

 **eKampus**
anadolum
e K a m p ü s
ve
anadolu mobil
dilediğin yerden,
dilediğin zaman,
öğrenme fırsatı!



(ekampus.anadolu.edu.tr)



(mobil.anadolu.edu.tr)

ekampus.anadolu.edu.tr



Açıköğretim Destek Sistemi
Açıköğretim Sistemi ile ilgili

merak ettiğiniz her şey AOS Destek Sisteminde...

- Kolay Soru Sorma ve Soru-Yanıt Takibi
- Sıkça Sorulan Sorular ve Yanıtları
- Canlı Destek (Hafta İçi Her Gün)
- Telefonla Destek

aosdestek.anadolu.edu.tr

AOS DESTEK Sistemi İletişim ve Çözüm Masası

0850 200 46 10

www.anadolu.edu.tr

AÖF Kitapları Öğrenci Kullanım Kılavuzu

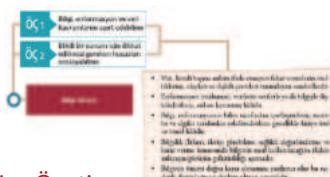
Bölüm 1

Temel Kavramlar

- | | | | |
|----------|--|----------|---|
| 1 | Önemli Kavramlar
1. Bilgi, bilinçlilik ve veri kavramlarının
arası farklılığı | 2 | Bilişim Modeli ve Biliş İşleme
Süreleri
2. Biliş işlemi sure ve aşamalarını
anımsalırmak |
| 3 | Bilgisayarların Dileşenleri
3. Bilgisayarın oluşturduğu bileylerin
sayısızlığı | 4 | Biliş İşleme ve Teknoloji
4. Biliş işlemi sureyeinde teknolojinin
oyunaklı roku söyleyebilme |
| 5 | Sosyal Hayatta Teknoloji
5. Teknolojinin sosyal yaşam üzerindeki
etkilerini konuşabilme | | |

Öğrenme çıktıları

Bölüm içinde hangi bilgi, beceri ve yeterlikleri kazanacağınızı ifade eder.



Bölüm Özeti

Bölümün kısa özetini gösterir.



- Maat houdt beperkt enkel de vormen van fijne condensatoren en dioden, omdat ze slecht presteren ten opzichte van verhitting.
 - Gevoelige temperatuur-sensoren werken uiterst goed bij hoge druktemperatuur, maar kunnen lekken.
 - De meest gebruikte fijne sensoren zijn piezoelektronische sensoren die reageren op mechanische veranderingen, zoals de piezoelektrische effecten van kristallen, glas en roestende metalen.
 - Piezoelektrische piezoelementen zijn ook geschikt voor de detectie van vaste en vloeibare stoffen en voor de detectie van kleine veranderingen in druk en temperatuur.
 - Piezoelektrische piezoelementen kunnen ook worden gebruikt voor de detectie van kleine veranderingen in druk en temperatuur.



TANIM

Bölüm içinde geçen
önemlı kavramların
tanımları verilir



Bölüm içinde geçen önemli kavramlardan oluşan sözlük ünite sonunda paylaştırılır.



Karekod

Bölüm içinde verilen karekodlar, mobil cihazlarınız aracılığıyla sizi ek kaynaklara, videolara veya web adreslerine ulaştırır.



Dikkat

Dikkat
Konuya ilişkin önemli uyarıları gösterir.



Neler Öğrendik ve Yanıt Anahtarları

Bölüm içeriğine ilişkin 10 adet
çoktan seçmeli soru ve cevapları
paylaşılır.

Öğrenme Çıktısı Tablosu

Araştır/İlişkilendir/Anlat-Paylaş

İlgili konuların altında cevaplayacağınız soruları, okuyabileceğiniz ek kaynakları ve konuya ilgili yapabileceğiniz ekstra etkinlikleri gösterir.

Yaşamla İlişkilendir

Bölümüne içeriğine uygun paylaşılan yaşama dair gerçek kesitler veya örnekleri gösterir.

Araştırmalarla İlişkilendir

Bölüm içeriği ile ilişkili araştırmaların ve bilimsel çalışmaları gösterir.

Mobil Uygulama Geliştirme

Editör

Arş.Gör.Dr. Hakan KILINÇ

Yazarlar

BÖLÜM 1 Doç.Dr. Hakan ALTINPULLUK

Prof.Dr. Kürşat ÇAĞILTAY

Dr. Serkan ALKAN

BÖLÜM 2 Dr.Öğr.Üyesi Fatih YAMAN

BÖLÜM 3 Dr. Özer ÇELİK

BÖLÜM 4 Doç.Dr. Onur DÖNMEZ

BÖLÜM 5 Dr.Öğr.Üyesi Ömer ARPACIK

BÖLÜM 6 Doç.Dr. Utku KÖSE

BÖLÜM 7 Doç.Dr. Serkan ÇANKAYA



T.C. ANADOLU ÜNİVERSİTESİ YAYINI NO: 4267
AÇIKÖĞRETİM FAKÜLTESİ YAYINI NO: 3037

Bu kitabın basım, yayım ve satış hakları Anadolu Üniversitesine aittir.
“Uzaktan Öğretim” teknüğine uygun olarak hazırlanan bu kitabı bütün hakları saklıdır.
İlgili kuruluştan izin alınmadan kitabı tümü ya da bölümleri mekanik, elektronik, fotokopi, manyetik kayıt
veya başka şekillerde çoğaltılamaz, basılamaz ve dağıtılamaz.

Copyright © 2022 by Anadolu University
All rights reserved

No part of this book may be reproduced or stored in a retrieval system, or transmitted
in any form or by any means mechanical, electronic, photocopy, magnetic tape or otherwise, without
permission in writing from the University.

Öğretim Tasarımcısı
Dr.Öğr.Üyesi Dönüş Çiçek

Grafik Tasarım ve Kapak Düzeni
Prof.Dr. Halit Turgay Ünalan

Dil ve Yazım Danışmanı
Öğr.Gör. Sebahat Yaşar

Ölçme Değerlendirme Sorumlusu
Mert Sarı

Grafikerler
Gülsah Karabulut
Ayşegül Dibek

Dizgi ve Yayıma Hazırlama
Beyhan Demircioğlu
Selin Çakır
Kader Abpak Arul
ZülfİYE Çevir
Gül Kaya
Diğdem Aydın
Burak Arslan
Halil Kaya

Mobil Uygulama Geliştirme

E-ISBN
978-975-06-4288-3

Bu kitabı tüm hakları Anadolu Üniversitesi'ne aittir.

ESKİŞEHİR, Mayıs 2022

3557-0-0-0-2209-V01

İçindekiler

BÖLÜM 1	Mobil Uygulama Geliştirme Bileşenleri	
---------	---------------------------------------	---

Giriş	3
Mobil Uygulama Tanımı, Kullanım Alanları ve Türleri	3
Mobil Uygulama Tanımları	3
Mobil Uygulama Kullanım Alanları	4
Mobil Uygulama Türleri	5
Mobil Uygulama Geliştirmek İçin Kullanılan Programlama Dilleri	8
Java	9
Javascript	9
HTML5	9
Swift	9
C#	10
C++	10
Objective C	10
Python	10
Kotlin	10
Scala	11
Dart	11
Ruby	12
Rust	12
Lua	13
PHP	13
Mobil Uygulama Geliştirmek İçin Kullanılan Platformlar	14
Solar 2D (Corona SDK)	14
Flutter	15
Apache Cordova (Phone Gap)	15
Codename One	15
React Native	16
Qt	17
Appery.io	17
Sencha Ext JS (Sencha Touch)	17
Xamarin	18
Mobil Uygulama Mağazaları	19
Google Play Store	19
App Store	20
Huawei AppGallery	21
Amazon Appstore	22
Microsoft Store	22

BÖLÜM 2	Mobil Arayüzlerde Kullanılabilirlik ve Mobil Uygulama Tasarımı	
---------	--	---

Giriş	33
Mobil Arayüzlerde Kullanılabilirlik	33
Kısa Tarihçe	33
Kullanılabilirlik Kavramının Önemi	34
Kullanılabilirliğin Temel Bileşenleri	34
Kullanılabilirliğin Tanımı	35
Kullanılabilirliğin Ölçülmesi	37
Tasarım Rehberleri	38
Kullanılabilirlik Testleri (Deneysel Yaklaşım)	38
Uzman Değerlendirmeleri ve Sezgiseller (Heuristics)	40
Model Temelli Yaklaşım	41
Mobil Uygulama Tasarımı	43
Mobil Uyumlu (Responsive) Tasarım ..	44
Mobil Site ve Mobil Uygulama Kavramları	45
Dikey Ekran - Yatay Ekran	46
Mobil Uygulamalarda Menü Yapısı....	47
Navigasyon/Gezinme	48
Dokunmatik Yüzeylerde Tasarım	49
Evrensel Tasarım ve Erişilebilirlik	52

BÖLÜM 3**Mobil Uygulama
Geliştirme
Platformları**

Giriş	61
Android Studio Kurulumu ve Arayüz Özellikleri	61
Android Studio'nun Bilgisayara Kurulumu	63
Android Studio Arayüz Özellikleri	71
Flutter Eklentisinin Kurulumu	76
Flutter'ın Android Studio'ya Kurulumu	77
Flutter Projesi Oluşturma	83
Emülatör Kurulumu	86
Örnek Uygulama Geliştirme	90

BÖLÜM 4**Mobil
Uygulamalarda
Arayüz Bileşenleri**

Giriş	99
Diyalog Ekranları	100
Alertdialog	100
Simpledialog	102
Showdialog	103
Buton Kontrolleri	103
Liste Kontrolleri	105
Listview	105
Listview.builder	107
Grafikler, Stiller ve Temalar	108
Grafikler	108
Stiller ve Temalar	110
İkon Kullanımı	112
Layout	114
Center	114
Row	115
Column	116
Spacer	116
Expanded	117
Container	117

BÖLÜM 5 Dart ile Kodlama



Giriş	125
Dart Kavramı	125
Dart Uygulamaları	126
Dart Dili	128
İşleçler (Operatörler)	128
Aritmetik İşleçler	129
Ön Ek ve Son Ek ve Birleşik Atama	
İşleçleri	129
İlişkisel İşleçler	130
Mantıksal İşleçler	130
Durumsal Atama	130
Anahtar Sözcükler	131
Değişkenler	131
Null Güvenliği (Null-Safety)	
Mekanizması	132
Akiş Yöneticiler	137
if-else	137
switch-case Yapıları	137
for Döngüleri	138
while Döngüleri	139
Fonksiyonlar	140
Değişkenlerin Etki Alanları	141
Dart ile Nesneye Yönelik Programlama	142
Cök Sayfali Uygulamalar	145
Başlangıç Sayfası	146
Giriş Sayfası	147
Çekmece Sayfası	148
pubspec.yaml Dosyası	150

BÖLÜM 6 Mobil Uygulamalarda Veri Tabanı Uygulamaları



Giriş	157
Asenkron (Async) İşlemler, Future ve Await	157
Flutter ile Sqlite İşlemleri	159
Sqlite Kurulumu	159
Veri Tabanı Oluşturma	160
Tablo Oluşturma	160
SQLite Veri Tipleri	161
Kayıt Ekleme (Insert)	161
Kayıt Getirme (Select)	162
ListView	164
Güncelleme – Update	165
Silme – Delete	166
Modellerle Sqlite İşlemleri	167
Flutter ile API İşlemleri	172
Kayıt Getirme (Get)	172
Kayıt Ekleme (Post)	174
Kayıt Güncelleme (PUT)	174
Kayıt Silme (Delete)	175

BÖLÜM 7**İleri Düzey
Uygulama
Seçenekleri**

Giriş	185
Sensör Tabanlı Uygulamalar	186
Flutter Ortamında Sensör Kullanımı ..	187
Kamera Kullanan Uygulamalar	194
Kamera Kullanım İzni	194
Kamera ile Fotoğraf Çekimi	197
Artırılmış Gerçeklik Uygulamaları	200
Flutter Ortamında Artırılmış Gerçeklik ...	203
Yapay Zeka Uygulamaları	206
Flutter Ortamında Yapay Zeka Çözümleri	208

BÖLÜM 8**Örnek Mobil
Uygulama
Geliştirme**

Giriş	221
Firebase Veri Tabanının Kurulması	222
Cloud Firestore	225
Telefon Defteri Uygulamasının Sayfalarını Tasarlama	227
Giriş Sayfasını Oluşturma	228
Kayıt Ol Sayfasını Oluşturma	232
Defter Sayfasını Oluşturma	234
Yeni Kayıt Sayfasını Oluşturma	236
Kayıt Düzenle Sayfasını Oluşturma	238

Önsöz

Sevgili Öğrenciler,

Teknolojik yakınsamanın neticesinde bir den fazla işlev sahip olan mobil cihazlar artık yaşamımızın vazgeçilmez bir unsuru haline gelmiştir. Buna bağlı olarak mobil uygulamaların da çok çeşitlendiği ve pek çok kategoride oldukça gelişmiş uygulamaların, uygulama mağazalarında yer aldığı görülmektedir. Gerçekleştirilen güncellemeler ile birlikte gün geçtikçe daha fazla özelliğe sahip olan ve insan hayatını oldukça kolaylaştıran mobil uygulamaların, Web sitelerinin yerini almaya başladığı da ifade edilebilir. Artık birçok kurumun daha fazla kullanıcıya ulaşmak, işlemlerini daha profesyonel bir şekilde gerçekleştirmek, marka değerini artırmak ve süreçlerini otomasyona taşımak için mobil uygulama geliştirmesi çok önemli bir hale gelmiştir. Özellikle akıllı telefonların herkesin cebinde olması ve yaygınlaşması bu alandaki girişimleri artırmaktadır. We Are Social 2022, Ocak raporuna göre tekil mobil telefon kullanıcı sayısı 2021 yılından bu yana 95 milyon daha artarak 5.31 milyar seviyesine ulaşmıştır. Aynı rapora göre bu sayı dünya nüfusunun %67.1 ine tekabül etmektedir. Bu veriler, mobil uygulama kullanımını bağlamında oldukça önemli bir potansiyelin olduğunu da gözler önüne sermektedir. Bu noktada içinde bulunduğumuz zamana uyum sağlayabilmek, çağın gerisinde kalmamak için gerek özel şirketlerin gerekse de kamuya bağlı kurumların mobil uygulama geliştirme konusunda kendisini geliştirmiş olan bireylere ihtiyaç duyduğu söylenebilir.

Mobil uygulama geliştirme bağlamında temel bilgilerden, ileri düzey bilgilere kadar oldukça kapsamlı bir içerik sunan ve son ünitesinde de örnek bir mobil uygulama geliştirme aşamasına degenen karekodlar ile desteklenmiş pratik uygulamalara dayalı bu kitabın, mobil uygulama geliştirme noktasında kendini ileriye taşımak isteyen herkese faydalı olmasını temenni eder tüm öğrenenlerimize başarılar dilerim.

Son olarak, editörlüğünü yapmış olduğum bu kitabın ortaya çıkışmasında büyük emek harcayan değerli yazarlara teşekkür ederim.

Saygı ve Sevgilerimle...

Editör

Araş.Gör.Dr. Hakan KILINÇ

Bölüm 1

Mobil Uygulama Geliştirme Bileşenleri

Öğrenme Çıktıları

Mobil Uygulama Tanımı, Kullanım Alanları ve Türleri

- 1 Mobil uygulamayı tanımlayarak kullanım alanlarını ve türlerini sıralayabilme

Mobil Uygulama Geliştirmek İçin Kullanılan Platformlar

- 2 En çok kullanılan mobil uygulama geliştirme platformlarını açıklayabilme

Mobil Uygulama Geliştirmek İçin Kullanılan Programlama Dilleri

- 3 Mobil uygulama geliştirmek için kullanılan programlama dillerinin özelliklerini ayırt edebilme

Mobil Uygulama Mağazaları

- 4 Mobil uygulama mağazalarını sıralayarak özelliklerini açıklayabilme

Anahtar Sözcükler: • Mobil Uygulama • Yerel Uygulamalar • Web Tabanlı Uygulamalar
• Hibrit Uygulamalar • Çapraz Platform Uygulamalar • Java • Scala • Flutter



GİRİŞ

Mobil cihazlar teknolojik yakınsamaya birlikte yaygınlaşmış, güçlü özellikler kazanmış ve her an her yerde yanımızda olan bilgi ve iletişim kaynaklarından en önemlisi haline gelmiştir. Mobil cihazlar yıllar geçtikçe ucuzlaşmış ve her yeni sürümde daha da gelişmiştir. Bu durum, mobil cihazları büyük kitlelerin hayatındaki vazgeçilmez unsurlardan biri haline getirmiştir. Buna bağlı olarak mobil uygulamaların da çok çeşitli olduğu ve pek çok kategoride oldukça gelişmiş uygulamaların, uygulama mağazalarında yer aldığı görülmektedir. Hatta mobil web sitelerinin günlük ortalama kullanım süresinin 12 dakikaya kadar düşüğü ve mobil uygulama kullanımının giderek daha fazla arttığı gözlenmektedir. Mobil web sitelerinden ziyade mobil uygulama kullanımının artmasında en büyük etkenin daha az problem yaşatması ve daha fazla dolaşım sağlanması olduğu söylenebilir. Artık işletmelerin daha fazla kullanıcıya ulaşmak, işlemelerini daha profesyonel bir şekilde gerçekleştirmek, marka değerini artırmak ve süreçlerini otomasyona taşımak için mobil uygulama geliştirmesi çok önemli bir hale gelmiştir. Özellikle akıllı telefonların herkesin身边inde olması ve yaygınlaşması bu alandaki girişimleri artırmaktadır. Mobil cihazların kullanımıyla ilgili çarpıcı istatistiklerden biri, sıradan bir akıllı telefon kullanıcısının, akıllı telefonunu içinde 63 kez kontrol etmesidir. Her gün insanların yaklaşık %49'u 11'den fazla kez bir uygulama açmaktadır. Ortalama olarak, akıllı telefon sahipleri her ay yaklaşık 30 farklı uygulama kullanmaktadır.

Global Stats Şubat 2022 verilerine göre, mobil cihazların özellikle akıllı telefonların etkisiyle pazar payının %56.05 olduğu, bunu %41.52 ile masaüstü ve %2.43 ile tabletlerin izlediği görülmektedir. Türkiye'de bu oran daha da yüksektir. Türkiye'de mobil cihazların pazar payı %70.23 iken, masaüstü %27.9, tablet %1.87 oranında kalmaktadır. Bu verilerden görüldüğü gibi akıllı telefonların önderliğinde mobil cihazların pazar payı oranı gün geçtikçe masaüstü pazar payı orANIyla arasındaki farkı açmaya devam etmektedir.

We are social (2022) tarafından hazırlanan Digital 2022 raporunda da mobil cihazların yükselişi dikkat çekmektedir. Ocak 2022 verilerine göre, dünya nüfusunun 5.31 milyarı (%67.1) mobil telefon kullanıcısıdır. Bu oranın internet kullanıcısı oranından (%62.5) yüksek olması dikkat çekicidir.

Bir önceki yıla göre mobil telefon kullanıcı sayısı 95 milyon artmıştır.

İnternete erişim sağlayan bireylerin %90.7'si akıllı telefonlardan internete girmektedir. Bu oranın laptop ve masaüstü bilgisayarlardan internete girenlerde %71.2'de olması akıllı telefonların geleceği hakkında bazı ipuçları vermektedir. Oyunlarda da durum farklı görünmemektedir. 16-64 yaş arası internet kullanıcılarının %68.1'i akıllı telefonlardan %36.8'i laptop ve masaüstü bilgisayarlardan %25.8'i oyun konsollarından, %17.2'si tabletlerden oyun oynamaktadır. Olanlara bakıldığında akıllı telefonların oyun oynama alışkanlıklarını da değiştirdiği ve mobil cihazlara doğru bir kayma gerçekleştiği söylenebilir.

Bu bölüm kapsamında mobil uygulama tanımları, kullanım alanları ve türleri açıklanıktan sonra, mobil uygulama geliştirmek için kullanılan programlama dillerinden bahsedilmektedir. Sonrasında mobil uygulama geliştirmek için kullanılan platformlar ve mobil uygulama mağazalarının tanıtılmasıyla bölüm tamamlanmaktadır.

MOBİL UYGULAMA TANIMI, KULLANIM ALANLARI VE TÜRLERİ

Bu başlık altında mobil uygulamaların tanımları, hangi kullanım alanlarına sahip olduğu ve mobil uygulamaların türleri inceleneciktir.

Mobil Uygulama Tanımları

Uygulamalar (application ya da kısaca app) genellikle cihazda yerel olarak çalışma da bir web tarayıcısı aracılığıyla da kullanılabilen yazılım türüdür. Uygulamalar bilgisayarda, akıllı telefonda, tablette, akıllı TV'ler ve akıllı saatler dahil diğer elektronik cihazlarda bulunabilmektedir. Uygulamalar genellikle internet bağlantısı ile çevrimiçi (online) olarak kullanıldığı gibi, internet bağlantısı olmadan çevrimdışı (offline) olarak ta kullanılabilmektedir. Masaüstü uygulamaları, fare ve klavye içeren bilgisayarlar için geliştirilirken, mobil uygulamalar, akıllı telefonlar ve dokunmatik ekranlar için tasarlanmıştır.

Mobil uygulama kavramı, telefon/akıllı telefon, tablet, akıllı saat veya akıllı gözlük gibi bir "taşınabilir" aygıttA çalışacak şekilde geliştirilen bir yazılım türü olarak tanımlanmaktadır. Masaüstü

bilgisayarlardaki yazılım ve uygulamaların aksine, mobil uygulamalar mobil cihazlara uyumlu olacak şekilde tasarlanmaktadır. Başlangıçta takvim, e-posta ve diğer iletişim bileşenleri için tasarlansa da daha sonra e-ticaret uygulamalarından, konum tabanlı uygulamalara, mobil oyunlardan sosyal medya araçlarına kadar çok geniş bir yelpazede kullanım alanları bulmuştur. Mobil uygulamalar genellikle sınırları çizilmiş fonksiyonlara sahip küçük yazılım uniteleri olsa bile, kullanıcılarla kaliteli hizmetler ve deneyimler sunma amacıyla taşımaktadır.

Bu uygulamalar, içinde yaşadığımız teknoloji odaklı dünyanın önemli bir parçası olarak bir bireyin yaşamını, üretkenliğini ve eğlenme düzeyini artırabilmektedir. Mobil uygulamalar işletmeler için de bazı avantajlar oluşturmaktadır. Mobil uygulamalar maliyeti düşük bir yazılım türü olduğu için işletmelerin verimliliğini artırabilmektedir. Üretimi kolaylaştmak ve işlerin kolayca yapılabililığını artırmak için hem büyük hem de küçük şirketler tarafından aktif bir biçimde kullanıldığı görülmektedir. Mobil uygulamalar işletmenin görünürüğünü artırmakta, müşteri sadakatini yükseltmekte, doğrudan bir pazarlama kanalı olarak kullanılabilmekte, daha iyi bir marka farkındalığı yaratmakta, kârı artırmakta ve daha iyi bir müşteri hizmeti sunma noktasında katkı sağlamaktadır. Ayrıca günümüzde mobil cihazların ve uygulamaların yoğun kullanımı göz önünde bulundurulduğunda, geniş bir etki alanı sağlamaktadır.

Mobil Uygulama Kullanım Alanları

Mobil uygulamalar kullanım alanları bağlamında çeşitli şekillerde sınıflandırılmaktadır. Bu bölümde bazı uygulamalara degenilse de, daha pek çok alanda mobil uygulamaların aktif biçimde kullanıldığı söylenebilir.

Eğitim uygulamaları: Eğitsel uygulamalar kullanıcıların ya da öğrenenlerin yeni bilgi ve beceriler kazanmasına destek olan uygulamaları içermektedir. Öğrenmede esnekliği sağlama ve öğrenme sürecini oyunlaştırması özellikleyle öğrenenler arasında popüler olması kadar, öğretim sürecinde organizasyonu kolaylaştırması ile öğretmenler arasında da etkin kullanım alanı bulmaktadır.

M-ticaret uygulamaları: Mobil ticaret uygulamaları, ürünlere kolay erişim ve sorunsuz ödeme gibi özellikleyle müşterilere/kullanıcılar en iyi alışveriş deneyimini sunmaktadır.

Sosyal ağ uygulamaları: Dünyada ve Türkiye'de mobil uygulamaların en popüler kategorilerindendir. Facebook, Instagram, Twitter gibi bilinen örnekleri vardır.

Oyun uygulamaları: Mobil uygulamaların en popüler kategorisi olarak, mobil uygulama geliştirici işletmeler için en fazla kar getiren uygulama türlerini kapsamaktadır. Hatta son yıllarda masaüstü video oyunlarının yerine daha çok mobil oyunların geliştirildiği gözlenmektedir. Son araştırmalara göre, mobil oyunlar tüm uygulama indirmelerinin %33'ünü, uygulama harcamalarının %74'ünü ve uygulamaları kullanarak harcanan tüm zamanın %10'unu oluşturmaktadır.

Eğlence uygulamaları: Bu uygulamalar, kullanıcıların yeni içerik üretmesine, sosyal aktivite aramasına, sohbet etmesine veya çevrimiçi müzik/video içeriği dinlemesine/izlemesine olanak tanımaktadır. Facebook veya Instagram gibi sosyal medya uygulamaları, YouTube, Netflix veya Amazon Prime Video gibi video akışı uygulamaları, tüm dünyadaki kullanıcılar arasında popüler olan en bilinen uygulamalardandır.

Yaşam tarzı (lifestyle) uygulamaları: Bu geniş uygulama kategorisi alışveriş, moda, egzersiz ve diyet gibi uygulamaları kapsamaktadır. Bu uygulamalar temel olarak kişisel yaşam tarzının çeşitli yönlerine odaklanmaktadır.

Seyahat ve harita uygulamaları: Bu uygulamaların gerisindeki temel düşünce, bireylerin rafatlıkla seyahat etmelerine yardımcı olmaktır. Google Haritalar, Airbnb, TripAdvisor, Bolt veya Uber gibi örnekler verilebilir.

Yardımcı (utility) uygulamalar: Yardımcı uygulamalar hemen hemen her gün ve çoğu kişi tarafından kullanılan, belirli ihtiyaçları hızlı bir şekilde karşılamak için yararlanılan uygulamaldır. Bu tür uygulamalara örnek olarak hesap makinesi, çalar saat, QR kod/barkod tarayıcı veya el feneri verilebilir.

