22
5.6.3. Platform Desteği.....	22
5.6.4. Hızlı Geliştirme (Hot-Reload)	23
5.6.5. Test	23
5.6.6. Yerellik ve Performans	23
5.6.7. Destek	24
5.6.8. Uygulama Boyutu	24
5.6.9. React Native Kullanan Büyük Firmalar	24
5.6.10. Flutter Kullanan Büyük Firmalar	24
5.6.11. Karşılaştırma Sonucu.....	24
ÜÇÜNCÜ BÖLÜM.....	25
UYGULAMANIN GEREKSİNİM VE TASARIM ANALİZİ	25
1. Proje Tanımı	25
1.1. Projeye Genel Bakış	25
1.2. Projenin Amacı	25
1.3. Ürün Kapsamı	25
1.4. Paydaşlar	25
1.5. Kısıtlar	26
1.5.1. Çözüm Kısıtları	26
1.5.2. Veri Tabanı	26
2. Gereksinimler	26
2.1 Kullanım Durum Diyagramı.....	26
2.2 Kullanım Senaryoları:	27
2.2.1 Giriş ve Çıkış (Login & Logout).....	27

2.2.2. Görev Oluşturma (Create Task).....	27
2.2.3. Görevleri Listeleme (List Tasks)	27
2.2.4. Tüm Görevleri Takvim’ de Göster (Show All Tasks On the Cal) ...	27
2.2.5. Görev Arama (Search Tasks).....	27
3. Tasarım	28
3.1. Tasarım Hedeflerinin Tanımlanması	28
3.1.1. Yüksek Performans Hedefi	28
3.1.2. Kullanıcı Dostu Olma Hedefi	28
3.1.3. En Az Hata Hedefi	28
3.1.4. Güvenirlilik Hedefi	28
3.2 Hedeflerin Takası (Trade-offs)	29
3.3 Önerilen Yazılım Mimarisi.....	30
3.3.1 Giriş	30
3.3.2. Sınıf Diyagramları.....	31
3.3.3. Dinamik Model	32
3.3.4 Alt Sistem Ayrıştırması	34
DÖRDÜNCÜ BÖLÜM.....	35
UYGULAMANIN GELİŞTİRİLMESİ	35
1. Flutter Eğitimi	35
2. Kurulum ve IDE	35
3. Proje Yapısı.....	36
4. Proje Konfigürasyonu ve Kullanılan Kütüphaneler	37
5. Uygulama Sınıfları.....	38
5.1. Main Metodu ve MyApp Sınıfı	38
5.2. Login Sınıfı.....	39
5.3. Homepage Sınıfı	40
5.4. TaskPage Sınıfı	41
5.5. CalendarPage Sınıfı.....	43
5.6. Models Sınıfları	44
5.7. TaskCardWidget Sınıfı.....	44
5.8. TodoWidget Sınıfı	45
5.9. DatabaseHelper_fs Sınıfı.....	46

6. My2do Uygulamasının Yazılım Kodu	47
7. Kullanıcı Ara yüzleri	47
SONUÇ	50
KAYNAKÇA.....	51
ÖZGEÇMİŞ.....	54

TABLÖLAR LİSTESİ

Sayfa No:

Tablo 1. Flutter 2.0 Platform Desteği	22
Tablo 2. Hedeflerin Takası.....	29

ŞEKİLLER LİSTESİ

Sayfa No:

Şekil 1. Şubat 2021’ de En Çok İndirilen Uygulamalar	7
Şekil 2. Yerel (Native) Uygulamalar	8
Şekil 3. Hibrit Uygulama Teknolojileri.....	9
Şekil 4. Flutter 2.0’ ın Desteklediği Platformlar	11
Şekil 5. React Native’ in Desteklediği Platformlar	12
Şekil 6. Xamarin’ in Desteklediği Platformlar	13
Şekil 7. Neden Flutter.....	14
Şekil 8. Flutter’ ın Mimarı Yapısı.....	16
Şekil 9. Widget Ağacı	17
Şekil 10. Kompozisyon Widget Yapısı	18
Şekil 11. Flutter’ın Yüksek Seviyede Platform Erişimi	19
Şekil 12. Flutter vs React Native	21
Şekil 13 React Native Mimarisi.....	23
Şekil 14. Flutter Mimarisi.....	23
Şekil 15. My2do Uygulamasının Kullanım Durum Diyagramı.....	26
Şekil 16. Dağıtım Diyagramı	30
Şekil 17. Sınıf Diyagramı	31
Şekil 18. Google Auth Etkileşimi Sıralı Diyagramı	32
Şekil 19. Temel Nesne Etkileşim Sıralı Diyagramı	33
Şekil 20. Bileşen Diyagramı	34
Şekil 21. Proje Yapısı.....	36
Şekil 22. Pubspec.yaml Yapılandırma Dosyası	37
Şekil 23. Giriş ekranı.....	47
Şekil 24. Ana Sayfa - Açık Görevler	47
Şekil 25. Görev Giriş Ekranı 1	48
Şekil 26. Görev Giriş Ekranı 2	48
Şekil 27. Görev Giriş Ekranı 3	48
Şekil 28. Görev Giriş Ekranı 4	48

Şekil 29. Menü.....	49
Şekil 30. Ana Sayfa – Görev Listesi 2	49
Şekil 31. Ana Sayfa – Arama	49
Şekil 32. Takvim.....	49

KISALTMALAR

- AOT** : Ahead Of Time
- APP** : Application
- BSD** : Berkeley Software Distribution
- CPU** : Central Processing Unit
- GC** : Garbage Collection
- GPU** : Graphics Processing Unit
- IDE** : Integrated Development Environment
- IOS** : iPhone/iPad Operation System
- JIT** : Just In Time
- JSX** : Javascript XML (Extensible Markup Language)
- RAM** : Random Access Memory
- SDK** : Software Development Kit
- SQL** : Structured Query Language
- VM** : Virtual Machine
- XML** : Extensible Markup Language

SÖZLÜK

Association	: İki sınıf arasındaki ilişki
Backwardcompatibility	: Geriye uyumluluk
Breadcrumbs	: Ekmek kırıntısı / sayfa menü göstergesi
Cache	: Ön bellek
Component Diagram	: Bileşen diyagramı
Composition	: Kompozisyon
Create Task	: Görev oluşturma
Cross-Platform	: Çoklu platform / Çapraz platform
Deployment Diagram	: Dağıtım diyagramı
Design Rule	: Tasarım kuralı
Due Date	: Bitiş tarihi
Focus	: Odaklanma
Garbage Collection	: Çöp toplayıcı
High Performance	: Yüksek performans
Hot-Reload	: Hızlı geliştirme
Hybrit App	: Mobil ve web teknolojilerin birlikte kullanıldığı uygulama türü
Layout	: Yerleşim düzeni
List Tasks	: Görev Listeleme
Login	: Uygulamaya giriş
Logout	: Uygulamadan çıkış
Multiplicity	: Çokluk sınıf ilişkisi

Native	: Yerel
Native App	: Yerel Uygulama
Platform Channels	: Platform kanalları
PWA	: İleri web uygulaması
Rapid Development	: Hızlı geliştirme
Search Tasks	: Görev arama
Sequence Diagram	: Sıralı diyagram
Show All Tasks on the Calendar	: Tüm görevleri takvimde göster
Use Case Diagram	: Kullanım durum diyagramı
User Friendliness	: Kullanıcı dostu
Widget	: Flutter ekran bileşeni
Widget Rendering	: Ekran bileşeni oluşturma

GİRİŞ

Günümüzde hayatımızın olmazsa olmazı hiç kuşkusuz mobil cihazlardır. Bununla birlikte teknolojinin gelişmesi ve internetin hızlanması ile beraber mobil uygulamaların kullanımı ve önemi de bir o kadar artmıştır. Mobil uygulamalar yer ve mekân gerektirmeden her an hizmet verebilmekte, klasik masa üstü ve web uygulamalarının verdiği hizmetlerin tamamını da yerine getirebilmektedir. Bunun yanında birçok internet kullanıcısı çoğunlukla mobil uygulamalar üzerinden internete erişim sağlamakta, günlük aktivitelerin hızına ancak mobil uygulamalar ile ayak yudurmaktadır. Hal böyleyken firmalar yeni müşteri kazanmak, kampanya yapmak veya mevcut müşteri potansiyellerini korumak için hizmetlerini mobil uygulamalar üzerinden vermeye adeta mecburdur. Günümüzde internet bankacılığı, e-ticaret, e-devlet, sosyal medya, iletişim ve haberleşme, oyun, kargo hizmetleri vb. gibi birçok alan ve sektörde mobil uygulamalar olmazsa olmaz olarak kullanılmaktadır.

Mobil platformlar olarak günümüzde çoğunlukla Andorid ve IOS platformları kullanılmaktadır. Her platform cihazının donanım ve geliştirme SDK' ları birbirinden farklıdır. Dolayısıyla mobil uygulamaların her iki platform içinde ayrı ayrı geliştirilmesi gerekmektedir. Bunun yanında web uygulamalarının da ayrıca geliştirilmesi gerekmektedir. Bu durum zaman, maliyet ve rekabet açısından uygulama sahiplerini zor durumda bırakmıştır.

Bu problem genelde iki şekilde çözülmektedir:

İlk çözümde uygulama web uygulaması olarak geliştirilir ve geliştirilen bu uygulama ilgili platformun yerel teknolojisinde geliştirilen bir kabuk uygulama içerisindeki tarayıcı bileşeninde gösterilir. Bu tarz uygulamalara Hibrit uygulama denir. Hibrit uygulamalar yerel uygulama özelliğine sahip olduğundan platformların uygulama mağazalarında yayınlanabilir. Ayrıca PhoneGap gibi çözümler sayesinde yerel uygulamalardaki fotoğraf çekme, medya oynatma gibi cihaz özelliklerine erişim kabiliyeti kazanırlar.

Sonuç olarak Hibrit uygulamalar zaman ve paradan tasarruf sağlar. Bunun yanında Hibrit uygulamaların en büyük dezavantajı performans sorunudur. İçerik web içeriği olduğundan ancak web görünümündeki kadar iyi gözükebilir. Bu nedene kullanıcı deneyimi yerel uygulama kadar iyi olmaz. Ayrıca her platform (Android,

IOS) için ayrı ayrı kabuk yazmak gerekecektir. Bu maliyetler bazen yerel uygulama geliştirme maliyetlerini geçebilir.

İkinci çözüm ise çoklu platform (Cross-Platform) teknolojisi ile uygulama geliştirme yaklaşımıdır. Bu yaklaşımda uygulama kodu çoklu platformun temel aldığı kod yapısında geliştirilir ve sonucunda ilgili platformun yerel uygulamasına dönüşür. Normal şartlarda tüm mobil işletim sistemlerinde çalışan yerel bir uygulama geliştirmek daha önceden bahsedildiği gibi çok maliyetlidir. Çoklu platform yaklaşımı ile kısa zamanda tüm mobil platformların yerel uygulama özelliklerinden ödün vermeden çalışabilen, kullanıcı deneyimi yüksek, uygulamalar geliştirilebilir. Günümüzde en yaygın kullanılan çoklu platform yaklaşımları React Native ve Flutter' dır.

Bu çalışmada mevcut mobil uygulama türlerinden ve teknolojilerinden bahsedilmiş, daha sonra Google tarafından geliştirilen çoklu platform bir teknoloji olan Flutter' ın özellikleri ve mimarisi anlatılmış, sonrasında gereksinim ve teknik analizi yapılmış olan bir To-Do uygulamasının Flutter ile geliştirilmesi amaçlanmıştır.

Bitirme projesinin birinci bölümünde literatürde yapılan çalışmalara yer verilmiş, ikinci bölümde mevcut mobil uygulama teknolojilerinden bahsedilmiş, bunun yanında Flutter' ın özellikleri, avantajları, dezavantajları, mimarisi ve temelleri detaylı olarak anlatılmıştır. Ayrıca en yaygın kullanılan çoklu platform teknolojileri React Native ile Flutter' ın karşılaştırılması yapılmıştır. Üçüncü bölümde ise Flutter ile geliştirilecek uygulamanın gereksinim ve teknik analizi yapılmış, dördüncü bölümde kodlama safhasında kullanılan IDE, yazılım proje yapısı, kullanılan kütüphaneler ve sınıf yapıları anlatılmıştır. Ayrıca son olarak Gitlab üzerinden kodun tamamı ve kullanıcı ara yüz şekilleri paylaşılmıştır.

BİRİNCİ BÖLÜM

ÇOKLU PLATFORM GELİŞTİRME ÜZERİNE LİTERATÜR TARAMASI

Literatürde Flutter ve diğer çoklu platform teknolojileri ile yapılan çalışmalar bulunmaktadır. Ancak Flutter ile tek kod tabanında Android, IOS ve Web platformlarında çalışan, gereksinim ve teknik analizinde yer aldığı bir çalışma bulunmamaktadır.

Bu bölümde literatürde yer alan ve daha önce gerçekleştirilmiş olan çalışmalar aşağıda incelenmiştir.

Mehmet İŞİTAN tarafından Aralık 2020 yılında hazırlanan "ÇAPRAZ PLATFORM MOBİL UYGULAMA GELİŞTİRME ARAÇLARININ KARŞILAŞTIRILMASI VE DEĞERLENDİRİLMESİ" konulu yüksek lisans tezinde Cross-Platform mobil uygulama geliştirme yazılım çatıları dahil olmak üzere her birinin artıları ve eksileri bir geliştiricinin bakış açısı temel alınarak ayrı ayrı değerlendirilip ölçümleri yapılmış ve işlemci, bellek, pil ve ağ kullanımı, kod yapısı, açılma (render) süreleri, popülerite, üçüncü parti yazılım desteği, hız-performans gibi konularda karşılaştırmalar yapılarak geliştiricilere kendi ihtiyaçlarına göre hangi yazılım çatılarının daha uygun olduğunu bulmalarına yardımcı olunması hedeflenmiştir.

Sonuç olarak çapraz platform uygulama geliştirme aracı seçiminde geliştirilen uygulamanın performanslı çalışması ve geliştirme desteğinin önemli olduğu vurgulanmıştır. Buna göre a Stackoverflow, GitHub ve Google Trends verilerine göre React Native ve Flutter öne çıkmaktadır. Kullanım olarak Flutter' ın React Native' in önüne geçtiği ve performans olarak da daha önde olduğu sonucuna varılmıştır. Eğer hali hazırda web teknolojilerine hâkimiyet varsa React Native' in Javascript kullanmasından ötürü tercih edilebileceği önerilmiştir. Flutter' da ise kendi geliştirme dili olan Dart dilinin öğrenilmesi gerektiği ifade edilmiştir. Son değerlendirmede React Native ve Flutter araçlarından herhangi birinin tercih edilmesi önerilmekle birlikte nihai karar yine geliştiricinin kendisine bırakılmıştır.¹

¹ Mehmet İŞİTAN, "ÇAPRAZ PLATFORM MOBİL UYGULAMA GELİŞTİRME ARAÇLARININ KARŞILAŞTIRILMASI VE DEĞERLENDİRİLMESİ", Yayınlanmamış Yüksek Lisans, Selçuk Üniversitesi, 2020, 1-58

Andreas Biørn-Hansen, Tor-Morten Grønli, Gheorghita Ghinea ve Sahel Alouneh tarafından Mayıs 2018 yılında hazırlanan "An Empirical Study of Cross-Platform Mobile Development in Industry" konulu araştırma makalesinde "Cross-Platform Mobile Development" hakkında endüstrinin perspektif ve görüşünü almak için 101 geliştiriciye 2 likert ölçeğinde, 3' ü çoktan seçmeli toplam 5 sorulu anket düzenlenmiştir.

Anket sonucunda katılımcılarının çoğu akademik literatürde 'de yaygın olarak tartışılan konuları bildirmiştir. Performans, kullanıcı deneyimi ve teknik çerçevelerin olgunluğuyla ilgili konular, anket katılımcıları tarafından en belirgin şekilde bildirilen konulardır. Özellikle performans ve kullanıcı deneyimi araştırmada sıkça tartışılan konulardır. Çerçevelerin olgunluğu veya eksikliği nadiren karşılaşılan bir konudur. Araştırmada daha kesin sonuçlara ihtiyaç duyulduğu çünkü çalışmalar genellikle birbiriyle çeliştiği ve araştırma destekli karar vermeyi zorlaştırdığı görülmüştür. Daha endişe verici bir not olarak çok az katılımcı platformlar arası uygulamalarda güvenliği bir sorun olarak algıladığı anlaşılmıştır. Araştırmada değinildiği gibi güvenliğin şüphe götürmez olarak mevcut olduğundan bahsedilmiştir. Popülerlik açısından katılımcılar sırayla React Native, PhoneGap, ve Ionic yazılım çatılarını seçerken profesyonel olarak en çok tercih edilen PhoneGap olduğu sonucu çıkmıştır. Ayrıca aynı sıranın hobi olarak ta geçerli olduğu sonucuna da varılmıştır.²

Gültürk KARLI tarafından Ekim 2014 yılında hazırlanan "CROSS PLATFORM MOBILE DEVELOPMENT" konulu yüksek lisans tezinde platform bağımsız mobil uygulama geliştiricilerine yardımcı olmak için tasarlanan ve geliştirilen yeni bir yazılım çerçevesi önerilmiştir.