Bu uygulamaların dışında aşağıdaki kategorilerde uygulama örnekleri de bulunmaktadır:

- İş veya üretkenlik (productivity) uygulamaları
- Mobil sağlık (m-health) uygulamaları
- Spor uygulamaları
- Hava durumu uygulamaları
- Alışveriş uygulamaları
- Kitap uygulamaları

- Finans ve bankacılık uygulamaları
- Yiyecek içecek uygulamaları
- Dergi ve gazete uygulamaları
- Müzik uygulamaları
- Haber uygulamaları
- Fotoğraf ve video uygulamaları

Mobil Uygulama Türleri

Mobil uygulamalar türlerine göre temel olarak şu şekilde kategorize edilebilir:

- Yerel uygulamalar
- Web tabanlı uygulamalar
 - İleri Web uygulamaları
- Hibrit uygulamalar
- Diğer mobil uygulama türleri
 - Çapraz platform uygulamalar

Yerel (Native) Uygulamalar

Yerel (native) uygulamalar, çoğu insanın “uygulama” kelimesini duyduğunda aklına gelen ilk uygulama türüdür. Yerel uygulamalar genellikle belirli bir işletim sistemi (iOS, Android, Windows vb.) üzerinde çalışacak şekilde özel olarak geliştirilen uygulama türüdür. Hedef kitleye erişim sağlamak için farklı işletim sistemleri için ayrı uygulamalar tasarlamak gerekmektedir. Yerel bir uygulama geliştirmek, her işletim sistemi için ayrı uygulamalar oluşturmayı ve bunların özelliklerini hesaba katmayı gerektirmektedir. İlgili işletim sistemine özel olarak geliştirilen bir yerel uygulama farklı işletim sistemi türlerinde kullanılmamaktadır. Başka bir deyişle, iOS uygulamaları Android işletim sistemine sahip telefonlarda kullanılmadığı gibi, bunun tersi de geçerlidir. Belirli bir işletim sistemi için özel olarak oluşturuldukları için, uygulamaların yerleşik olduğu programlama dilleri de işletim sistemine özeldir.

Doğası gereği bu uygulamalar daha yüksek geliştirme maliyetlerine ve daha uzun bir pazara inme süresine yol açmaktadır. Ancak aynı zamanda uygulamaların her cihazın ve işletim sisteminin özelliklerinden tam olarak yararlanması anlamına gelmektedir.

Yerel uygulamaların bazıları internet bağlantısı gerektirse de, internet bağlantısından bağımsız olarak ta çalışabilmektedir. Bu uygulamalar yerel cihaz kullanıcı arayüzüünü kullanmaktadır. Kotlin, Swift, Python, Java, Objective-C gibi çeşitli programlama

dilleriyle yerel bir uygulama geliştirilebilmektedir. Bu uygulamalar, işletim sisteme özel mobil uygulama mağazalarında kolayca bulunabilmektedir.

Cihaza ve işletim sisteme özel basit kod yapısı sayesinde hızlı ve sezgisel bir performans göstermektedirler. İşletim sisteminin ve cihaza özgü işlevlerin daha iyi kullanıldığı görülmektedir. Daha az bellek kaplayan bu uygulamalar, hızlıdır ve daha az pil gücü kullanırlar. Oldukça güvenilir olan bu uygulamalar cihaz kullanıcısının izniyle cep telefonunun donanımına erişmelerine izin verilmektedir. Bu da kullanıcıların verilerinin korunmasını sağlamaktadır.



dikkat

Bu uygulamalar, cihaz özelliklerine erişerek uygulamanın genel deneyimini iyileştirir.

Verimli kodlama sistemleri sayesinde, optimum donanım gereksinimlerine ihtiyaç duyar. Yapılandırması daha kolaydır ve uyumluluk sorunları daha azdır. Kişiselleştirmeye uygundur. Duyarlıdır ve sorunsuz, hızlı ve etkileşimli bir kullanıcı arayüzüne sahiptir. Arayüz tanıdık bir görünüm ve his verir.

Bu uygulamaların bazı dezavantajları da bulunmaktadır. İşletim sisteme özel uygulamalar oluşturmak zaman alabilmektedir. Ayrıca, Swift ve Java gibi işletim sistemine özel programlama dillerini öğrenmek oldukça zordur. Belirli bir işletim sistemi ile sınırlı olduğundan dolayı, hedef kitleyi de sınırlamaktadır. Uygulamanın bakımı ve güncellenmesi yoğun bir maliyet ve emek gerektirir. Farklı platformlar için farklı uygulamalar oluşturulması gereklidir. Yeni özellikler için ayrı bir kod tabanı gereklidir.

Özetle yerel uygulamalar, yüksek güvenlikli ve tek bir kod tabanına sahip olacak uygulamalar için en uygun seçimidir. Haber veya blog gibi içerik açısından zengin durumlarda yerel uygulamaların geliştirilmesi uygun bir seçenek olmayacağıdır. Bu uygulamaların geliştirilmesi maliyet gerektirdiğiinden dolayı düşük bir bütçe de yerel bir uygulama tasarlama noktasında problem oluşturacaktır.



dikkat

WhatsApp, Spotify, SoundCloud yerel uygulamalara örnek olarak verilebilir.

Web Tabanlı (Web-Based) Uygulamalar

Web tabanlı uygulamalar yerel uygulamalara benzer özellikler taşımamasına rağmen, mobil cihazda ki web tarayıcıları (Opera, Chrome, Firefox, Safari vb.) aracılığıyla erişilen uygulamalardır. Yerel uygulamalardan farklı olarak cihazın donanımına erişememekte ve HTML5, CSS, Javascript gibi popüler programlama dilleriyle daha kolay geliştirilmektedir. Yerel uygulamalarla arasındaki en büyük farklardan biri, yerel uygulamaların aktif bir internet bağlantısı kullanıp kullanmamasıyla ilgilidir. Yerel uygulamalar hem aktif bir internet bağlantısı olmadan çevrim dışı modda hem de çevrim içi modda çalışabilirken, web uygulamalarının çalışması için aktif bir internet bağlantısı gerekmektedir.

Uygulamayı kullanmak için herhangi bir depolama alanı veya yükleme işlemi gerekmemektedir. Bir kullanıcının uygulamayı indirmesi veya doğrudan cihazında bilgi depolaması gerekmez.



dikkat

Web tabanlı uygulamalar herhangi bir uygulama mağazasında satılamazlar.

Bu uygulamalar bilgisayarınızda veya akıllı telefonunuzda yüklü olmadığından, web'de barındırılan sunucularda kendilerini güncelledikleri için web uygulamasını güncellemeye gerek yoktur. Web uygulamaları düşük işletme maliyeti, kurulum gerektirmemesi, her yerden erişilebildiği için daha iyi erişim sağlama, daima güncel kalması gibi avantajlara sahiptir. Ancak, çevrimdışı olarak çalışması, yerel uygulamalara kıyasla sınırlı sayıda işlevsellik içermesi, geliştirilmesinin uzun zaman alması ve güvenlik riskleri nedeniyle web uygulamaları bazı dezavantajlara sahiptir.

Mobil web uygulamaları, çeşitli ekran boyutlarına ve cihazlara kolayca uyum sağlamaktadır. Duyarlı (responsive) web uygulamaları (siteleri), bir mobil cihazdan erişildiğinde farklı bir tasarıma geçmektedir. Uyarlanabilir (adaptive) web uygulamaları ise, mobil cihazların farklı ekran boyutlarına uyacak şekilde ölçeklenmektedir. Bu uygulamalar için tasarım değişimmemektedir.

İleri (Progressive) Web Uygulamaları (PWAs)

Bu uygulamalar teknik anlamda tamamen ayrı bir mobil uygulama türü olmasa da web tabanlı uygulamaların altında sınıflandırılabilir. Bazı kaynaklarda hibrit uygulamalarla da benzerliklerine atıf yapılmaktadır. Bu bölüm kapsamında web tabanlı uygulamaların bir alt kategorisi olarak değerlendirilmektedir. İleri web uygulamaları, normal web uygulamalarına benzer web teknolojilerini kullanmasına rağmen, çok daha üstündür ve mobil web'in geleceği olarak kabul edilmektedir. Gelişmiş bir kullanıcı deneyimi sunmalarına yardımcı olan standart web uygulamalarından daha fazla işlevselliğe sahiptir. Kullanıcılar, yerel bir uygulama gibi, cihazlarına bu mobil uygulamaları yükleyebilir ve başlatabilir. HTML5, CSS ve JavaScript üzerinde çalışıkları gibi, ReactJS, Vue.js ve benzeri çerçeveleri de kullanabilirler.

İleri web uygulamalarıaslında bir tarayıcı içinde çalışan yerel uygulamalardır. Bu uygulamalar, cihazlar arasında yerel uygulama benzeri bir deneyim sunmak için web tarayıcısı **API**'lerini ve işlevlerini kullanmaktadır. Mobil cihazlara kaydedilebilen ve bir uygulama gibi kullanabilen web sitesinin uzantıları gibi düşünülebilir.

✓ API: Application Programming Interface kısaltması ya da Türkçe karşılığı Uygulama Programlama Arayüzü olarak, API, diğer uygulamaların, hizmetlerin, yazılımların veya işletim sistemlerinin verilerine ve özelliklerine erişen uygulamaların oluşturulmasına izin veren bir dizi işlev ve prosedürdür. Örneğin, çevrimiçi sinema bilet satışı alırken, web sitesine erişerek; film, ad soyad ve kredi kartı gibi bilgileri girerek biletlerin çıktısını almak API kullanımına örnektir.

İşletim sistemleri ve cihaz türlerinden bağımsız olarak hızlı çalışan bu uygulamalar, çok az veri kullanırlar. Kuruluma gerek yoktur. Çevrimdışı olarak çalışabilirler. Bu yaklaşımmda kullanılan ana fikir yerel uygulamanın fonksiyonlarının web ortamına taşınması olarak belirtilebilir. Anlık bildirimler, tit-

reşim gibi mobil donanım erişimi, dokunma hareketleri gibi özelliklerin eklenmesi bu uygulamaların kapsamına girmektedir.

Bu uygulamalar, cihaz türü veya işletim sistemi ne olursa olsun hızlı çalışmaktadır. Arama motorları tarafından kolayca bulunabilirler. Bu uygulamalar sürekli güncel kalırlar ve güvenlidirler. Bir URL kullanarak kolayca paylaşılabilir ve yararlı uygulamalar kaydedilebilir. Uygulama web ortamında ve mobil cihazlarda neredeyse aynı göründüğünden dolayı, kullanıcı için bu uygulamalar birden çok cihazda sorunsuz bir deneyim sağlamakta ve hibrit uygulama hissi yaratmaktadır.

Bu uygulamaların bazı dezavantajları da bulunmaktadır. Bazı üreticilerin varsayılan tarayıcılarında tam destek mevcut değildir. Ayrıca, en son donanım gelişmeleri (parmak izi tarayıcı gibi) bu uygulamalarda kullanılamayabilir.

Hibrit (Hybrid) Uygulamalar

Hibrit uygulamalar, web uygulamaları ile yerel uygulamaların bir karışımı olarak bu iki tür uygulamanın en iyi yönlerini sunmaktadır. Yerel uygulamalar gibi görünen ve hissettiren web uygulamalarıdır. Yerel uygulamalardan farklı olarak hibrit uygulamaların birçok cihaz ve platformda çalışabilmesi en büyük üstünlüklerinden biridir. Bu durum uygulamaların geliştirilmesinde zaman kazandırmaktadır.



Araştırmalarla İlişkilendir

Anadolu Üniversitesi Bilgisayar ve Araştırma Uygulama Merkezi tarafından geliştirilmiş olan “Anadolu Mobil” uygulaması, Anadolu Üniversitesi Web Sitesi (www.anadolu.edu.tr) aracılığıyla, Anadolu Üniversitesi öğün/AÖF öğrencileri ile personele hizmet veren Android ve iOS işletim sistemine sahip mobil cihazlarda çalışan bir hibrit uygulamadır. Özellikle Açık öğretim sistemine kayıtlı öğrencilerine ders kitaplarına erişme, deneme sınavı olma, sınav giriş belgelerini ve ders programlarını görüntüleme gibi çok çeşitli hizmetler sunar. Ayrıca uygulamada kütüphane, yemekhane menüsü, e-gazete gibi birçok araç yer almaktadır. IOS 7.1 ve üzeri işletim sistemine sahip iPhone, iPad ve iPod Touchlar'a, ve Android sürümü 4.0 ve üzeri olan akıllı telefon ve tabletlere indirilebilir.

Kaynak: Özdamar, N. ve Kılınç, H. (2015). Mobil öğrenme uygulamalarına yönelik geliştirme platformlarının karşılaştırılması ve örnek uygulamalar. Açıköğretim Uygulamaları ve Araştırmaları Dergisi, 1(3), 68-90.

Kullanıcılar, uygulama mağazalarından hibrit uygulamaları indirebilmekte ve kullanabilmektedir. Ayrıca, cihazda bulunan bir tarayıcıda da kullanılabilirlerdir. Geliştiriciler, bu uygulamaları geliştirmek için Objective C, HTML5, Ionic, Swift gibi programlama dillerinden yararlanabilir. Hibrit uygulamalar, web teknolojilerinin ve yerel API'lerin bir karışımını kullanmaktadır.

Çoklu platform desteği sunması, hızlı bir şekilde geliştirilmesi, daha az maliyet gerektirtmesi ve değişikliklerin ve güncellemlerin kolay bir şekilde yapılabilmesi hibrit uygulamaların avantajlarıdır. Geliştiricilerin, mobil uygulama geliştirme sürecinde yalnızca tek bir kod tabanını yönetmesi yetерlidir. Bu bağlamda, bakımı gereken çok az kod bulunduğu söylenebilir. Hibrit uygulamalar çok fazla API gerektirmediği için çok fazla entegrasyon sorunu da yaşanmamaktadır. Hızlı yüklenikleri için internet hızı düşük ülkelerde kullanımı yaygındır. Önceliği içerik sunmak olan uygulamalar için idealdir.

Hibrit uygulamalar daha sınırlı yeteneklerinin olması, daha düşük düzeyde kullanıcı deneyimi sunması gibi dezavantajlara da sahip olabilmektedir. Performans olarak yerel uygulamalardan daha yavaş oldukları için, performans odaklı uygulamalar için uygun değildir. Daha fazla kaynak tüketirler ve bazı özellikler bazı cihazlarda kullanılamayabilir. Hataların düzeltilmesi zaman alabilir.

Düzenleme ve Planlama

Düzenleme ve Planlama

Bu üç temel mobil uygulama türü dışında alanlarında çapraz platform uygulamalar bulunmaktadır. Bu alt başlıkta bu türün özelliklerine yer verilmektedir.

Çapraz Platform (Cross-Platform) Uygulamalar

Çapraz platform veya platformlar arası ya da farklı bir ifadeyle çok platformlu mobil uygulamalar son zamanlarda popüler hale gelen bir diğer mobil uygulama türü olarak karşımıza çıkmaktadır. Hibrit uygulamalara benzemekle birlikte bazı farklılıklara sahiptir.

Çapraz platform uygulamalar, iOS, Android ve Windows gibi birden çok işletim sisteminde ve cihaz üzerinde çalışabilir. Bu uygulamalar, birden fazla platform veya işletim sisteminde sorunsuz bir

şekilde çalışacak şekilde çoklu uyarlanabilir yazılım özelliklerine sahiptir. Kullanılan kodlar yeniden kullanılabilirliktedir. "Bir kez yaz, her yerde çalıştır (write-once-run-everywhere)" ilkesiyle çalışmaktadır. Bu uygulamalar ayrıca, sorunsuz işlevsellik, kolay uygulama ve uygun maliyet özellikleri taşımaktadır.

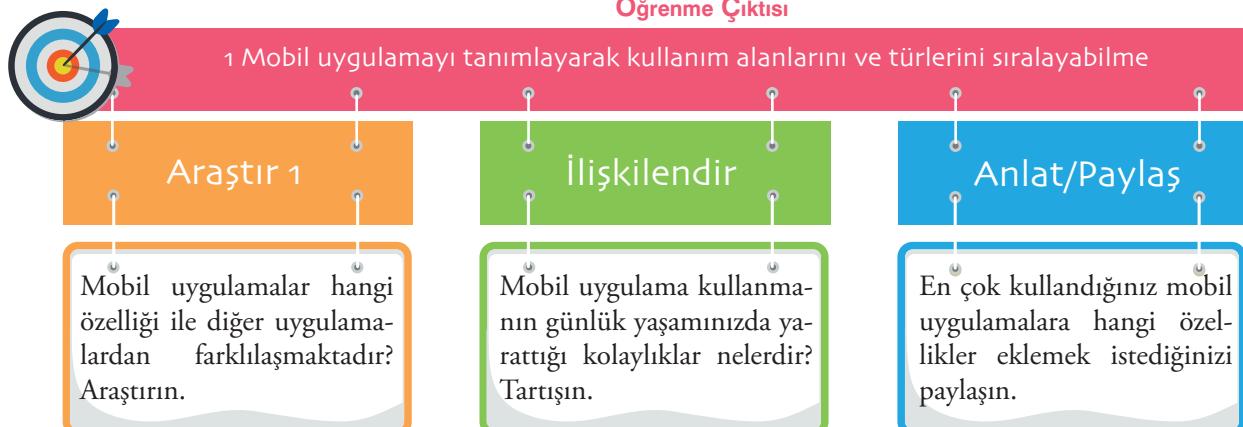
Çapraz platform uygulamalarının dezavantajlarına bakıldığından, işletim sistemlerine erişimi düşüktür. Uygulama performansı yavaş kalabilmektedir. Ayrıca, yerel uygulamalarla etkileşimi sınırlı kalmaktadır.



dikkat

React Native, Qt, Flutter, Xamarin, Apache Cordova ve Ionic çapraz platform uygulamalara örnek olarak verilebilir.

Öğrenme Çıktısı



MOBİL UYGULAMA GELİŞTİRMEK İÇİN KULLANILAN PROGRAMLAMA DILLERİ

Bu kısımda mobil uygulama geliştirmek için kullanılan programlama dillerinden bazlarına yer verilmektedir. Bu bölümde değişimyeni daha pek çok programlama dili olsa da bu başlık altında şu programlama dilleri açıklanmaktadır:

- Java
- Javascript
- HTML5
- Swift
- C#
- C++
- Objective C
- Python
- Kotlin
- Scala

- Dart
- Ruby
- Lua
- PHP

Java

İlk olarak 1995 yılında geliştirilen Java, James Gosling tarafından Sun Microsystems adına tasarlanmıştır. Günümüzde Oracle firması tarafından geliştirme ve güncellemeleri devam etmektedir. Mobil uygulama geliştirme bağlamında en çok kullanılan dillerden biri Java'dır. PYPL endeksine göre, Eylül 2021 itibarıyle en popüler programlama dilleri arasında Phyton'dan sonra ikinci sıradadır. Özellikle Android adlı mobil işletim sisteminin Java ile geliştirilmesi nedeniyle Google Play Store adlı mobil uygulama mağazasındaki uygulamaların büyük bölümü Java programlama diliyle yazılmıştır. Android Yazılım Geliştirme Kiti (Software Development Kit) ile mobil uygulama geliştirmek isteyen geliştiriciler tercih edebilir.

Javascript

Üst düzey yorumlanmış bir programlama dili olan JavaScript, nesne yönelimli ve işlevsel programmayı destekleyen çok paradigmalı bir dildir. 1995 yılında ilk olarak kullanılmaya başlanmıştır. Web sitelerinin %97'sinden fazlası Javascript kullanılarak tasarlanmıştır. En çok kullanılan web tarayıcılarında, Javascript kodlarını kullananların cihazlarında çalıştırmak için özel JavaScript motoru bulunmaktadır. Bu motorlar başlangıçta sadece web tarayıcılarında kullanılmasına rağmen, Node.js ile birlikte bazı sunucuların ve uygulamaların temel bileşeni haline gelmiştir.



dikkat

JavaScript motorları başlangıçta yalnızca web tarayıcılarında kullanılmaktaydı. Ancak şimdi bazı sunucuların ve çeşitli uygulamaların temel bileşenleridir. Bu kullanım için en popüler çalışma zamanı sistemi Node.js'dir.

Öncelik olarak mobil uygulama geliştirmekten ziyade, tarayıcılar tarafından çalıştırılan ve web tabanlı sayfaları geliştirmek için kullanılan bir dildir.

JavaScript, CSS, HTML ve AJAX ile birlikte kul lanıldığı taktirde web tabanlı bir mobil uygulama geliştirmek mümkün olabilmektedir. JavaScript'te uygulama geliştirmeyi kolaylaştıran en önemli faktör, uygulamayı ya da mobil uyumlu web siteyi yalnızca bir kez kodlamanızın yeterli olmasıdır. Böylece Android, iOS ve Windows dahil tüm platformlarda yayınlanabilmesi mümkün olmaktadır.



dikkat

Java ve JavaScript, başta isimlendirme noktasında olmak üzere, sözdizimi (syntax) ve bazı kütüphaneler açısından benzer özellikler taşısa da, bu iki dil tasarım açısından önemli ölçüde farklı özellikler göstermektedir.

HTML5

Web tabanlı mobil uygulamalar geliştirmek için kullanılan HTML dilinin 5. sürümüdür. Tam anlamıyla bir mobil uygulama geliştirme dili olmaktan ziyade, CSS ve Javascript gibi diğer dillerle birlikte kullanıldığından mobil uyumlu web tabanlı siteler ve uygulamalar programlanabilmektedir. Düşük düzeyde donanıma sahip cihazlarda bile sorunsuz bir şekilde çalışabilmektedir. API'lerle birlikte kul lanılabilen HTML5 uygulamaları oldukça duyarlı olduğundan dolayı platformlardaki tüm cihazlarda sorunsuz bir şekilde çalışmaktadır. Mobil ve masaüstü tarayıcılar tarafından oldukça benimsenen HTML5 tabanlı uygulamalar çapraz platform desteği sayesinde başarılı bir şekilde kullanılabilmektedir. En önemli üstünlüğü basit, kolay anlaşılabilir ve temiz bir kodlamadan oluşmasıdır.

Swift

Swift, özellikle iOS uygulama geliştirmeye ilgilenen geliştiriciler için kullanılabilecek Apple ekosistemine yeni giren açık kaynak kodlu, ücretsiz, nesne tabanlı bir programlama dildir. Apple firması, Worldwide Developers Conference adlı etkinlikte Swift'i tanıtarak, daha ilk haftada 60.000'den fazla geliştiricisinin katkıda bulunmasını sağlamıştır.

Objective-C adlı programlama diliyle beraber çalıştırılması ve entegre edilmesi mümkündür. Bu dile oranla biraz daha basit bir söz dizimine sahip olduğu söylenebilir. Hem Python hem de Objective-C'ye oranla daha hızlıdır.

Yeni olmasına rağmen kısa sürede popülerleşen bir dildir. Modern iOS geliştirme süreçlerine ve gereksinimlerine uygun şekilde en baştan yazılmış olan Swift ile yazılmış bir uygulama iOS tüm mobil cihazlarda çalışabilmektedir. iOS için mobil uygulama geliştirme süreçleri gerçekleştirilebilediği gibi açık kaynaklı olduğu için Linux üzerinde de uygulamalar çalışabilmektedir.



dikkat

Swift ile mobil uygulama geliştirmek için Xcode adlı uygulama kullanılabilir. Bu uygulama App Store uygulama mağazasından indirebilir.

C#

C# (C Sharp), özellikle Microsoft firması için uygulama ürünleri oluşturmak için kullanılan C programlama ailesinin bir ürünüdür. Nesne yönelimli, güvenli ve çok paradigmalı bir dildir. Xamarin platformunda yerel mobil uygulama geliştirmek için kullanılabilir. Bu platformda ayrıca çapraz platformlu mobil uygulamalar geliştirmek için de uygundur. Objective-C Apple firmasının uygulamalarını geliştirirken kullanılmıştır, C# Microsoft firmasında oldukça popüler olan bir dildir. Eski adıyla Windows Store, yeni adıyla Microsoft Store uygulama mağazasında pek çok uygulama bu dille oluşturulmuştur.



dikkat

C# Java diline söz dizimi bağlamında benzese de ondan daha temiz ve basit bir kullanımı vardır.

C++

C programlama dili ailesinin bir diğer üyesi C++ dilidir. Düşük seviyeli bellek işleme özgünlüğine sahip genel amaçlı nesne yönelimli bir programlama dilidir. iOS uygulama geliştirilebilmekle birlikte, daha çok Android ve Windows işletim sistemleri için mobil uygulama geliştirme imkânı sunmaktadır. Çapraz platform uygulamalar geliştirmek için uygundur. C++, mobil uygulamaları geliştirmenin çok ötesinde bir potansiyele sahip ol-

duğu için güçlü masaüstü uygulamalarını ve video oyunları geliştirmek için de kullanılabilir.

Objective C

C programlama dilinin bir türevi olan Objective – C, nesne yönelimli genel amaçlı bir programlama dili olarak, Apple firmasının iOS ve MacOS işletim sistemleri için uygulama geliştirmek için oldukça uygundur. Swift geliştirilmeden önce Apple tarafından iOS uygulamalarının geliştirilmesinde temel programlama dili olarak kullanılmıştır.

Swift, son yıllarda popülerliğiyle ve hızlı olmasıyla Objective – C dilinin yerini alma yolunda olsa da, Objective - C, iOS geliştiriciler arasındaki popülerliğini hala korumaktadır. Kullanım kolaylığı sunan ve C++ ile entegre çalışabilen Objective-C iOS için hala önemli bir dil olarak değerlendirilmektedir.

Python

Python genel amaçlı üst düzey bir programlama dili olarak son yılların en gözde dili olarak dikkat çekmektedir. 1991 yılında geliştirilen bu dil, PYPL endeksine göre, Eylül 2021 itibarıyle en popüler programlama dilleri arasında ilk sıradadır. Öğrenmesi kolay, etkileşimli bir dil olarak, YouTube, Instagram, Spotify, Dropbox, Calibre, Reddit, BitTorrent, Quora gibi popüler uygulamalar dahil pek çok uygulama Python dilinde yazılmıştır. Python, yapay zekâ, finansal hizmetler ve veri bilimi gibi gözde teknolojik alanlar dahil olmak üzere çeşitli şekillerde kullanım alanına sahiptir.

Kotlin

Kotlin statik, çapraz platformlu ve birinci sınıf bir programlama dilidir. Bu dil, Java sanal makinesini (JVM) destekleyen ve JavaScript kaynak koduya derlenebilen istatistik temel alınarak yazılmış bir programlama dilidir. JetBrains tarafından ilk olarak 2011 yılında ortaya çıkarılmıştır. Java ile birlikte çalışıldığı gibi Java'ya oranla daha basit bir yazımı vardır. Java kodları Kotlin'e dönüştürülebilmektedir.



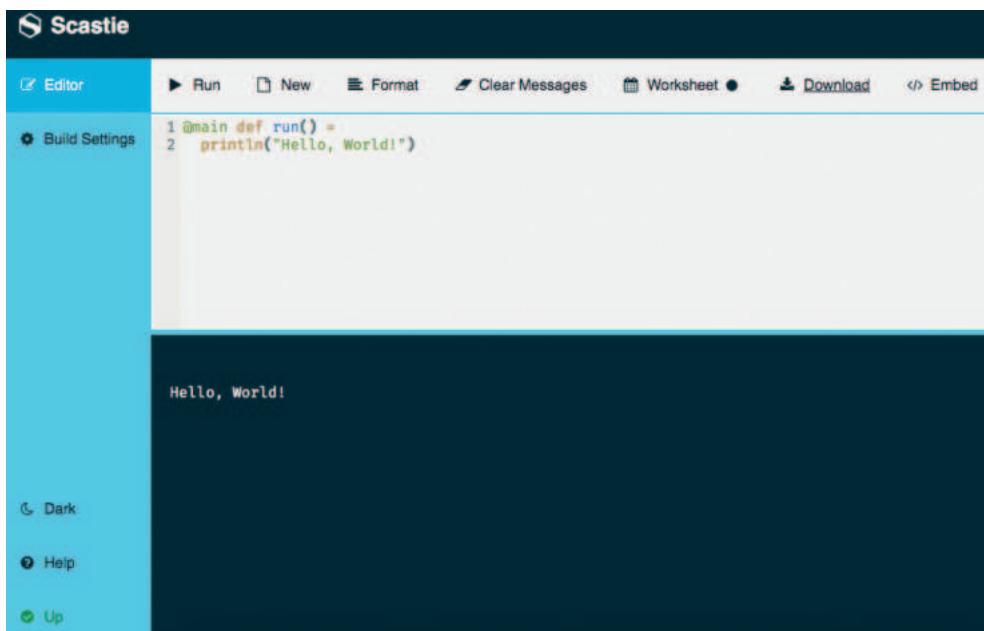
dikkat

2019 yılında Google'ın Android uygulama geliştirme için artık tercih ettiği dil Kotlin olmuştur.

Yeni sayılabilecek bu dil temiz bir kodlamaşa sahiptir. Bu da beraberinde netliği ve kolaylığı getirdiği için kodlamada çok az hata yapılmasını sağlamaktadır. Android Yazılım Geliştirme Kiti ile mobil uygulama geliştirmeyi desteklemektedir. Kotlin ile programlanan uygulamaların arayüzlerinin daha iyi göründüğü söylenebilir. Google tarafından da desteklenen Kotlin, platform bağımsız bir dildir. Geliştiricilerin Kotlin ile karşılaşabileceği dezavantaj, dilin sınırlı öğrenme kaynaklarına sahip olmasıdır.

Scala

Scala, genel amaçlı (güçlü statik tipte) bir programlama dili olup, hem işlevsel programlama hem de nesne yönelimli programlama dili olmayı yüksek seviyeli tek bir dilde birleştirmektedir. Çok paradigmalı Scala'nın statik türleri, karmaşık uygulamalardaki hataları önlemeye yardımcı olmaktadır. Özellikle Java'da yaşanan problemlerin ve eleştirilerin giderilmesi için tasarlanmıştır. Bu dil Java'ya oranla basitlik, kodun daha küçük boyutta olması, performans, statik türde olması, daha fazla sözdizimi (syntax) kullanımı, topluluk ekosisteminin sürekli genişlemesi gibi özellikleyle üstünlük sağlamaktadır.

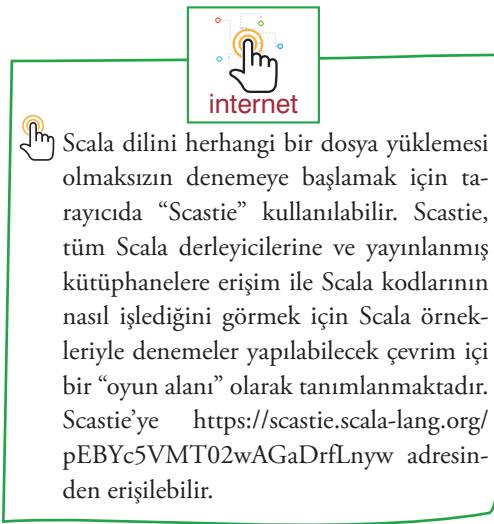


Görsel 1.1 Scastie Platformu

Dart

Dart, herhangi bir platformda hızlı uygulamalar geliştirmek için istemci tarafından optimize edilmiş bir dildir. Dart ücretsiz ve açık kaynak kodlu bir dil olarak Google tarafından desteklenmektedir. Et-

Kotlin gibi Scala'da temeli Java diline dayanan ve Java ile birlikte çalışabilen bir dildir. JVM ve JavaScript ile Scala entegrasyonu, devasa kütüphane ekosistemlerine kolay erişim ile yüksek performanslı sistemler oluşturmaya olanak tanımaktadır.

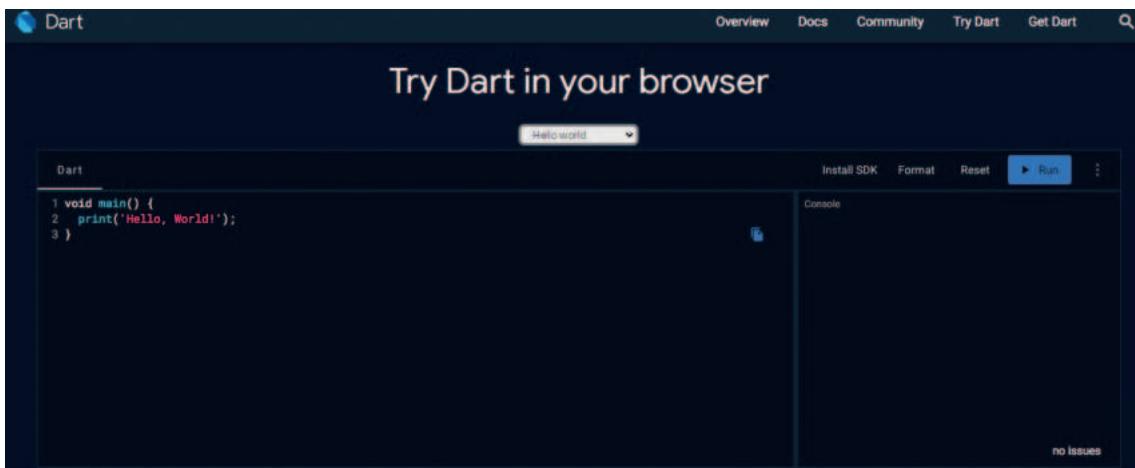


Scala dilini herhangi bir dosya yüklemesi olmaksızın denemeye başlamak için tarayıcıda “Scastie” kullanılabilir. Scastie, tüm Scala derleyicilerine ve yayınlanmış kütüphanelere erişim ile Scala kodlarının nasıl işlediğini görmek için Scala örnekleriyle denemeler yapılabilecek çevrim içi bir “oyun alanı” olarak tanımlanmaktadır. Scastie'ye <https://scastie.scala-lang.org/pEBYc5VMT02wAGaDrfLnyw> adresinden erişilebilir.

kileşimli kullanıcı arayüzü oluşturma ihtiyaçlarını giderebilecek en uygun dillerden biridir. Öğrenmesi kolay ve hızlı bir dildir. Dart dili ayrıca Flutter platformunun temelini oluşturur. Mobil ve masaüstü uygulamalar için ARM ve x64 makine koduna derlenebileceği gibi web tabanlı sayfalar ve uygulamalar için JavaScript'te derlenebilmektedir.



 Dart programlama diliyle denemeler yapmak ve örnek kodlar yazarak dili öğrenmek için <https://dart.dev/#try-dart> adresinden yararlanılabilir.



Görsel 1.2 Tarayıcıda Dart Dilinin Denenmesi

Ruby

Ruby, çok paradigmalı, işlevsel, nesne yönelimli, dinamik bir dil olarak verimlilik ve sadelik üzerine odaklanmıştır. Açık kaynak kodlu, yazımı kolay ve anlaşılır bir sözdizimine sahiptir. Ruby'nin yaratıcısı Yukihiro "Matz" Matsumoto en sevdiği diller olan Perl, Smalltalk, Eiffel, Ada ve Lisp'in en iyi özelliklerini harmanlayarak işlevsel programlama ile imperatif programlamaya dayanan yeni bir dil yaratmayı amaçladığını belirtmektedir. 1995 yılında başladığı geliştirme sürecinde Ruby'i yaşamın içinden, dengeli ve doğal bir dil olarak tasarlamıştır.

İşletim sistemi izin verdiği sürece harici dinamik kütüphaneler yüklenebilmektedir. Ruby işletim sisteminde bağımsız iş parçacığı özelliği sunmaktadır. Yani işletim sisteminin desteklemesine bakılmaksızın, MS-DOS üzerinde bile çalışılsa çoklu iş parçacıkları kullanılabilmektektir.

Ruby, otomatik bellek yönetimine sahip dinamik tip bir sisteme sahiptir. Bu şekilde, web uygulamaları ve mobil uygulamalar için standart bir yapı sunmaktadır. Ancak Ruby'nin dezavantajı, düşük çalışma süresi (run time) hızına sahip olması ve iyi dokümantasyon bulmanın zor olabileceğidir.

 En son kararlı sürüm olan Ruby 3.1.1.'i <https://www.ruby-lang.org/tr/downloads/> adresinden indirebilirsiniz.

Rust

Rust 2010 yılında geliştirilen Mozilla vakfı tarafından desteklenen, güvenliği ön plana alan işlevsel, genel amaçlı ve çok paradigmalı bir sistem programlama dilidir. Belleği verimli kullanan hızlı derlenen bir dildir. Kodlama yapısı C dili ailesine benzemektedir. Ancak diğer programlama dillerine de kolayca entegre olabilir. Rust, kodların derlenmesi sırasında hataların tespit edilmesi ve geliştiricilerin uygun kodu yazmasına olanak tanımaması noktasında avantaj sağlamaktadır. Dezavantaj olarak ise, Windows işletim sistemine yüklemesinin zor olması söylenebilir. Rust dilinde kod yazarak denemeler yapmak için <https://play.rust-lang.org/> adresindeki alandan yararlanılabilir.





Görsel 1.3 Rust Kodları Deneme Platformu



Rust dilini öğrenmek için <https://doc.rust-lang.org/book/> adresindeki dökümantasyondan yararlanılabilir.

Lua

Lua, özellikle gömülü uygulamalar için geliştirilmiş hafif bir çoklu paradigma dilidir. Brezilya'daki Rio de Janeiro şehrinde Papalık Katolik Üniversitesi PUC-Rio'daki bir ekip tarafından tasarlanmıştır.

Dilin göze çarpan özellikleri arasında hız, taşınabilirlik, genişletilebilirlik ve güvenilirlik yer almaktadır. Prosedürel, nesne yönelimli, işlevsel programlamayı ve veriye dayalı programlamayı desteklemektedir. Lua, gömülü sistemler (örneğin Brezilya'da dijital TV için Ginga ara yazılımı) ve oyunlar (örneğin World of Warcraft ve Angry Birds) başta olmak üzere birçok endüstriyel uygulamada (örneğin Adobe Photoshop Lightroom) kullanılmıştır. Roblox,Lua'da kodlanmış birden çok türde kullanıcı tarafından oluşturulan oyunları barındıran bir çevrimiçi oyun platformu ve oyun oluşturma sistemidir. Solar2D, yineleme ve kullanım kolaylığına odaklanan ücretsiz bir Lua tabanlı oyun motorudur. Lua programlama dili hızlı, taşınabilir, gömülebilir, güçlü, basit, küçük boyutlu, ücretsiz ve açık kaynak kodlu bir dil olarak tanımlanmaktadır.



“Lua” Portekizce’de “Ay” anlamına gelmektedir.



⌚ 26 Ocak 2022 itibarıyla <https://www.lua.org/versions.html#5.4> adresinden Lua 5.4.4. indirilebilmektedir.

PHP

PHP (Hypertext Preprocessor) 1995 yılında özellikle dinamik web siteleri için geliştirilmiş, açık kaynak kodlu sunucu taraflı bir betik dilidir. HTML 5 ile birlikte çalışabilmektedir. Günümüzde genel amaçlı olarak kullanılabilirmektedir. Hatta Android ve iOS işletim sistemi için mobil uygulama geliştirmek için de tercih edilmektedir. Hızlı, esnek ve pragmatik olarak tanımlanan PHP, basit bir blog sayfasından dünyyanın en popüler web sitelerine kadar kullanım alanı bulabilmektedir.



⌚ İngilizce, Brezilya Portekizcesi, Çince, Fransızca, Almanca, Japonca, Rusça, İspanyolca dilleriyle birlikte Türkçe'nin de yer aldığı PHP kılavuzuna <https://www.php.net/manual/tr/> adresinden ulaşılabilirmektedir.

PHP sunucu taraflı dinamik web siteleri için en popüler dillerden biridir. W3Techs, Mart 2022 itibarıyla sunucu taraflı tüm web sitelerinin %77,9'unda PHP kullanıldığını raporlamıştır. PHP Sürüm 7 ise, PHP kullanan tüm web sitelerinin %70.8'i tarafından kullanılmaktadır. Facebook, Wikipedia, Zoom, Instagram, Slack ve OpenSea gibi dünyaca ünlü bilinen siteler PHP ile programlanmıştır.



MOBİL UYGULAMA GELİŞTİRMEK İÇİN KULLANILAN PLATFORMLAR

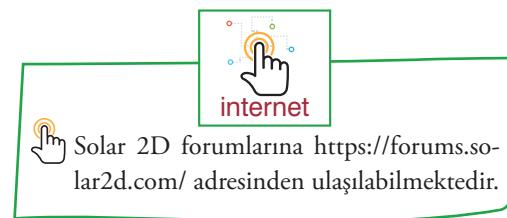
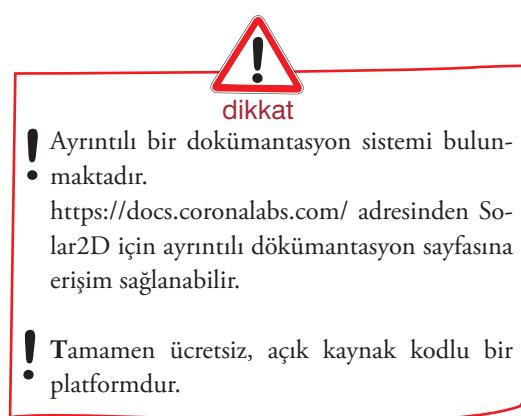
Mobil uygulama geliştirmek için pek çok platform bulunmaktadır. Bu başlık altında şu platformlara yer verilmektedir:

- Solar 2D
- Flutter
- Apache Cordova
- Codename One
- React Native
- Qt
- Appery.io
- Sencha Ext JS
- Xamarin

Solar 2D (Corona SDK)

Solar2D (eski adıyla Corona SDK), yineleme ve kullanım kolaylığına odaklanan Lua tabanlı bir oyun motoru ve yazılım geliştirme kitidir. Solar2D, artık ticari olarak desteklenmeyen ve yaygın olarak kullanılan Corona SDK oyun motorundan türetilmiş tamamen açık kaynaklı bir projedir. 2D oyunlar ve mobil uygulamalar için en kolay geliştirme aracıdır. Solar2D ile geliştirme sürecinde kod güncellendikçe en son sonuçlar arasında güncelleme Simülatöründe anında görülmektedir. Bu da geliştirme sürecini hızlandırmaktadır.

Geliştirme süreci, Corona Labs Inc.'de eski teknik baş mühendis olan Vlad Shcherban tarafından yönetilmektedir. Kodları MIT lisansı altında korunmaktadır. Çapraz platform desteğiyle tek bir kod tabanıyla mobil (iOS, Android, Windows), masaüstü (Windows, MacOS) ve bağlı TV cihazları (Android TV, Apple TV) için uygulama geliştirmek mümkündür.



kündür. Yerel, web tabanlı ve hibrit uygulamalar geliştirilebilir. Uygulama geliştirmek hızlı, esnek ve kolaydır. 1000'den fazla işlevi kapsayan ve her öğeyi çok hızlı bir şekilde çalıştırılmaya olanak tanıyan tutarlı bir Uygulama Programlama Arayüzü (Application Programming Interface/API) bulunmaktadır.

Flutter

Flutter 2017 yılında Google tarafından piyasaya sürülen açık kaynak kodlu bir çerçeveye (framework) olarak mobil başta olmak üzere, web, masaüstü ve gömülü uygulamalar geliştirmek için kullanılabilen bir platformdur. Özellikle hibrit uygulamalar geliştirmek için oldukça uygundur. Tek bir kod tabanı kullanan Flutter, yerel ve çok platformlu uygulamalar geliştirmek için uygundur. Herhangi bir hata durumunda hatanın tespiti ve çözümü oldukça nettir. JavaScript'in yanı sıra ARM veya Intel makine kodunu derlemektedir. Google tarafından desteklendiği için pek çok geliştirici ve dünyaca ünlü marka tarafından güvenilir izlenimi bulunan Flutter, küresel geliştiricilerden oluşan bir topluluk tarafından sürekli desteklenmektedir.

Flutter, herhangi bir platformda hızlı uygulamalar için optimize edilmiş bir programlama dili olan Dart tarafından desteklenmektedir. Android ve iOS'ta uygulama geliştirmek için Dart dilini temel almaktadır. Ayrıca iOS ve Android'e özgü programlama dillerini bilmeksizin tek bir platformda eş zamanlı olarak uygulama geliştirebilmektedir. Dart diğer dillere göre oldukça kolay bir dil olarak betimlenmektedir.



dikkat

Flutter'daki widget'ları kullanarak rahatlıkla uygulama geliştirmek mümkündür. <https://docs.flutter.dev/development/ui/widgets-intro> adresinden yararlanılabilir.



ipuçları



internet

Flutter, Windows, MacOS, Linux ve Chrome OS işletim sistemlerine indirilip kurulabilmektedir. <https://docs.flutter.dev/get-started/install> adresinden Flutter'in güncel sürümlerini uygun işletim sistemini seçerek indirmek mümkündür.

Apache Cordova (Phone Gap)

Apache Cordova (eski adıyla PhoneGap), açık kaynaklı, ücretsiz bir mobil geliştirme çerçevesi olarak 2009 yılında piyasaya sunulmuştur. Platformlar arası geliştirme için standart web teknolojileri olan HTML5, CSS3 ve JavaScript'i kullanmaya olanak tanır. Mobil hibrit web uygulamaları geliştirmek için uygundur. Çevrimdışı olarak çalışan uygulamaları geliştirmek için de uygun olan bu platformda yerel cihaz API'lerine erişim sağlanmasıdır. Kodlar platformlar arasında yeniden kullanılabilmektedir.



internet



Apache Cordova dökümantasyon sayfasına erişerek geliştirme süreçlerine başlamak için <https://cordova.apache.org/docs/en/latest/> adresinden yararlanabilirsiniz.

Codename One

Codename One açık kaynak kodlu ve ticari / ticari olmayan kullanım için telif ücreti veya kısıtlama olmaksızın kullanılabilen bir çapraz platform çerçevesidir. Codename One'in ücretsiz ve ücretli sürümleri bulunmaktadır. Codename One, Java geliştiricilerinin mobil uygulamalarını Java veya Kotlin kullanarak yazmasına olanak tanımaktadır. Sun Microsystems bünyesinde 2007 yılında başlatılan bir açık kaynak projesine dayalı olarak eski Sun/Oracle mobil geliştiricileri tarafından ortaya çıkarılmıştır. Codename One'in temel vizyonu, mobil cihazlar çağında Java'nın WORA (Write Once Run Anywhere/ Bir Kez Kodla, Her Yerde Çalıştır) mottosunu gerçekleştirmektir. Codename One, Java geliştiricilerinin tek bir kod tabanıyla yerel mobil işletim sistemi uygulamaları oluşturmasına olanak tanımaktadır.



dikkat

Codename One, tüm büyük Java geliştirici ortamları (IDE'ler) NetBeans, Eclipse, IntelliJ/ IDEA veya VSCode ile birlikte çalışmaktadır.

✓ **IDE:** Integrated Development Environment (IDE) veya Entegre Geliştirme Ortamı, bir yazılım geliştirme projesi için gereken tüm araçları tek bir platformda birleştiren bir yazılım uygulamasıdır. Daha temel düzeyde, IDE'ler, kullanıcıların kod yazması, metin gruplarını düzenlemesi ve programlama şartlarını otomatikleştirmesi için arabirimler sağlamaktadır. Basit bir kod düzenleyici yerine IDE'ler, birden çok programlama komutunun işlevsellliğini tek bir işlemde birleştirmektedir. Visual Studio, Eclipse, Android Studio örnek olarak verilebilir.



👉 <https://codenameone.teachable.com/> adresinde yer alan Codename One Academy eğitimleriyle Java kullanarak iOS, Android ve Windows için yerel mobil uygulamalar oluşturma hakkında bilgi edinilebilir. Ayrıca, <https://www.codenameone.com/videos.html> adresindeki videolardan da yararlanılabilir.

React Native

React Native Facebook (Meta) tarafından desteklenen ve Android ve iOS için yerel uygulamalar geliştirilebilen açık kaynaklı bir uygulama çerçevesidir. Facebook, React Native'i 2015'te piyasaya sundığından beri desteklemeyi sürdürmektedir. Javascript kütüphaneleri etkin şekilde entegre edilebilmektedir. Tek bir kod tabanıyla platformlar arası uygulamalar geliştirilebilmektedir. Yapılan işlemler, yazılan kodlar eş zamanlı ve hız bir şekilde görüntülenebilmektedir.

```

import React from 'react';
import {Text, View} from 'react-native';
import {Header} from './Header';
import {heading} from './Typography';

const WelcomeScreen = () => (
  <View>
    <Header title="Welcome to React Native"/>
    <Text style={heading}>Step One</Text>
    <Text>
      Edit App.js to change this screen and turn it
      into your app.
    </Text>
    <Text style={heading}>See Your Changes</Text>
    <Text>
      Press Cmd + R inside the simulator to reload
      your app's code.
    </Text>
  </View>
)

```

Görsel 1.4 React Native Ekran Görüntüsü

React Native, Callstack, Expo, Infinite Red, Microsoft ve Software Mansion dahil olmak üzere dünyanın dört bir yanındaki kişilerden ve şirketlerden gelen katkılarla desteklenmektedir. Popüler uygulamalar dan Facebook'un bir kısmı ve Instagram, Tesla ve Pinterest ve Skype uygulamaları React Native ile geliştirilmiştir. Oculus'ta bulunan sanal gerçeklik uygulamalarının bir kısmı da bu platformda tasarılmaktadır. Diğer kullanımları görmek için <https://reactnative.dev/showcase> adresinden yararlanılabilmektedir.



👉 <https://reactnative.dev/docs/accessibilityinfo> adresinden React Native API'lerine ulaşmak mümkündür.

Qt

Qt, bir uygulama çerçevesi ve bir kod tabanıyla herhangi bir platformda çalışabilen uygulamalar geliştirmek için kullanılan widget aracıdır. Günümüzde The Qt Company adlı şirket tarafından geliştirilme süreci devam etmektedir. Yüksek performanslı gömülü sistemler, masaüstü ve mobil uygulamalar geliştirmek mümkündür.

Qt ile mobil uygulama geliştirme süreci kullanımı kolay olduğundan dolayı eğlencelidir. Dökümantasyon oldukça yeterlidir ve tam olarak aranılan konuları içermektedir. Arayüzü etkileşimli görünen, farklı öğeleri birkaç dakika içinde uygulayan ve kullanıcı arayüzü öğelerini 3B nesnelerle yüksek hassasiyetle senkronize eden esnek ve duyarlı bir kullanıcı arabirimini hızlı bir şekilde tasarlamaya olanak sağlamaktadır. Ayrıca, hatalı SSL sunucuları için çözümler sunmaktadır.



👉 Qt 6 QML kitabı, <https://www.qt.io/product/qt6/qml-book> adresinden erişilebilir.

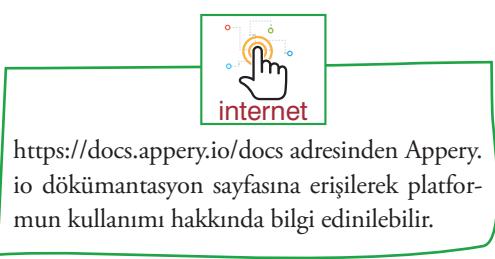
Appery.io

Appery.io **düşük kodlu** ve çok amaçlı bir platform sunan bulut tabanlı bir uygulama geliştiricisidir.

Düşük Kod: Düşük Kod (Low-Code), sürece hızlandırmak için tüm geliştirme sürecini optimize eden yazılım geliştirmeye yönelik görsel bir yaklaşımındır. Bu yaklaşımında grafik tabanlı bir kullanıcı arayüzü kullanılarak kod yazımı en aza (hatta çoğu zaman sıfıra) indirgenir ve kullanım kolay bir ortamda anlaşıılır nesneler ve sürükle bırak gibi özelliklerle uygulama geliştirmek mümkün hale gelir. Gartner'a göre, 2024'e kadar, düşük kodlu uygulama geliştirmenin, uygulama geliştirmeye etkinliğinin %65'inden fazlasından sorumlu olacağı öngörlmektedir.

Exadel tarafından ilk olarak 2012 yılında piyasaya sürülmüştür. Bu düşük kodlu uygulama geliştirme platformu, tamamen yeni başlayanlar için bile kolay bir arayüze sahiptir. Öte yandan, deneyimli uygulama geliştiricileri, üretkenliklerini artırmak için görsel araçlarla zenginleştirilmiş Ionic ve Angular gibi uygulama çerçeveleriyle çalışmaya devam edebilmektedir. Bulut veritabanı, sunucu tarafı komut dosyası oluşturma, kimlik doğrulama, anında iletme bildirimleri gibi entegre arka uç (back-end) hizmetlerini kullanmak mümkündür.

Bulut tabanlı uygulama oluşturucusu oldukça kolay bir arayüze sahiptir. Sürükle bırak işlevlerinin hızından ve Javascript'in verimliliğinden yararlanmaktadır. Hibrit uygulamalar dışında en çok web tabanlı uygulamalar ve ileri (progressive) web uygulamaları (PWA) geliştirmek için de kullanılabilir. Appery.io ile oluşturulmuş platformlar arası uygulamalar sorunsuz çalışmaktır ve tüm popüler cihazlar da ve işletim sistemlerinde aynı şekilde görünmektedir. Mobil uygulamalar tek bir kod tabanıyla App Store ve Google Play mağazalarında yayınlanabildiği gibi, web uygulamaları ve ileri web uygulamaları olarak da sunulabilmektedir. 500.000+ kayıtlı kullanıcı, 10.000+ yayınlanan uygulaması vardır. Appery.io uygulamalarının tarayıcılar ve çeşitli cihazlar arasında aynı anda paylaşılması mümkündür. Böylece, ekip çalışmalarında gerçek zamanlı olarak işbirlikçi yöntemler kullanılabilmektedir. Rol tabanlı ekip yönetimi ve izinler gibi işlevleri kullanarak kurumsal verilere kimlerin erişeceğini yönetimi gerçekleştirilebilmektedir. Ayrıca, sürüm oluşturma ve yedeklemelerle ilerleme sürekli kayıt altına alınmaktadır.



Sencha Ext JS (Sencha Touch)

Sencha Ext JS, herhangi bir modern cihaz için veri yoğun, platformlar arası web ve mobil uygulamalar oluşturmaya yönelik en kapsamlı JavaScript çerçevelerinden biridir.

Platformlar arası web ve mobil uygulamalar oluşturmak için ihtiyaç duyulan bütün bileşenleri sağlayan Sencha Ext JS, Sencha Touch ile birleştirildiği için Sencha Touch artık desteklenmemektedir.

Sencha Ext JS ile JavaScript kullanarak veri yoğun HTML5 uygulamaları oluşturulabilmektedir. Sencha Ext JS bir Javascript kütüphanesi olarak belirtilebilir. 140'tan fazla önceden entegre edilmiş ve test edilmiş yüksek performanslı kullanıcı arayüzü bileşeni içermektedir. Yerel uygulama hissi yaratan web tabanlı mobil uygulamalar geliştirmek olanaklıdır. Fortune 100 şirketlerinin %60'ı Sencha kulandığını belirtmektedir. 30 gün ücretsiz deneme sürümü olan Topluluk (Community) Sürümü dışında, Enterprise ve Pro adlı ücretli sürümleri de bulunmaktadır.



dikkat

Hala Sencha Touch'ı kullanmak isteyen geliştiriciler için son sürüm Touch 2.4.2 ücretsiz olarak indirilebilmektedir.



dikkat

NET ücretsizdir ve Xamarin platformu da buna dahildir. Ticari kullanım da dahil olmak üzere hiçbir ücret veya lisans maliyeti bulunmamaktadır.