Sonuç olarak Android ve IOS platformlarında kullanılabilen tek bir yazılım çerçevesi tasarlanmış ve geliştirilmiştir. Veri katmanında C++, ara yüzler için iş katmanları, Javascript ve HTML5 teknolojileri kullanılmıştır. SWIG ve Android NDK kod oluşturma ve derleme için kullanılmıştır. Kod oluşturma aracı Android ve IOS projeleri için ayrı ayrı gerekli sınıf ve dosyaları oluşturmaktadır. Android ve IOS platformlarında web içerikli kullanıcı ara yüzleri doğal görünümde olduğu fakat

² Abdalbasit Mohammed QADIR, "CROSS PLATFORM CARGO TRACKING SYSTEM", Yayınlanmamış Yüksek Lisans, Fırat Üniversitesi, 2020, 1-45

platformların yereline göre "CROSS PLATFORM MOBILE DEVELOPMENT" performansının çok iyi olmadığına çıkarımı yapılmıştır.³

T. F. Bernardes ve M. Y. Miyake tarafından Nisan 2016 yılında hazırlanan "Cross-platform Mobile Development Approaches: A Systematic Review" konulu araştırma makalesinde 4 mobile platformda (iPhone, Android, BlackBerry ve Windows Phone) ayrı ayrı çözüm yollarının zorluğundan bahsedilmiş ve Cross-Platform yaklaşımlarının biz dizi platform için tek bir adımda tekrardan kaçınıp üretkenliği artırarak bu zorluğun üstesinden gelmek için ortaya çıktığına değinilmiştir.

Yapılan çalışmada Cross-Platform yaklaşımlarının her birinin güçlü ve zayıf yönlerini ortaya koyan sistematik bir inceleme yapılmıştır. İnceleme sonucunda web uygulamalarının dezavantajlarından bahsedilmiş, PhoneGap aracılığıyla geliştirilenler gibi Hibrit uygulamalarında web teknolojisine dayalı olduğunu ve bu nedenle bu tür araçlarla geliştirmenin karmaşık olduğundan bahsedilmiş fakat uygulamanın görünümünün gerçek bir mobile uygulama görünümünde olduğu, ana avantajının da bu olduğu ifade edilmiştir. Ancak ortaya çıkan mobil uygulamalarda performans sınırlamalarının kullanıcı deneyimlerine olumsuz etkilemesinden ve ayrıca mobil cihazların donanım ve SDK' larının hızlı geliştiğinden ve seçilen Cross-Platform geliştirme aracının da bu nedenle sürekli güncellenebilmesi gerektiğinin öneminden bahsedilmiştir.⁴

Çağlar ÇINAR' ın Mayıs 2019 yılında "ÇOKLU PLATFORM MOBİL UYGULAMALAR OLUŞTURMAK İÇİN UYGUN YAKLAŞIM SEÇME ANALİZİ" konulu yüksek lisans tezinden öncelikle mobil cihazlara ait platformlar hakkında genel bilgi verilmiş ve çoklu platform araçları detaylandırılmıştır. Sonrasında, mobil uygulama geliştirirken gereken ihtiyaçlar bir matriste toplanmış ve bu ihtiyaçlara ağırlık katsayıları verilmiştir. Oluşturulan matriste ağırlıklı ortalama yöntemiyle geliştirilecek uygulamaya ait bir skor üretilerek geliştirme yaklaşımı seçilmiştir.

³ Gültürk Karlı, "CROSS PLATFORM MOBILE DEVELOPMENT", Yayınlanmamış Yüksek Lisans, Dokuz Eylül Üniversitesi, 2014, 1-64

⁴ T. F. Bernardes, M. Y. Miyake, "Cross-platform Mobile Development Approaches: A Systematic Review", 2019, <https://www.hindawi.com/journals/wcmc/2019/5743892/>, 02.05.2021

Çalışma sonucunda 12 adet kıstas analiz edilmiş, öncelikle her bir kıstas 1 (çok zayıf) ve 5 (çok iyi) arasında her bir geliştirme yöntemine göre derecelendirilerek değerlendirme matrisi oluşturulmuştur. Derecelendirmeler, herhangi bir yöntemi genel olarak en iyi ilan etmek için birbirine çok yakın olduğu görülmüştür.⁵

⁵ Çağlar ÇINAR, “ÇOKLU PLATFORM MOBİL UYGULAMALAR OLUŞTURMAK İÇİN UYGUN YAKLAŞIM SEÇME ANALİZİ”, Yayınlanmamış Yüksek Lisans, İstanbul Üniversitesi, 1-30

İKİNCİ BÖLÜM

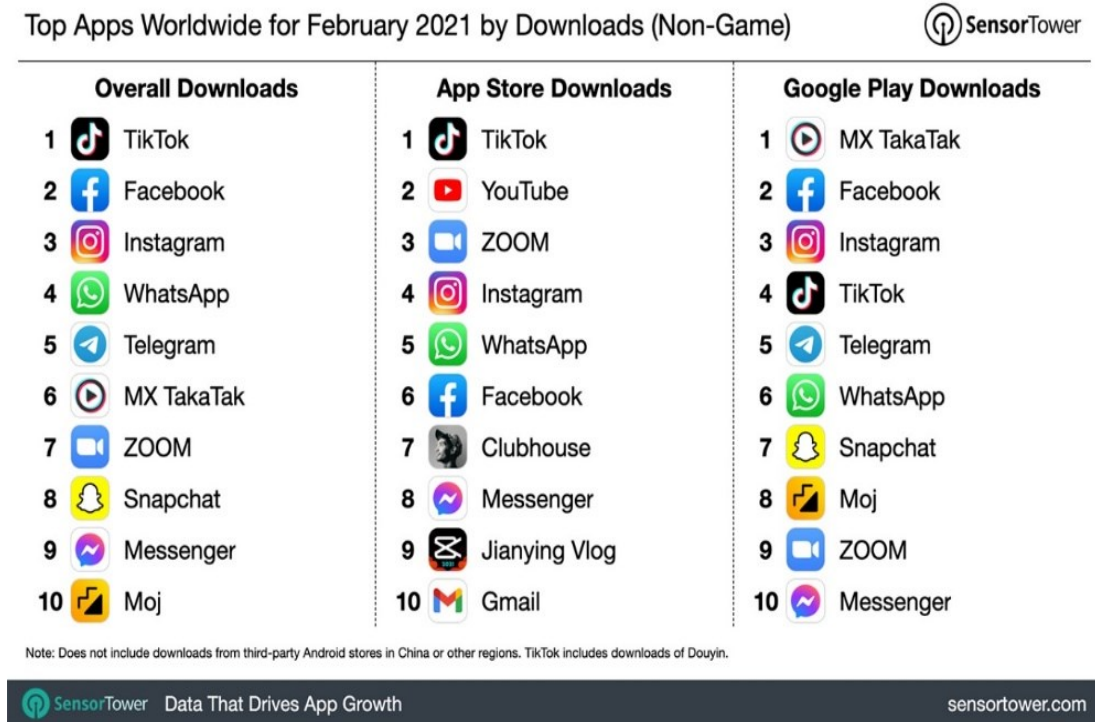
MOBİL UYGULAMA GELİŞTİRME TEKNOLOJİLERİ

1. Sayılarla Mobil Uygulama Kullanımları

Pandemi döneminde evine kapanan insanlar artık daha da fazla mobil uygulama kullanmaya başlamıştır. 2020 verilerine göre ortalama bir kullanıcı mobil cihazlarda 3 saat’ den fazla zaman harcarken, mobil web sitelerinin payı ise her geçen gün daha da azalmaktadır. Mobil web sitelerine harcanan ortalama günlük zaman ise sadece 12 dk’ dır.

Şekil 1.’ de Şubat 2021 indirilen uygulamaların en çok indirilme sırasına göre listesi verilmiştir. Görüldüğü gibi sosyal medya uygulamaları neredeyse listenin tamamında yer almaktadır. Buna göre TikTok uygulaması sadece Şubat 2021’ de 56 milyon, ikinci sırada yer alan Facebook uygulaması ise 45 milyon kez indirilmiştir.

Sayılardan da anlaşıldığı gibi mobil uygulamalar hayatın içerisinde çok fazla yerini almıştır.



Şekil 1. Şubat 2021’ de En Çok İndirilen Uygulamalar

Kaynak: Sensor Tower, “Top Apps Worldwide for February 2021 by Downloads”, 2021, <https://sensortower.com/blog/top-apps-worldwide-february-2021-by-downloads>

2. Mobil Uygulama Geliştirme Türleri

Mobil uygulama geliştirmenin dolaylı olarak birkaç türü vardır. Aşağıda bu türler incelendikten sonra avantaj ve dezavantajlar karşılaştırılmıştır.⁶

2.1. Yerel Uygulamalar (Native App)

Yerel mobil uygulamalar belirli bir işletim sistemine özgü yazılan uygulama olarak ta tanımlanır. Uygulama sadece geliştirme yapılan işletim sisteminde çalışır. Bu yaklaşımın en büyük avantajı cihazın sunduğu yazılım ve donanımsal özelliklerinden tam bir şekilde yararlanılmasıdır. IOS için Objective C ve Swift, Android için Java ve Kotlin dilleri kullanılır.

Bununla birlikte yerel mobil yazılım geliştirme yaklaşımı tüm platformlar için çok maliyetlidir. Aynı özelliklere sahip mobil uygulamalar birbirinden farklı mobil işletim sistemlerinde ve o sistemlerin birbirinden farklı SDK' larında alana özgü tecrübesi olan kaynaklar tarafından tekrardan yazılır. Mobil teknolojilerinin de sürekli geliştiği bir dünyada buna ayak uydurmak gerçekten zordur. Bunların üstesinden gelmenin tek yolu, mobil geliştirmeyi dış kaynak kullanarak yürütmektir, ancak o zaman da proje kontrolü, sürüm sonrası destek ve iyileştirmeler gibi yeni sorunlar ortaya çıkmaktadır.



Şekil 2. Yerel (Native) Uygulamalar

⁶ Webrazzi, “Mobil Uygulama Geliştirme Türleri: Yerel, Hibrit, Platform Tabanlı Yaklaşımlar ve Daha Fazlası”, 2020, <https://webrazzi.com/2020/12/22/mobil-uygulama-gelistirme-turleri-yerel-hibrit-platform/>, Erişim Tarihi: 01.05.2021

2.2. Hibrit Uygulamalar (Hybrit App)

Hibrit uygulamalarda web ve mobil yaklaşımları bir arada kullanılır. Karma veya melez olarak ta adlandırılır. Uygulama mobil uygulama görünümündedir. Bu sayede uygulama mağazalarından dağıtılabilir. Bu yaklaşım genelde web uygulaması fonksiyonları yeterli olup kısıtlı bütçeye sahip işletmeler tarafından tercih edilir.

Hibrit uygulamalar gerçek bir mobil uygulaması işlevselliğinde değildir. Uygulamanın içeriği aslında web içeriği olarak sunulduğundan donanımsal özelliklere erişim ancak PhoneGap gibi yazılım çatıları tarafından sağlanır. Bu nedenlerden dolayı Hibrit uygulamaların performansı yerel mobil uygulamalara göre daha yavaştır. Bu durum kullanıcı deneyimini etkilemektedir.

Sonuç olarak Hibrit uygulamaların çözebileceği görevler sınırlıdır. Mesela oyun gibi donanımsal özelliklerin kullanıldığı bir uygulama geliştirmek gerçekten zor olacaktır.



Şekil 3. Hibrit Uygulama Teknolojileri

2.3.İleri Web Uygulamaları (PWA)

Bu yaklaşımda mobil uygulama işlevselliği en modern tarayıcıların desteği ile web uygulamasına taşınır. Bu yöntem kullanılarak geliştirilen uygulamalar ile mobil tecrübenin faydaları kullanıcılara sunulmuş olur.

2.4. Çoklu Platform Uygulamaları (Cross-Platform App)

Çoklu platform uygulamaları farklı mobil işletim sistemlerinde yerel olarak çalıştırılabilecek şekilde geliştirilen uygulamalardır. Bu yaklaşımda geliştirilen uygulama kullanılan aracın sağladığı tek kod yapısında geliştirilir. Daha sonra geliştirilen uygulama kullanılan araç seti ile birden fazla mobil işletim sisteminde çalıştırılabilen yerel bir uygulamaya dönüşür. Çoklu platform yaklaşımı uygulama kodunun yeniden kullanmayı varsayar.

2.4.1 Avantajlar

- Tüm mobil işletim sistemleri için ayrı ayrı uygulama geliştirmek çok maliyetli ve zaman alıcıdır. Bunun yerine tek kod yapısı ile uygulamayı geliştirip birden fazla mobil işletim sisteminde çalışabilen uygulamalar elde edilebilir.
- Kullanıcı deneyimi gibi yerel uygulama avantajlarından ödün vermez.
- Kısa süre içinde geniş kitlelere erişim sağlanabilir.
- Uygulamaya eklenen yeni özellikler ve hata gibi iyileştirmeler tek kod yapısında geliştirilip tüm platformlara dağıtıldığından uygulamanın tutarlılığı sağlanır.

2.4.2. Dezavantajlar

- Kullanılan çoklu platform araçlarına göre farklı sonuçları çıksa da genelde yerel uygulamalar ile karşılaştırıldığında küçük performans sorunları olabilir. Bununla birlikte, performanstaki bu farklılıklar, özellikle basit uygulamalar söz konusu olduğunda genellikle küçüktür.
- Çoklu platform uygulamaları, tasarımlarını ve işlevlerini yalnızca belirli cihazlara değil, aynı zamanda birçok farklılığı olan platformlara da uyarlamak zorundadır. Sonuç olarak, özellikle daha karmaşık özelliklerle, çeşitli cihazlar ve platform farklılıkları için belirli istisnaları ele almak

zorunda olan geliştiriciler için fazladan iş yaratır. Bu sorunlar yerel uygulamalarda o kadar sık görülmez, bu nedenle geliştiriciler kullanıcıların sorunlarını çözmeye odaklanabilir.

- Google veya Apple, Android veya IOS için her yeni özellik sunduğunda, bu yeni özelliği desteklemek için uygulamaları güncellenmesi biraz zaman alır. Yerel uygulamalarda, yeni SDK' lar, güncellemelerle çoklu platformlardan daha önce sağlanır.

3. Çoklu Platform Uygulama Geliştirme Teknolojileri

3.1. Flutter

Google tarafından geliştirilen Flutter, mobil, masaüstü ve web platformlarında yerel uygulamalar oluşturmak için kullanılan açık kaynaklı bir teknolojidir. Flutter, yerel uygulamaların daha kısa bir sürede oluşturulmasına yardımcı olan, tamamen özelleştirilebilir geniş bir Widget ögesi sağlar. Dahası, Flutter, görseller geliştirmek için Skia adı verilen 2D oluşturma motorunu kullanır ve katmanlı mimarisi, bileşenlerin verimli çalışmasını sağlar. Flutter hakkında daha detaylı bilgi ilgili bölümlerde verilecektir fakat aşağıda başlıca faydalarından bahsedilmiştir.

- Tam Yerel Uygulama Performansı Sağlar
- Esnek Kullanıcı ara yüzü (UI)
- Güçlü Widget Desteği Sağlar
- Yerleşik Tasarım Kabiliyeti
- Hızlı Uygulama Geliştirme

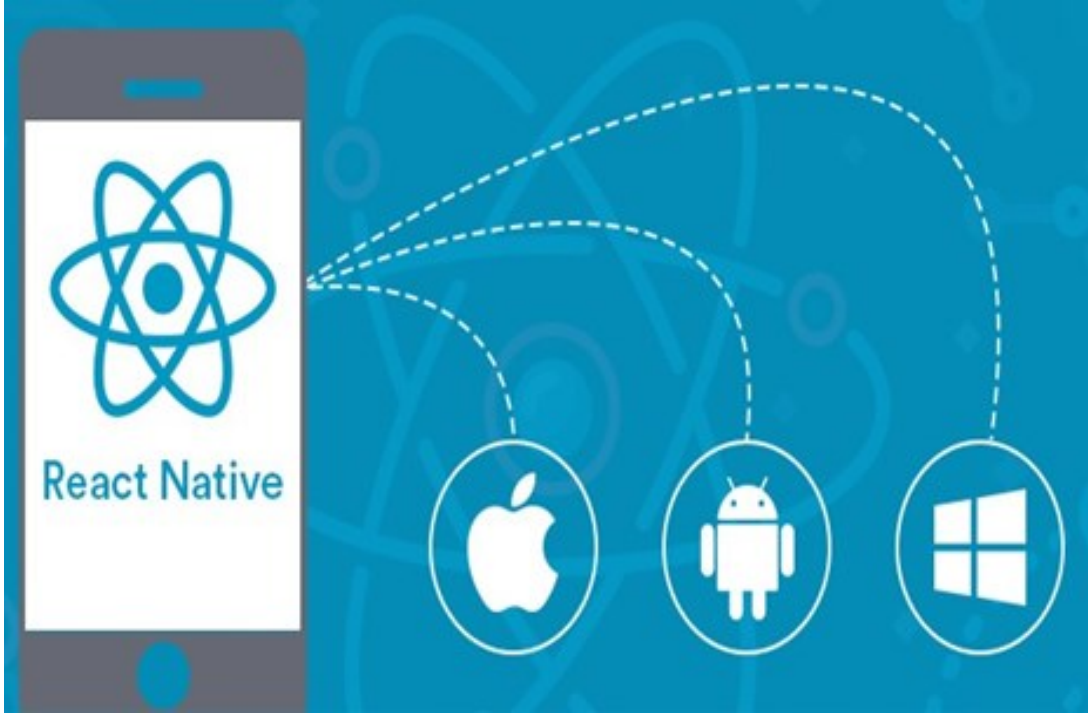


Şekil 4. Flutter 2.0' ın Desteklediği Platformlar

3.2. React Native

React Native, geliştirme endüstrisinde en çok tavsiye edilen mobil uygulama araçlarından biridir. Facebook tarafından oluşturulan araç, Android ve IOS platformları için yerel mobil uygulamalar geliştirmesini sağlayan açık kaynaklı bir teknolojidir. React Native bir web görünümünde çalışan Hibrit uygulamalar üzerinden yerel uygulamalar geliştirmeyi amaçlayan, React ve JavaScript' e dayanmaktadır. Ayrıca, hem Android hem de IOS uygulamaları için tek kod tabanını kullanan bir çoklu platform geliştirme aracıdır. React Native' in başlıca faydalarından bazıları aşağıda belirtilmiştir:

- Kod Yeniden Kullanılabilirliği ve Uygun Maliyetli
- Üçüncü taraf eklentilerle uyumlu
- Optimum performans için yeniden kullanılabilir bileşenler
- Sıcak dağıtım (Hot Deployment) özellikleri sağlar
- Bakım kolaylığı



Şekil 5. React Native' in Desteklediği Platformlar

3.3. Xamarin

Xamarin mobil uygulamalar geliřtirmek için kullanılan en popüler açık kaynaklı araçlardan biridir. Microsoft tarafından desteklenir ve dot.net tabanlıdır. Android, IOS ve Windows platformları için yerel uygulamalar oluşturulmasına olanak tanır. Xamarin, yerel uygulamalar oluşturmak için gereken hemen hemen her gerekli araç ve kitaplıkla birlikte gelir ve yerel kullanıcı arabirimi öğelerini kullanarak zengin deneyimler oluşturulmasını sağlar. Ayrıca Xamarin, geliştirme sürecini daha verimli ve uygun maliyetli hale getirmek için ortak kod tabanını paylaşma özelliğini de destekler.⁷

Xamarin' in çeřitli faydaları vardır, bunlardan bazıları aşağıda belirtilmiştir;

- Yaklaşık 1,4 milyon geliřtiriciden oluşan büyük Topluluk
- Yerel API Eriřimi ve UI Desteęi
- Daha kolay API Entegrasyonu
- Tüm Platformların Hedeflenmesi
- Uygun Maliyetli ve Daha Hızlı Geliřtirme Süreci



řekil 6. Xamarin' in Destekledięi Platformlar

⁷ Argenova, “Xamarin Nedir - Artıları ve Eksileri Nelerdir?”, 2019, <https://www.argenova.com.tr/xamarin-nedir-artilari-eksileri-nelerdir>, Eriřim Tarihi: 04.05.2021

4. Flutter Teknolojisi

Flutter, Google tarafından geliştirilip ve Mayıs 2017'de piyasaya sürülen ücretsiz ve açık kaynaklı bir mobil uygulama teknolojisidir. Yalnızca tek bir kod tabanıyla yerel mobil ve web uygulaması oluşturmasını sağlar. İlk yayınlandığından itibaren bazı büyük firmalar Flutter ile uygulama geliştirdiklerini duyurdular. Bunlardan bazıları Groupon, Philips Hue, Alibaba ve Tencent firmalarıdır. Yazılım geliştirme dili olarak Dart dili kullanılır.^{8 9 10}



Şekil 7. Neden Flutter

- Öğrenmesi ve kullanması kolay
- Flutter modern bir yazılım teknolojisidir! Mobil uygulamalar oluşturmak gerçekten kolaydır. Java, Swift veya React Native kullanıldıysa, Flutter' ın ne kadar farklı olduğu hemen anlaşılır.
- Hızlı derleme ve maksimum verimlilik sağlar.
- Flutter ile kod değiştirildiğinde gerçek zamanlı olarak değişiklik çalışan uygulamaya yansır. Bu özellik Hot-Reload olarak adlandırılır.

⁸ Argenova, “Flutter Nedir ve Neden Öğrenmek Gerekir?”, 2020, <https://www.argenova.com.tr/flutter-nedir-ve-neden-ogrenmek-gerekir> , Erişim Tarihi: 04.05.2021

⁹ Murat Öner, “Flutter Nedir ve Neden Flutter?”, 2019, <https://www.muratoner.net/flutter/flutter-nedir-ve-neden-flutter> , Erişim Tarihi: 04.05.2021

¹⁰ Vayes, “Flutter Nedir? Nasıl Kullanılır? Özellikleri Nelerdir?”, 2020, <https://www.vayes.com.tr/tr/blog/flutter-nedir-nasil-kullanilir-ozellikleri-nelerdir#:~:text=burada%20bu%20konudan%20bahsedece%C4%9Fiz%3A,verimli%20%C5%9Fekilde%20uygulamalar%20yapabilece%C4%9Finizi%20d%C3%BC%C5%9F%C3%BCn%C3%BCn> , Erişim Tarihi: 01.05.2021

Kaydettikten sonra uygulamanın kendisini güncellemek yalnızca kısa bir süre alır.

- Her türlü ideal olması
 - o Tek kod yapısı ile Andorid, IOS ve WEB platformlarında çalışan uygulama geliştirme olanağını sağlar.
 - o Performansı yerel bir uygulama hızındadır.
 - o Flutter ile yazılan uygulama çalıştığı mobil işletim sisteminin makine koduna dönüşür. Bu nedenle diğer çoklu platform sistemlerine göre özellikle hız açısından ayrışır.
- pub.dev ile gelişmiş ve çok sayıda Widget bileşenleri ile değerli kullanıcı ara yüzleri oluşturabilme yeteneği sağlar.
- Geniş dokümantasyon içeriği vardır.
- Büyüyen bir topluluktur.
- Flutter için farklı IDE' ler mevcuttur. Bunlar Android Studio (IntelliJ) ve VS Code' dur. Android Studio, zaten entegre edilmiş her şeyi içeren eksiksiz bir yazılımdır. Başlamak için Flutter ve Dart eklentilerini indirilmelidir. VS Code ise daha basit bir araçtır ve eklentiler aracılığıyla Flutter geliştirilebilir.
- Dart dili
 - o Dart, ilk kez Google tarafından geliştirilen ve daha sonraları ECMA tarafından standart haline getirilen açık kaynaklı ve genel amaçlı bir programlama dilidir. Dart dili kullanılarak web, sunucu, mobil uygulamalar ve IoT cihazları geliştirilebilir.¹¹

5. Flutter' ın Mimarisi ve Temelleri

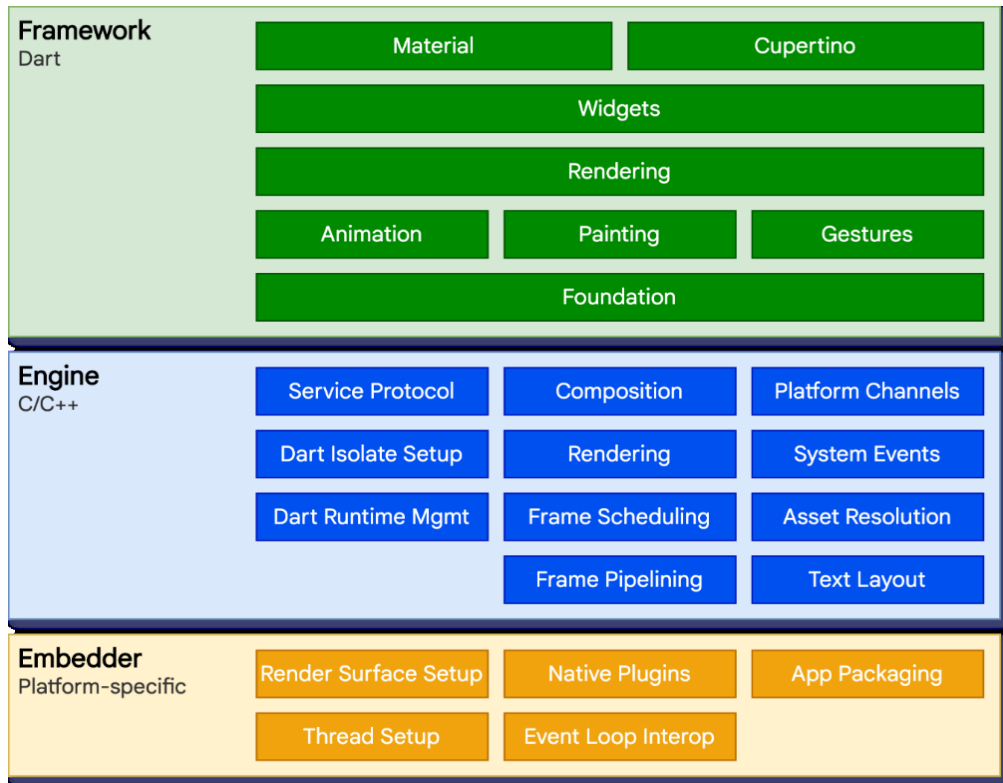
Flutter, IOS ve Android gibi işletim sistemlerinde kodun yeniden kullanımına izin verirken, aynı zamanda uygulamaların doğrudan temel platform hizmetleriyle ara yüz oluşturmalarına izin vermek için tasarlanmış, platformlar arası bir UI araç setidir. Amaç, geliştiricilerin farklı platformlarda doğal hissettiren yüksek performanslı uygulamalar sunmalarını sağlamaktır.

¹¹ Vikipedi, “Dart (programlama dili)”, 2020, [https://tr.wikipedia.org/wiki/Dart_\(programlama_dili\)](https://tr.wikipedia.org/wiki/Dart_(programlama_dili)) , Erişim Tarihi: 04.05.2021

Geliştirme sırasında, Flutter uygulamaları, yeniden derlemeye gerek kalmadan değişikliklerin yeniden yüklenmesini sağlayan bir sanal makinede çalışır. Sürüm için Flutter uygulamaları, ister Intel x64 veya ARM talimatları olsun, doğrudan makine koduna veya web platformu hedefleniyorsa javascript' e derlenir. Flutter alt yapısı izin verilen bir BSD lisansına sahip açık kaynak kodludur ve temel kitaplık işlevini tamamlayan, gelişen üçüncü taraf paketlerinden oluşan bir ekosisteme sahiptir.

5.1. Mimari Yapı

Flutter, genişletilebilir, katmanlı bir sistem olarak tasarlanmıştır. Şekil 8.' de gözüktüğü gibi her biri temeldeki katmana bağlı olan bir dizi bağımsız kitaplık olarak sunulur. Hiçbir katmanın alt katmana ayrıcalıklı erişimi yoktur ve çerçeve düzeyinin her bölümü isteğe bağlı ve değiştirilebilir olacak şekilde tasarlanmıştır.¹²



Şekil 8. Flutter' ın Mimari Yapısı

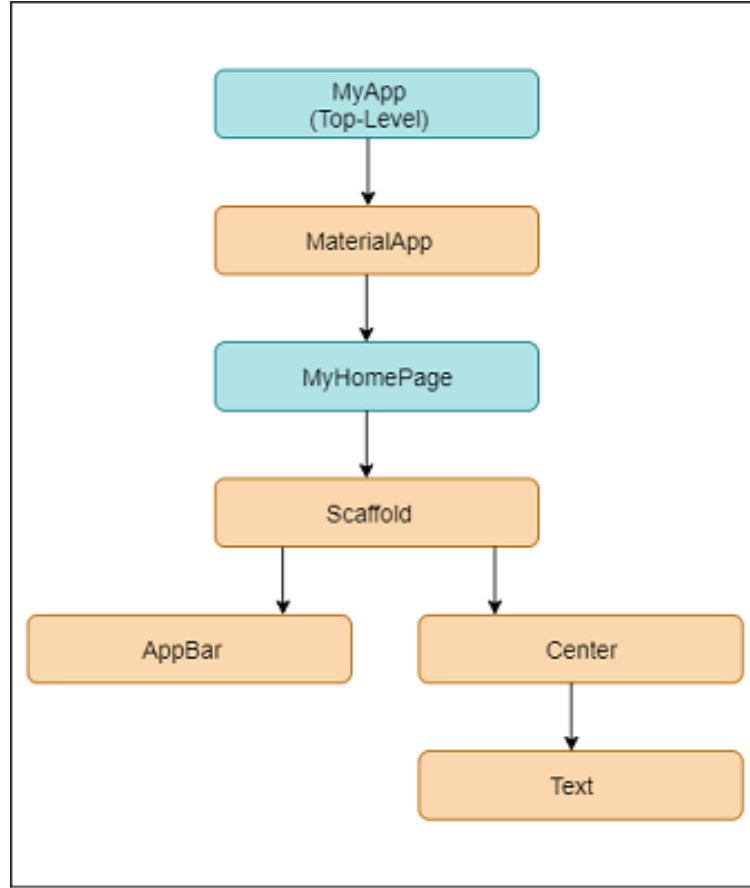
Kaynak: Flutter.dev, “Flutter architectural overview”, 2021, <https://flutter.dev/docs/resources/architectural-overview>

¹² Flutter.dev, “Flutter architectural overview”, 2021, <https://flutter.dev/docs/resources/architectural-overview> , Erişim Tarihi: 05.05.2021

5.2. Widgets

Kısaca Flutter' da her şey widget' dır. Flutter' daki widget, temelde uygulamanın görünümünü ve ara yüzünü etkileyen ve kontrol eden bir kullanıcı ara yüz bileşenidir. Kullanıcı ara yüzünün değişmez bir açıklamasını temsil eder ve widget' lar oluşturulan grafikleri, metinleri, şekilleri ve animasyonları içerir. widget' lar React bileşenlerine benzer.

Aşağıdaki şekil 9.' da tüm bileşenlerin alt widget' lar içeren widget' lar olduğunu görebiliriz. Bu nedenle, Flutter uygulamasının kendisi bir widget' tır. Bu özellik, karmaşık bir kullanıcı arabirimini çok kolay bir şekilde oluşturmanıza yardımcı olur.¹³



Şekil 9. Widget Ağacı

Kaynak: JavatPoint, “Flutter Architecture”, 2021, , <https://flutter.dev/docs/resources/architectural-overview>

¹³ Javat Point,” Flutter Architecture”, 2021, <https://www.javatpoint.com/flutter-architecture> , Erişim Tarihi: 05.05.2021

5.3. Kompozisyon (Composition)

Kompozisyon karmaşık nesneler oluşturmak için basit nesneleri birleştirmenin bir yoludur. Örneğin, bir bilgisayar yaparken ana kartı, CPU'yu, GPU'yu, RAM'i ve bir sabit sürücü bir araya getirilir. Bu kompozisyonudur. Programlamada kompozisyon, bir sınıfın diğer sınıfların örnekleri olan özelliklere sahip olduğu yerdir.¹⁴

Flutter kullanıcı ara yüzü oluşturmada esneklik sağlar. Uygulamada istenilen kullanıcı ara yüzünü elde etmek için istenilen sayıda widget bir araya getirilebilir.