Xamarin

Xamarin açık kaynaklı ücretsiz çapraz platform desteği sunan bir uygulama platformudur. .NET ve C# programlama dilleriyle Android ve iOS uygulamaları oluşturmak mümkündür.

Xamarin ile Objective-C veya Java'da yapılabilecek uygulamalar C#'ta gerçekleştirilmektedir.



3 En çok kullanılan mobil uygulama geliştirme platformlarını açıklayabilme

Araştır 3

Mobil bir uygulama geliştirilebilecek platformları araştırın.

Öğrenme Çıktısı

İlişkilendir

Tek bir kod tabanı kullanan platformların diğerlerine olan üstünlüğünü tartışın.

Anlat/Paylaş

Mobil uygulama geliştirme platformlarının dökümantasyonlarını inceleyerek, güçlü ve zayıf olanları paylaşın.



internet

Xamarin kurulum ve yüklemesinden, Android, iOS ve Mac uygulamalarının geliştirilmesine kadar tüm dokümantasyona https://docs.microsoft.com/tr-tr/xamarin/?WT.mc_id=dotnet-35129-website adresinden ulaşılabilir.

MOBİL UYGULAMA MAĞAZALARI

Mobil uygulama mağazaları (app store veya app marketplace) mobil uygulamaların sunulduğu bir dijital dağıtım platformudur. Bazı kaynaklarda mobil uygulama marketleri şeklinde belirtildiği de görülmektedir. Masaüstü uygulamalardan ziyade mobil cihazlara (tablet, akıllı telefon gibi) yönelik uygulamalar içermektedir. Her uygulama mağazasının kendi içinde kategorileri bulunmaktadır. Böylece kullanıcı istediği uygulamaya gerek arama bölümünden gerekse kategoriler yardımıyla ulaşabilmektedir. Uygulamalar ilgili işletim sisteminde ve dolayısıyla ilgili uygulama mağazasında çalışacak şekilde geliştirilmektedir.

Bu bölüm kapsamında,

- Google Play Store,
- App Store,
- Huawei AppGallery,
- Amazon Appstore,
- Microsoft Store

mobil uygulama mağazaları işlenecektir. Bunlar dışında

- Opera Mobil Store,
- Samsung Galaxy Store,
- BlackBerry World,
- OpenStore for Ubuntu Touch

gibi mobil uygulama mağazaları da kullanılmaya devam etmektedir. Ayrıca,

- Firefox Marketplace,
- Windows Phone Store,
- Nokia Store,
- HP App Catalog

gibi yayın hayatı sona eren ve kullanılmayan uygulama mağazaları da bulunmaktadır.

Google Play Store

Google Play Store (eski adıyla Android Market) olarak da adlandırılan, 2012 yılında faaliyetlerine başlayan Google tarafından yürütülen bir mobil uygulama mağazasıdır. Ekim 2008'de Android Market adıyla piyasaya sürüldükten sonra Google Play Store (Google Play Mağazası) ismini almıştır. Bir dijital dağıtım platformu olan Play Store, Android mobil işletim sistemi ve Chrome OS uygulamalarının sunulduğu resmi uygulama mağazasıdır. Geliştiriciler Android Yazılım Geliştirme Kiti

(SDK) ile geliştirdikleri uygulamaları bu platforma sunabilir ve diğer kullanıcılar da bu uygulamaları güvenli bir şekilde indirip kullanabilir. Uygulamalar ücretli veya ücretsiz sunulabilir.



dikkat

Android, Global Stats Şubat 2022 verilerine göre, dünya çapında mobil işletim sistemi akıllı telefon pazar payı noktasında, 70.97% ile birinci sırada yer almaktadır. Türkiye'de ise 82.31% orANIYLA ANDROID DÜNYA ORTALAMASINDAN DA YUKARIDA YER ALMAKTADIR. TABLETLERDE İSE BU ORAN 46.13%'TUR VE IOS'TAN SONRA ANDROID İKİNCİ SİRADA YER ALMAKTADIR.

Nisan 2021 itibarıyla, dünya çapında en çok hasılat yapan Android uygulamaları sıralamasında, Coin Master, Garena Free Fire ve PUBG mobile ile sıralamada liderlige mobil oyunlar hakimdir. Oyun uygulamalarının 2024'te Google Play uygulama gelirlerinin yüzde 70'inden fazlasını oluşturacağı tahmin edilmektedir. Oyunların çoğu ücretsiz olarak indirilebiliyor olsa da, uygulama içi satın alma veya uygulama içi reklamcılık ile ücret alınması mümkün olabilmektedir.



dikkat

PEGI etiketleri oyunlar için kategorilendirme yapmaktadır. Örneğin PEGI 3 tüm yaş grupları için uygun oyunların etiketlendiği kategoridir. Bunun dışında PEGI 7, 12, 16, 18 ve OK etiketleri de bulunmaktadır

Mart 2022 istatistiklerine göre, Google Play Store'da 10 milyardan fazla kez indirilerek en çok indirilen 5 uygulama şu şekildedir:

- Google Play Hizmetleri
- YouTube
- Google Haritalar
- Google Chrome
- Gmail

Statista verilerine göre, Google Play Store'daki mevcut uygulama sayısı, Temmuz 2013'te ilk defa 1 milyon uygulamayı aşından sonra Şubat 2022 itibarıyle en son 2,6 milyon uygulamayı geçmiştir.

Bu rakamlarla en yüksek uygulama sayısının Google Play Store'da olduğu görülebilmektedir.

Google Play Store'da "Size Özel" bölümyle kullanıcının tercihlerine özgü önerilen uygulamalar görüntülenebilmektedir.



"Üst Sıralar" bölümünde ücretsiz ve ücretli en popüler ve en çok kazanan uygulamalar yer almaktadır. Çocuklar kısmında çocuklara özel 5 yaşa kadar, 6-8 yaş ve 9-12 yaş kategorilerine uygun oyun seçenekleri sunulmaktadır. Kategoriler kısmında alışverişten artırılmış gerçekliğe, eğitimden haberleşmeye kadar pek çok kategoride uygulama listelenmektedir. Ayrıca, editörün seçimi kısmında editörlerin özenle seçtiği en iyi uygulamalarдан oluşan ve sürekli güncellenen bir uygulama listesi de bulunmaktadır. Uygulamalar sekmesi dışında oyunlar, filmler ve kitaplar sekmeleri de bulunmaktadır.

App Store

App Store, Apple Inc. tarafından iOS ve iPadOS işletim sistemlerindeki mobil uygulamalar için geliştirilen bir uygulama mağazası ve dijital dağıtım platformudur. Mağazada, geliştiricilerin iOS Yazılım Geliştirme Kitinde geliştirdikleri onaylı uygulamalar kullanıcılar tarafından güvenlik ön plana alınarak indirilip kullanılabilmektedir.

App Store, 10 Temmuz 2008'de açıldığından yalnızca 500 uygulama mevcuttu. Mart 2022 ve rilerine göre Dünya çapında 1,8 milyon uygulama, 40'tan fazla dilde 175 farklı tasarımla sunulmaktadır. Tüm dünyada 150'den fazla uzman editörün kontrolünden geçen uygulamalar her aşamada güvenlik testlerinden geçerek kullanıcıya ulaşmaktadır. Uygulamalar kötü amaçlı yazılım testlerinden geçerek otomatik olarak taramaktadır. Apple yönergelerine uymayan, eski, sakıncalı, zararlı, yasa dışı içeriğe sahip ve güvenlik riskleri barındıran uygulamaların temizlenmesiyle 2017'de 2.2 milyon civarında olan uygulama sayısı azalmıştır. Yalnızca 2021'de 215.000'den fazla uygulama başvurusu gizlilik kurallarını ihlal ettiği için reddedilmiştir. Her hafta dünya çapında 500'den fazla uzman 100.000'in üzerinde uygulamayı incelemektedir. 2020'de, spam olarak kabul edilen 80 milyondan fazla müsteri yorumu silinmiştir.

Apple'a göre 2019'da en çok indirilen ilk on ücretsiz iPhone uygulaması şu şekildedir:

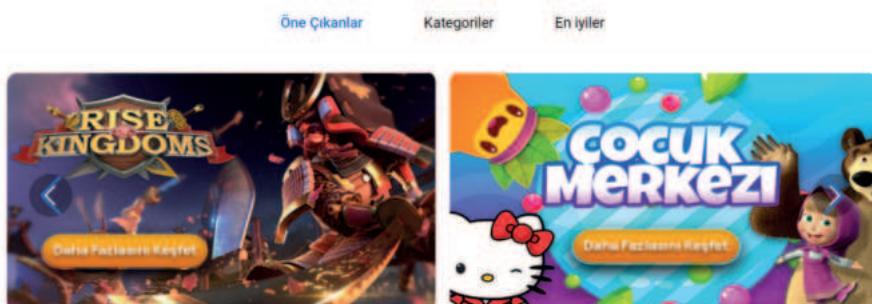
- | | |
|---------------------|--------------|
| 1. YouTube | 2. Instagram |
| 3. Snapchat | 4. TikTok |
| 5. Messenger | 6. Gmail |
| 7. Netflix | 8. Facebook |
| 9. Google Haritalar | 10. Amazon |

2020 itibarıyle, en popüler App Store kategorisi, uygulamaların yüzde 22'sinden fazmasını içeren oyun kategorisidir. İkinci en popüler uygulama kategorisi iş, ardından eğitim ve yaşam tarzı uygulamaları olmuştur. Mobil oyun uygulamaları genellikle gelirlerinin çoğunu reklamlar ve uygulama içi satın almalar yoluyla elde etmekte ve oyunların yalnızca üçte birinden biraz fazla ücretli indirmelerden oluşmaktadır. App Store'daki popüler oyun uygulamaları arasında Clash Royale, Candy Crush Saga ve en çok gelir getiren oyunlardan biri olan Clash of Clans yer almaktadır.

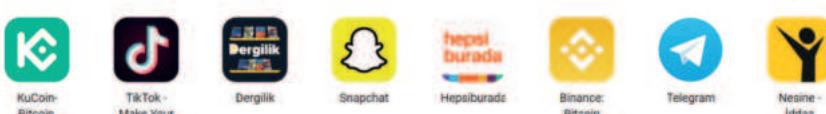
Huawei AppGallery

Huawei AppGallery (kısaltılmış haliyle AppGallery), Huawei'nin HarmonyOS adlı mobil işletim sistemi ve Android için, Huawei Technologies şirketi tarafından geliştirilen bir uygulama mağazasıdır. İlk olarak 2011 yılında Çin'de, daha sonra 2018 yılında tüm dünya genelinde kullanılmaya başlanmıştır. Türkçe dahil 80 dilde desteklenmektedir. Yalnızca Huawei değil Honor cihazlarında da kullanılmaktadır. AppGallery, 700 milyon Huawei cihazında 420 milyon aktif kullanıcı tarafından kullanılmaktadır. Huawei'nin yeni cihazları, Mayıs 2019'da Çin-ABD ticaret savaşı nedeniyle Google Mobil Hizmetlerine ve diğer Google uygulamalarına erişimi kaybettiğinde, şirket bazı yeni telefonlarında Google hizmetlerini kullanamamıştır. AppGallery ile Google Play Store ve üzerlerinde yüklü diğer tüm Google uygulamaları olmaksızın kendi tescilli Huawei Mobil Hizmetlerini kullanarak uygulamaları piyasaya sürmeye başlamıştır.

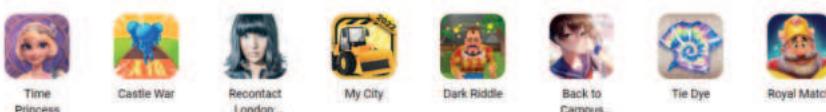
AppGallery "Öne Çıkanlar" sekmesinde yeni uygulama ve oyunlar, en çok oylanınan uygulamalar, en eğlenceli oyunlar, en popülerler, yenilikçi uygulamalar, beğenecinizinizi düşündüklerimiz, olmazsa olmazlar, editörün tavsiyeleri, bu ayın en iyisi gibi bölümler sunmaktadır. Uygulama kategorilerinde haber ve okuma, iş dünyası, otomobiller, sosyal, eğitim, finans, yiyecek ve içecek, çocuklar, yaşam tarzı, eğlence, navigation ve ulaşım, kişiselleştirilmiş temalar, fotoğraf ve video, alışveriş, spor ve sağlık, araçlar, seyahat yer almaktadır. Oyun kategorilerinde spor oyunları, aksiyon, kart ve kutu, bulmaca ve diğer, rol yapma, strateji listelenmektedir. En iyiler sekmesinde ise derecelendirme yapılan uygulama ve oyunların bir listesi bulunmaktadır.



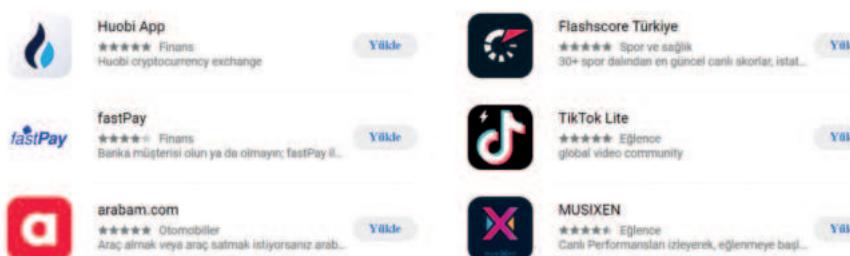
Bayıldığımız yeni uygulamalar



Bayıldığımız yeni oyunlar



En çok oylanınan uygulamalar



Görsel 1.6 Huawei AppGallery Ekran Görüntüsü

Amazon Appstore

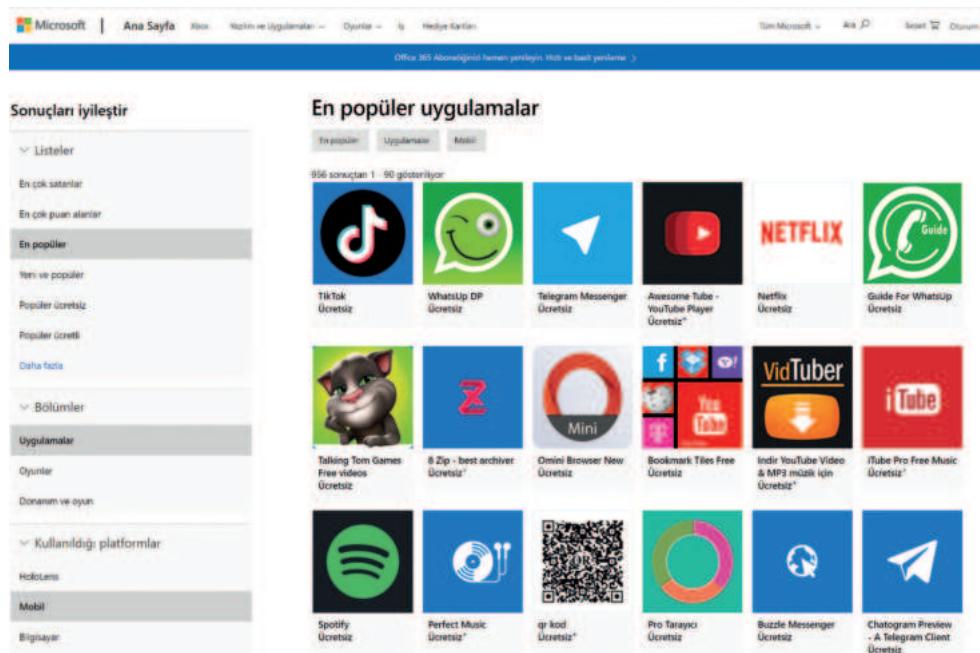
Amazon AppStore, e-ticaret devi Amazon.com tarafından işletilen Android işletim sistemi için tasarlanmış bir uygulama mağazasıdır. 2011 yılında piyasaya sürülmüşdür. Amazon Appstore Android cihazları desteklemektedir. Google Play Store'a en uygun alternatiflerden biridir. Ayrıca, Amazon'un Fire OS adlı işletim sistemi için paket uygulama mağazasıdır. Amazon Appstore, Fire Tabletler için diğer Android cihazlarda kullanılan Google Play uygulama mağazası yerine resmi uygulamalar sağlamaktadır. 460 bini aşkın uygulama içermektedir.



Amazon AppStore, şu anda yalnızca Android mobil cihazları desteklediği için masaüstü bilgisayarlarda açıldığında çalışmamayacaktır.

Microsoft Store

Microsoft Store (eski adıyla Windows Store), Microsoft'a ait bir dijital dağıtım platformu ve uygulama mağazası olarak 2012 yılında piyasaya sunulmuştur. Microsoft'un Windows işletim sistemi için uygulamalar yer almaktadır. Uygulamada Listeler bölümünde en çok satanlar, en çok puan alanlar, yeni ve popülerler, popüler ücretli/ücretsiz uygulamalar yer almaktadır. Bölümler kısmı uygulamalar, oyunlar, donanım ve oyun olmak üzere 3 kısma ayrılmaktadır. Kullanıldığı platformlar bölümünde Hololens, mobil, bilgisayar, Xbox gibi seçenekler yer almaktadır. Sadece mobil uygulamalarla yetinilmediği, ayrıca, Hololens karma gerçeklik gözlüğü uygulamaları ve Xbox oyun konsolu uygulamalarının da bu mağazada yer aldığı görülmektedir. 2021 verilerine göre mağazada 669 bin uygulama bulunmaktadır. En fazla sayıda uygulamayı içeren kategoriler "Kitaplar ve Başvurular", "Eğitim", "Eğlence" ve "Oyunlar"dır. Google Play Store ve Apple App Store'da olduğu gibi Microsoft bu uygulamaları güvenlik testlerinden geçirerek onaylamaktadır.



Görsel 1.7 Microsoft Store Ekran Görüntüsü

**Öğrenme Çıktısı**

4 Mobil uygulama mağazalarını sıralayarak özelliklerini açıklayabilme

Araştır 4

Aktif olarak kullanılan mobil uygulama mağazalarını araştırınız.

İlişkilendir

Google Play Store'daki uygulama sayısının diğer uygulama mağazalarından daha fazla olmasının nedenlerini yorumlayınız.

Anlat/Paylaş

Geçmişte kullandığınız ancak şu an aktif olarak kullanılmayan uygulama mağazası deneyiminiz varsa anlatınız. Uygulama mağazasının neden kapatılmış olabileceğini yorumlayınız.

1

Mobil uygulamayı tanımlayarak kullanım alanlarını ve türlerini sıralayabilme

Mobil Uygulama Tanımı,
Kullanım Alanları ve Türleri

Mobil uygulama, akıllı telefon, tablet, akıllı saat veya akıllı gözlük gibi bir taşınabilir cihazda çalışacak şekilde tasarlanan yazılım türüdür. Masaüstü bilgisayarlardaki uygulamaların tersine, mobil uygulamalar mobil aygıtlara uyumlu olacak biçimde geliştirilmektedir. Mobil uygulamalar eğitimden seyahate çeşitli şekillerde sınıflandırılmaktadır. Tür olarak ise yerel, web ve hibrit uygulamalar temelinde değerlendirilirse de çapraz platform uygulamalar da son zamanlarda ayrı bir tür gibi düşünülmektedir. Yerel uygulamalar, çoğu kullanıcının uygulama sözcüğünü işittiğinde aklına gelen ilk uygulama türü olarak genellikle belirli bir işletim sistemi üzerinde çalışacak şekilde özel olarak geliştirilir. Hedef kitleye ulaşmak için farklı işletim sistemleri için ayrı uygulamalar tasarlamak gerekmektedir. Web tabanlı uygulamalar yerel uygulamalara benzer özellikler taşımasına rağmen, mobil cihazdaki web tarayıcıları aracılığıyla ulaşılan uygulamalardır. Yerel uygulamalardan farklı olarak cihazın donanımına erişememekte ve HTML5, CSS, Javascript gibi popüler programlama dilleriyle geliştirilmektedir. İleri web uygulamaları, normal web uygulamalarına benzer web teknolojilerini kullanmasına rağmen, çok daha üstündür. Gelişmiş bir kullanıcı deneyimi sunmalarına yardımcı olan standart web uygulamalarından daha fazla işlevselligi sahiptir. Hibrit uygulamalar, web uygulamaları ile yerel uygulamaların bir karışımı olarak bu iki tür uygulamanın en iyi yönlerini sunmaktadır. Yerel uygulamalardan farklı olarak hibrit uygulamaların birçok cihaz ve platformda çalışabilmesi en büyük üstünlüklerinden biridir. Çapraz platform uygulamalar, birden fazla platform veya işletim sisteminde sorunsuz bir şekilde çalışacak şekilde çoklu uygulanabilir yazılım özelliklerine sahiptir.

2

Mobil uygulama geliştirmek için kullanılan programlama dillerinin özelliklerini ayırt edebilme

Mobil Uygulama Geliştirmek
İçin Kullanılan Programlama
Dilleri

Mobil uygulama geliştirme bağlamında en çok kullanılan dillerden biri olan Java özellikle Android işletim sisteminin Java ile geliştirilmesi nedeniyle Google Play Store mağazasındaki uygulamaların büyük bölümü Java programlama diliyle yazılmıştır. Üst düzey yorumlanmış bir programlama dili olan JavaScript, nesne yönelimli ve işlevsel programlamayı destekleyen çok paradigmalı bir dildir. Swift, özellikle iOS uygulama geliştirmeye ilgilenen geliştiriciler için kullanılabilecek Apple ekosistemine yeni giren açık kaynak kodlu, ücretsiz, nesne tabanlı bir programlama dilidir. C#, özellikle Microsoft firması için uygulama produklerini oluşturmak için kullanılır. C programlama dilinin bir alt kategorisi olan Objective – C, nesne yönelimli genel amaçlı bir programlama dilidir. Python, yapay zekâ, finansal hizmetler ve veri bilimi gibi gözde teknolojik alanlar dahil olmak üzere çeşitli şekillerde kullanım alanına sahip olduğundan dolayı geliştiriciler arasında en popüler dil olarak belirtilmektedir. Kotlin statik, çapraz platformlu ve birinci sınıf bir programlama dili olarak Java sanal makinesini destekleyen ve JavaScript kaynak koduyla derlenebilen istatistik tabanlı programlama dilidir. Dart, herhangi bir platformda hızlı uygulamalar geliştirmek için istemci tarafından optimize edilmiş ücretsiz ve açık kaynak kodlu Google tarafından desteklenen bir dildir. Ruby, işlevsel, nesne yönelimli, dinamik bir dil olarak verimlilik ve sadelik üzerine odaklanırken, Rust ise güvenliği ön plana alan işlevsel, genel amaçlı ve çok paradigmalı bir sistem programlama dilidir.

Mobil Uygulama Geliştirme

3

En çok kullanılan mobil uygulama geliştirme platformlarını açıklayabilme

Mobil Uygulama Geliştirmek İçin Kullanılan Platformlar

Solar2D'nin eski adı Corona SDK'dır. Lua tabanlıdır ve bir oyun motoru / yazılım geliştirme kiti olarak sınıflandırılmaktadır. Flutter Google destekli açık kaynak kodlu bir çerçeveye olarak mobil başta olmak üzere, web, masaüstü ve gömülü uygulamalar geliştirmek için kullanılabilen bir platformdur.

Dart dilini temel almaktadır. Apache Cordova açık kaynaklı, ücretsiz bir mobil geliştirme çerçevesi olarak platformlar arası geliştirme için standart web teknolojileri olan HTML5, CSS3 ve JavaScript'i kullanmaya fırsat tanımaktadır. Codename One açık kaynak kodlu bir çapraz platform çerçevesidir. React Native ise Facebook (Meta Inc.) tarafından desteklenen ve Android / iOS için yerel uygulamalar geliştirilebilen açık kaynaklı bir uygulama çerçevesidir. Qt, bir widget aracı iken, Appery.io ise düşük kodlu ve çok amaçlı bir platform sunan bulut tabanlı bir uygulama geliştiricisidir. Sencha Ext JS, herhangi bir cihaz için veri yoğun, çapraz platformlu uygulamalar oluşturmaya yönelik JavaScript çerçevesidir. Xamarin ise açık kaynaklı ücretsiz çapraz platform desteği sunan bir uygulama platformu olarak, .NET ve C# programlama dilleriyle mobil uygulamalar geliştirmek için kullanılabilen bir platformdur.

4

Mobil uygulama mağazalarını sıralayarak özelliklerini açıklayabilme

Mobil Uygulama Mağazaları

Mobil uygulama mağazalarına Google Play Store, App Store, Huawei AppGallery, Amazon Appstore, Microsoft Store örnek olarak verilebilir. Google Play Store, 2012 yılında faaliyetlerine başlayan Google tarafından yürütülen bir mobil uygulama mağazası olarak Şubat 2022 itibarıyle en son 2,6 milyon uygulamayı barındırmaktadır. En yüksek uygulama sayısının Google Play Store'da olduğu görülebilmektedir. İkinci sırada bulunan App Store, Apple tarafından iOS ve iPadOS işletim sistemlerindeki mobil uygulamalar için geliştirilen bir dijital dağıtım platformudur. Huawei AppGallery, Huawei'nin HarmonyOS adlı mobil işletim sistemi ve Android için, geliştirilen bir uygulama mağazası olarak 2011 yılında Çin'de, daha sonra 2018 yılında tüm dünya genelinde kullanılmaya başlanmıştır. Amazon AppStore, e-ticaret sitesi Amazon tarafından işletilen Android işletim sistemi için tasarlanmış bir uygulama mağazasıdır. Bu mağaza, şu anda sadece Android mobil cihazları desteklediği için masaüstü bilgisayarlarda açıldığında çalışmamaktadır. Microsoft Store ise eski adıyla Windows Store olarak bilinmektedir. Microsoft'a ait bir dijital dağıtım platformudur.

1 I. Bu uygulamaları geliştirmek yüksek geliştirme maliyetlerine ve daha uzun bir pazarla inme süresine yol açmaktadır.

II. Cihazın donanımına erişememekte ve HTML5, CSS, Javascript gibi popüler programlama dilleriyle daha kolay geliştirilmektedir.

III. Belirli bir işletim sistemi üzerinde çalışacak şekilde özel olarak geliştirilen uygulama türüdür.

Yerel uygulamalarla ilgili yukarıda verilen bilgilerden hangisi ya da hangileri doğrudur?

- A. Yalnız I
- B. Yalnız II
- C. Yalnız III
- D. I ve III
- E. I, II ve III

2 Diğer uygulamaların, hizmetlerin, yazılımların veya işletim sistemlerinin verilerine ve özellikle erişen uygulamaların oluşturulmasına izin veren bir dizi işlev ve prosedüre ne ad verilmektedir?

- A. Entegre Geliştirme Ortamı (IDE)
- B. İleri Web Uygulamaları (PWA)
- C. Hibrit (Melez) Uygulamalar
- D. Yazılım Geliştirme Kiti (SDK)
- E. Uygulama Programlama Arayüzü (API)

3 Alışveriş, moda, egzersiz ve diyet gibi uygulamaları kapsayan mobil uygulama kullanım alanı aşağıdakilerden hangisidir?