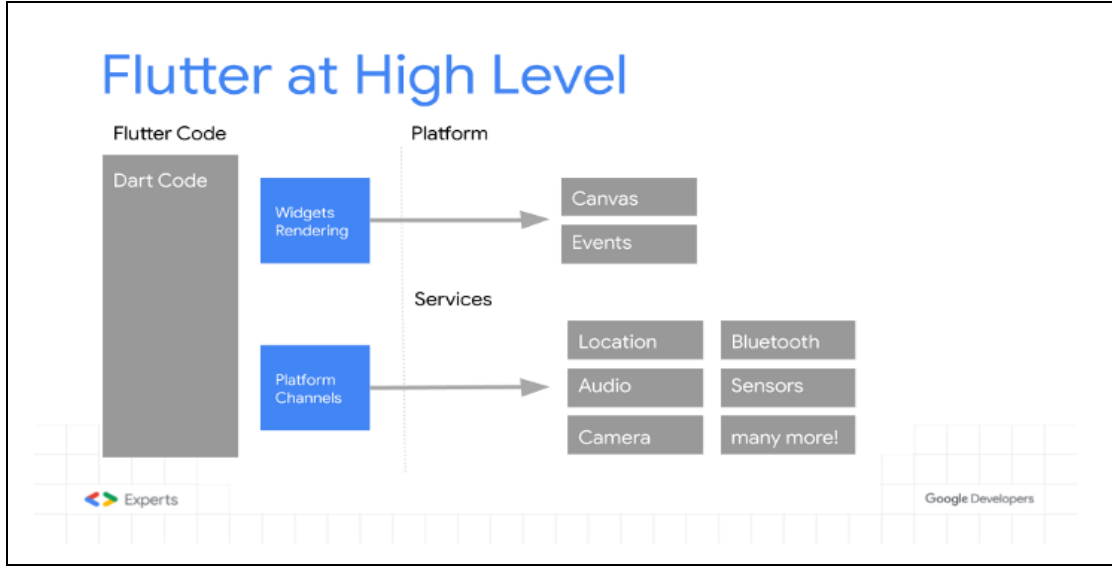
Aşağıda Şekil 10' da BusinessCardWidget widget' ını oluşturmak için Kompozisyon kullanıldığı bir örnek verilmiştir.

```
class BusinessCardWidget extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      home: Scaffold(  
        body: Column(  
          children: [  
            CircleAvatar(  
              backgroundImage: AssetImage(''),  
            ),  
            Row(  
              children: [  
                Text('Name'),  
                Text('John Doe'),  
              ],  
            ),  
            Row(  
              children: [  
                Text('Email'),  
                Text('John@example.com'),  
              ],  
            ),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

Şekil 10. Kompozisyon Widget Yapısı

¹⁴ Kinya Beatrice, "Composition in Flutter", 2020, <https://kinya.hashnode.dev/composition-in-flutter-ckepqbojt02g7kps1224cbrdg>, Erişim Tarihi: 05.05.2021

5.4. Flutter'ın Çalışma Şekli



Şekil 11. Flutter'ın Yüksek Seviyede Platform Erişimi

Kaynak: Muhammed Salih Guler, "Flutter'ın Mimarisi ve Dart'ın Flutter üzerindeki etkisi", 2020 , <https://medium.com/flutter-t%C3%BCrkiye/flutter%C4%B1n-mimarisi-ve-dart-%C4%B1n-flutter-%C3%BCzerindeki-etkisi-b9b652d0525>

5.4.1. Widget Oluşturma (Widget Rendering)

Yukarıdaki Şekil 11' de gözüktüğü gibi Dart kodu platform ile haberleşirken ilgili platformdan ekrana çizmek için canvas, ekran dokunuşları gibi kullanıcı etkileşimlerini yakalayabilmek içinde event (olay) listesini ister.

Ekrana widget' ları çizmek için canvas yeterlidir. Yapısında bulunan grafik motoru Skia cihaz ile iletişime geçerek ekrana çizimi gerçekleştirir.

Event' lar tüm kullanıcı işlemleri olarak düşünülebilir. Ekrana dokunma, uzun dokunma, çift tıklama gibi hareketlerdir. ¹⁵

5.4.2. Platform Kanalları (Platform Channels)

Mobil ve masaüstü uygulamaları için, Flutter, Dart kodu ile ana bilgisayar uygulama platformu arasında iletişim kurmak için basit bir mekanizma olan bir platform kanalı aracılığıyla özel kod çağırılmasına olanak tanır.

¹⁵ Muhammed Salih Guler, "Flutter'ın Mimarisi ve Dart'ın Flutter üzerindeki etkisi", 2020, <https://medium.com/flutter-t%C3%BCrkiye/flutter%C4%B1n-mimarisi-ve-dart-%C4%B1n-flutter-%C3%BCzerindeki-etkisi-b9b652d0525> , Erişim Tarihi: 05.05.2021

Platform Kanalları (Platform Channels) sayesinde cihaza ait olan kamera, ses, Bluetooth, lokasyon işlemleri gibi servislere ve daha fazlasına erişim imkânı olur. ¹⁶

5.5. Dart' ın Etkisi

Flutter' ın yazılım dilidir. Dil 2011 yılında Google tarafından oluşturulmuştur. Dart dili zaman içerisinde komitelerinde etkisiyle modern bir dil haline gelmiştir. Söz dizimi mevcut java, c# gibi dillere benzediğinden kolayca öğrenilebilir.

Dart dilinin kendisine ait bir sanal makinesi vardır. Geliştirme zamanı yapılan her bir değişiklik **JIT** tarafından algılanır ve sanal makinede derlenip anında hedef platforma yüklenir. Bu sayede Hot-Reload ve Hot-Restart özellikleri elde edilmiş olur.

Dart dili üretim anında **AOT** derlemesi yapar. AOT derleme tipinde Dart dili önce Assembly diline çevrilir daha sonra bulunduğu platformun makine diline çevrilir. Bu çok önemli bir özelliğidir. Özetle Flutter **JIT** ve **AOT** sayesinde önemli performans getirisi elde eder. ¹⁷

5.5.1. Çöp Toplayıcı (Garbage Collection)

Modern olan diğer dillerde olduğu gibi Dart dilinde de çöp toplayıcı özelliği vardır. GC, artık bir uygulama tarafından kullanılmayan "ölü" bellek bölgelerini bulmak ve geri almak için öbek arama işlemidir. Bu işlem, belleğin yeniden kullanılmasına izin verir ve bir uygulamanın belleğinin bitmesi riskini en aza indirir. Çöp toplama, Dart VM tarafından otomatik olarak gerçekleştirilir. DevTools' ta, GC düğmesine tıklayarak isteğe bağlı olarak ta çöp toplama işlemi gerçekleştirebilirsiniz.

18

¹⁶ Sait Banazlı, “Flutter nedir, nasıl çalışır, ne olacak? (Dikkat React N t ve çıkabilir)”, 2019, <https://medium.com/kodcular/flutter-nedir-nas%C4%B1l-%C3%A7a%C4%B1%C5%9F%C4%B1r-ne-olacak-dikkat-react-n-t-ve-%C3%A7%C4%B1kabilir-685c41977a88> , Erişim Tarihi: 05.05.2021

¹⁷ Muhammed Salih Guler, “Flutter’ın Mimarisi ve Dart’ın Flutter üzerindeki etkisi”, 2020, <https://medium.com/flutter-t%C3%BCrkiye/flutter%C4%B1n-mimarisi-ve-dart-%C4%B1n-flutter-%C3%BCzerindeki-etkisi-b9b652d0525> , Erişim Tarihi: 05.05.2021

¹⁸ Muhammed Salih Guler, “Flutter’ın Mimarisi ve Dart’ın Flutter üzerindeki etkisi”, 2020, <https://medium.com/flutter-t%C3%BCrkiye/flutter%C4%B1n-mimarisi-ve-dart-%C4%B1n-flutter-%C3%BCzerindeki-etkisi-b9b652d0525> , Erişim Tarihi: 05.05.2021

5.6. Flutter ile React Native' in Karşılaştırılması

Yeni başlayanlar için Flutter, kod tabanında Dart programlama dilini kullanırken React Native, JavaScript ve XML anlamına gelen JSX' i kullanıyor. Her iki dil de C tarzı sözdizimi türüne dayanır ve nesneye yönelik ilkeleri takip eder. Bu ortak nokta, Flutter ve React Native'in tasarım açısından temelde birbirine benzediği ve kodun da çok benzer olduğu anlamına gelir. Aşağıda ise her iki teknolojinin farklılıkları aynı başlıklarda ele alınmıştır.¹⁹



Şekil 12. Flutter vs React Native

5.6.1. Dilin Dinamik veya Statik Olması

React Native dinamik bir dil olan javascript dilini kullanırken, Flutter hem dinamik hem statik bir dil olan Dart dilini kullanır. Dinamik dillerde veri türleri değiştirilebilirken statik dillerde veri türü (type-safe) baştan belirtildiği için farklı bir veri değeri yüklenemez, bu nedenle daha güvenlidir. Örneğin sayısal bir veri türüne metinsel bir değer yüklenemez. Bu nedenle statik dillerde daha az hata meydana gelir. Flutter' da her iki yaklaşımda kullanılır. React Native javascript' in bir üst kümesi olan TypeScript ile statik olma özelliği kazanabilir.

¹⁹ Andrew Baisden ,”Flutter vs. React Native”, 2021, <https://blog.logrocket.com/flutter-vs-react-native/#:~:text=Key%20differences%20between%20Flutter%20and%20React%20Native,-Flutter%20and%20React&text=For%20starters%20Flutter%20uses%20the,and%20follow%20object%20Dorientated%20principles.>, Erişim Tarihi: 05.05.2021

5.6.2. Yerleşim Düzeni (Layout)

Flutter, kullanıcı ara yüzünü oluşturmak için bir pencere ögesi stili kullanırken, React Native JavaScript ve JSX kullanır. Flutter widget' ları önceden derlenmiştir, bu nedenle istenilen sürece teknik olarak özel widget' lar oluşturmaya gerek kalmaz. Google tarafından oluşturulup test edildikleri için uyumsuzluk sorunları konusunda endişelenmeye gerek yoktur.

5.6.3. Platform Desteği

Flutter 1.0 ile geliştirilen uygulamalar Android, IOS ve Web platformlarında çalışabilirken React Native ile geliştirilen uygulamalar Android ve IOS platformlarında çalışır.

Bunun yanında ayrıca Flutter 2.0 sürümü ile aşağıdaki platformlarda desteklenecektir.²⁰

Tablo 1. Flutter 2.0 Platform Desteği

Platform	Sürüm
Android	API 16 & üstü
iOS	iOS 8 & üstü
Linux	Debian 10 & üstü
macOS	El Capitan & üstü
Web	Chrome 84 & üstü
Web	Firefox 72.0 & üstü
Web	Safari on El Capitan & üstü
Web	Edge 1.2.0 & üstü
Windows	Windows 7 & üstü

²⁰ Flutter.dev, “Supported platforms”, 2021, <https://flutter.dev/docs/development/tools/sdk/release-notes/supported-platforms>, Erişim Tarihi: 05.05.2021

5.6.4. Hızlı Geliştirme (Hot-Reload)

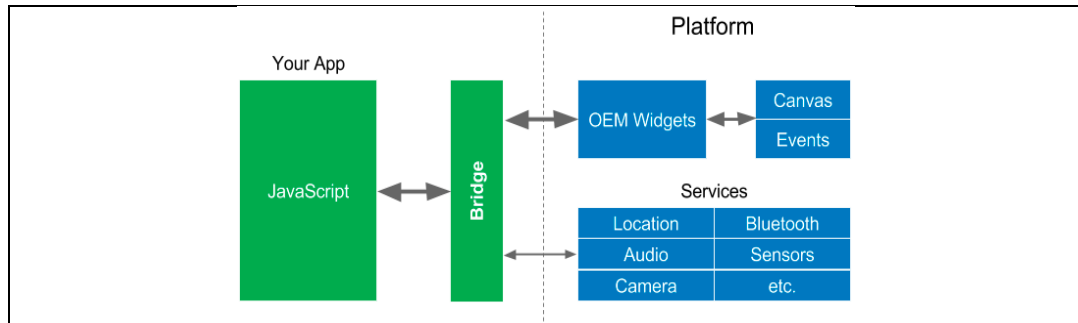
Her iki teknolojide kod değişikliklerinin anında gözükmesini sağlayan hızlı uygulama (Hot-Reload) geliştirme özelliklerini destekler.

5.6.5. Test

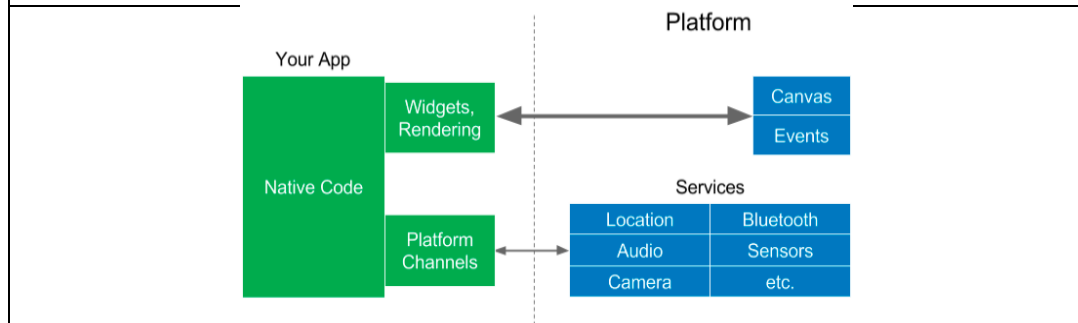
Flutter’ da iki farklı platform aynı anda test edilebilirken React Native’ de testler sırayla yapılır.

5.6.6. Yerellik ve Performans

Flutter uygulamaları bulunduğu platformda tamamen yerel olarak (Native) çalışır. Kendi içerisindeki yapılar tamamen yerel olduğundan arada köprü kullanmaz. React Native ise javascript kullandığından yerel bileşenlerle etkileşmek için arada köprü yapısını kullanması gerekmektedir. Bu nedenle Flutter, React Native’ e göre daha performanslıdır.²¹



Şekil 13 React Native Mimarisi



Şekil 14. Flutter Mimarisi

²¹ Merve Duran, “React Native vs Flutter”, 2019, <https://medium.com/batech/react-native-vs-flutter-536c8cfbec11> , Erişim Tarihi: 04.05.2021

5.6.7. Destek

Flutter uygulaması Google tarafından geliştirilmiştir. Google tam olarak destekler. Google desteğini çekmesi Flutter açısından çok risklidir fakat Flutter bu açığını hızla kapatmaktadır. React Native’ in arkasında ise Facebook ile beraber geniş bir topluluk vardır.

5.6.8. Uygulama Boyutu

Flutter ve React Native uygulamalarının her ikisi de yerel uygulamalara göre daha geniş boyuttadır.

5.6.9. React Native Kullanan Büyük Firmalar

FaceBook, Instagram, Pinterest, Uber Eats, Skype, Tesla, Wix

5.6.10. Flutter Kullanan Büyük Firmalar

Google Ads, Toyota, eBay Motors, Alibaba, Groupon, Square, The New York Times

5.6.11. Karşılaştırma Sonucu

Her iki teknolojiye çok kullanılan ve kendini ispatlamış teknolojilerdir. Flutter mimarisi sayesinde React Native’ den daha performanslıdır. Hatta yerel uygulamalarla hemen hemen aynı hızda çalışır. Bu durum kullanıcı deneyiminin Flutter açısından daha başarılı olmasını sağlar.

React Native’ in kullandığı dil javascript’ dir. Javascript web teknolojilerinde çok kullanıldığından genelde geliştiricilerin aşına olduğu bir dildir. Dart ise yeni bir dildir fakat C dili söz dizimin den geldiği için çabuk öğrenilebilir. Bu açıdan bakıldığında React Native duruma göre biraz daha avantajlıdır.

Flutter 2.0 sürümü ile beraber Linux, Windows, MacOS işletim sistemlerinde de çalışacak uygulamalar geliştirilebilecektir. Ayrıca Google tarafından gelecekte Android işletim sisteminin yerini alacak olan Fuchsia’ nın sistem uygulamaları Flutter’ ın Dart dili ile direkt çalışacak. Bu nedenlerden dolayı da Flutter, React Native’ e göre birkaç adım önde gözükmektedir.

ÜÇÜNCÜ BÖLÜM

UYGULAMANIN GEREKSİNİM VE TASARIM ANALİZİ

1. Proje Tanımı

1.1. Projeye Genel Bakış

To-Do uygulaması kullanıcıların görev ve bu görevlere bağlı alt görevler oluşturabildiği ve bu sayede kullanıcıların görevlerini takip edebildiği, Flutter teknolojisi ile yazılan, Android, IOS ve Web platformlarında çalışan bir uygulamadır. Uygulamanın ismi To-Do olarak çok genel algılandığından **My2Do** olarak isimlendirilmiştir.

1.2. Projenin Amacı

My2Do uygulaması özellik ve fonksiyonları açısından basit bir uygulamadır. Bunun yanında bulut ortamındaki veri tabanı ve Google kimlik doğrulama gibi özellikleri ile de Flutter 'ın özelliklerini deneyimlemek açısından yeterlidir.

Projenin amacı kullanıcıların Google kimlik yönetimi ile uygulamaya giriş yapabildiği, görev ve buna bağlı alt görevleri girip takip edebildiği kısaca görev takip uygulaması amacındadır. Uygulama İngilizce ara yüzde hazırlanmıştır.

1.3. Ürün Kapsamı

- Projenin kapsamında Google kimlik doğrulaması dışında ayrıca kullanıcı ve şifre girişi olmayacaktır.
- Tutulan veri, web platformunu etkilediğinden yerel veri tabanında kullanılmayacaktır.
- Uygulamanın 1.0 sürümünde sadece İngilizce dil kullanılacaktır.
- Bildirim özellikleri kapsam dışıdır.

1.4. Paydaşlar

My2Do uygulaması bireysel bir uygulamadır. Bu nedenle tek paydaş kullanıcıdır.

1.5. Kısıtlar

1.5.1. Çözüm Kısıtları

Uygulama mobil ve web platformlarında çalışacaktır. Bu nedenle tüm platformların farklı sürümlerinde tasarlandığı gibi görüntülenmesi ve çalışması ayrıca test edilmelidir.

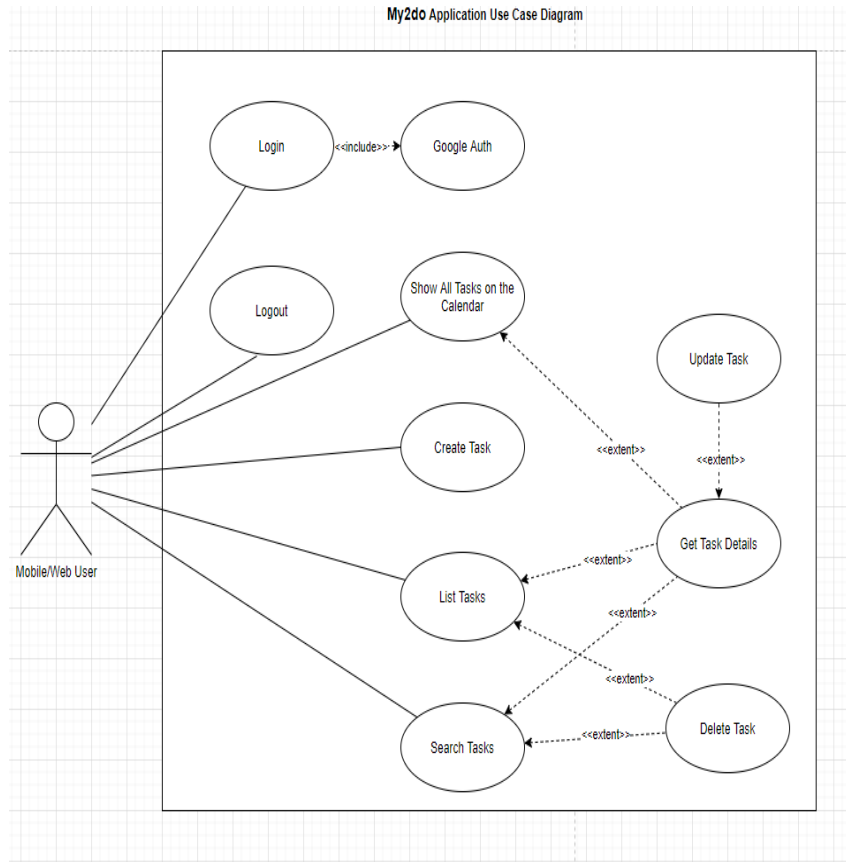
1.5.2. Veri Tabanı

Uygulamanın veri tabanı bulutta tutulduğu için performans sorunu kullanıcının internet hızına bağlıdır. Bu nedenle veri tabanı işlemlerinde performansa ayrıca dikkat edilerek SQL cümleleri yazılmalıdır.

2. Gereksinimler

2.1 Kullanım Durum Diyagramı

Uygulamanın kullanım durum diyagramı Şekil 15’ de aşağıda gösterilmiştir.



Şekil 15. My2do Uygulamasının Kullanım Durum Diyagramı

2.2 Kullanım Senaryoları:

2.2.1 Giriş ve Çıkış (Login & Logout)

Mobil/ Web Kullanıcı uygulamaya ilk giriş yaptığında Google kimlik bilgileri ile sisteme giriş yapar. Giriş yapan kullanıcı ana sayfadan oturumu bitirebilir.

2.2.2. Görev Oluşturma (Create Task)

Kullanıcı sisteme giriş yaptıktan sonra ana sayfadan bir link aracılığıyla yeni görev oluşturma sayfasına giriş yapar. Görev konusu girilip giriş onaylanmasıyla beraber girilen diğer alanlarda varsayılan olarak kaydedilir. Diğer alanlar onaylama işlemi sonrası ekranda gözükür. Kullanıcı bu alanları da isterse güncelleyebilir.

2.2.3. Görevleri Listeleme (List Tasks)

Kullanıcı sisteme giriş yaptıktan sonra ana sayfada daha önceden girilen ve açık olan görevleri listeler. Kullanıcı menüden isterse tamamlanan görevler ya da tüm görevleri listeleyebilir. Listedden seçtiği görevi detayına bakabilir, güncelleyebilir, silebilir ya da tamamlayabilir.

2.2.4. Tüm Görevleri Takvim’ de Göster (Show All Tasks On the Cal)

Kullanıcı sisteme giriş yaptıktan sonra ana sayfadan takvim üzerinde tüm görevleri listeler. Seçtiği görevin özetini görüntüleyip görev detay sayfasına geçiş yapabilir.

2.2.5. Görev Arama (Search Tasks)

Kullanıcı sisteme giriş yaptıktan sonra ana sayfadan tüm görevleri arar. Arama sonucu listede görüntülenir. Listedden seçtiği görevin detayına bakabilir, güncelleyebilir, silebilir ya da tamamlayabilir.

3. Tasarım

3.1. Tasarım Hedeflerinin Tanımlanması

3.1.1. Yüksek Performans Hedefi

Kullanıcının uygulamaya giriş yaptıktan sonra ana sayfanın beklemeden hızlı açılması ve içerikleri anında gösterilmesi hedeflenmiştir. Bu hedefe ulaşmak için performans testlerinin majör sürüm yayınlarında yapılması ve statik verinin cache (ön bellek) işlemine tabi tutulması şarttır.

3.1.2. Kullanıcı Dostu Olma Hedefi

Uygulama ara yüzlerinin tüm platformlarda (Android, IOS ve Web) tasarlanan şekilde görüntülenmesi kullanıcıların memnuniyeti açısından önemlidir. Bunun yanında göz yormayan tema, kolay okunabilen font, ekranda hangi dizinde olduğunu gösteren ekmek kırıntısı (Breadcrumbs) ve aradığını bulabilme gibi diğer kullanıcı dostu özelliklerinde sağlanması gerekmektedir. Bunun yanında “Design Rule” yönetmeleri kullanıcı dostu bir uygulama olma açısından uygulanmalıdır.

3.1.3. En Az Hata Hedefi

Uygulamanın çalışma zamanında minimum hata ile çalışabilmesi için birim, bütünleşme, fonksiyonel, performans ve yük testlerinin her sürümde yapılması, ayrıca hataların kaydedilmesi, izlenmesi ve hata anında teknik ekibe bildirim yapılması gerekmektedir.

3.1.4. Güvenirlilik Hedefi

Tüm çalışanların aynı zaman aralığında uygulamayı kullanabilmesi ve operasyonel işlemlerin bu zaman aralığında yapılabilmesi güvenirlilik açısından önemlidir. Güvenirlilik ölçümleri her sürümde yapılmalı ve güvenirlilik düşük ise iyileştirmeler sağlanmalıdır. Güvenirlilik ölçümü koşulan test sayısı / kırılma (crash) sayısı’ dır.

3.2 Hedeflerin Takası (Trade-offs)

Tablo 2. Hedeflerin Takası

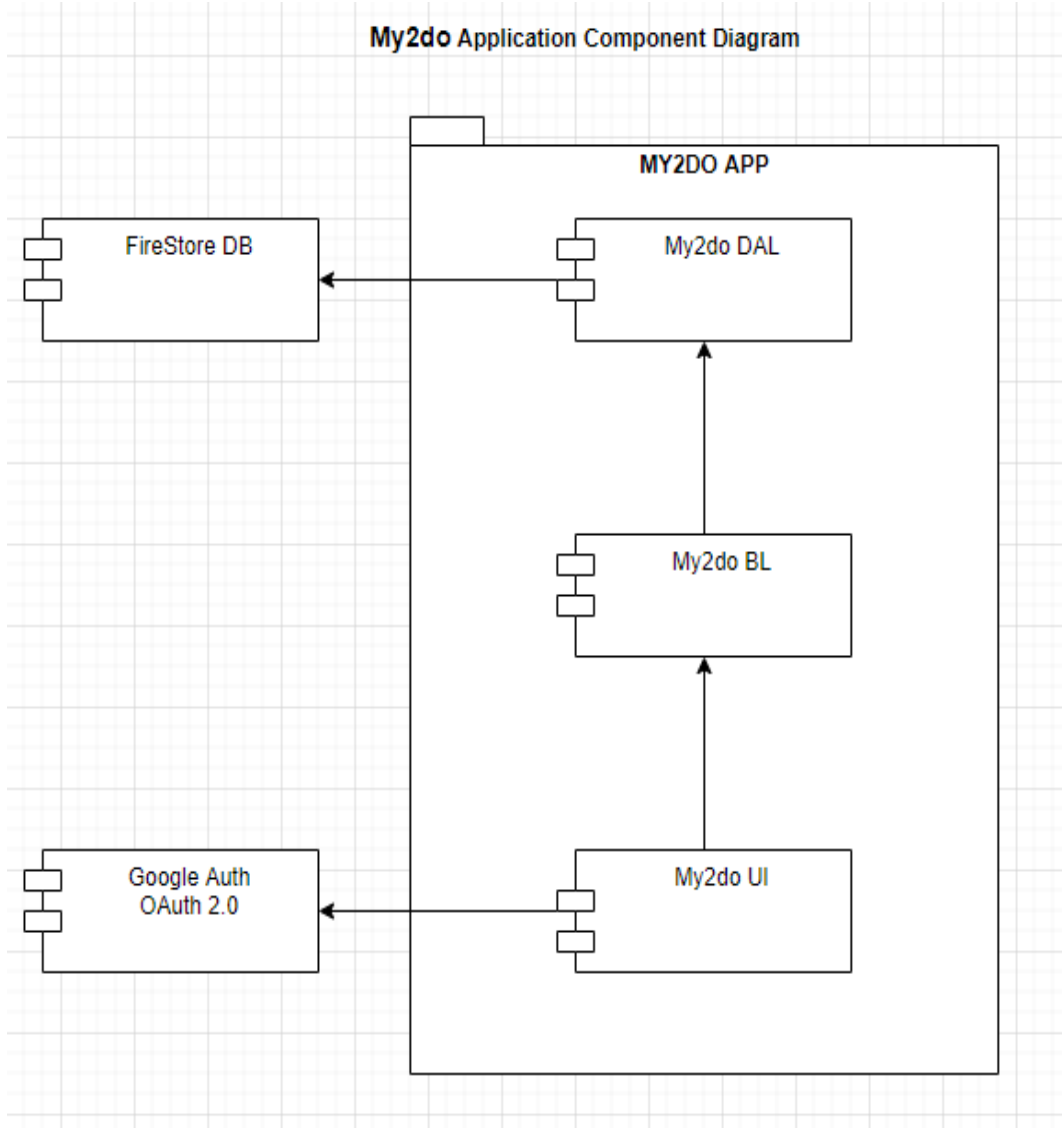
Seçilen Tasarım Hedefi	Potasiyel Takas (Trade-Off)	Açıklama
Yüksek Performans (High Performance)	Geri Dönük Uyumluluk (Backwardcompatibility)	Yüksek performans hedefi uygulamanın geriye dönük uyumluluğuna basitlik açısından engeller. Uygulamanın geriye dönük uyumlu olması örneğin fazladan veri tabanı sorgularının çalışmasına neden olabilir. Bu durum uygulamanın her sürümde daha da yavaşlamasına sebep olacaktır.
Kullanıcı Dostu (User Friendliness)	Hızlı Geliştirme (Rapid Development)	Uygulamanın kullanıcı dostu (User Friendliness) olabilmesi için birçok ara yüz özelliğinin geliştirilmesi gerektiğinden hızlı geliştirme hedefi başlangıçta sağlanamayacaktır. Ayrıca belirlenen kullanıcı gereksinimlerinde de hızlı geliştirme olabilmesi açısından kapsam daraltma yapılmayacaktır.

3.3 Önerilen Yazılım Mimarisi

3.3.1 Giriş

My2Do uygulaması Google Firebase ile entegredir. Kimlik doğrulaması Firebase Google OAuth 2.0 ile yapılır. Veriler Firebase entegrasyonu ile Firestore katalog veri tabanında tutulur.

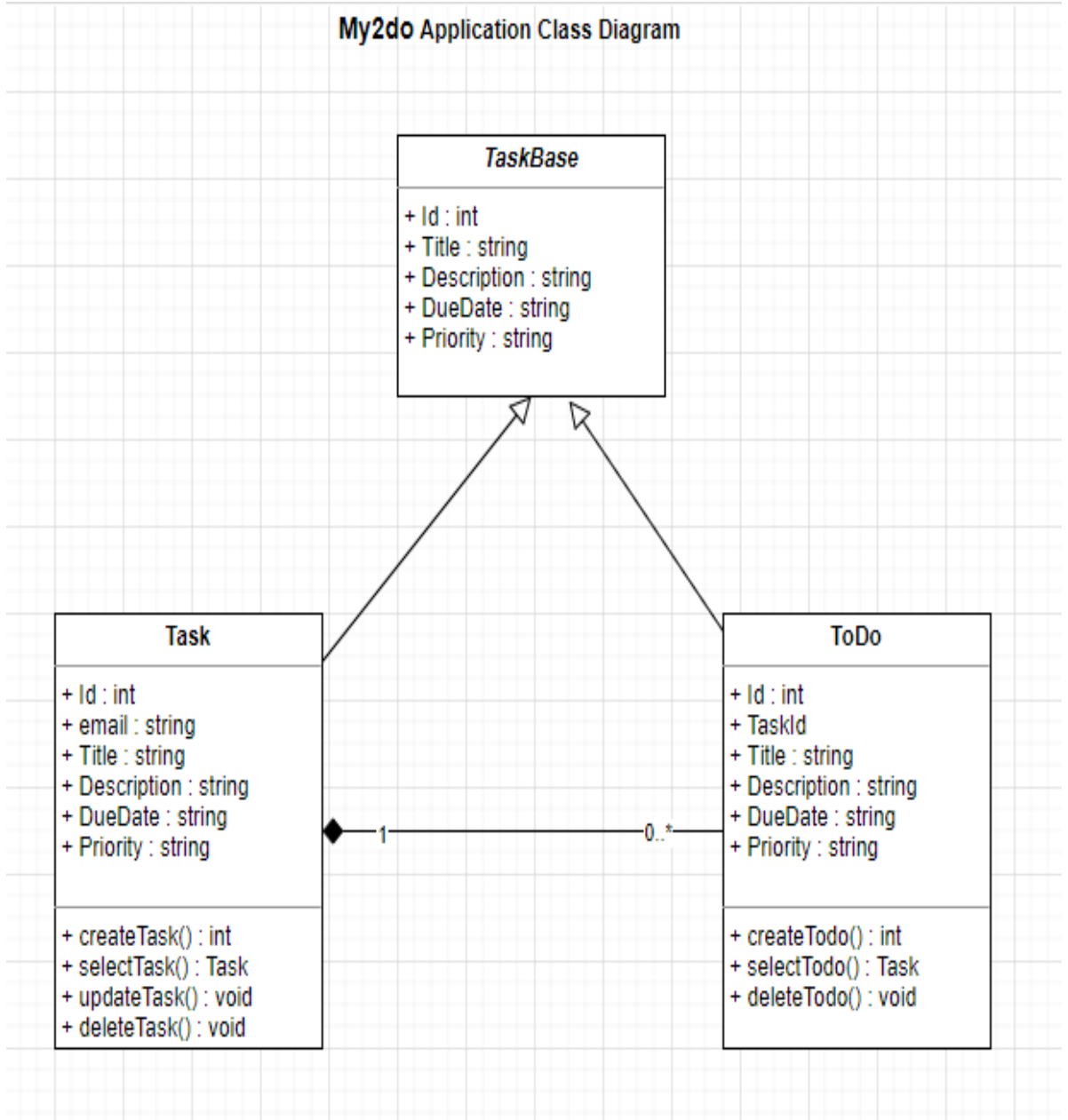
Buna göre uygulamanın bileşenlerine göre dağıtım diyagramı (deployment diagram) aşağıda Şekil 16’ da gösterilmiştir.



Şekil 16. Dağıtım Diyagramı

3.3.2. Sınıf Diyagramları

Kullanım durum (Use Case Diagram) diyagramları ve senaryolarına göre aşağıda Şekil 17’ de gösterilen sınıf diyagramı oluşturulmuş, sınıflar arasındaki association, multiplicity ve composition ilişkileri diyagramda gösterilmiştir.

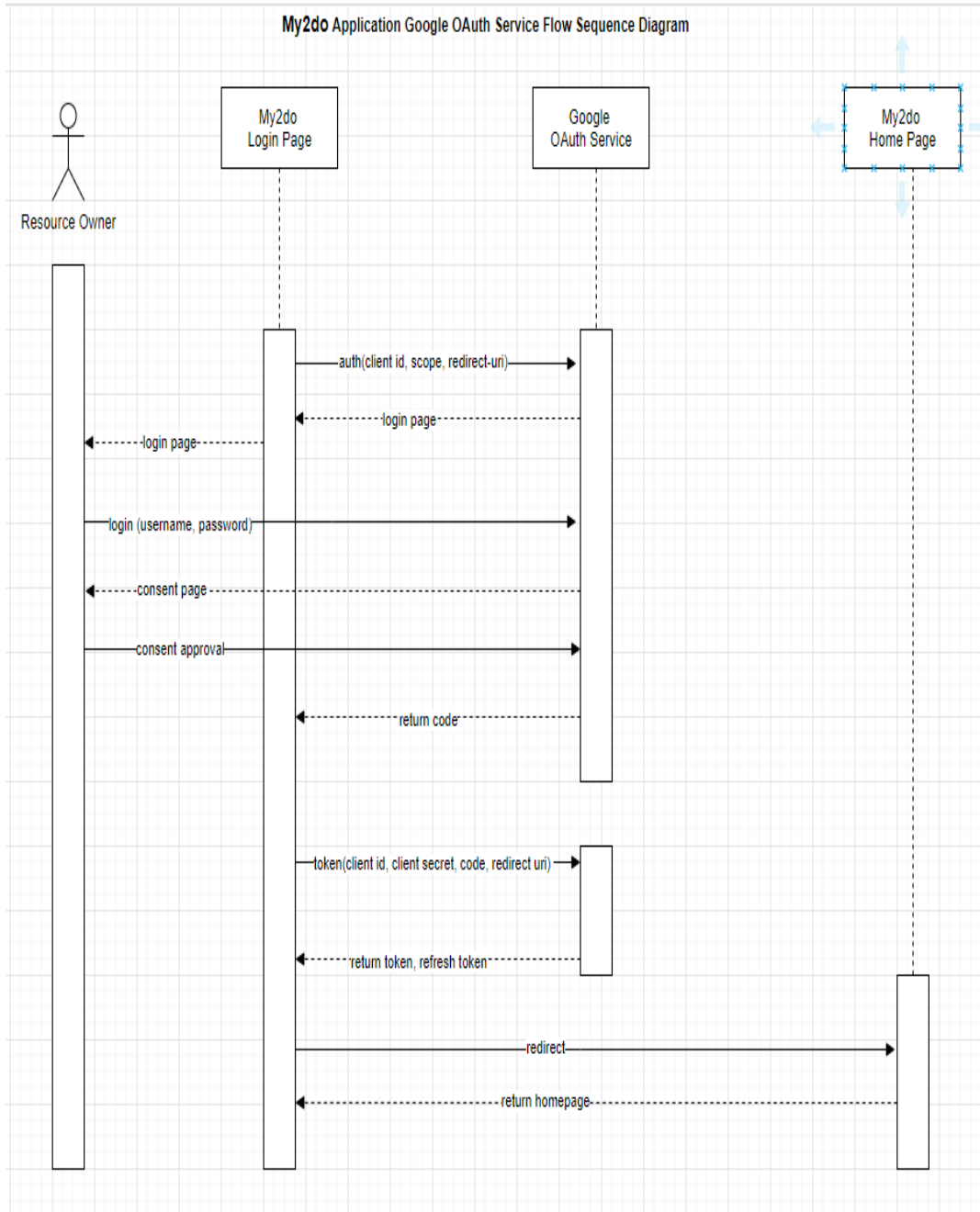


Şekil 17. Sınıf Diyagramı

3.3.3. Dinamik Model

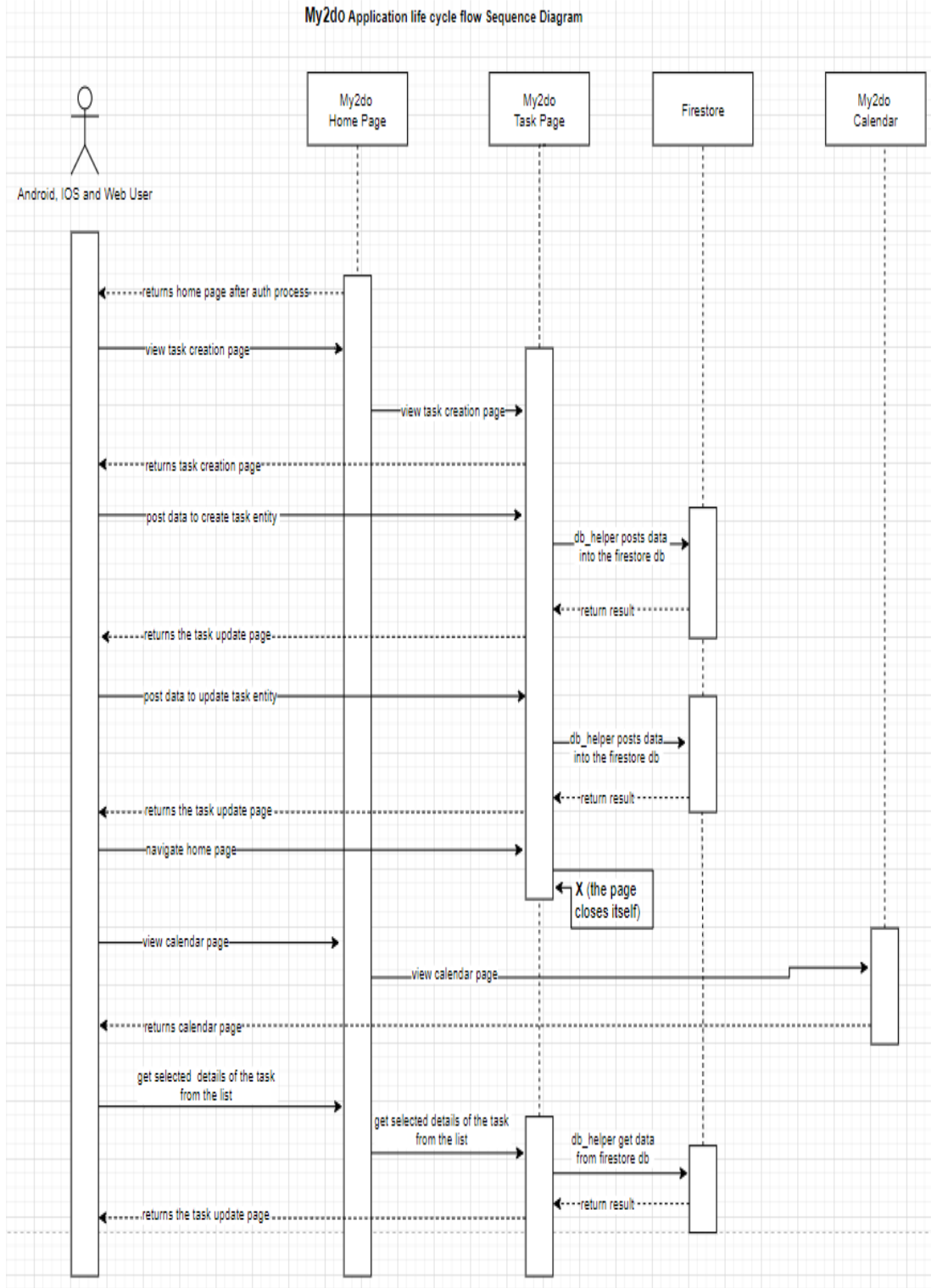
Sınıf diyagramlarına göre uygulamanın dinamik nesne etkileşimi aşağıdaki sıralı diyagramlarda gösterilmiştir.

Aşağıda Şekil 18’ de uygulamanın Google OAuth servisi ile etkileşime girerek kaynak sahibinin uygulamaya giriş nesne etkileşimi sıralı diyagramda (Sequence Diagram) gösterilmiştir.



Şekil 18. Google Auth Etkileşimi Sıralı Diyagramı

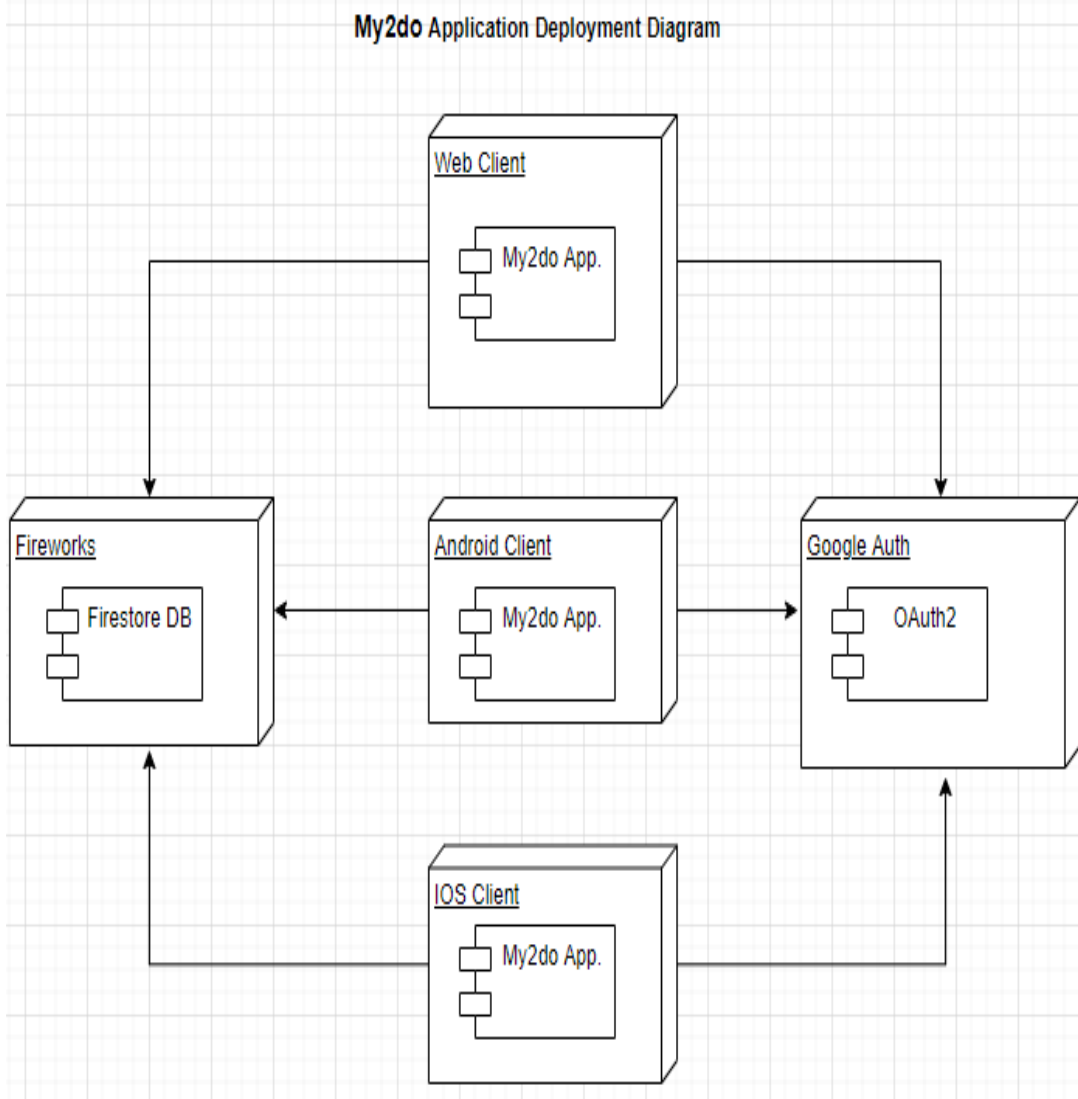
Aşağıda Şekil 19’ da gösterilen sıralı diyagramda (Sequence Diagram) uygulama kullanıcısının uygulama temel fonksiyonları ile nesne etkileşimi gösterilmiştir.



Şekil 19. Temel Nesne Etkileşim Sıralı Diyagramı

3.3.4 Alt Sistem Ayrıştırması

Aşağıda Şekil 20’ de uygulamasının alt sistem yapısını gösteren bileşen (Component Diagram) diyagramları açıklamalarıyla gösterilmiştir.



Şekil 20. Bileşen Diyagramı

DÖRDÜNCÜ BÖLÜM

UYGULAMANIN GELİŞTİRİLMESİ

1. Flutter Eğitimi

Uygulamanın Flutter ile geliştirilmesi öncesinde Flutter ve Dart dili üzerine eğitim veren kaynaklardan gerekli eğitimler alınmış, ayrıca seri olarak atölye çalışmaları yapılmıştır.^{22 23 24}

2. Kurulum ve IDE

Projenin yazılımı için öncelikle Flutter SDK, Android Studio, Mobil Emulator kurulumu ve ayarlamaları <https://flutter.dev/docs/get-started/install/windows> linki temel alınarak yapılmıştır.²⁵

2.1. Sistem gereksinimleri

Flutter'ı kurmak ve çalıştırmak için geliştirme ortamı minimum aşağıdaki gereksinimleri karşılamalıdır:

İşletim Sistemleri: Windows 7 SP1 veya üstü (64-bit), x86-64 tabanlı

Disk Alanı: 1.64 GB (IDE / araçlar için disk alanı içermez).

Araçlar: Flutter, bu araçların ortamda mevcut olmasına bağlıdır.

Windows PowerShell 5.0 veya daha yenisi (bu, Windows 10'a önceden yüklenmiştir)

Windows Komut İstemi'nden Git Kullan seçeneğiyle Windows 2.x için Git.

Windows için Git zaten yüklüyse, komut isteminden veya PowerShell'den git komutlarını çalıştırabildiğinizden emin olunur.

²² The Net Ninja, “Flutter Tutorial for Beginners”,2019, <https://www.youtube.com/watch?v=1ukSR1GRtMU&list=PL4cUxeGkcC9jLYp2Aoh6hcWuxFDX6PBJ/> , Erişim Tarihi: 6-21.04.2021

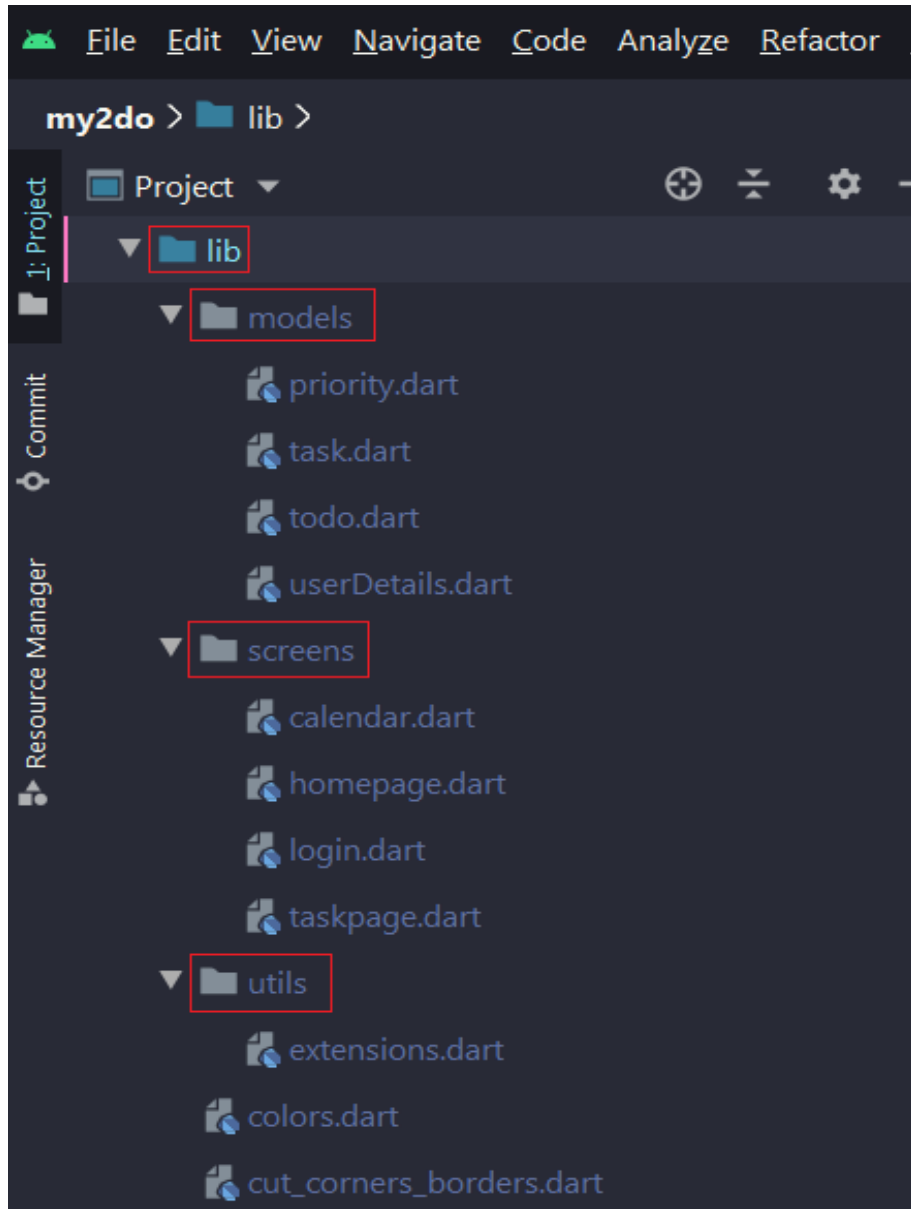
²³ TVAC Studio, “Flutter - Todo App from Scratch”, 2020, <https://www.youtube.com/watch?v=mOiXndQAZpw&list=PLGCjw1lRrtcSIUrd-Z-924b3ahWISiDh/> , Erişim Tarihi: 21-30.04.2021

²⁴ OA Başaran, “Flutter ile Yazılım Kariyerini Başlat | Kodlama Programlama Yazılım ve Uygulama Geliştirme Eğitimi”, 2020, https://www.youtube.com/watch?v=C3c_sBPXDkQ&list=PLNRtC6HXL3EDnn4naNGqlQSFqGyMXXWow , Erişim Tarihi:11-22.04.2021

²⁵ Flutter.dev, “Windows install”, 2021, <https://flutter.dev/docs/get-started/install/windows> , Erişim Tarihi: 05.05.2021

3. Proje Yapısı

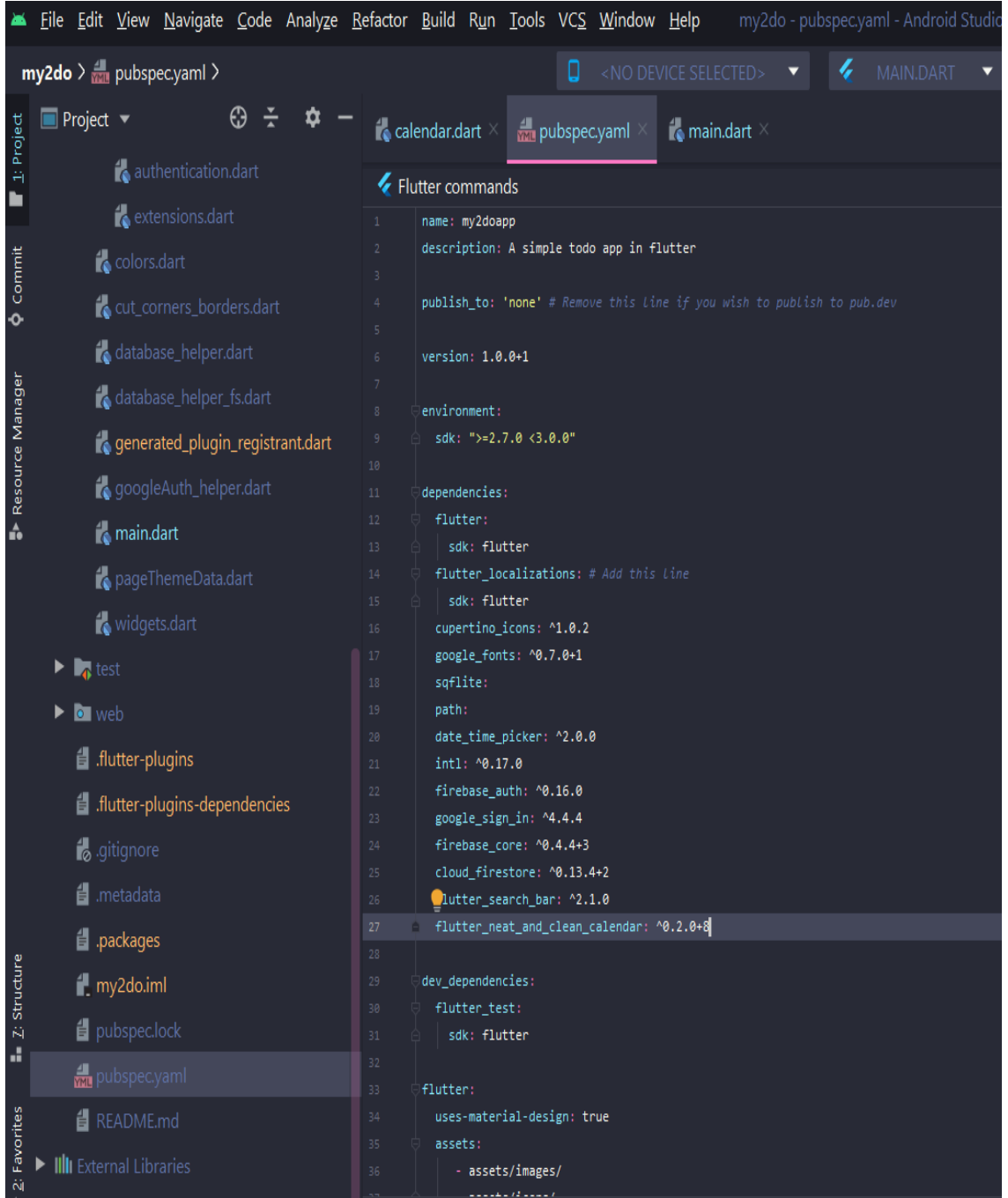
Android Studio aracı kullanılarak öncelikle Flutter projesi oluşturulmuş ve proje yapısı aşağıdaki Şekil 21’ deki gibi inşa edilmiştir. Buna göre uygulama sayfaları screens dizini altında, veri tabanı varlık nesneleri (entity) models dizini altında, Google kimlik doğrulama ve harici (extension) metotlar utils dizini altında ve ortak olan tüm nesneler ise kök dizin olan lib altında konumlanmıştır.



Şekil 21. Proje Yapısı

4. Proje Konfigürasyonu ve Kullanılan Kütüphaneler

Flutter projelerinde aşağıdaki Şekil 22’ de gösterildiği gibi yapılandırma ayarları pubspec.yaml dosyasında tutulmuştur. Projenin ismi, sürüm numarası, projenin geçerli olduğu Flutter SDK sürüm aralığı, bağımlı olduğu kütüphaneler ve diğer yapısal ayarlar pubspec.yaml dosyasında ayarlanmıştır.



Şekil 22. Pubspec.yaml Yapılandırma Dosyası

5. Uygulama Sınıfları

5.1. Main Metodu ve MyApp Sınıfı

Flutter uygulamasının yazılım dili olan Dart, diğer benzer dillerdeki gibi **main** metodu ile başlar. Yani uygulama kullanıcı tarafından çalıştırıldığında ilk main metodu çalışır. Daha sonra main metodu içinde runApp metoduna My2do uygulamasının ana sınıfı olan MyApp sınıfının nesne hali (instance) parametre olarak atanır.

MyApp sınıfı StatelessWidget sınıfından miras alır. StatelessWidget sınıfı statik bir sınıftır, yani durum (state) değişikliklerinden etkilenmez. StatelessWidget sınıfı' da Widget sınıfından miras alır. **Flutter' da herşey bir widget' tır.** Kullanıcının uygulama ile etkileşime geçtiği giriş alanları, menü, sekmeler, düğmeler, diyalog kutuları gibi bileşenlerde aslında birer widget' dır. Uygulamanın kendisi de diğer bileşenleri içeren en üst düzeyde bir widget' dır.

Aşağıdaki kodda MyApp sınıfının build metodu temel bir sınıf olan MaterialApp widget sınıfını döndürür. Bu sınıf uygulamanın temasının, lokalizasyon parametrelerinin ve sayfa geçişlerinin ayarlandığı görünmez bir sınıftır. home parametresine Login() sınıfı parametre olarak atanmıştır. Bu sayede uygulama ilk çalıştığında uygulama giriş (login) ekranı açılacaktır.