- A. M-ticaret uygulamaları
- B. Yaşam tarzı uygulamaları
- C. Seyahat ve harita uygulamaları
- D. Yardımcı uygulamalar
- E. Artırılmış gerçeklik uygulamaları

4 Aşağıdakilerden hangisi, Java programlama dili ile ilgili **yanlıştır**?

- A. İlk olarak 1995 yılında geliştirilmiştir.
- B. Android SDK ile mobil uygulama geliştirmek isteyen geliştiriciler tercih edebilir.
- C. JavaScript ile eş anlamlıdır.
- D. Java uygulamalarının dosya uzantısı .java, .class, .jar, .jmod olabilmektedir.
- E. James Gosling tarafından geliştirilmiştir.

5 Aşağıdakilerden hangisi iOS uygulama geliştirmeyle ilgilenen geliştiriciler için kullanılabilcek Apple ekosisteminde bulunan açık kaynak kodlu, ücretsiz, nesne tabanlı bir programlama dilidir?

- A. PHP
- B. HTML5
- C. JavaScript
- D. Swift
- E. Kotlin

6 Portekizce'de "Ay" anlamına gelen Brezilya'daki Rio de Janeiro şehrinde Papalık Katolik Üniversitesi PUC-Rio'daki bir ekip tarafından tasarlanan açık kaynak kodlu çok paradigmalı dil aşağıdakilerden hangisidir?

- A. Rust
- B. Lua
- C. Ruby
- D. Dart
- E. Objective C

7

- I. Eski adı PhoneGap olarak bilinmektedir.
- II. Exadel tarafından ilk olarak 2012 yılında piyasaya sürülmüştür.
- III. Düşük kodlu ve çok amaçlı bir platform sunan bulut tabanlı bir uygulama geliştiricisidir.

Appery.io ile ilgili yukarıda verilenlerden hangisi ya da hangileri doğrudur?

- A. Yalnız I
- B. Yalnız II
- C. I ve II
- D. II ve III
- E. I, II ve III

8

Google tarafından desteklenen ve Dart dili temel alınarak uygulama geliştirilebilen platform aşağıdakilerden hangisidir?

- A. Apache Cordova
- B. Codename One
- C. Solar 2D
- D. Xamarin
- E. Flutter

9

Google Play Store ile ilgili aşağıda verilenlerden hangisi **yanlıştır**?

- A. Chrome OS uygulamaları da sunulmaktadır.
- B. HarmonyOS adlı mobil işletim sistemi için özel geliştirilmiştir.
- C. 2012 yılında faaliyetlerine başlamıştır.
- D. En yüksek uygulama sayısına sahiptir.
- E. Eski adıyla Android Market olarak da adlandırılmaktadır.

10

- I. Yalnızca Android mobil cihazları desteklediği için masaüstü bilgisayarlarda açılığında çalışmamaktadır.
- II. Çin-ABD ticaret savaşı nedeniyle popülerliği artmış ve Çin'den dünyaya yayılmıştır.
- III. iOS ve iPadOS işletim sistemlerindeki mobil uygulamalar için bir uygulama mağazasıdır.

Apple AppStore ile ilgili yukarıda verilen bilgilerden hangisi ya da hangileri doğrudur?

- A. Yalnız I
- B. Yalnız II
- C. Yalnız III
- D. I ve III
- E. II ve III

1. D	Yanıtınız yanlış ise “Mobil Uygulama Tanımı, Kullanım Alanları ve Türleri” konusunu yeniden gözden geçiriniz.	6. B	Yanıtınız yanlış ise “Mobil Uygulama Geliştirmek İçin Kullanılan Programlama Dilleri” konusunu yeniden gözden geçiriniz.
2. E	Yanıtınız yanlış ise “Mobil Uygulama Tanımı, Kullanım Alanları ve Türleri” konusunu yeniden gözden geçiriniz.	7. D	Yanıtınız yanlış ise “Mobil Uygulama Geliştirmek İçin Kullanılan Platformlar” konusunu yeniden gözden geçiriniz.
3. B	Yanıtınız yanlış ise “Mobil Uygulama Tanımı, Kullanım Alanları ve Türleri” konusunu yeniden gözden geçiriniz.	8. E	Yanıtınız yanlış ise “Mobil Uygulama Geliştirmek İçin Kullanılan Platformlar” konusunu yeniden gözden geçiriniz.
4. C	Yanıtınız yanlış ise “Mobil Uygulama Geliştirmek İçin Kullanılan Programlama Dilleri” konusunu yeniden gözden geçiriniz.	9. B	Yanıtınız yanlış ise “Mobil Uygulama Mağazaları” konusunu yeniden gözden geçiriniz.
5. D	Yanıtınız yanlış ise “Mobil Uygulama Geliştirmek İçin Kullanılan Programlama Dilleri” konusunu yeniden gözden geçiriniz.	10. C	Yanıtınız yanlış ise “Mobil Uygulama Mağazaları” konusunu yeniden gözden geçiriniz.

1

Araştır Yanıt Anahtarı

Araştır 1

Uygulamalar genellikle cihazda yerel olarak çalışsa da bir web tarayıcısı aracılığıyla da kullanılabilen yazılım türüdür. Bunlar bilgisayarda, akıllı telefonda, tablette, akıllı TV'ler ve akıllı saatler dahil diğer elektronik cihazlarda kullanılabilirlerdir. Örneğin, masaüstü (desktop) uygulamaları, fare ve klavye içeren bilgisayarlar için geliştirilirken, mobil uygulamalar, akıllı telefonlar ve dokunmatik ekranlar için tasarlanmıştır. Mobil uygulama kavramı, telefon/akıllı telefon, tablet, akıllı saat veya akıllı gözlük gibi bir "taşınabilir" aygıtta çalışacak şekilde geliştirilen bir yazılım türü olarak tanımlanmaktadır. Masaüstü bilgisayarlardaki yazılım ve uygulamaların aksine, mobil uygulamalar mobil cihazlara uyumlu olacak şekilde tasarlanmaktadır. Yani, masaüstü için hazırlanan bir uygulama mobil cihazda çalışmayacağı gibi, mobil cihazlar için geliştirilmiş bir uygulama masaüstü bilgisayarlarda kullanılamamaktadır.

Araştır 2

Python genel amaçlı üst düzey bir programlama dili olarak son yıllarda en çok tercih edilen programlama dilidir. PYPL endeksine göre, Eylül 2021 itibarıyle en popüler programlama dilleri arasında ilk sıradadır. Öğrenmesi kolay, etkileşimli bir dildir. Ayrıca, Python, yapay zekâ, finansal hizmetler ve veri bilimi gibi gözde teknolojik alanlar dahil olmak üzere çeşitli şekillerde kullanım alanına sahiptir.

Araştır 3

Mobil uygulama geliştirmek için pek çok platform bulunmaktadır. Solar 2D, Flutter, Apache Cordova, Codename One, React Native, Qt, Appery.io, Sencha Ext JS, Xamarin bu platformlardan bazlarıdır.

Araştır 4

Mobil uygulama mağazaları mobil uygulamaların sunulduğu bir dijital dağıtım platformu olarak mobil cihazlara (tablet, akıllı telefon gibi) yönelik uygulamalar içermektedir. Google Play Store, App Store, Huawei AppGallery, Amazon Appstore, Microsoft Store mobil uygulama mağazaları yanında Opera Mobil Store, Samsung Galaxy Store, BlackBerry World, OpenStore for Ubuntu Touch uygulama mağazaları da aktif kullanılmaktadır.

Kaynakça

- Akbaş, B. (2020). Mobil Uygulama Geliştirme Nedir? Mobil Uygulama Geliştirme için Hangi Programlama Dilleri Kullanılmalıdır? <https://enprobilisim.com/mobil-uygulama-gelistirme-nedir-mobil-uygulama-gelistirme-icin-hangi-programlama-dilleri-kullanilmalidir/>
- Cinto (2021). 6 Reasons Scala is Better Than Java. <https://levelup.gitconnected.com/6-reasons-scala-is-better-than-java-c328cfb410d1>
- Demirci, M. (2015). Mobil Uygulama Türleri. <https://muratdemirci.me/2015/09/01/mobil-uygulama-turleri/>
- Dinakar (2021). Types of Mobile Apps: Native, Hybrid, Web and Progressive Web Apps. <https://www.pcloudy.com/blogs/types-of-mobile-apps-native-hybrid-web-and-progressive-web-apps/>
- Fireart (2021). Types of Mobile Apps. <https://fireart.studio/blog/types-of-mobile-apps/>
- Girdhar, A. (2022). 25 Best Programming Languages for Mobile Apps & Top Mobile App Development Tools & Frameworks. <https://www.appypie.com/top-programming-languages-for-mobile-app-development>
- Indeed Editorial Team (2021). What Is an App? Types of Apps and Examples. <https://www.indeed.com/career-advice/career-development/what-is-an-app>
- Karch, M. (2021). A Beginner's Guide to Mobile Apps. <https://www.lifewire.com/what-are-apps-1616114>
- Kumar, A. (2021). The Mobile App Primer. <https://blog.vsoftconsulting.com/blog/comparing-different-types-of-mobile-application-development>
- Kushnir, A. (2021). Main Categories and Types of Mobile Apps. <https://bambooagile.eu/insights/main-categories-and-types-of-mobile-apps/>
- Lardinois, F. (2019). Kotlin is now Google's preferred language for Android app development. <https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/?guccounter=1>
- Mroczkowska, A. (2021). What Is a Mobile App? | App Development Basics for Businesses. <https://www.thedroidsonroids.com/blog/what-is-a-mobile-app-app-development-basics-for-businesses>
- Özdamar Keskin, N. & Kılınç, H. (2015). Mobil öğrenme uygulamalarına yönelik geliştirme platformlarının karşılaşılması ve örnek uygulamalar. *AUAd*, 1(3), 68-90. <https://dergipark.org.tr/tr/download/article-file/35552>
- Samsukha, A. (2021). What Is A Mobile Application? <https://www.emizentech.com/blog/what-is-a-mobile-application.html>
- Siegel, C. & Dorairajan, A. (2020). What is an API? <https://blog.axway.com/amplify-products/api-management/what-is-an-api>
- Statista (2021a). Number of apps available in leading app stores as of 1st quarter 2021. <https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>
- Statista (2021b). Number of available apps in the Apple App Store from 2008 to 2021. <https://www.statista.com/statistics/268251/number-of-apps-in-the-itunes-app-store-since-2008/>
- Statista (2022). Number of available applications in the Google Play Store from December 2009 to December 2021. <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>
- Stolyar, B. (2021). Apple unveils the most popular iPhone apps of 2019. <https://mashable.com/article/apple-most-popular-iphone-apps-2019>
- Waldellon, L. (2020a). What Are the Different Types of Mobile Apps? And How Do You Choose? <https://clevertap.com/blog/types-of-mobile-apps/>
- Waldellon, L. (2020b). What is an SDK? Everything You Need to Know. <https://clevertap.com/blog/what-is-an-sdk/>
- Walker, A. (2021). What Is an IDE? How It Helps Developers Code Faster. <https://www.g2.com/articles/ide>
- We are social. (2022). Digital 2022. <https://wearesocial.com/uk/blog/2022/01/digital-2022-another-year-of-bumper-growth-2/>
- Webrazzi (2020). Mobil Uygulama Geliştirme Türleri: Yerel, Hibrit, Platform Tabanlı Yaklaşımlar ve Daha Fazlası. <https://webrazzi.com/2020/12/22/mobil-uygulama-gelistirme-turleri-yerel-hibrit-platform/>
- Wetzler, T. (2021). What are the different app types and how do they work? <https://www.adjust.com/blog/different-app-types-and-how-they-work/>

■ Internet Kaynakları

<https://appery.io/>
<https://appgallery.huawei.com/Featured>
<https://bilginc.com/tr/blog/257/hibrit-karma-vs-native-yerli-mobil-uygulama-gelistirme>
<https://blog.vertexplus.com/types-of-mobile-application/>
<https://cordova.apache.org/>
<https://dart.dev/>
<https://dotnet.microsoft.com/en-us/apps/xamarin>
https://en.wikipedia.org/wiki/Amazon_Appstore
https://en.wikipedia.org/wiki/App_store
[https://en.wikipedia.org/wiki/App_Store_\(iOS/iPadOS\)](https://en.wikipedia.org/wiki/App_Store_(iOS/iPadOS))
https://en.wikipedia.org/wiki/Google_Play
https://en.wikipedia.org/wiki/Huawei_AppGallery
[https://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language))
<https://en.wikipedia.org/wiki/JavaScript>
https://en.wikipedia.org/wiki/List_of_mobile_app_distribution_platforms
https://en.wikipedia.org/wiki/List_of_most-downloaded_Google_Play_applications
https://en.wikipedia.org/wiki/Microsoft_Store
https://en.wikipedia.org/wiki/Mobile_app
https://en.wikipedia.org/wiki/Mobile_app_development
<https://flutter.dev/>
<https://gs.statcounter.com/os-market-share/mobile/worldwide>
<https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet>
<https://pypl.github.io/PYPL.html>
<https://reactnative.dev/>
<https://solar2d.com/>
<https://w3techs.com/technologies/details/pl-php>
<https://www.apple.com/tr/app-store/>
<https://www.charterglobal.com/understanding-the-3-types-of-mobile-apps-development-services/>
<https://www.codenameone.com/about-us.html>
<https://www.gurukafa.net/blog/en-iyi-mobil-uygulama-gelistirme-dilleri-nelerdir/>
<https://www.innoraft.com/blogs/different-types-mobile-app-development-platforms>
<https://www.lua.org/about.html>
<https://www.mendix.com/low-code-guide/>
<https://www.microsoft.com/tr-tr/store/most-popular/apps/mobile>
<https://www.mobiluygulama.com.tr/mobil-uygulama-turleri-nelerdir>
<https://www.qt.io/>
<https://www.ruby-lang.org/tr/about/>
<https://www.rust-lang.org/>
<https://www.scala-lang.org/>
<https://www.sencha.com/products/extjs/>
<https://www.webiotic.com/different-types-of-mobile-app-development/>

Bölüm 2

Mobil Arayüzlerde Kullanılabilirlik ve Mobil Uygulama Tasarımı

Öğrenme Çıktıları

1

Mobil Arayüzlerde Kullanılabilirlik

- 1 Kullanılabilirliğin temel bileşenlerini ve ölçüm yöntemlerini tanımlayabilme

2

Mobil Uygulama Tasarımı

- 2 Mobil uygulama tasarımının genel özelliklerini açıklayabilme

3

Evrensel Tasarım ve Erişilebilirlik

- 3 Evrensel tasarım ve erişilebilirlik kavramlarını anlatabilme

Anahtar Sözcükler: • Kullanılabilirlik • Kullanıcı Deneyimi • Kullanılabilirlik Testi • Sezgiseller
• Mobil Uygulama Tasarımı • Responsive Tasarım • Erişilebilirlik
• Kullanılabilirlik Standartları ve Rehberleri



GİRİŞ

9 Ocak 2007. Mobil cihazlarda yeni bir dönemin başladığı tarih. O gün iPhone®, Steve Jobs tarafından yapılan bir sunumla kullanıcıları ile tanıştırdı. 29 Haziran 2007'de satışa sunulacak iPhone'nun tanıtıldığı MacWorld 2007 konferansında ise 'Apple Bilgisayar' ismi 'Apple'a dönüştürdü. Bu değişim Apple için kullanıcı elektroniki sektöründe odağın bundan sonra ürünlerin içindeki teknolojiden ziyade, kullanıcı deneyimi olacağının işaretiydi.

2008'de *iOS App Store* sadece 500 uygulama ve üçüncü taraflar için yazılım geliştirme seti ile birlikte geliştiricilerin ve kullanıcıların hizmetine sunulmuştur. Takip eden 10 yıl içinde ise *Apple App Store*'da yer alan uygulama sayısı yaklaşık 2 000 000 seviyelerine ulaşarak, o tarihten itibaren benzer düzeyi korumaktadır (apple.com/app-store/). 2022 itibarıyla diğer büyük uygulama sağlayıcısı olan *Google Play*'in Android tabanlı akıllı telefon kullanıcıları için sunduğu uygulama seçenekleri ise biraz daha yüksektir, yaklaşık 2 500 000.

Bu bölümün amacı, mobil uygulama tasarımları veya mobil arayüzlerde kullanılabilirlik konusunda bilgi edinmek isteyen okurlara temel bilgiler yanında, dilerlerse daha fazla uzmanlaşabilecekleri için alanda yürütülen güncel tartışmaları genel hatları ile sunmaktadır. Mobil uygulama tasarımlının ilkeleri ile kullanılabilirlik konusunda daha fazla uygulamaya yönelik bilgi veya tecrübe edinmek isteyen okurlar bölüm sonunda verilen kaynaklara başvurabilirler.

MOBİL ARAYÜZLERDE KULLANILABİLİRLİK

Mobil uygulama geliştirme çığlığını ile birlikte birçok şirket ve/veya bağımsız geliştirici için kullanılabilirlik özellikle üzerinde durulması, geliştirme sürecinde ele alınması gereken bir kavram olmuştur. Bugün, mobil uygulamalarda var olan kullanılabilirlik ilkelerinin birçoğu insan bilgisayar etkileşimi alanında yürütülen arayüz çalışmaları sonucunda tanımlanmıştır.

Kısa Tarihçe

Günümüzde kullanıldığı anlamda, kullanılabilirlik kavramının öncüsü sistematik olarak yürütülen ergonomi (insan faktörü) çalışmalarıdır. Ergo-

nomi, genel anlamıyla insanların yaptığı iş ile insan arasında işleyen yasaların anlaşılması çalışılması olarak tanımlanabilir.

Ergonomi çalışmalarının tarihi biraz daha eskilere uzansa da kullanılabilirlik testi yapılan laboratuvarlar pilotlar ile çalışmalar yapmak için 20. yüzyılın ilk yarısında ABD'de kurulmuştur. Kurulduğu yıllarda kullanılabilirlik laboratuvarlarının amacı bugün olandan daha farklıydı. Kullanılabilirlik testlerinin amacı teknolojinin gelişimi ile daha da karmaşık hale gelen işler için "doğru türde yeteneği olan insanları" bulmaktı. Ancak İkinci Dünya Savaşı, üretimi zor ve maliyeti çok yüksek olan bu uçaklar için "doğru pilotları bulma" stratejisini sürekli bir çözüm sağlamadığını gösterdi. Uçaklara entegre edilen her yeni teknoloji, o teknolojiyi kullanacak daha da iyi adayların bulunmasını gerektiriyordu. Sonuçta, pilotlar her ne kadar en yetenekliler arasından seçilse de kullanım sırasında insan hatasının bu yöntemle engellenmesinin mümkün olmadığını ortaya çıktı (Soegaard, 2012).

1945 yılında Vannevar Bush tarafından ortaya atılan ve günümüzde kullandığımız bilişim sistemlerinin nerede ise birebir karşılığı olan, teorik analog bilgisayar fikrini (MEMEX, *MEMory Extender* – Bellek Genişletici) bu alandaki güncel yaklaşımı temsil eden adımlardan ilki olarak adlandırmak yanlış olmaz (Bush, 1945). Bush, Atlantic Monthly dergisinde yayınlanan "As We May Think – Düşündüğümüz Gibi" makalesinde, MEMEX adlı bir makina üzerinden bahseder. Makalede, bu makinanın insanın bilişsel sistemini nasıl destekleyeceğini tanımlar. MEMEX sadece insan bilgisayar arasındaki etkileşimi değil aynı zamanda, insan-insan arasındaki etkileşimi de gündeme getirmektedir. Bu makalenin hayali araştırmacıları tarihte çok ünlü olan Türk ok ve yayları konusunda araştırma yapmakta, MEMEX'in zengin kütüphanesi üzerinden gerekli bilgiye ulaşmaktadır. Araştırmacılar bu bilgileri birbirleri paylaşarak ileride kullanmak üzere MEMEX'de depolamaktadırlar (<https://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881/>).

Dijital teknolojilerin gelişmesi ile birlikte 1960'lardan sonra insan bilgisayar etkileşimi alanında akademik yayınlar ve çalışmaları düzenlenmeye başlamıştır. 1970'lere gelindiğinde ise İngiltere'de Brian Shackel tarafından HUSAT (*Human Science and Advanced Technology Research Institute*), ABD'de ise Xerox PARC (*Palo Alto Research*

Center) kurularak bilgisayar teknolojisile ilgili ürünler ve donanım sistemleri geliştiren başlıca iki araştırma grupları oluşturulmuştur.

1980'ler, 20. yüzyılın başlarında ve II. Dünya Savaşı boyunca güçlü bir etkiye sahip olan ergonomi ve insan faktörleri yaklaşımının, bugün bildenimiz anlaşıyla kullanılabılırlik çalışmalarına dönüştüğü yıllarda. (i) Sistemlerin kullanılabılırliği çalışmalarını planlamak, (ii) gerçekleştirmek ve (iii) geçerlemek amacıyla kullanılan tanım, kavram ve tekniklerinin birçoğuna kullanılabılırlik mühendisliği öncülük yapmıştır (Çağiltay, 2018).

Mobil arayüzler için, ergonomiden ve insan bilgisayar etkileşimi çalışmalarından elde edilen bilgilerin birçoğu bugün de geçerliğini koruduğundan, bir sonraki başlıkta kullanılabılırlik kavramını incelemek için temel bileşenler konusunda insan-bilgisayar etkileşimi literatüründen yerleşik tanımlar incelenec, mobil uygulamalara özgü kullanılabılırlik nüanslarına metin içinde yeri geldiğinde degeinilecektir.

Kullanılabilirlik Kavramının Önemi

Kullanılabılırlığın en genel anlamı bir sistem ile ilgili görevlerin kullanıcılarına güvenli, etkili ve verimli biçimde sunulmasıdır. Aynı zamanda bu görevlerin yerine getirilmesi sırasında kullanıcının deneyimden keyif almasını sağlayacak koşulların sağlanması da kullanılabılırliğin hedefleri arasındadır. Kullanılabılırlik kavramında sistem bir makine, bir süreç, bir uygulama, bir web sitesi, bir kitap, bir araç, bir oyun veya bir insanın etkileşimde bulunduğu herhangi bir şey olabilir.

Kullanılabılırlik terimi aslında günlük hayatımızın içinde yer almaktadır. Yeni aldığımız bir cihazın nasıl çalıştığını anlamak, bir bankanın mobil uygulamasında EFT işlemini nasıl yapacağımızı keşfetmek ya da kiraladığınız yeni bir aracın dokunmatik multimedya sistemini telefonunuz ile eşleştirmek gibi davranışların hepsi kullanılabılırlik ile doğrudan ilgilidir. Verilen örnekler elektronik cihazlar ile ilgili olsa da unutulmamalıdır ki, kullanılabılırlik dijital dünya ile sınırlı değildir. İçinde bulunduğuımız çevre, okul, şehir, kaldırımlar; hergün gelip geçtiğiniz yolların da kullanılabılır olup olmaması günlük yaşamımıza doğrudan etki eder.

İnsan-bilgisayar etkileşimi ve bilgisayar bilişinde kullanılabılırlik ise, daha spesifik olarak bir bilgisayar programı, mobil uygulama veya web arayüzü ile etkileşimin nasıl tasarlandığını, görsel-

liğini ve fonksiyonel anlaşılabilirliğini inceler. Kullanılabılırlik genel olarak kullanıcı memnuniyetini ve faydayı kalite bileşenleri olarak kabul ederken, tekrarlayan tasarım ile kullanıcı deneyimini iyileştirmeyi amaçlar.

Günlük hayatımızın her zaman içinde olan kullanılabilirlik çalışmalarının neden önemli olduğunu listeleyeceğimiz olursak:

- Kullanıcı memnuniyetinin arttırılmasının yanı sıra, ürüne ve organizasyona yönelik olumlu algının olmasını sağlar
- Ürün geliştirme sürecindeki harcamaların azaltılmasını sağlar
- Daha eksiksiz bir ürünün geliştirilmesine olanak verir
- Ürüne yönelik olumsuz gelişmelerin oluşma riskini azaltır
- Geliştirme sürecinin ilk aşamalarında, kavramların, tasarımın, akış ve içeriğin geçerliğinin test edilmesine olanak verir
- Uygulamadaki değişikliklerin ve problemlerin sayısının azaltılmasını sağlar.

Kullanılabılırlığın Temel Bileşenleri

Mobil uygulamalarda arayüz ile etkileşim dört ana bileşeni kapsar. Bu bileşenler (i) kullanıcı (*user*), (ii) araç/arayüz (*tool*), (iii) görev (*task*) ve (iv) bağlam (*context*)’dır (Şekil 2.1). Bu dört bileşen arasındaki etkileşim bir bütün olarak kullanıcılar açısından kullanılabılırlik deneyiminin kendisidir.

Kullanıcı ya da daha geniş bir bakış açısı ile ‘*insan*’ etkileşimde başroldeki bileşendir. Kullanıcı belirli bir hizmeti, ürün ya da sistemi kullanması beklenen hedef kitledir. Kullanıcı tek bir kişi veya grup olabilir. Örneğin sosyal ağların özellikle mobil cihazlarda en çok zaman geçirilen uygulamalar arasında olması nedeni ile bu konuda yapılacak bir çalışmada kullanıcı bir kişiyi değil bir grubu ifade edecektir!

Araç/Arayüz ise kullanımının gerçekleştirmek istediği görevi yerine getirmek için kullandığı cihazdır. Bu cihazlara örnek olarak bilgisayarlar, akıllı telefonlar veya taşınabilir müzik çalarlar verilebilir.

Üçüncü temel bileşen olan **görev** ise kullanımın gerçekleştirdiği iştir. Kullanılabılırlik çalışmaları içinde bu görevlerin daha iyileştirilmesi ve kullanıcının zorluk çekmeden bu görevleri yerine getirmesi öncelikli konulardır.

Son olarak **bağlam** ise kullanıcının bir görevi bir araç ile gerçekleştirken içinde bulunduğu ortamıdır. Bu ortam kullanıcının ofisi, otomobili, evi, sınıfı, sokak veya bir park olabilir. Eğer ortamda cihaz ve kullanıcı dışında var olan üçüncü kişiler ile etkileşim varsa sosyal dinamikler de ortama dahil edilir. Mobil uygulamalar, bilgisayarlardan farklı olarak günlük yaşamımızın her anında yanımızda olduğu için bağlam mobil uygulama tasarımda daha fazla çeşitlilik göstermektedir.

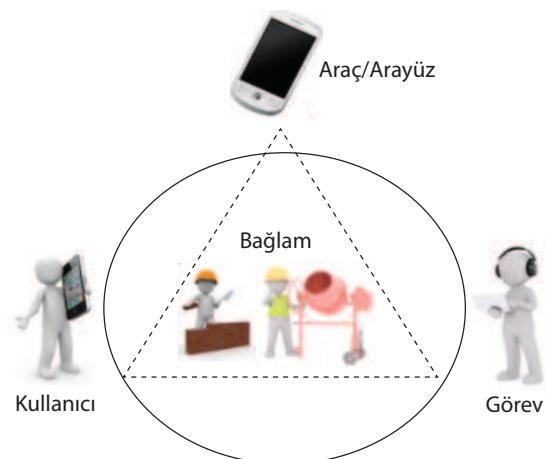
Örneğin bir cep telefonu ve onun dokunmatik ekranı bu sistemin aracı ve arayüzüdür. Diğer yan dan telefonu kullanan kişi çok farklı bedensel veya bilişsel özelliklere sahip olabilir; yaşlı veya çocuk, eğitimli veya okuma-yazma bilmeyen gibi... Telefonun ve kişilerin değişkenliğine ek olarak, telefon ile yapılacak görevler de kullanıcılar gibi geniş yelpaze ye sahip olabilir; konuşmak, oyun oynamak, mesaj yazmak, not almak, fotoğraf çekmek gibi. Telefonun hangi bağlam ya da ortamda kullanıldığı da önemli temel bir bileşenlerden biridir. Kişi evde, otobüs te, sokakta, karanlıkta, yatakta, sessiz veya gürültülü bir ortamda telefonu kullanıyor olabilirler. Kullanılabilirlik çalışmalarında bu dört bileşen (kullanıcı, araç/arayüz, görev, bağlam) göz önüne alınarak, her koşulda tüm kullanıcılar tarafından, tüm görevlerin en az çaba ve sorunsuzca kullanılabilmesine yönelik etkileşimin tasarlanması için atımların araştırılması gereken adımların araştırılmasına yer verilir.

Kullanılabilirliğin Tanımı

Kullanılabilirlik günlük dilde genellikle, bir ürünün kullanıcı dostu, kolay kullanılır, kolay öğrenilir, sezgisel olarak anlaşılabılır gibi özelliklerini tanımlayan bir terim olarak düşünülür. Fakat bir çalışma alanı olarak kullanılabilirlik günlük dildeki anlamından biraz daha farklıdır. Bazı uzmanlar kullanılabilirliği, ergonomi (insan faktörü) alanının bir dalı, yazılım uzmanlığı olarak değerlendirir. Bazıları da ergonominin fizyolojik konulara, kullanılabilirliğin ise psikolojik konulara daha çok odaklılığını ileri sürer. İnsan bilgisayar etkileşimi açısından bakıldığına ise kullanılabilirlik daha dar bir anlama sahiptir, web siteleri, yazılımlar, mobil uygulamaların kullanıcı arayüzü deneyimindeki kaliteyi tanımlar. Biraz daha geniş bir açıdan bakıldığına ise, kullanılabilirliğin kapsayıcı ve evrensel tasarım ile olan ilişkisini dikkate almak gereklidir.

Kullanılabilirliğin günlük dildeki anlamı bilim insanların üzerinde çalışması için yeterince belirlenmiş sınırlara sahip değildir. Örneğin bir uygulamanın kolay anlaşılıbıldığı ileri sürüldüğünde akla hemen; - kim için kolay anlaşılır, her koşul ve ortam da bu uygulamanın kullanımı kolay mıdır, uygulamadaki bütün görevlerde hedefe ulaşmak için atılan adımların doğruluğu ve bütünlüğü nelerdir, uygulamayı kullanılabılır kabul etmek için hangi ölçütler kullanılmıştır? - gibi birçok soru gelir. Bir kullanılabilirlik uzmanı için "kolay kullanılır" ifadesinin anlam taşıyılmasını için yukarıdaki soruların cevabı ile birlikte verilmesi gereklidir. Bu nedenle, günlük dile yerleşmiş olan kolay kullanım ile ilgili kalıplar dışında niteliksiz veya niceliksiz bir ölçüm yapabilmek için kullanılabilirliğin kişiden kişiye değişmeyen nesnel bir tanımının yapılması gereklidir.

Kullanılabilirlik ile ilgili çeşitli tanımlar olmasına rağmen, Uluslararası Standartlar Enstitüsü (*International Standards Organization- ISO*) tarafından yapılan kullanılabilirlik tanımı herkes tarafından kabul edilmiş ve yaygınlaşmıştır. Kullanılabilirlik (ISO 9241 Bölüm 11) sayılı dokümanda aşağıdaki gibi tanımlanmaktadır.

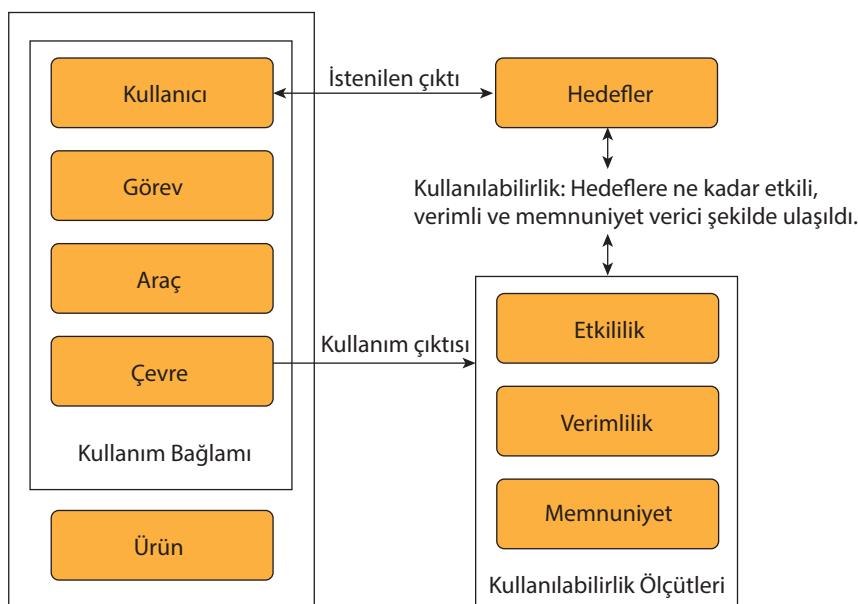


Şekil 2.1 Kullanılabilirliğin Ana Bileşenleri

Kullanılabilirlik: Belirli bir kullanıcı grubunun, belirli görevleri, belirli bir bağlamda etkili, verimli ve memnuniyet ile yerine getirmeleridir.

Kaynak: ISO Ergonomi gereksinimleri, ISO 9241 bölüm 11: Kullanılabilirliğin belirlenmesi ve ölçütleri için kılavuz.

ISO 9241-11 standart dokümanında belirlenen kullanabilirlik tanımının bir de görsel gösterimi hazırlanmıştır. Bu gösterimde, tanımlardaki tüm kavramların birbirleri ile olan ilişkileri belirtilmiştir (Şekil 2.2).



Şekil 2.2 ISO 9241-Bölüm 11 Standardına Göre Kullanılabilirliğin Gösterimi

Kullanılabilirliğin araştırmacılar tarafından en sık referans verilen temel unsurlarının verimlilik, etkililik ve memnuniyet ölçüsü cinsinden ifade edildiği yukarıda belirtilmiştir. Bu üç temel unsuru tanımlama gereklisi **etkililik**, kullanıcıların uygulamayı kullanarak yapması beklenen işleri ne ölçüde başarabildiğini ifade eder. Bu anlamda etkililik, kullanıcının kendisine verilen görevi ne kadar doğru yaptığı ve veya tamamlayabildiğini ölçer. Belirlenen işi yapmak için kullanılan zaman, çaba ve maliyet gibi unsurlar **verimlilik** ölçümü ile değerlendirilir. Mobil bir uygulama örneğinde verimlilik, kullanıcının belirlenen işi ne kadar sürede yaptığı ya da hangi adımları izlediği gibi ölçümler ile belirlenebilir. **Memnuniyet** ise kullanıcının bir uygulamayı kullanırken oluşan duygularını (beğendikleri, beğenmedikleri, tutumu, vb.) ifade eder. Memnuniyet, genellikle bu amaçla düzenlenen ankетler veya sözlü geri bildirimlerin analizi ile belirlenir. Bir ürünün kullanılabilirliğini anlamak için, etkililik ve verimlilik verileri tek başına yeterli olamamakta, bunların yanı sıra, memnuniyet verileri ile de bir değerlendirmenin yapılması gerekmektedir.

ISO'dan farklı olan kullanabilirlik tanımlara Nielsen'in (1993) beş farklı bileşenden oluşan tanımı örnek verilebilir. Bu beş bileşen aynı zamanda Shneiderman (1998)'ın hedeflerinin değerlendirilmesinde 'beş ölçülebilir insan faktörü' sınıflaması ile birebir örtüşmektedir. ISO'nun 3 olan bileşen sayısının 5'e yükseltilmesi kullanılabilirlik kavramının uygulabilirliğini artırmaktadır. ISO, Nielsen ve Shneiderman'in kullanılabilirlik bileşenleri Tablo 2.1'de karşılaştırılmış olarak sunulmuştur.

Tablo 2.1 ISO 9241-11, Shneiderman ve Nielsen'in Kullanılabilirlik Tanımları

ISO 9241-11	Shneiderman	Nielsen
Verimlilik	Performansın Hızı	Verimlilik
	Öğrenme zamanı	Öğrenilebilirlik
Etkilik	Zaman içinde hatırlama	Akılda kalıcılık
	Kullanıcıların hata oranı	Hatalar/Güvenlik
Memnuniyet	Öznel memnuniyet	Memnuniyet

Kaynak: (van Welie, van der Veer ve Elins' den adapte edilmiştir - 1999)

Kullanılabilirliğin daha iyi anlaşılmasına için kullanılabılırlik ile bugün yaygın olarak kullanılan kullanıcı deneyimi (*User eXperience - UX*) arasındaki ilişkinin de tanımlanmasında yarar vardır. Kullanıcı deneyimi ifadesi, kullanılabılırlik kadar nesnel ölçütlerde sahip olmasa da günümüzde kullanılabılırlikten daha kapsayıcı bir terim olarak alan arasında yer almaktadır. Kullanıcı deneyimi kullanıcı merkezli tasarım kavramı ile yakından ilintili olup, kullanıcının ürün ile olan etkilileşiminin tamamını, kullanıcının ürünü ne kadar faydalı gördüğünü ve daha sonrasında bu ürünü kullanmaya ne kadar istekli olacağını içerir. Kullanıcı deneyiminin yaygınlaşması genelde Norman'a atfedilse de Norman bu terimi çalışma alanını tanımlamak için değil de Apple'da görev yaptığı yıllarda yönettiği grubu ifade etmek için kullanmıştır (Norman, 2013).



Kullanıcı deneyimi balpeteği modeline <https://www.usability.gov/what-and-why/user-experience.html> adresinden erişebilirsiniz (Morville, 2004)



UX - UI: Kullanıcı deneyimi (*User eXperience - UX*) bir kullanıcının bir ürün veya hizmetle ilgili deneyimini, kullanıcı arayüzü (*User Interface - UI*) ise insanlar ve cihazlar arasında etkileşimin gerçekleştiği yeri ifade eder.

Kullanılabilirliğin Ölçülmesi

Mobil uygulama marketlerinde milyonlarca mevcut olan uygulamaların marketten indirilmesi, yüklenmesi, ilk defa kullanılması, gerekiyorsa kullanıcının kayıt olması, telefondan silinmemesi, hatta ilerleyen zamanda tekrar kullanılması birçok faktöre bağlıdır. Birçok uygulama yükleme sonrası çeşitli nedenlerden dolayı kaldırılmaktadır. Örneğin Türkiye'de, 30 gün sonra mobil uygulama kaldırma oranı %62'dir. ABD, Japonya ve Birleşik Krallık gibi gelişmiş ülkelerde bu oran %40 civarındadır (Ceci, 2021).

Uygulama geliştiriciler ister bireysel çalışmalar ister büyük bir ekibin parçası olsunlar, uygulamanın kullanıcılarına sunduğu kullanılabılırlik deneyiminin uygulamanın market sonrası yaşam döngüsüne olan doğrudan etkisi konusunda bilgi sahibi olmalıdır. Bu nedenle, kullanılabılırlik testleri, uygulama geliştirme çalışmalarının en önemli parçalarından birisidir. Bu testler, bir uygulamanın geliştirme sürecinin başlangıcından itibaren uygulanabileceği gibi, tamamlanmış bir ürünün kullanılabılırlığını test etmek için de kullanılabılırler.

Kullanılabılırlikle ilgili, kolayca tahmin edilebileceği gibi birçok test türü ve ölçüm yaklaşımı vardır. Test türlerinde ve yaklaşımlarda var olan çeşitlilik nedeniyle, birçok araştırmacı kendi bakış açısına göre farklı parametreler kullanarak gruplama önermiştir. Örneğin, testler uygulandığı mekâna göre sınıflandırılabilir; uzaktan veya laboratuvar gibi kontrollü bir ortam olabilir. Aynı zamanda uygulanan test sırasında bir moderatörün varlığı veya kullanıcının yalnız olması da ayrı bir parametre olarak karşımıza çıkabilir. Araştırma sırasında nitel veya nicel verinin toplanması hedefi de ayrı bir parametredir. Seçilen kullanılabılırlik araştırması, ürünün daha geliştirilme aşamasında olabileceği gibi, uygulama yayınlandıktan sonra A/B testi gibi bir yöntem de tercih edilebilir. Kullanılabılırlik testleri ile ilgili seçenekler daha da çoğaltılabılır, kısa sürede uygulayabilecek bir teste mi ihtiyacınız var, yoksa planlanmış, sistematik bir çalışma mı hedefliyorsunuz? Kullanılabılırlik araştırması için kullanıcı temelli bir test mi, uzman görüşüne dayalı sezgiselleri mi kullanacaksınız?

Burada bilinmesi gereken nokta kullanıcı deneyimi ile ilgili yapılan her çalışmanın, gerçek kullanıcılar ile yapılmış deneysel bir kullanılabılırlik testi ile ulaşılabilen işlevsel sonuçların yerine geçmeyeceğinin bilinmesidir. Her çalışmanın avantajlı ve dezavantajlı yönleri olabilir. Önemli olan bunların farkında olmak, amaca göre doğru testi seçmektir. Örneğin bir odak grup çalışması, ürün hakkında kişilerin görüşlerini almak konusunda veri sağlarken, ürünün gerçekte nasıl kullanılacağı hakkında bir bilgi kaynağı değildir. Yukarıda de濂ilen A/B testi ile birden çok sürümü olan bir uygulamanın hangisinin en etkili olduğunu test edilebilir. A/B testini belirli bir yaklaşımın işe yarıp yaramadığını doğrulamaya yardımcı olması için de kullanabilirsiniz, ancak A/B testinin kullanıcıların neden bir tasarıyı diğerine tercih ettiğini konusunda hiçbir

şey söyleyemediğinin farkında olmalısınız. Diğer yandan, maliyetten veya zamandan tasarruf etmek için gerçek kullanıcılar yerine firma personeli kullanıyorsanız, gerçek kullanıcı testi ile ulaşılan sonuçlar kadar tarafsız olmayacağı bilmeniz gereklidir (Medlock, 2018).

Sonuç olarak, kullanılabılırlik çalışmaları sırasında yürütülen testleri gruplamak için farklı yaklaşımalar mevcuttur. Eğer testler elde edilen verinin kaynağı ile ilişkilendirilirse, bu başlıkta, (i) tasarım rehberleri, (ii) kullanıcı, (iii) uzman ve (iv) model temelli olmak üzere dört ayrı grup oluşturulabilir. Testler nasıl değerlendirildiklerine göre de gruplanabilir. (a) süreç içi (*formativel/ diagnostic*) veya (b) süreç sonu (*metrication/ summative*) zamanlama seçilebilir. Fakat her kullanılabılırlik testi için belki bir veri kaynağı ve değerlendirme zamanı seçilmek durumundadır (Çağiltay, 2018). Örneğin, bir oyun firması geliştireceği mobil oyunun senaryosundan itibaren (süreç içi), oyuncular ile (kullanıcı temelli) kullanılabılırliği düzenli olarak test etmeyi tercih edebileceği gibi, diğer bir uygulama için de uygulamanın mevcut sorunlarını ortaya çıkarmak için uzman görüşüne başvurabilir (süreç sonu, uzman temelli yaklaşım).

Tasarım Rehberleri

Tasarım rehberleri genellikle, bilişim sistemleri geliştiren büyük firmalar tarafından, kendi ürünlerine ait tasarımlarda uygulanmak üzere hazırlanır (*Apple Human Interface Guidelines, Android Design Guidelines, Google's Material Design* gibi). Bu rehberlerin temel amacı, hazırlanan tasarımlarda tutarlılığı ve uyumluluğu sağlamak. Ancak, uyumluluğun kullanılabılırlik ile aynı anlama gelmediği konusuna özellikle dikkat edilmelidir. Tasarım rehberlerini, gerçekleştirilmesi planlanan arayuzlerde kullanmak, her zaman mümkün olmayabilir. Bu-

nun yanında, tasarımcıların kendi tasarımlarının kullanılabılırlığı hakkında, sağlıklı görüşler geliştirmediği göz önüne alınırsa (Bailey, 1993), tasarım rehberlerinin yorum ve uygulamayı sadece tasarım-ciya bırakması, kullanılabılırlik açısından sakıncalar yaratılabilmektedir.

Türkçe olarak ve çok kapsamlı hazırlanan bir diğer tasarım dokümanı “Kamu Kurumları İnternet Siteleri Rehberi” KAMİS’dir. Bu rehberde kullanılabılırlik ile ilgili hemen hemen tüm konular, örneklerle ele alınmaktadır (kamis.gov.tr).



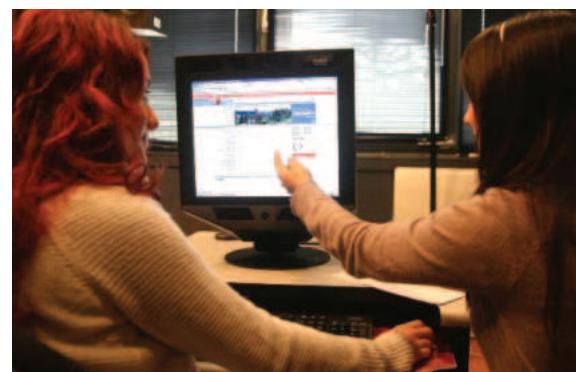
Kullanılabilirlik Testleri (Deneysel Yaklaşım)

Deneysel yaklaşım, gerçek kullanıcıları, gerçek bir arayüzü kullanırken, gerçek görevler ile, tercihen gerçek uygulama ortamında test etmeyi ve toplanan verileri analiz ederek tasarımını kullanılabılırlik açısından değerlendirmeyi amaçlar. Kullanılabılırlik testleri, kullanılabılırlik kriterlerinin belirlenmesinin ardından hedef kullanıcı kitlesine uygulanır ve sonuçlar, tasarımın yeniden değerlendirilmesi için birer girdi olarak kullanılır. Kullanılabılırlik testlerinden elde edilen deneysel veriler kullanılarak, tasarımın daha kullanılabılır bir hale getirilmesi sağlanır.

Gerçek kullanıcılarla yapılan kullanılabılırlik testleri için farklı yöntemler tercih edilebilir. Kullanıcıların uygulamayı kullanırken sergilediği bedensel davranışlar değerlendirilebileceği gibi, izledikleri yolu sesli düşünme ile (*think aloud*) kullanılabılırlik uzmanlarına aktarması da istenebilir. Böylece elde edilen görüntü ve ses kayıtları, diğer test verileri ile birlikte değerlendirilebilir. Bunların yanı sıra, göz-izleme (*eye-tracking*) cihazları kullanılarak elde edilen veriler testler ile ilgili çok daha detaylı bilgi verebilir (Görsel 2.1).



A



B

Görsel 2.1 Kullanılabilirlik testleri farklı deney ortamlarında uygulanabilir A) Kâğıt Prototip Üzerinden Yapılan Bir Kullanılabilirlik Testi B) ODTÜ İBE Laboratuvarında Bilgisayar Başında Yapılan Bir Kullanılabilirlik Testi

Deneysel yaklaşım temelli kullanılırılık testleri, tasarımın kullanılırılığı hakkında en gercekçi verileri veren ve çok yaygın kullanılan bir yöntemdir. Ancak, bunlar emek yoğun testlerdir ve oldukça detaylı bir şekilde planlanması gereklidir. Bu testler aynı zamanda en çok hatanın yapıldığı test türü olarak adlandırılabilir. Çok iyi planlanması ve uygulanması gereken bu testler sonucunda elde edilen veriler çok dikkatli bir şekilde yorumlanmalıdır. Bu nedenle, bu çalışmaların maliyetleri diğer testlere göre daha yüksektir.

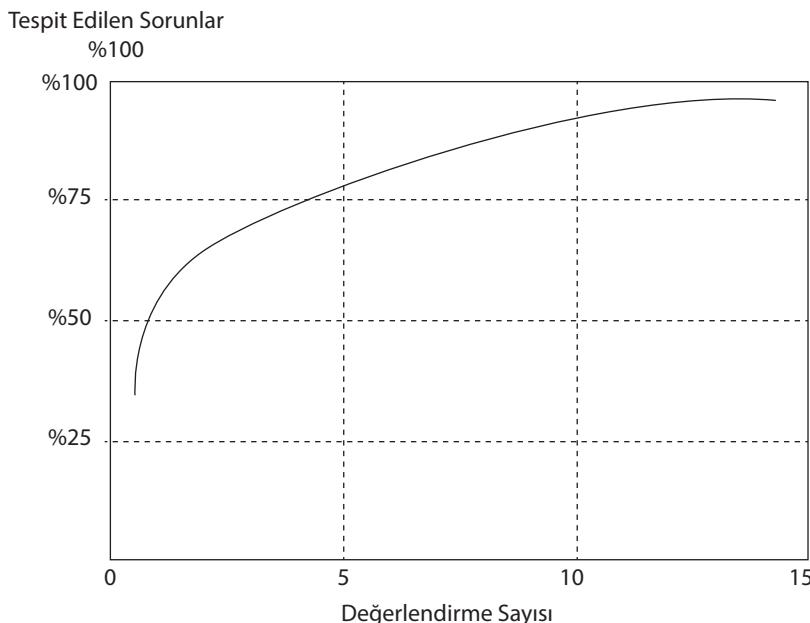
Son kullanıcı ile kullanılırılık testlerini gerçekleştirirken takip edilmesi gereken adımlar başlıklar halinde aşağıda özetlenmiştir. Bu başlıklarla ilgili daha ayrıntılı bilgi için Kürşat Çağiltay'ın (2018) "Teoriden Pratiğe İnsan Bilgisayar Etkileşimi ve Kullanılabilirlik Mühendisliği" kitabı incelenbilir.

a. Test Öncesi Belirlenmesi Gerekenler:

Kullanılabilirlik testleri için ihtiyaç duyulan ideal kişi sayısı konusunda en çok kabul gören görüşlerden Nielsen'e göre, kullanılırılık testlerinde bir sistem ile ilgili temel sorunları ortaya çıkartabilmek için gerekli kişi sayısı en az beş olmalıdır (Şekil 2.3).

Nielsen, iyi tasarılanmış testlerle kullanılırılık problemlerinin %75'inin beş denek ile ortaya çıkarılabileceğini belirtmiştir. Nielsen'in testlerde kullanılacak kişi sayısını ve testin edilen kullanılırılık problemlerinin yüzdesi ile ilgili olarak aşağıdaki şekildeki ilişkiye vermiştir (Nielsen, 1993). Buna göre, denek sayısını beş yerine on yapmak, bulunan sorunların sadece yaklaşık %15'ini ortaya çıkarmaktadır.

- b. Kullanılabilirlik Çalışmaları Sürecinde İzlenmesi Gereken Adımlar
 - Uygulamadan Beklenen Hedeflerin Bellirlenmesi
 - Hedef Kitlenin Siteden Beklentilerinin Analizi
 - Kullanıcı Profillerinin Çıkarılması
 - Test öncesi ve sonrası kullanılacak ensüremanlar neler olacak?
- c. Testin Gerçekleştirilmesi
- d. Veri analizi
- e. Bulguların Raporlanması



Şekil 2.3 Nielsen'e Göre Kullanılabilirlik Problemleri ve Kullanıcı Sayısı

Yukarıda sayılan beş madde bir kullanılabılırlik testi için bir araştırmancın takip etmesi gereken temel adımlardır. Kullanılabilirlik testleri genellikle takım çalışması ile gerçekleştirilir. Bu nedenle bir kullanılabılırlik testi uzmanı için en önemli özelliklerden biri takım çalışmasına yatkınlıktır. Testin tasarlanması aşamasında deneklerin deneyimleri, demografik özellikleri dikkate alınmalıdır. Kullanılabilirlik testlerinde potansiyel kullanıcı profili (persona) çıkarılması analiz için yardımcı olur. Böylece sistem testleri herhangi birisi ile değil, gerçekten o sistemi kullanacak hedef kitleyi temsil eden kişilerle yapılır.

Test esnasında denekler için deney süreçlerini açıklayan bir protokol hazırlanması ve bilgilendirilmiş onam alınması, ileride testin tekrarlanabilirliği ve etik problem ile karşılaşılmaması için önemlidir.

Uzman Değerlendirmeleri ve Sezgiseller (Heuristics)

Arayüz tasarımını ve testinde takip edilen çok yaygın diğer bir yaklaşım ise arayüzün uzmanlarca değerlendirilmesidir. Değişik uzman değerlendirmeleri arasında en yayğını, iyi bir arayüz tasarımının sahip olması gereken özellikleri veren sezgisellerin (*heuristics*) kullanılmasıdır. Sezgiseller genelde platform (hem donanım ve hem de yazılım olarak) bağımsızdır ve kullanılabılırlığı artırmaya yönelik-

tir. Günümüzde en çok kullanılan sezgisel rehberlerden birisi Jacob Nielsen tarafından önerilmiştir. Jacob Nielsen'in insan bilgisayar etkileşimi ve kullanılabılırlik mühendisliği tecrübelerine dayanarak oluşturduğu bir dizi değerlendirme ilkesi 1994'de yayımlanmıştır (<https://www.nngroup.com/articles/ten-usability-heuristics/>). Yayınlandığı günden itibaren on yıllar geçmiş olmasına rağmen temelde kullanılabılır bir ekran tasarımları için oluşturulan olan bu ilkeler, bugün oldukça yaygınlaşmış mobil uygulamalarda da güçlü bir değerlendirme aracı seçeneği olarak yerini korumaktadır.

Nielsen'in 10 Kullanılabilirlik Sezgiseli (*Nielsen's Ten Usability Heuristics*) adı verilen bu ilkelerde göre, kullanıcı arayüzleri tasarımında aşağıdaki noktalara dikkat edilmeli ve kullanılabılırlik bu çerçevede sorgulanmalıdır.

- **Sistem durumunun görünürlüğü:** Sistem, kullanıcıları o anki durumla ilgili olarak, ne olduğunu konusunda, sürekli bir şekilde ve uygun dönütler eşliğinde bilgilendirmelidir.
- **Sistem ile gerçek dünyadan eşleşmesi:** Sistem, kullanıcılar ile onların anlayabileceği dilde konuşmalı, kullanılan terimler, kelimeler ve kavramlar kullanıcıya tanıdık olmalıdır. Bilgilendirmelerin kullanıcı tarafından doğal ve mantıksal bir şekilde görülmesi sağlanmalıdır.