```
import 'package:flutter/material.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:my2doapp/screens/login.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primaryColor: Colors.indigo,
        textTheme: TextTheme(subtitle1: TextStyle(color: Colors.black)),
        outlinedButtonTheme: OutlinedButtonThemeData(
          style: OutlinedButton.styleFrom(primary: Colors.indigo),
        )
      )
    );
  }
}
```

5.2. Login Sınıfı

Login sınıfı uygulama giriş sayfasını kod olarak uygulayan sınıftır. Bu sınıftaki “Sign in with Google” düğmesinin ilgili olayından (event) **GoogleAuthHelper.signIn(context)** metodunun çağrılmasıyla beraber Google’ ın kullanıcı doğrulama sayfa açılır. Bu sayfada kullanıcı Google mail hesap bilgilerini girer, eğer başarılı ise uygulamanın ana giriş sayfası açılır.

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:my2doapp/models/userDetails.dart';
import 'package:my2doapp/screens/homepage.dart';
import '../googleAuth_helper.dart';
import '../widgets.dart';

class Login extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;
    return Scaffold(
      body: Builder(
        builder: (context) => Stack(
          fit: StackFit.expand,
          children: <Widget>[
            Container(
              height: size.height,
              width: size.width,
              decoration: BoxDecoration(
                color: Colors.indigo,
              ),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  RoundedRaisedButton(
                    size: size,
                    press: () async {
                      UserDetails userDetails = await
GoogleAuthHelper.signIn(context);
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => Homepage(userDetails),
                        ),
                      );
                    },
                    icon: "assets/icons/googleIcon30px.png",
                    text: "Sing in with Google",
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

5.3. Homepage Sınıfı

Homepage sınıfı uygulama ana sayfasını kod olarak uygulayan, StatefulWidget' dan miras alan bir sınıftır. StatefulWidget sınıflar durum değişikliklerine karşı etkileşim gösterir. Örneğin butona tıklayınca ekranın ortasındaki sayının 1 artması gibi.

Homepage sınıfta görevlerin listelendiği ListView widget, ekranın üst kısmında liste adı, arama ve takvim ikonlarının gösterildiği AppBar widget, diğer görev listelerine link verilen ve kullanıcı profilinin gösterildiği Drawer widget, yeni görev giriş sayfasını açan GestureDetector widget ve bu widget' ların hepsinin bağlı olduğu Scaffold widget yer alır. Implementasyon açısından söz konusu widget' larada bağlı olan alt widget bileşenleri vardır.

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import
'package:flutter_neat_and_clean_calendar/flutter_neat_and_clean_calendar.d
art';
import 'package:flutter_search_bar/flutter_search_bar.dart';
import 'package:my2doapp/database_helper_fs.dart';
import 'package:my2doapp/googleAuth_helper.dart';
import 'package:my2doapp/models/task.dart';
import 'package:my2doapp/models/userDetails.dart';
import 'package:my2doapp/screens/taskpage.dart';
import 'package:my2doapp/widgets.dart';
import '../widgets.dart';
import 'calendar.dart';

enum MenuItem { AllTasks, DoneTasks, OpenTasks }

class Homepage extends StatefulWidget {
  final UserDetails userDetails;

  Homepage([this.userDetails]);

  @override
  _HomepageState createState() => _HomepageState();
}

class _HomepageState extends State<Homepage> {
  DatabaseHelper_fs _dbHelper = DatabaseHelper_fs();
  Future<List<Task>> tasks;
  MenuItem menuItem = MenuItem.AllTasks;
  String selectedMenuItemText = "";
  SearchBar searchBar;

  @override
  void initState() {
    menuItem = MenuItem.OpenTasks;
```

```

getTasksByMenuItem();

searchBar = new SearchBar(
  inBar: false,
  buildDefaultAppBar: buildAppBar,
  setState: setState,
  onSubmitted: onSubmitted,
  onCleared: () {
    print("cleared");
  },
  onClosed: () {
    print("closed");
  });

super.initState();
}

void getTasksByMenuItem() {
  if (menuItem == MenuItem.AllTasks) {
    selectedMenuItemText = "All Tasks";
    tasks = _dbHelper.getTasks();
  } else if (menuItem == MenuItem.OpenTasks) {
    selectedMenuItemText = "Open Tasks";
    tasks = _dbHelper.getOpenTasks();
  } else if (menuItem == MenuItem.DoneTasks) {
    selectedMenuItemText = "Done Tasks";
    tasks = _dbHelper.getDoneTasks();
  }
}
...