- **Kullanıcı kontrolü ve özgürlük:** Kullanıcılar sık sık sistem fonksiyonlarının seçiminde hata yaparlar ve bu istenmeyen durumdan çok detaya girmeden çıkmak için açıkça belirtilmiş bir “acil çıkış”a ihtiyaç duyarlar. Geri alma (undo) ve yeniden yapma (redo) seçenekleri bu amaçla sunulmalıdır.
- **Tutarlılık ve standartlar:** Kullanıcılar farklı kelimelerin, durumların ve eylemlerin aynı anlamda gelip gelmediğini düşünmemlidirler. Uygulama kendi içinde tutarlı olmalıdır.
- **Hataları önleme:** Kullanıcıların iyi bir hata mesajı ile karşılaşması yerine, dikkatli bir tasarım ile, hatanın oluşması önlenmelidir.
- **Hatırlamak yerine tanıma (ya da bellek yükünün en aza indirilmesi):** Nesneler, aktiviteler ve seçenekler görünür yapılmalıdır. Kullanıcı diyalogun bir bölümünden diğerine olan geçişlerde, önceki kısmı hatırlamak zorunda kalmamalıdır. Sistemin kullanımı için gerekli talimatlar görünür ve kolayca ulaşılabilir olmalıdır.
- **Esneklik ve kullanım verimliliği:** Acemi kullanıcılar tarafından görülemeyen hızlandırıcılar kullanılmalıdır. Genellikle sistemin deneyimli ve deneyimsiz kullanıcıları farklı kullanım davranışını gösterirler. Her iki grubuda da hitap etmek için, uzman kullanıcılar için etkileşimi hızlandıracı yöntemler kullanılmalıdır. Kullanıcılara sık kullandıkları fonksiyonları isteklerine göre ayarlayabilmeleri için imkanlar sunulmalıdır.
- **Estetik ve sade tasarım:** Diyaloglar alakasız ya da pek ihtiyaç duyulmayacak bilgiler içermemelidir. Bir diyaloga eklenen her ilave bilgi, daha gerekli bilgilerin görülmemesini engelleyip, karmaşa yaratır. Eğer bir bilgi ya da resim arayüzden kaldırıldığında kullanım açısından birsey farketmiyorsa, gereksiz demektir. Gereksiz ise kullanma prensibi izlenmelidir.
- **Kullanıcılar hata ile karşılaşmaları durumunda hatayı teşhis etmeleri, onaramları ve kurtulmaları olanağı tanınmalıdır:** Hata geri dönütleri, sade dilde (kodsuz) olmalı, sorunu açıklamalı ve yapıcısı bir çözüm önerisi sunmalıdır.
- **Yardım ve dokümantasyon:** Dokümantasyon olmadan sistemi kullanabilmek daha tercih edilir olmasına rağmen, kullanıcıya dokümantasyon ve yardım servisi sunmak gerekli olabilir. Yardım sisteminde gereken bilgiyi aramak kolay olmalı, yardım dokümanı kullanıcının görevine odaklı olmalı, çözümler listelerken somut adımları göstermeli ve çok uzun olmamalıdır.

Nielsen'in önerdiği sezgisellerin ayaryüz tasarımı dışında video oyunları, karmaşık sistemler gibi farklı alanlara uygulanması ile ilgili örnekler ve en son güncellemelere Nielsen Norman Group'un web sitesinden (<https://www.nngroup.com/>) ulaşılabilir.

Model Temelli Yaklaşım

Geçerleştirme yöntemi ve uygulama alanı açısından model temelli kullanılabilirlik testleri, sezgisel ve test yaklaşımına göre daha farklıdır. Bu yaklaşımda hedef, kullanıcıların biliş ve davranışların modellenmesidir. Modelleñerek oluþturulan bir uygulamada, davranışların modele uyup uymadığı ya da modelin nasıl daha da iyileştirilebileceği varolan model üzerinden değerlendirilebilir. Model temelli yaklaşım klasik insan faktörü çalışmalarındaki süreçlere benzer olsa da mühendislik ve bilişsel psikoloji geleneklerini takip ederek, prototip oluşturmadan önce kullanılabilirlik sorunları hakkında yordama ve doğrusal yineleme yapma olanağı tanır. Geleneksel bir insan faktörü çalışmasında görevler insanın fiziksel performansı açısından temsil edilir. Görev öğesinin tamamlanma süresi, hata oranları, iş yükü, görev zaman çizelgelerinin vb. Model için diğer önemli parametre ise bilişsel alandan gelmektedir. Yapay zeka veya bilişsel psikolojinin bilgi işleme yaklaşımının göre oluşturulmuş zihin modelleri kullanılarak performans simule edilebilir. Aşılması gereken sorun hem psikoloji hem de mühendislik yaklaşımlarını bir arada kullanmaktadır (Kieras, 2005).

Model temelli bir yaklaşım psikoloji ve fizyolojik sınırları her zaman dikkate almalıdır. İnsan davranışının, hafızasının veya algısının limitleri vardır. Tasarımcı modeli oluştururken her zaman bu limitleri dikkate almalı, sistem psikolojinin ihtiyaçlarına uyum göstirmelidir. Eğer model insan odaklı geliştirilirse, kullanılabilirlik çalışmaları tasarım hatalarına odaklanabilir. Evans (2017) kullanıcı psikolojisi ile kullanıcı deneyimi tasarnımını bir araya getirdiği kitabında dijital yeniliklerin ç-

ğalması için dikkat, algı, hafiza, tutum, motivasyon ve sosyal etkinin insanın limitlerini belirleyen psikolojik ve fizyolojik darboğazlardan uzak olması gerektiğini iddia eder. Ona göre, bir uygulamanın sunulduğu biçimde şekilde kabul edilmesi, uygulamanın kullanılması ve hatta başkalarına tavsiye edip edilmediği, sunulan dijital ürünün bu psikolojik sınırlar içinde kalıp kalmaması ile doğrudan ilgilidir.

Ancak, model tabanlı bu yaklaşım, her türlü sistemde kullanmaya çok uygun değildir. Model tabanlı yaklaşımın görevlerin çok iyi sınırlarla belirlendiği, belli akış içinde giden ve rutin uygulamalarda, kullanılması daha uygundur. Bu modelin

kullanım alanına en uygun örnek, büyük firmaların çağrı merkezlerinde kullanılan arayüzlerdir. Bilindiği gibi bankalar ya da telekom firmaları, müşterilerinin soru ve sorunlarına binlerce operatörün çalıştığı çağrı merkezleri ile yardımcı olurlar. Bu süreçte, operatör müşterinin bilgilerini sisteme girip ona en kısa sürede cevap vermek durumundadır. Bu durumda, sorunlu arayüz tasarımlı ile fazladan konulmuş bir ekrana gidip gelmek ya da gereksiz tuşlara basmak toplamda çok ciddi kaynak ve zaman kaybına neden olur. Dolayısıyla, bu tür sistemlerde tasarruf edilecek tek bir saniye bile hem maliyetlerin azaltılması hem de müşteri memnuniyetini yükseltme açısından önemli olmaktadır.



Araştırmalarla İlişkilendir

Dokunmatik Ekran Mobil Arayüz Tasarımı için Bilişsel Modelleme ve Kullanıcı Performans Analizinin Karşılaştırılması

Mobil cihazların artan kullanımı, İnsan Bilgisayar Etkileşimi (HCI) ve iletişim alanlarında önemli değişiklikler ve trendler getirdi. Akıllı telefon kullanımının artması, mobil uygulamaların ve mobil uygulama geliştiricilerinin sayısının artmasına neden olmuş, kullanıcılar ve mobil uygulamalar arasındaki artan etkileşim sonucunda bu tür uygulamalara olan talep artmıştır. Bu etkileşimi etkin ve verimli hale getirmek için yapılacak mobil kullanılabilirlik çalışmaları büyük önem taşımaktadır. Bu çalışmada bilişsel modelleme yöntemi olarak CogTool kullanılmış ve dokunmatik ekranlı cep telefonları için gerçek kullanıcıların performansları karşılaştırılarak tahminlerin doğruluğu araştırılmıştır.

Mobil uygulama olarak ise Turkcell Cüzdan uygulaması seçilmiştir. Uygulama geliştiricileri ile görüşülerek en çok kullanılan ve kritik olduğu düşünülen 8 görev belirlenmiştir. Uygulamayı daha önce kullanmamış 10 kullanıcı seçilerek kullanılabilirlik laboratuvarında kullanıcılarla test gerçekleştirilmişdir. CogTool deneyimli kullanıcı performans tahmin sonuçları verdiği için çalışma iki aşamalı olarak gerçekleştirılmıştır. Kullanıcılar ilk önce görevleri gerçekleştirerek uygulama üzerinde deneyim sahibi olmuş, kullanıcılarla yapılan ikinci testin sonuçları CogTool sonuçları ile karşılaştırılmıştır.

Analizler sonucunda, yetenekli kullanıcıların kullanılabilirlik testlerindeki performans sonucu ile CogTool'un tahmini performans sonuçlarının birbirine yakın değerlerde olduğu tespit edilmiştir. Ayrıca görevler adım bazında tek tek analiz edildiğinde, gerçek kullanıcılarından ve CogTool'dan elde edilen sonuçların bir görevin toplam süresi sonuçlarından daha fazla farklılık gösterdiği görülmüştür.

Kullanıcının tutuş şekli ve telefonun ekran boyutu, kullanıcının parmağının konumunu etkilemiştir Testlerde iPhone 3GS kullanıldığı için katılımcılar parmakları telefonun alt kısmında bulunan ev tuşuna yakınlık testlere başlamışlardır. Parmağın konumu, adımların ve görevlerin performans süresini doğrudan etkilemiş, kaydırma eyleminden, kullanıcılar ekranı yukarı itmiş ve parmaklarını önceki konuma yeniden konumlandırmışlardır. Dokunmatik ekranlı arayüz cihazları için CogTool uygulaması hala geliştirme aşamasındadır. Yalnızca arayüzün prototipleri tasarlanmış olsa bile, geliştirmenin çok erken aşamalarında CogTool kullanarak prototiplerin kullanılabilirliğini değerlendirmek mümkündür. Böylece uygulamada olabilecek olası kullanılabilirlik sorunları, uygulama geliştirme aşamasının en başında keşfedilebilir ve ele alınabilir. Bu nedenle, geliştiriciler, UI'nin gerçek uygulamasından önce tasarım fikirleriyle CogTool'u kullanabilirler.

Daha sonraki çalışmalar için, katılımcıların uygulamayı daha uzun süre kullanma ve keşfetme fırsatına sahip olmaları için çalışmadan önce eğitim verilmesi önerilebilir İlkinci olarak, bu çalışmada kullanıcı testleri bir kullanılabilirlik laboratuvarında yapıldığından, akıllı telefonların doğası gereği, kullanıcılar hareket halindeyken telefonlarını kullandıkları için kullanıcıların mobil olduğu alanda daha ileri çalışmaların yapılması önerilir.

Kaynak: Nihan Ocak & Kursat Cagiltay (2017) Comparison of Cognitive Modeling and User Performance Analysis for Touch Screen Mobile Interface Design, International Journal of Human–Computer Interaction, 33:8, 633-641, DOI: 10.1080/10447318.2016.1274160



CogTool'a <https://www.cogtool.org> adresinden ulaşabilirsiniz.



1 Kullanılabilirliğin temel bileşenlerini ve ölçüm yöntemlerini tanımlayabilme

Araştır 1

Kullanılabilirlik ile kullanıcı deneyimi yaklaşımını temel bileşenler açısından karşılaştırınız.

Öğrenme Çıktısı

İlişkilendir

Kullanılabilirlik ölçüm yöntemlerinin diğer bilim alanlardaki araştırmalardan farklı yönleri nelerdir?

Anlat/Paylaş

Telefonunuzda yüklü olan bir programı yükledikten sonra ilk kullanımda zorluk çektiniz mi; nasıl?

MOBİL UYGULAMA TASARIMI

Önceki kısımlarda kullanılabilirliğin tanımı ile birlikte, arayüz tasarımını için neden önemli olduğu incelendi. Tasarımcılar için kullanılabilirlik çalışmalarının amacı bir ürünün kullanıcılar tarafından nasıl değerlendirildiğini görmektir. Tasarımcılar ve geliştiriciler ürünü kullanacak hedef kitle olmadıkından, kullanılabilirlik testleri tasarımcılar bir üründe nelerin bekleniği gibi çalışmadığını gösterir. Daha günlük bir dil ile ifade etmek gerekirse kullanılabilirlik uzmanlarının görevi "uygulamanın geliştirilmesi aşamasında neler olmamış, neler yanlış gitmiş veya neler atlanmış" ise onları geliştiriciler ve tasarımcılar ile paylaşmaktadır. Tasarımcılar ve geliştiriciler de kullanılabilirlik testlerinden elde edilen bulgulara göre, üründe veya arayüzde iyileştirme yapabilirler.



dikkat

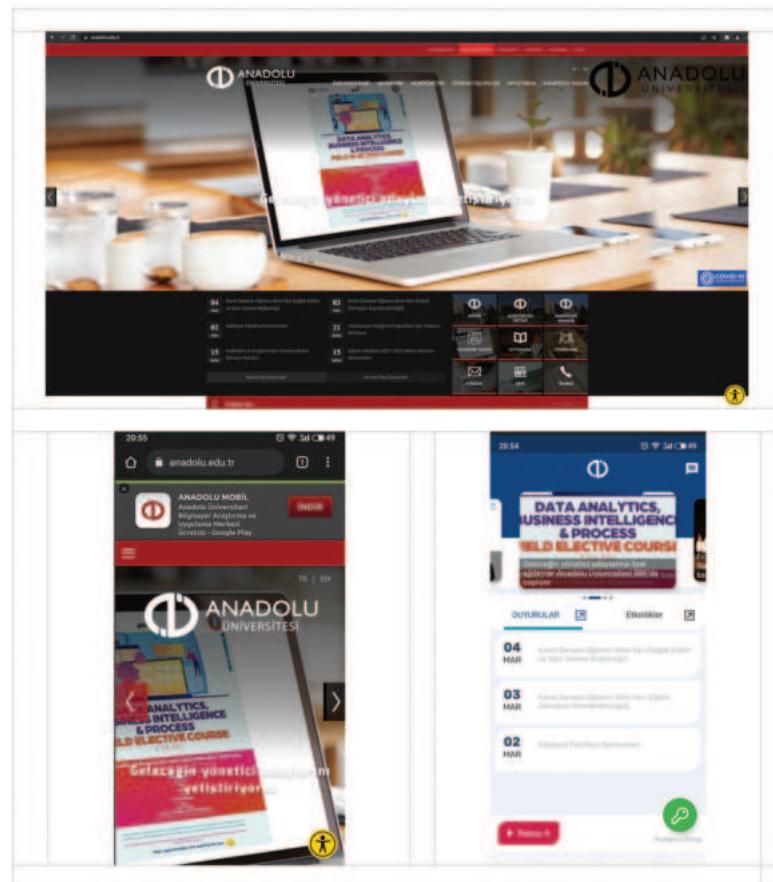
Arayüz tasarımını ve kullanılabilirlik iç içe geçmiştir. Görsel tasarım arayüzün anlaşılmasına ve kullanmasına yardımcı olur. Kullanıcı için ise arayüz ile kurduğu etkileşim, uygulamadan edindiği deneyimdir. Bir arayüz tasarımının, kullanılabilirlik çalışmalarından başarı elde etmesi için önce güçlü grafik arka plana sahip olması gereklidir. [mizanpj (denge, yakınlık, hizalama) hiyerarşi, beyaz alan, hareket, renk ve kontrast] arayüzün grafik tasarımının konusunda temel kavamlardır ve kullanılabilirlik uzmanlarının görsel tasarımcılar ile iletişim kuracak düzeyde de olsa bu konular hakkında fikirleri olmalıdır.

Arayüz tasarımının iyileştirilmesi için tasarımcıların veya yazılımcıların mobil uygulamalar ile ilgili geliştirme adımlarını platformlardan başlayıp arayüz bileşenlerine; kodlama aşamasından ileri düzey uygulama seçeneklerine kadar dikkatle incelemeleri gereklidir. Son on yılda mobil cihazların yaygınlaşması ile birlikte, insan bilgisayar etkileşiminin lokomotifi olan web arayüzleri, bugün küçülen ekranlar, değişen kullanım içerikleri ve yeni etkileşim biçimleri nedeni ile varlıklarını sürdürmekteki değişimlerin önemini vurgulamaktadır. Bu bölümde, web arayüzünden mobil uygulamalara geçiş sürecinde, mobil uygulama tasarımında dikkat etmesi gereken genel ilkelerde değişikliklerin olabileceğini, eğer varsa bu ilkelerin diğer geleneksel insan bilgisayar etkileşimi (taşınabilir dizüstü bilgisayarlar dahil) cihazlarındaki arayüzlerden nasıl farklılığından alıcı çizilecektir.

Mobil Uyumlu (Responsive) Tasarım

Mobil uygulama tasarım ilkelerinden bahsetmeden önce söz edilmesi gerekliliğin yaklaşım “responsive web tasarım”ı olmalıdır. Responsive (duyarlı, uyumlu ve esnek anlamına gelse de metin içindeki anlatıma “ekran uyumlu-mobil uyumlu” ifadesi daha yakındır) tasarım hem geleneksel masaüstü/dizüstü bilgisayarlarla hem de mobil cihazlarda hangi platformda çalıştığını gözetmeksizin kullanıcıların ihtiyaçlarını karşılayan web siteleri için kullanılmaktadır. Responsive web siteleri her ekranda görüntülenebilen web sitesi anlamına gelmeyip görüntüleneceği ekran ve platformu tanıarak kendisini o ekranın özelliklerine göre ayarlayabilen tasarımını ifade eder.

Aşağıdaki örnekte (Görsel 2.2) Anadolu Üniversitesi web sitesinin bilgisayar ve mobil bir cihazdaki ana sayfası ile mobil uygulamasının açılış ekranı görülmektedir. Mobil web sitesinde menü, hamburger menü haline dönüşmüş, resim ekran çözünürlüğü nedeniyle kesilmiştir. Yine mobil web sitesi, aynı zamanda kullanıcıyı ANADOLU MOBİL uygulamasına yönlendirmektedir ki bu yönlendirme mobil kullanıcılar yapmak istedikleri işlemler için farklı bir alternatif daha olduğunu hatırlatmaktadır. Web sayfasının her ikisinde de sağ alt köşedeki erişilebilirlik butonu varlığını korumuştur. ANADOLU MOBİL uygulamasında ise sadece üniversitenin ismi çıkarılmış, amblem kullanıcılarla hangi uygulamada olduklarını hatırlatmak amacıyla bırakılmıştır. Haber ile ilgili resim daha da kesilmiş, erişilebilirlik butonu yerini yüzen kullanıcı giriş anahtarına terketmiştir. Bu değişiklik uygulamayı yükleyen kullanıcıların kullanım amaçlarına daha uygun bir tercihtir.



Görsel 2.2 Anadolu Üniversitesi Web Sitesi ve Mobil Uygulama Örnekleri
(2022, Mart)

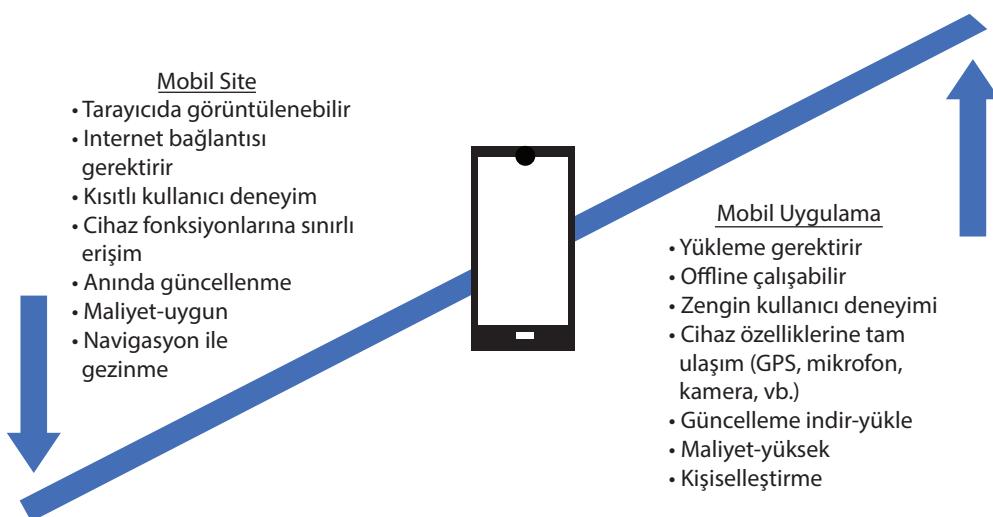
Mobil Site ve Mobil Uygulama Kavramları

Web sitelerinin mobil cihazlar geliştirilmeden önceki varlığı mobil uygulamaların yaygınlaşması sonrası ikinci plana itilmiş görünse de responsive tasarımın birçok cihaz ve platforma kolaylıkla sağladığı uyum, sitelerin mobil cihazlarda da varlık göstermeye başlamasına yol açmıştır. Responsive tasarım aynı zamanda mobil web sitelerine uygulamalardan işlevsel olarak farklılaşma olağanı tanımlamıştır. Bugün bir görevin yerine getirilmesi için genelde uygulamalar tercih edilirken, mobil web siteleri daha çok bilgi sunmaya ve/veya almaya yönelik işlemlerde tercih edilmektedir. Mobil web sitelerinde hedef; kullanıcıların sunulan bilgiyi hemen almaları, bu bilgiden yararlanmaları ve gerekiyorsa web sitesinde kolaylıkla hedeflerine yönelik yönlendirme seçeneklerini bulabilmeleridir.

Mobil web siteleri ve mobil uygulamalar bugün birlikte varlıklarını sürdürdüğü için tasarımçılardan beklenen karar, geliştirme aşamasında hedeflenen ürünün mobil cihazlar için tarayıcıda görüntülenecek bir web sitesi mi olacağı veya yaygın platformlar için yepyeni bir uygulama olarkı mı kullanıcıların hizmetine sunulacağıdır. Mobil bir web sitesi ile uygulama arayüz açısından birbirine benzetilebilir olsa da geliştiriciler tarafından değerlendirilmesi gereken avantajları ve dezavantajları mevcuttur. Bir uygulama veya web sitesi geliştirmeden önce bu farklılıkların bilinmesi zaman ve para kaybını engelleyebilir. Bu konudaki temel farklara dikkat çeken en erken yayınlardan biri,

2012 yılında Nielsen tarafından mobil siteler ve mobil uygulamalar ile arasındaki farkların sunulduğu rapordur (<https://www.nngroup.com/articles/mobile-sites-vs-apps-strategy-shift/>).

Nielsen'e göre mobil cihazlar için geliştirme yapmak isteyenlerin düşünmeleri gereken kriter maliyet olmalıdır. Eğer maliyet karşılanabiliyorsa, testlerde daha iyi performans gösteren uygulamalar tercih edilmelidir. Çünkü uygulamalar mobil cihazlar için geliştirilirken, web siteleri mobil cihazlar için optimize edilmektedir. Nielsen gelecekte mobil sitelerin, uygulamalar karşısında daha yaygın olacağını öngörürken, uygulamaların platform bağımlı olduğunu, uygulamalarda kullanıcı deneyiminin ekran büyülüğüne bağlı olduğunu ve tabletler ile telefonların kullanım amaçlarındaki fark nedeniyle bir uygulamanın her ikisinde de aynı performansı göstermediğini ileri sürmüş, bu nedenle ileride mobil web sitelerinin daha avantajlı konuma geleceğini ifade etmiştir (Nielsen, 2012). Şekil 2.4 mobil siteler ve mobil uygulamaların arasındaki temel farkları göstermektedir. Örneğin mobil platformlar için bir web arayüzü, uygulamadan daha kolay ve ucuz geliştirilebilir, ayrıca mobil cihaza yüklenmesi gerekmeli gibi güncellenmesi de kolaydır; diğer yandan bir uygulama da mobil web arayüzünden farklı olarak kişisel tercihlere göre özelleştirilebilir, offline çalışabilir ve gerektiğinde oyunlarda olduğu gibi farklı etkileşim biçimlerine izin verir. Mevcut kitabın içeriği mobil uygulama geliştirme üzerine olduğu için takip eden kısımlarda mobil uygulamalar için temel arayüz tasarımları ilkeleri açıklanacaktır.



Şekil 2.4 Mobil Web Siteleri ile Mobil Uygulamalar Arasındaki Temel Farklar

DİKEY EKRAN - YATAY EKRAN

Mobil cihazlarda geleneksel yatay ekranların yerini görece daha küçük ve dikey kullanılan ekranlara bırakması, ekranlarda yer alabilecek arayüz bileşenlerin sayısını ve yerleşimini etkilemiştir. Mobil cihazların günlük yaşamı istila etmesi masaüstü bilgisayarlardaki klavye ve farenin yerini dokunmatik ekranlara ve parmaklarımıza bırakmasına neden olmuştur.

Nielsen Norman Group tarafından 2006 yılında göz izleme tekniğini kullanarak yapılan orjinal çalışma kullanıcıların web sayfalarını genellikle iki

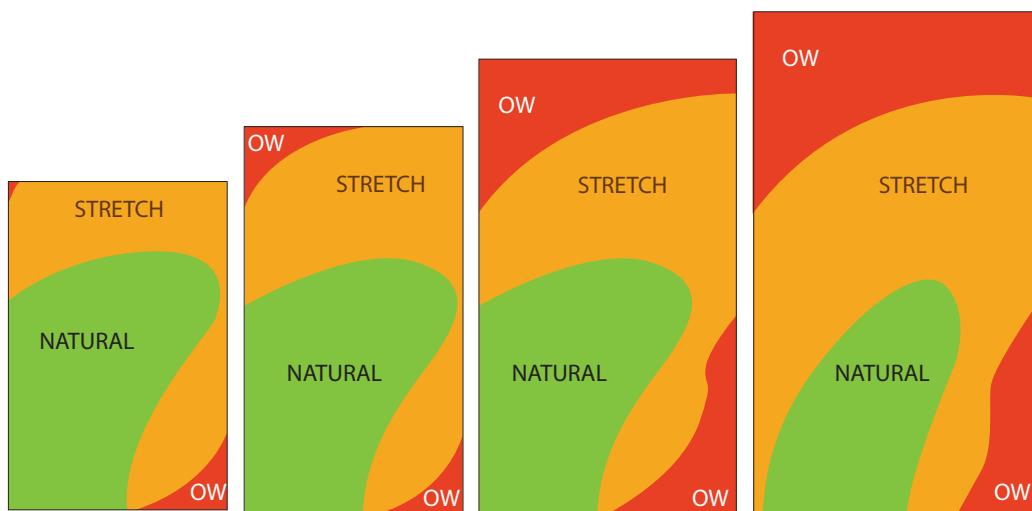
yatay ve bir dikey şeritten oluşan F harfi şeklinde okuduklarını göstermiştir (Şekil 2.5). Bu çalışmayı takip eden yıllarda akıllı telefonlar ile birlikte dikey olarak kullanılmaya başlanan ekranlar için bu çalışmadan çıkarılacak ders, kullanıcıların ekranı piksel-uyumlu takip etmedikleri, aksine genel hatları izledikleridir. Aynı konuda 2017 yılında Pernice tarafından Nielsen group sayfasında yayınlanan bir makalede ise F örüntüsünün tasarımcılar tarafından bazen yanlış yorumlandığı, F örüntüsünün pozitif olarak algılandığı, -negatif- kaçınılmaması gereken bir örüntü olduğunun dikkate alınmadığı belirtilmiştir.



Eyetracking by Nielsen Norman Group nngroup.com NN/g

Şekil 2.5 Göz İzleme Araştırmaları, Kullanıcıların Web Sayfalarını Genellikle İki Yatay ve Bir Dikey Şeritten Oluşan F Harfi Şeklinde Okuduklarını Göstermiştir.