```

5.4. TaskPage Sınıfı

TaskPage sınıfı uygulamanın ana sayfasından açılan, görev tanımlama ve güncelleme sayfasını kod olarak uygulayan sınıftır.

Görev başlığı girilip onaylanması ile görev Firestore veri tabanına kaydedilir. Daha sonra Visibility widget kullanılarak başlangıçta gizlenen diğer giriş alanları görev başlığı güncellendik sonra gösterilir. Sayfadaki her giriş özelliğindeki widget alanı onaylandıktan sonra bir sonraki widget' a otomatik odaklanması () sağlanır.

Göreve bağlı alt görevler eklenebilir. Eklendiğinde bu görevler aynı sayfa da listelenir. Listedenden seçilen görev güncellenebilir ya da silinebilir.

Görev sayfasından ana sayfa ya dönüş yapıldığında ana sayfa da eklenen ya da güncellenen görevin son hali listede gösterilir. Bu ana sayfanın miras aldığı StatefullWidget sınıfın setState methodu çağrılarak yapılır. Söz konusu setState

metodu çağrıldığında StatefullWidget' ın build metodu tekrar çağrılır. Sadece değişen widget' ların yapısı tekrar oluşturularak (Render) güncelleme sağlanır.

```
import 'dart:ui';
import 'package:date_time_picker/date_time_picker.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:my2doapp/database_helper.dart';
import 'package:my2doapp/database_helper_fs.dart';
import 'package:my2doapp/googleAuth_helper.dart';
import 'package:my2doapp/models/task.dart';
import 'package:my2doapp/models/todo.dart';
import 'package:path/path.dart';
import '../widgets.dart';

class TaskPage extends StatefulWidget {
  final Task task;

  TaskPage({@required this.task});

  @override
  _TaskPageState createState() => _TaskPageState();
}

class _TaskPageState extends State<TaskPage> {
  String _taskId = "";
  String _taskTitle = "";
  String _taskDescription = "";

  DatabaseHelper_fs _dbHelper = DatabaseHelper_fs();

  FocusNode _titleFocus;
  FocusNode _descFocus;
  FocusNode _dueDateFocus;
  FocusNode _priorityFocus;

  FocusNode _todoDescFocus;
  FocusNode _todoDueDateFocus;
  FocusNode _todoPriorityFocus;

  TextEditingController _titleController = TextEditingController();
  TextEditingController _descController = TextEditingController();
  TextEditingController _dueDateController = TextEditingController();
  TextEditingController _priorityController = TextEditingController(text:
"1");

  TextEditingController _todoDescController = TextEditingController();
  TextEditingController _todoDueDateController = TextEditingController();
  TextEditingController _todoPriorityController =
TextEditingController(text: "1");
  ...
}
```

5.5. CalendarPage Sınıfı

CalendarPage sınıfı uygulamanın ana sayfasının uygulama üst çubuğundaki takvim ikonundan açılan takvim sayfasını kod olarak uygulayan sınıftır. Bu sınıf yapısında görevler görev son tarihine (due date) göre takvim deki günlerde gösterilir. İlgili tarih tıklandığında görevin açıklaması ve önceliği takvimin açıklama bölümünde gösterilir. Bu alandan görev tıklandığında ise görev sayfa açılır.

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import
'package:flutter_neat_and_clean_calendar/flutter_neat_and_clean_calendar.d
art';
import 'package:my2doapp/models/task.dart';
import 'package:my2doapp/screens/taskpage.dart';
import "package:collection/collection.dart";

import '../database_helper_fs.dart';

class CalendarPage extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return _CalendarPage();
  }
}

class _CalendarPage extends State<CalendarPage> {
  DatabaseHelper_fs _dbHelper = DatabaseHelper_fs();
  Map<DateTime, List<NeatCleanCalendarEvent>> _events;

  @override
  void initState() {
    _events = generateEvents();
    super.initState();
  }

  Map<DateTime, List<NeatCleanCalendarEvent>> generateEvents() {
    List<NeatCleanCalendarEvent> _neatCleanCalendarEvents = [];
    Map<DateTime, List<NeatCleanCalendarEvent>> r = {};
    List<Task> tasks = [];

    _dbHelper.getTasks().then((element) {
      element.forEach((element) {
        var d = DateTime.now();
        tasks.add(element);
      });
    }).whenComplete(() {
      var newMap = groupBy(tasks, (obj) => obj.dueDate);
      newMap.forEach((k, tasks) {
        DateTime dueDate = DateTime.parse(k);
        List<NeatCleanCalendarEvent> events = [];
        tasks.forEach((element) {
          ...
```

5.6. Models Sınıfları

Uygulamanın ara yüzleri ile veri tabanı arasında köprü olan sınıflardır. Veri tabanındaki veriyi temsil ederler.

```
class Task {
  final String email;
  final String id;
  final String title;
  final String description;
  final String dueDate;
  final int priority;
  final int isDone;

  Task({this.email, this.id, this.title, this.description, this.dueDate,
    this.priority, this.isDone});

  Map<String, dynamic> toMap() {
    return {
      'email': email,
      'id': id,
      'title': title,
      'description': description,
      'dueDate': dueDate,
      'priority': priority,
      'isDone': isDone,
    };
  }
}
```

5.7. TaskCardWidget Sınıfı

Uygulama ana sayfasında listelenen her bir görevin kart içerisinde gösterilmesini sağlayan widget' dır. Bu widget diğer widget lardan oluşturulmuş özel bir widget' dır.

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

import 'models/priority.dart';

class TaskCardWidget extends StatelessWidget {
  final String title;
  final String desc;
  final String dueDate;

  TaskCardWidget({this.title, this.desc, this.dueDate});

  @override
  Widget build(BuildContext context) {
    return Container(
```

```

        margin: EdgeInsets.only(bottom: 20.0),
        width: double.infinity,
        padding: EdgeInsets.symmetric(vertical: 20.0, horizontal: 14.0),
        decoration: BoxDecoration(color: Colors.white, borderRadius:
BorderRadius.circular(20.0)),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(title ?? "(Unnamed Task)", style: TextStyle(fontSize:
20.0, fontWeight: FontWeight.bold)),
            Padding(
              padding: const EdgeInsets.only(top: 5.0),
              child: Text(desc ?? "No Description Added.", style:
TextStyle(fontSize: 18.0, height: 1.5)),
            ),
            Text(DateFormat("dd/MM/yyyy").format(DateTime.parse(dueDate)))
          ],
        ));
    }
}

```

5.8. TodoWidget Sınıfı

Görev sayfasındaki listelenen her bir alt görevin alanlarının gösterildiği widget' tır. Bu widget diğer widget' lardan oluşturulmuş özel bir widget' tır.

```

class TodoWidget extends StatelessWidget {
  final String text;
  final String dueDate;
  final int priority;
  final bool isDone;

  TodoWidget({this.text, this.dueDate, this.priority, @required
this.isDone, this.onUpdate, this.onDelete});

  final Function onUpdate;
  final Function onDelete;

  @override
  Widget build(BuildContext context) {
    final AlertDialog deleteDialog = AlertDialog(
      title: Text('DELETE ?'),
      content: Text('Do you want to delete this ToDo item?'),
      ...

```

5.9. DatabaseHelper_fs Sınıfı

Uygulama verilerinin tutulduğu bulut ortamındaki Firestore veri tabanı ile uygulama arasındaki verilerin oluşturulması, güncellenmesi, silinmesi ve listelenmesi için gerekli fonksiyonları barındıran yardımcı bir sınıftır.

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:my2doapp/googleAuth_helper.dart';
import 'models/task.dart';
import 'models/todo.dart';

// ignore: camel_case_types
class DatabaseHelper_fs {
  final _firestoreInstance = Firestore.instance;

  Future<String> insertTask(Task task) async {
    String docId = "";
    var docRef = (await
    _firestoreInstance.collection("Task").add(task.toMap()));
    docId = docRef.documentID;
    print("created doc id:$docId");
    return docId;
  }

  Future<void> insertTodo(String taskId, Todo todo) async {
    _firestoreInstance.collection("Task").document(taskId).collection("Todos")
    .add(todo.toMap());
  }

  Future<void> updateTaskTitle(String id, String title) async {
    _firestoreInstance.collection("Task").document(id).updateData({"title":
    title});
  }

  Future<void> updateTaskDescription(String id, String description) async
  {
    _firestoreInstance.collection("Task").document(id).updateData({"descriptio
    n": description});
  }

  Future<void> updateTaskDueDate(String id, String dueDate) async {
    _firestoreInstance.collection("Task").document(id).updateData({"dueDate":
    dueDate});
  }

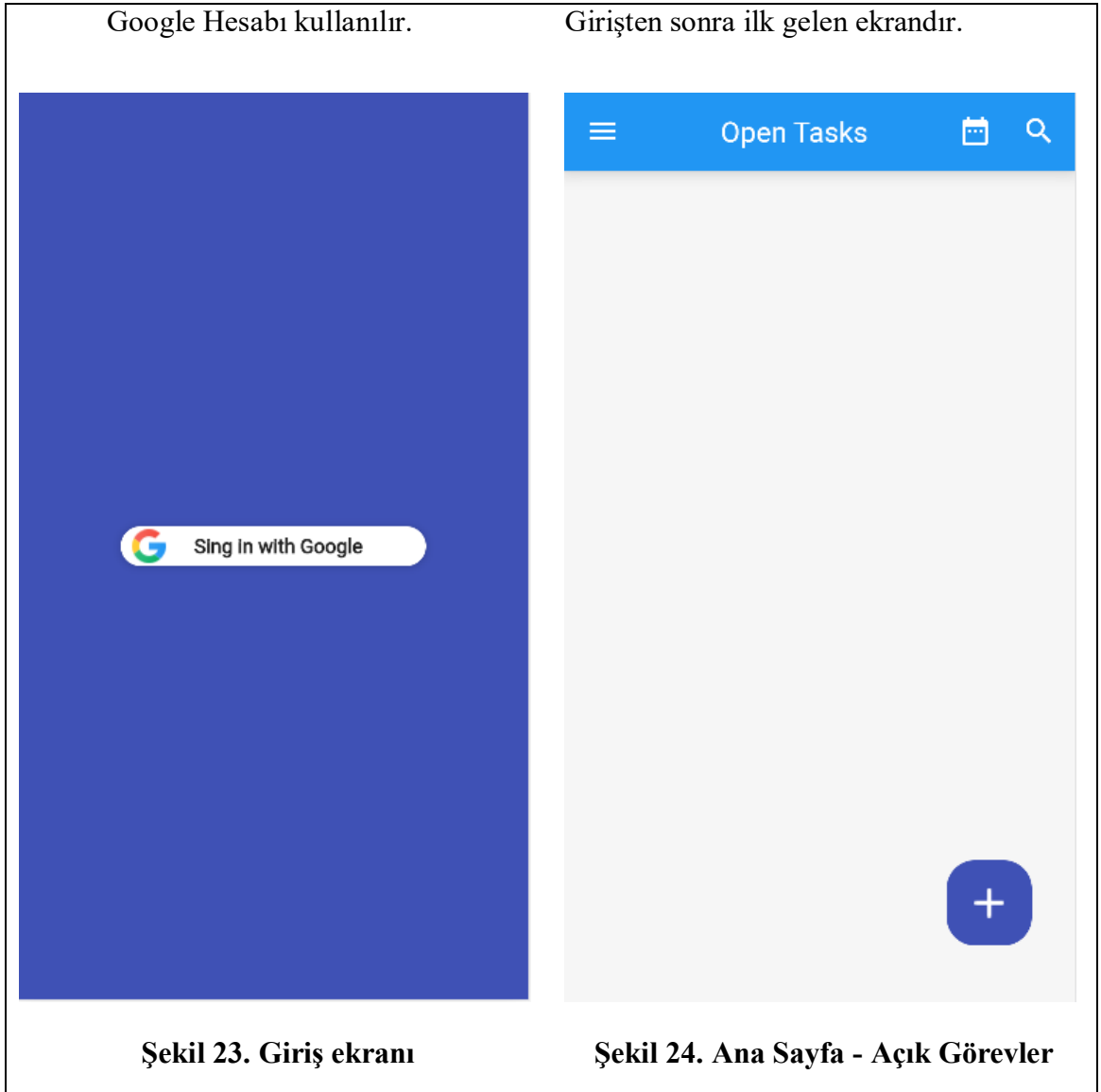
  Future<void> updateTaskPriority(String id, int priority) async {
    _firestoreInstance.collection("Task").document(id).updateData({"priority":
    ...
```

6. My2do Uygulamasının Yazılım Kodu

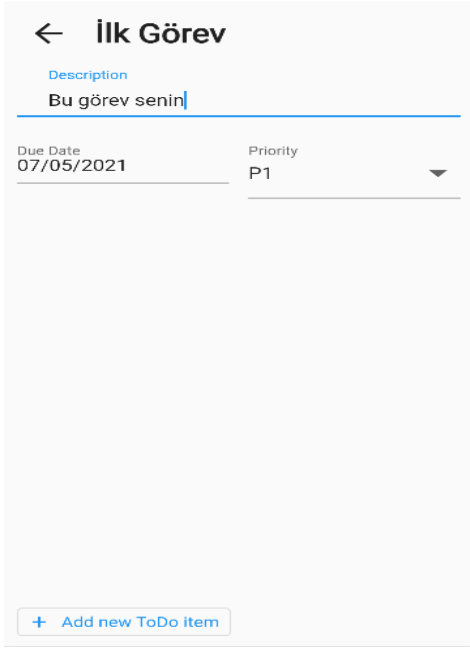
My2do uygulamasının tam kodu Gitlab’ da private olarak <https://gitlab.com/mguney78/my2do.git> isimli depoda yer almaktadır. Bitirme projesi kapsamında değerlendirilmesi gerektiğinde paylaşım yapılacaktır.

7. Kullanıcı Ara yüzleri

Aşağıdaki ekran şekillerinde çalışan uygulama sayfaları gösterilmiştir. Ayrıca https://drive.google.com/file/d/18eOsD_omAqo7c_GUgaTe6HZNhTVI37OO/view?usp=sharing linkinden uygulamanın çalışma anı canlı görüntülenebilir.

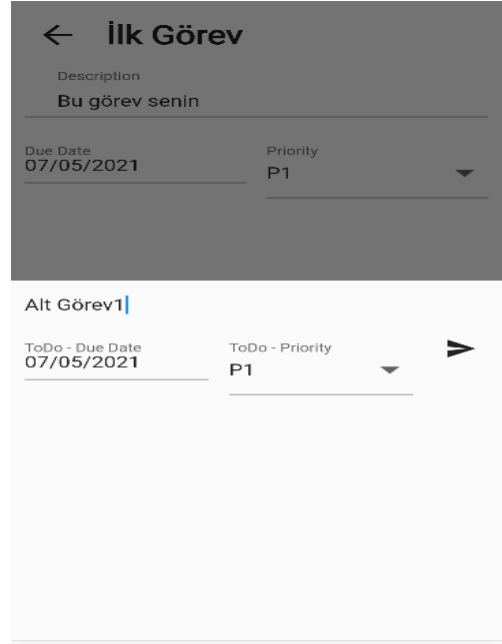


Ana ekrandan açılır Görev kaydeder.



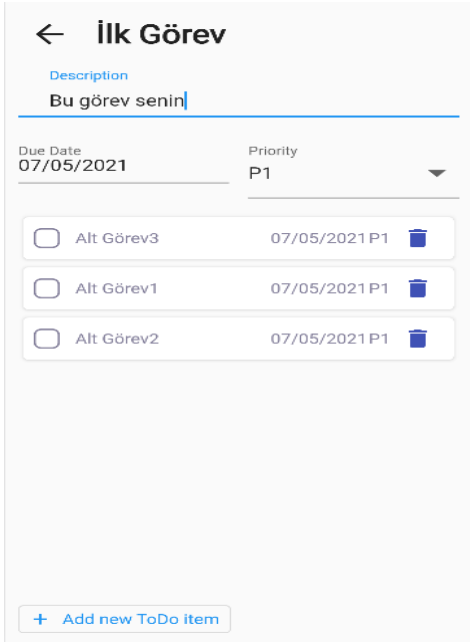
Şekil 25. Görev Giriş Ekranı 1

Göreve alt görev ekler.



Şekil 26. Görev Giriş Ekranı 2

Alt görev eklenmiş görev.



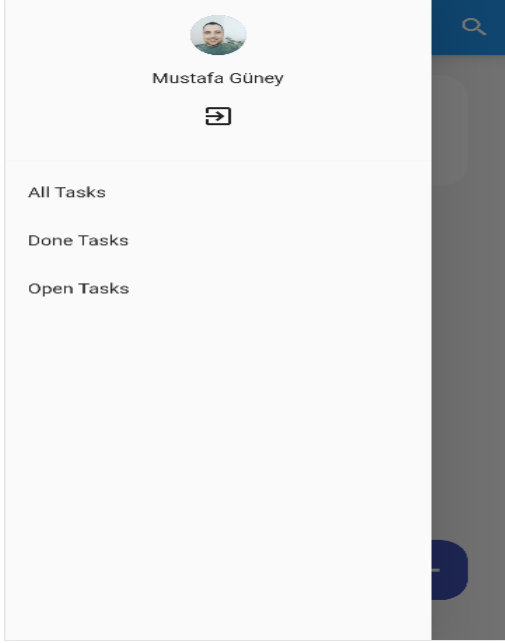
Şekil 27. Görev Giriş Ekranı 3

Tamamlanmış alt görevler.



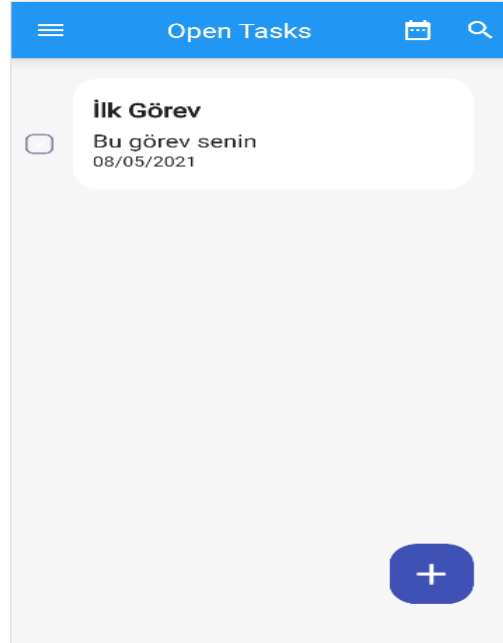
Şekil 28. Görev Giriş Ekranı 4

Kullanıcı profil ve diğer görev listeleri



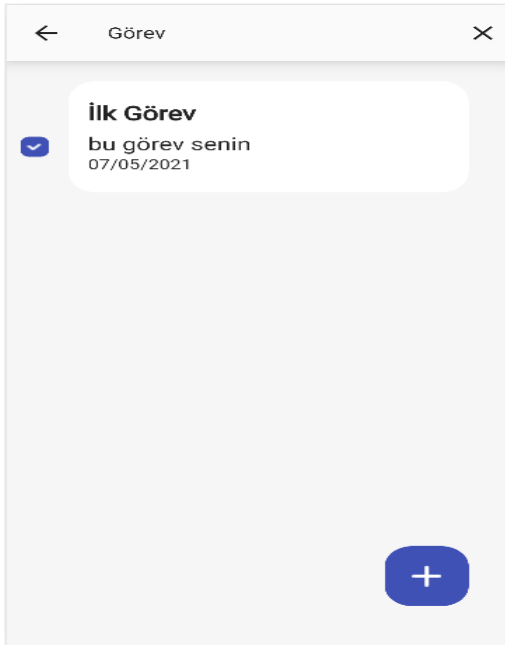
Şekil 29. Menü

Menüden açılan açık görevler



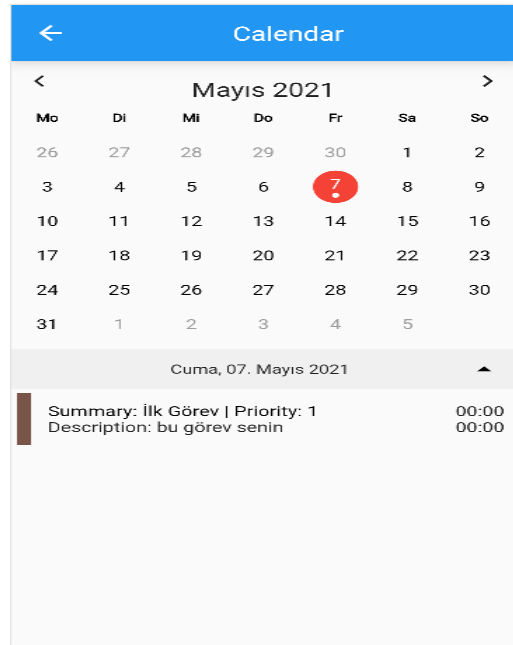
Şekil 30. Ana Sayfa – Görev Listesi 2

Üst uygulama çubuğundan açılan arama



Şekil 31. Ana Sayfa – Arama

Üst uygulama çubuğundan açılan takvim



Şekil 32. Takvim

SONUÇ

Bu çalışmada mevcut mobil uygulama türlerinden ve teknolojilerinden bahsedilmiş, daha sonra Google tarafından geliştirilen çoklu platform bir teknoloji olan Flutter' ın özellikleri ve mimarisi anlatılmış, sonrasında gereksinim ve teknik analizi yapılmış, ismi My2do olan bir To-Do uygulaması Flutter ile geliştirilmiştir.

Kodlama aşamasından önce Youtube' da kaynakları verilen alıştırma ve laboratuvar çalışmaları yapılmış, ayrıca çeşitli kaynak dokümanlar incelenmiştir.

Dördüncü bölümde hazırlanan analiz ve tasarım dokümanına göre, uygulama Android Stüdyo kullanılarak Windows işletim sisteminde Android, IOS ve Web platformları hedeflenerek yazılmıştır. İlk aşamada yerel bir dosya' da verilerin ilişkisel olarak tutulmasını sağlayan SqfLite veri tabanı kullanılmış fakat bu veri tabanı web platformuna uygun olmadığı anlaşıldığından bir bulut teknolojisi olan Firestore veri tabanı kullanılmıştır. Bunun için Firebase hesabı oluşturulmuş ve verilen talimatlara göre uygulama ile Firebase entegre edilmiştir.

Söz konusu durumlardan anlaşılacağı üzere yazılan uygulamanın hedeflenen tüm platformlara göre uygun alt yapıda geliştirilmesi gerektiği anlaşılmaktadır. Ayrıca mobil uygulamalarda kullanıcı deneyiminin önemli olduğu bilhassa anlaşılmıştır. Çünkü kullanıcı uygulamada istenmeyen bir hata ya da farklı bir durumla karşılaşması durumunda geri bildirim yapmaksızın uygulamayı kullanmaktan vazgeçebilmektedir. Bu nedenle bir sonraki sürümde uygulamanın hata ve kullanım kayıtlarının tutulması gerektiği anlaşılmıştır.

Sonuç olarak geliştirilen My2do isimli To-Do uygulamasının tüm hedeflenen Android, IOS ve Web platformlarında çalıştığı platform emülatör' leri ve tarayıcı üzerinden görülmüştür.

KAYNAKÇA

Argenova, **“Flutter Nedir ve Neden Öğrenmek Gerekir?”**, 2020, <https://www.argenova.com.tr/flutter-nedir-ve-neden-ogrenmek-gerekir>, Erişim Tarihi: 04.05.2021

Argenova, **“Xamarin Nedir - Artıları ve Eksileri Nelerdir?”**, 2019, <https://www.argenova.com.tr/xamarin-nedir-artilari-eksileri-nelerdir>, Erişim Tarihi: 04.05.2021

Baisden, A., **“Flutter vs. React Native”**, 2021, <https://blog.logrocket.com/flutter-vs-react-native/#:~:text=Key%20differences%20between%20Flutter%20and%20React%20Native,-Flutter%20and%20React&text=For%20starters%20Flutter%20uses%20the,and%20follow%20object%2Dorientated%20principles.>, Erişim Tarihi: 05.05.2021

Banazlı, S., **“Flutter nedir, nasıl çalışır, ne olacak? (Dikkat React N t ve çıkabilir)”**, 2019, <https://medium.com/kodcular/flutter-nedir-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-ne-olacak-dikkat-react-n-t-ve-%C3%A7%C4%B1kabilir-685c41977a88>, Erişim Tarihi: 05.05.2021

Başaran, O.A., **“Flutter ile Yazılım Kariyerini Başlat | Kodlama Programlama Yazılım ve Uygulama Geliştirme Eğitimi”**, 2020, https://www.youtube.com/watch?v=C3c_sBPXDkQ&list=PLNRtC6HXL3EDnn4naNGqlQSFqGyMXXWow, Erişim Tarihi: 11-22.04.2021

ÇINAR, Ç., **“ÇOKLU PLATFORM MOBİL UYGULAMALAR OLUŞTURMAK İÇİN UYGUN YAKLAŞIM SEÇME ANALİZİ”**, Yayınlanmamış Yüksek Lisans, İstanbul Üniversitesi, 2019

Duran, M., “**React Native vs Flutter**”, 2019, <https://medium.com/batech/react-native-vs-flutter-536c8cfbec11> , Erişim Tarihi: 04.05.2021

Flutter.dev, “**Flutter architectural overview**”, 2021, <https://flutter.dev/docs/resources/architectural-overview> , Erişim Tarihi: 05.05.2021

Flutter.dev, “**Supported platforms**”, 2021, <https://flutter.dev/docs/development/tools/sdk/release-notes/supported-platforms>, Erişim Tarihi: 05.05.2021

Flutter.dev, “**Windows install**”, 2021, <https://flutter.dev/docs/get-started/install/windows> , Erişim Tarihi: 05.05.2021

Guler, M.S., “**Flutter’ın Mimarisi ve Dart’ın Flutter üzerindeki etkisi**”, 2020, <https://medium.com/flutter-t%C3%BCrkiye/flutter%C4%B1n-mimarisi-ve-dart-%C4%B1n-flutter-%C3%BCzerindeki-etkisi-b9b652d0525> , Erişim Tarihi: 05.05.2021

İŞİTAN, M., “**ÇAPRAZ PLATFORM MOBİL UYGULAMA GELİŞTİRME ARAÇLARININ KARŞILAŞTIRILMASI VE DEĞERLENDİRİLMESİ**”, Yayınlanmamış Yüksek Lisans, Selçuk Üniversitesi, 2020

Javat Point, “**Flutter Architecture**”, 2021, <https://www.javatpoint.com/flutter-architecture>, Erişim Tarihi: 05.05.2021

Karlı, G., “**CROSS PLATFORM MOBILE DEVELOPMENT**”, Yayınlanmamış Yüksek Lisans, Dokuz Eylül Üniversitesi, 2014

Kinya Beatrice, “**Composition in Flutter**”, 2020, <https://kinya.hashnode.dev/composition-in-flutter-ckepqbojt02g7kps1224cbrdg> , Erişim Tarihi: 05.05.2021

Öner, M., “**Flutter Nedir ve Neden Flutter?**”, 2019, <https://www.muratoner.net/flutter/flutter-nedir-ve-neden-flutter>, Erişim Tarihi: 04.05.2021

QADIR, A.M., **“CROSS PLATFORM CARGO TRACKING SYSTEM”**,
Yayınlanmamış Yüksek Lisans, Fırat Üniversitesi, 2020

Sensor Tower, **“Top Apps Worldwide for February 2021 by Downloads”**, 2021,
<https://sensortower.com/blog/top-apps-worldwide-february-2021-by-downloads>

T. F. Bernardes, M. Y. Miyake, **“Cross-platform Mobile Development Approaches:
A Systematic Review”**, 2019,
<https://www.hindawi.com/journals/wcmc/2019/5743892/>, 02.05.2021

The Net Ninja, **“Flutter Tutorial for Beginners”**, 2019,
<https://www.youtube.com/watch?v=1ukSR1GRtMU&list=PL4cUxeGkcC9jLYyp2Aoh6hcWuxFDX6PBJ/> , Erişim Tarihi: 6-21.04.2021

TVAC Studio, **“Flutter - Todo App from Scratch”**, 2020,
https://www.youtube.com/watch?v=mOiXndQAZpw&list=PLGCjw1lRrtcSlUrd_-Z-924b3ahWISiDh/ , Erişim Tarihi: 21-30.04.2021

Vayes, **“Flutter Nedir? Nasıl Kullanılır? Özellikleri Nelerdir?”**, 2020,
<https://www.vayes.com.tr/tr/blog/flutter-nedir-nasil-kullanilir-ozellikleri-nelerdir#:~:text=burada%20bu%20konudan%20bahsedece%C4%9Fiz%3A,verimli%20%C5%9Fekilde%20uygulamalar%20yapabilece%C4%9Finizi%20d%C3%BC%5%9F%C3%BCn%C3%BCn> , Erişim Tarihi: 01.05.2021

Vikipedi, **“Dart (programlama dili)”**, 2020,
[https://tr.wikipedia.org/wiki/Dart_\(programlama_dili\)](https://tr.wikipedia.org/wiki/Dart_(programlama_dili)) , Erişim Tarihi: 04.05.2021

Webrazzi, **“Mobil Uygulama Geliştirme Türleri: Yerel, Hibrit, Platform Tabanlı Yaklaşımlar ve Daha Fazlası”**, 2020, <https://webrazzi.com/2020/12/22/mobil-uygulama-gelistirme-turleri-yerel-hibrit-platform/>, Erişim Tarihi: 01.05.2021

ÖZGEÇMİŞ

1978 yılında İstanbul' da doğdu. Eğitimini sırasıyla, Sarıgazi İlkokulu (1984-1989) Ümraniye/İstanbul, 60.Yıl Sarıgazi Ortaokulu (1989-1992) Ümraniye/İstanbul, Mehmetçik Lisesi - Fen Bölümü (1992-1995) Ümraniye/İstanbul ve Anadolu Üniversitesi/Açık Öğretim Fakültesi/Halkla İlişkiler (2006) bölümünde tamamlamıştır. Sonrasında Anadolu Üniversitesi/İşletme Fakültesi/İşletme bölümünü (2008) tamamlamıştır. 1998 yılından bu yana Lojistik ve Sigortalık sektörlerinde, yazılım mimarı olarak çalışmaktadır. 2017 yılından beri Türkiye Hayat Emeklilik firmasında Yazılım Mimarı olarak görevini sürdürmektedir. Yüksek Lisans programına 2020 yılında başlamıştır. Yabancı dili İngilizcedir. Evli ve 3 çocuk babasıdır.

Mustafa GÜNEY