Mobil uygulamalarda gözün ekranın dışında, dokunmatik özelliklerden dolayı parmakların ekran ile olan etkileşimini de kontrol etmek gerekir. Mobil cihazlarda, özellikle akıllı telefonlarda tek el kullanımında ekran ile etkileşim başparmak ile yapıldığı için başparmağın ekran üzerindeki bölgelere ulaşım kolaylığı, ekranda etkileşimli öğelerin yerleşimine doğrudan etki yapmaktadır. Son yıllarda akıllı telefonların ekran büyülüklüğünün de 6 inç üzerine çıktıığı düşünülürse cihaz kullanımında ilgili navigation veya işlem menülerinin başparmağın ulaşabilecegi bölgelerde olması önemlidir. Aşağıdaki Şekil 2.6 ekran büyülükleri arttıkça başparmak ile rahat işlem yapılabilecek bölgelerin ekran üzerindeki payının nasıl azaldığı gösterilmektedir. Steven Hoober 1333 kullanıcı ile yaptığı çalışmada kullanıcıların telefonlarını kullanırken nasıl tuttuklarını araştırılmıştır (<https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php>). Sonuçlar kullanıcıların %49'unun tek el ile kullandıklarını gösterirken, diğer yarısının iki elini kullandığı, fakat ikinci el kullanımının işaret parmağı veya başparmak arasında değişiklik gösterdiğini bulmuştur. Bu bulgular göstermektedir ki, herhangi bir tasarım yapmadan önce kullanıcıların mobil cihazları kullanırken nasıl tuttukları, kullanım tercihlerinin uygulamalara göre nasıl değiştiği araştırılmalıdır; veriler, görevler ile kullanım türleri arasındaki doğru korelasyona ulaşmayı sağlar, sonuçta yanlış varsayımlar ile kullanımına uygun olmayacak etkileşim biçimleri tercih edilmez.



Şekil 2.6 Akıllı Telefonlarda Ekran Büyüdükle Başparmak İle Ulaşılabilen Payı Azalmaktadır.

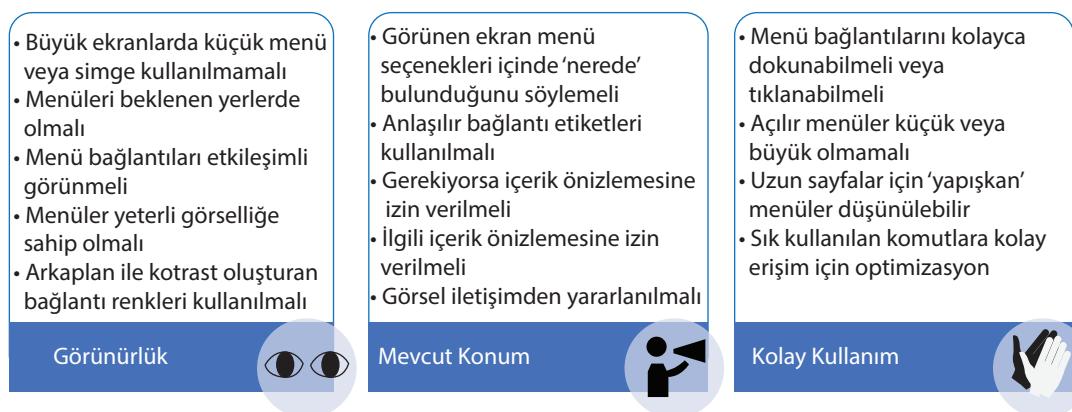
Mobil Uygulamalarda Menü Yapısı

Kullanıcıların uygulama içinde kolay gezmesini sağlamak her tasarımcı için yüksek bir öncelik olmalıdır. Menülere aşağı yukarı her web sitesinde veya uygulamada karşılaşıyor olsak da; sıklıkla anlaşılması, yönetilmesi zor menüler ile karşılaşıyoruz. Menü tasarımını kritik yapan unsur nedir? İçerik.

Menüler kullanıcıların uygulama içinde gezinmesini, A noktasından B noktasına gitmesini sağlamak üzere tasarlanırlar. Amaç kullanıcıların en kısa sürede gitmesi değil aynı zamanda kolay gitmesini de sağlamaktır. Menülerin uygulamanın yapısal iskeleti olarak düşünülebilir. Tasarımcılar için çeşitli menü olasılıkları vardır. Ekranın üst

veya alt bölgelerine yerleştirilen birinci veya ikinci menü öğeleri dışında, hamburger menü, kart menüler, sekmeler veya harekete duyarlı menüler gibi seçenekler kullanılabilir.

Nielsen kullanılabilecek menüleri için yöneleri ve rehberleri izlemeyi güvenilir bir yol olarak önerir. Bazen yenilikçi veya ilginç menüler ile etkileşim tasarımcıyı cezbetse de kullanıcı açısından beklenen etki oluşmayabilir. Bu nedenle kullanıcılar harika görünen özel efektler ile etkilemek yerine, tanıdıkları bir menü kullanarak kolayca erişilebilen bir içerik sunmak sonuçta daha fazla memnuniyet sağlayabilir. Bu amaca ulaşmak için Nielsen grubu tarafından Whitenton tarafından derlenen 15 kriter aşağıda verilmektedir (Şekil 2.7) (<https://www.nngroup.com/articles/menu-design/>).



Şekil 2.7 Nielsen'in Menü Tasarımı İçin Önerdiği Rehber Liste

Navigasyon/Gezinme

İyi bir uygulamanın temel niteliklerinden biri kusursuz gezinmeye (navigasyon) sahip olmasıdır. Kullanıcıyı bir noktadan diğerine taşıyan bir araç olan navigasyonun yüzey alanı sınırlı olan bir cihazda kullanımını web sitelerinden farklılıklar gösterir. Günümüzde mobil cihazlar için çeşitli çözümler mevcut olsa da hiçbir sorunlardan muaf değildir. Bu bölümde güncel uygulamalar için kullanılan bazı navigasyon biçimleri, avantajları ve olası sorunlarına değinilecektir (Babich, 2020).

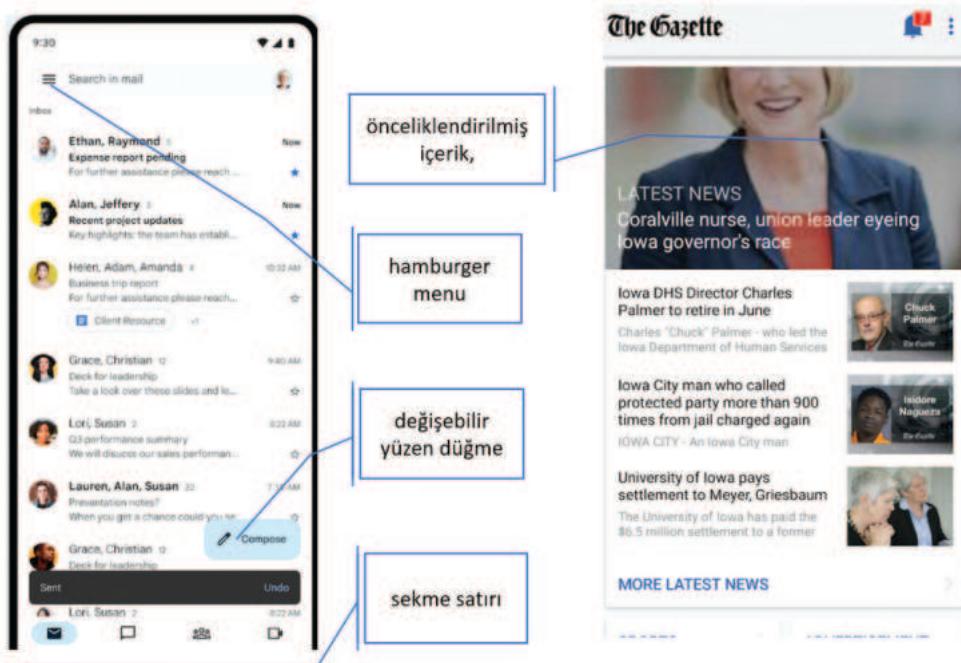
Hamburger menüler uygulamalarda sıkılıkla karşımıza çıkmaktadır. Yukarıda belirtildiği gibi dokunmatik ekranlarda çok da verimli kullanılamayan kenar bölgeler için popüler bir tercihtir. Hamburger menüler içeriği saklar ve gerektiğinde kullanıcı tarafından etkinleştirilebilir. Hamburger menülerin ekranada kapladığı yer ve konum açısından keşfedilmeleri problem olabilir. Fakat bu özellik anahtar olmayan ikincil navigasyon işlemleri için aynı zamanda avantaj sağlayabilir. Hamburger menü birçok seçeneği içinde gizlediği için kullanıcıya aynı zamanda temiz bir tasarım sunarken, belirli bir işleme ulaşmak için fazladan adım gerektirmektedir.

Uygulama arayüzünde bazı kullanıcılar öncelendirmek istenirse, içeriğini gizleyen hamburger menüler yerine her zaman ana ekranada yer alacak sekülerler (tabs) tercih edilebilir. Uygulamada atılmak istenen adının görünürlüğü kullanıcı memnuniyetini artıracağı için fonksiyonların önceliklendirilip uygulamanın odağını artırılması menü ile gezmenin olası karmaşıklığını giderecektir. Hedefe doğrudan ulaşmayı sağlayan sekülerler, masaüstü tasarımdan ödünç alınmıştır. En fazla beş sekmeye kadar verimli sonuç alınan mobil tasarımda, kullanıcı hangi işlem bölgesinde olduğunu kolaylıkla keşfedebilir. Sekme kullanımın sıkıntısı ise sayı olarak limitli olması ekranın üst kısmında kullanıldığından başparmak erişim bölgesi dışında olmasıdır.

Mobil uygulamalar için tercih edilebilecek diğer bir menü biçimini ise gezinme için tam ekranın kullanılmasıdır. Diğer mobil gezinme modellerinde, navigasyon sistemlerinin kapladığı alanı en azı indirmeye çalışırken, tam sayfa gezinme yaklaşımı ana sayfayı yalnızca gezinmeye ayırrı. Bu model, özellikle kullanıcılar tek bir oturumda gezinmek ve yalnızca bir konu ile sınırlanma eğiliminde olduklarıda iyi çalışır. Kullanıcıları ana sayfalarından ayrıntı sayfalarına yönlendirir. Tam ekran gezinme uygulamada sadelik ve tutarlılık sağlanırken büyük tutarda bilgiyi düzenli halde kullanıcıya sunabilir. Bu tür bir menünün dezavantajı ise gezinme seçenekleri sunulabilecek farklı içeriğin görüntülenme olasılığını sınırlar.

Eğer tasarımcı gezinmeyi değil de belirli bir içeriği kullanıcıların dikkatine sunmak için öncelendirmek istiyorsa responsive tasarımdan esinlenen öncelik+ yaklaşımını kullanabilir. En önemli gezinme öğeleri olarak değerlendirilenler ana sayfada önceliklendirir ve diğerleri “daha fazla” düğmesinin arkasına gizlenir. Bu model, yoğun ve çeşitli içeriğe sahip uygulamalar veya web siteleri (örneğin bir haber sitesi) için kullanışlı bir çözüm sunar.

Uygulamalarda menü kullanımı açısından arayüz üzerinde yüzen bir düğme de kullanılabilir. Android kullanıcılarının aşina olduğu bu tasarım uygulamanın arayüzünde süzülen bir düğme kullanıcıya gerçekleştirmek istediği eylem açısından destek sağlamaktadır. Örneğin, Gmail uygulaması ekran üzerinde yer alan “compose” düğmesi ekran kullanılırken küçülmekte, sabit ekranda tekrar belirmektedir. Eylem düğmesi, kullanıcıların çoğu zaman belirli bir eylemi gerçekleştireceği varsayımlına dayandığından Gmail uygulamasında e-posta oluşturma üzeri düğme olarak seçilmiştir. Üzer düğmeler kullanıcıların yapmasını istediğiniz en önemli eylemi ekranда az bir yer kaplayarak önceliklendirmenin bir yoludur. Fakat bu düğmelerin içeriğin üzerini kapattığını, farklı içeriklerde farklı anlamaya gelebileceğini ve dikkat dağıtabileceğini bilerek sınırlı ve ekran başına sadece bir adet kullanılması tavsiye edilir (Şekil 2.8).



Şekil 2.8 Günümüzde Mobil Uygulamalarda Kullanılan En Yaygın Menü Biçimlerine Örnekler

Mobil cihazların, masaüstü çözümlerden yapısal farklarından biri arayüz tasarımda el mimiklerinin (gesture) kullanılabilmesidir. iPhone'nun piyasaya çıkması ile birlikte hızla baskın hale gelen dokunmatik ekranların ayrılmaz bir parçası olan arayüzü parmak veya cihaz hareketleri ile kontrol edebilme seçeneği tasarımcılar arasında kısa sürede popüler olup deneyel tasarımlara olanak sağlamıştır. Bugün bir uygulamanın başarısı parmak hareketlerinin kullanıcıya ne kadar iyi deneyim sunduğuna bağlıdır. Uygulamalarda geçirilen zamanın çoğu içerik ile ilgili olduğu için mimikler içeriği kontrol ettiği ölçüde kullanıcılar tarafından kabul görmektedir. Arayüzde menü kullanımını kaldırın mimikler, kullanıcılar açısından doğal ve yerleşik olarak hissedilmelidir. Mimiklerin kullanımında dikkat edilmesi gereken en önemli nokta, uygulama mimiğin yerleşik algısı açısından yeni ve farklı bir deneyim veya işlev sunuyorsa, muhtemelen kullanıcılar tarafından kabul görmeyeceğidir.

Dokunmatik Yüzeylerde Tasarım

Geleneksel olarak masaüstü ekranların yatay kullanımı, küçülen mobil cihaz ekranları ile dikeye dönüştürülmüştür. Mobil cihazların çözünürlükleri ve yüzey alanları masaüstü ekranlardan daha azdır. Bu nedenle mobil cihazlarda ekranда aynı anda sunu-

lacak için miktarına dikkat edilmeli, mobil cihazların kolaylıkla kullanılabilen kaydırma ve zoom fonksyonlarına imkân sağlanmalıdır.

Mobil cihazların kullanıldığı bağlam, ev veya ofis bilgisayarlarının kullanıldığı bağlamdan çok daha genişir. Sokakta, otobüste, sınıfta hatta bu satırları okurken bile mobil bir cihaz kullanıyor olabilirsiniz. Mobil cihazların kullanıldığı bağlamın geniş olması nedeniyle mobil uygulamalarda çoklu-görev etkisine dikkat etmek gereklidir (Hodent, 2018). Mobil cihazların kullanıldığı ortamların çeşitliliği nedeniyle mobil cihazlarda çoklu görev, sadece ekranın bölünerek birden fazla uygulamanın aynı anda çalıştırılması olarak algılanmamalıdır. Mobil cihazlarda dikkatin uygulama ile mobil cihazdan tamamen bağımsız diğer bir öğede de olabileceği hatırlanmalıdır. Örneğin, telefonda sosyal medyayı kontrol ederken bir yandan da televizyon seyredilebilir veya ofiste günlük raporlar kontrol edilirken aynı zamanda telefonda bir video açık olabilir vb. Bu noktada bilinmesi gereken dikkat gerektiren iki işlemin aynı anda yapılmasının genelde mümkün olmadığıdır.

Sınırlı ekran yüzeyi ve çoklu görevler için ihtiyaç duyulan zihinsel eforun yüksekliği nedeniyle mobil cihazlarda çoklu-görev uygulamaları yaygınlaşmamıştır. Diğer yandan mobil cihazların bir-

den fazla uygulamayı aynı anda çalıştırması veya ekranda göstermesi, kullanıcıların çoklu görevleri gerçekleştirdiği anlamına gelmez. Kullanıcılar daha kolay tanımlanan geçiş hareketleri veya kısa yolları nedeniyle uygulamalar arasında daha hızlı geçiş yapabilirler.

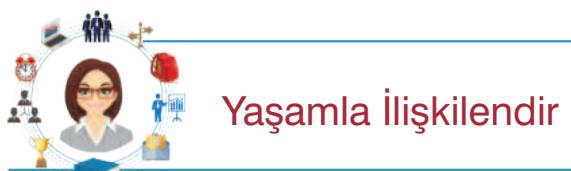
Aşağıda “Mobil Arayüzler için Dokunmatik Tasarım” kitabının yazarı Steven Hoober'in (2021) bazı önerileri yer almaktadır.

- Mobil cihazlarda ekranın merkezi en iyi ve en hızlı şekilde okunduğu ve etkileşim bu bölgelerde daha kolay olduğu için, anahtar bilgiler ortadaki büyük kaydırma alanında yer almmalıdır.
- Kaydırma (scroll) hareketi genel kabul olarak kolaydır ve çok yaygın biçimde kullanılmaktadır. Kullanıcı daha fazla içerik olduğun anladığı zaman, bu içeriği keşfetmek için kaydırma hareketini fark etmeden uygulayacaktır.
- Ekran kullanımı sırasında istenmeyen dokunuşların olması kaçınılmazdır. Bu nedenle kullanıcıların yapmasını istediğiniz davranışların, istemediklerinizden uzak olması tercih edilmelidir. Örneğin çıkış işlemi için masaüstü sistemlerde “emin misiniz?” gibi bir soru koruma sağlar iken, mobil sistemlerde çıkış işlemi koruma yerine geri alma seçeneğinin olması daha kullanışlıdır.
- Mobil sistemler, masaüstü bilgisayarlarda olduğu gibi görev odaklı değildir. Gerçek dünyada mobil bir uygulama işte, sokakta, evde kullanılabilir; kullanıcıların dikkatinin dağılması çok kolaydır. Bu nedenle mobil sistemlerde bildirimlerin zaman aşımına uğratmamak daha akılcı bir seçenekdir. Eğer zaman aşımına uğrayacaksa eylemi gerçekleştirmek için belirli bir zaman tanımlayıp, bu zamanı kullanıcıya sunmak gereklidir.
- Mobil dokunmatik cihazların kenarları ve köşeleri dokunmak için, fare imleci gibi tercih edilen bölgeler değildir. Hızlı ulaşım için menüler masaüstü sistemlerde kenarlara yakınen, mobil sistemlerde daha az kullanılan bağlantıları ve menüleri gizlemek için çok uygundurlar. Fakat buraya yerleştirilen, örneğin hamburger bir menü kolayca dokunabilmeyi sağlayacak kadar büyük olmalıdır.
- Kullanıcılara sunulacak içerik arayüze entegre edilmelidir. Mobil arayüz tasarımda kullanıcıların dikkatini çekmek için açılan pencerelerden uzak durulmalı, gerekiyorsa akordiyon menu veya çekmeler (drawer) kullanılmalıdır.
- Mobil bir arayüzde kullanıcıya seçme şansı verilmelidir. Seçimleri ile ilgili yeterli bilgi sunulmalı, bu seçimi yapıp yapmamak konusunda bir an düşünmeleri sağlanmalıdır. Yanlış bir işlem yapmaktansa, yapılacak işlem konusunda gecikmek daha iyidir.
- Düğme gibi etkileşimli öğeleri, ekranda beklenen yerlere ve gerektiği kadar büyütükte yerleştirin. Eğer bir etiket veya yazı kullanılması gerekiyorsa açık ve seçilebilir olmalıdır.
- Dairesel menüler teorik olarak en verimli sonuçlara ulaştırması gerekiyken beklenmedik bir yerleşim olduğu için kullanımda teorik değerlerini kaybeder.

Dikkatli okuyucular, Hoober'in listelediği tavsiyelerin günlük olarak kullandığımız mobil uygulamalar için geçerli olduğunu, fakat mobil oyunlar için farklı dinamiklerin iş başında olabileceğini fark etmişlerdir. Mobil uygulama geliştirmek isteyenler oyunları sadece keyifli zaman geçirmek için değil aynı zamanda incelenmesi gereken ürünler olarak ele almalıdır. Yaşamla ilişkilendir kutusu oyuncu deneyiminin kullanıcı deneyiminden hangi açılardan farklılığı konusunda Alistair Greo tarafından 2017 yılında UXPA konferansında yapılan sunumun ana hatlarını içermektedir. Oyun geliştirme konusunda uzmanlaşmak isteyenler genel arayüz tasarımları ve kullanılabilirlik ilkelerinin mobil oyunlar ile bağlantılı olduğunu, fakat birebir transfer edilemediğini bilmelidirler. Örneğin oyunu çok kolaylaştırmak, oyuncu açısından olumlu bir deneyim midir; kaçınılması gereken mi?



Oyun geliştirme ve animasyon teknolojileri alanlarına ilgisi olan ekiplere çeşitli destekler sağlayarak sektörün güçlenmesini ve sektör paydaşlarının sahiləli bir ekosistem oluşturmasını sağlamayı amaçlayan merkez... atom.org.tr



Yaşamla İlişkilendir

Kullanıcı deneyimi mi, oyuncu deneyimi mi?

Aşağıdaki bilgiler Alistair Greo'nun UXPA konferansında yaptığı sunumdan derlenmiştir.

Ana akım kullanıcı deneyiminden oyuncu deneyimine geçildiğinde iki özellik kaybolur, dört özellik kazanılır.

Kaybedilenler;

- Oyuncu deneyimi çalışmaları daha yendir.
- Oyun geliştiren tasarımcılar, kullanıcı deneyimi çalışmalarına ihtiyaç duymaya yaklaşık kadar yeteneklidirler yaklaşımı, kullanıcı deneyimi çalışmalarının zaman kaybı olarak değerlendirilmesinde neden olmuştur.
- Kısıtlı bütçeler nedeniyle kullanıcı deneyimi çalışmaları göz arı edilmiştir.
- Oyuncu deneyimi için kullanılabilcek araştırma yöntemleri, ana akım yöntemlerinden daha azdır.
- Oyuncu deneyimi araştırmaları gizliliğe daha fazla önem verir. Oyun geliştiren firmalar oyunları veya geliştirme süreçleri ile ilgili bilgi birikimleri rakipleri ile paylaşma konusunda isteksizdirler. Hatta büyük firmalar, kendi çalışanları arasındaki bilgi paylaşımına bile sınır getirirler.

Kazanılanlar;

- Oyuncu deneyimi çalışmaları daha ilginç sorular ile uğraşır. Oyun araştırmalarında mevcut yöntemlerin ya çalışmanın amacına

göre güncellenmesi ya da yeni yöntemlerin bulunması gerekmektedir.

- Oyuncu deneyimi duyguları çalışır.
- Oyuncular ve geliştiricilerin tutkusu yüksektir. Oyun geliştiriciler yetenekleri ile başka sektörlerde daha fazla gelir edebilecek olsalar da, oyun geliştirmek onlar için her zaman birinci uğraştır.
- Oyunlar her zaman en son teknoloji ürünleri kullanır, tabii ki savunma sektöründen sonra; stylus, hareket kontrolü, ikinci ekranlar, dokunmatik yüzeyler, sanal gerçeklik, göz hareketleri izleme, biyometrik. Menülerde vücut hareketleri ile gezinme gibi birçok tasarım yeniliği oyunlar için geliştirilmiştir.

Günümüzde oyuncu araştırmaları için güncel olan başlıklar ise aşağıdaki gibidir;

- Duyguları ve oyuna dahil olma nasıl ölçülür?
- Sanal gerçeklik uygulamalarında tasarım ve değerlendirme nasıl olmalıdır?
- Bir oyunun senaryosunu nasıl test ederiz?
- Oyuncu deneyimi nasıl kişiselleştirilir?

Sonuç olarak, kullanıcı deneyimini araştırmak için geliştirilen yöntemler oyunlarda da kullanılabilir olsa da ulaşılan çözümler yeni olacaktır.

Kaynak: Alistair Greo (2017) 'Mainstream' UX and Games UX'

<https://www.youtube.com/watch?v=XxZeFJpqIDY> Erişim tarihi: 2022

Öğrenme Çıktısı

2 Mobil uygulama tasarımının genel özelliklerini açıklayabilme

Araştır 2

Sanal ve artırılmış gerçeklik teknolojileri mobil uygulama tasarımını nasıl etkileyecektir?

İlişkilendir

En sık kullandığınız mobil uygulamanın arayüzünde metin içinde belirtilen özelliklerden hangileri kullanılmıştır?

Anlat/Paylaş

Cep telefonunuzda yüklü olan ve yakın zamanda güncellenmiş bir uygulamada nelerin değiştigini arkadaşınızla tartışınız.

EVRENSEL TASARIM VE ERİŞİLEBİLİRLİK

1.000.000.000.-, dünya nüfusunun yaklaşık %15'i. Dünya Sağlık Örgütü'ne göre engelli nüfusunun genel nüfusa oranı ve bu sayı hergün Dünya nüfusu ile birlikte artıyor. Bu bölüme kadar tartışılan kullanılabılırlik kavramları ile ve arayüz tasarım ilkeleri standartlarda ve tasarım rehberlerinde yer aldığı biçimde ele alınmıştır. Standartlar ve rehberler tasarımcılar için vazgeçilmez olmasına rağmen kapsayıcı veya evrensel kullanılabılırlik sonuçlarına ulaşmak için her zaman yardımcı olmazlar, çünkü standartlar ve rehberler teknik belgeler olup, her tasarımcının takip etmesi ve içeriğine hâkim olması için zor olabilir. Diğer yandan tasarımcı, ürününü evrensel veya kapsayıcı tasarım ilkelerine göre oluşturarak mümkün olan herkes için erişilebilirlik sağlamak istiyor olsa bile, ISO belgelerinde olduğu gibi ergonomik ve erişilebilirlik standartların güncellemesine rağmen, ticari olabilirler ve ücretsiz olarak sunulmazlar.

Standartlara hâkim olmayan tasarımcılar, en azından Tim Berners-Lee tarafından MIT ve CERN bünyesinde kurulmuş olan Dünya Çapında Ağ Konsorsiyumu (World Wide Web Consortium ya da kısaca W3C) gibi web ve mobil tasarım hakkında açık erişime sahip standartlar geliştiren toplulukların farkında olmalılar. Örneğin World Wide Web Konsorsiyumu'nun Web Erişilebilirlik Girişimi'nin (WAI) web sitelerinin erişilebiliği konusunda ayrıntılı rehberlerine w3.org sayfasından

ulaşılabılır. WAI mevcut standartların büyük ölçüde mobil cihazlara uygun olduğunu belirtse de mobil erişilebilirliğe yönelik grafik ve kullanıcı etkileşimi gibi başlıklarını kapsayan yol haritasını 2020 yılında yayınlamıştır (<https://www.w3.org/2020/09/web-roadmaps/mobile/>). Web İçeriği Erişilebilirlik Rehberi (Web Content Accessibility Guidelines - WCAG) olarak bilinen bu ilkelerin çeşitli uygulama düzeyleri vardır. Erişilebilirlik konusunda daha fazla bilgi edinmek isteyen okuyucular için W3C-WAI Dijital Erişilebilirlik Temelleri hakkında ücretsiz çevrimiçi kurslar mevcuttur.

Bölümü kapatırken Çağiltay'ın (2018) İnsan bilgisayar etkileşimi yaklaşımına göre kullanıcıların genel özelliklerini tanımladığı listeyi hatırlamakta fayda olacaktır. Tasarımcıların bir ürünü geliştirmeye aşamasında kullanıcıların sahip olduğu bu özellikleri dikkate alması, hedeflerin daha geniş bir grup tarafından benimsenmesini sağlayacaktır. Unutulmamalıdır ki kullanıcılar;

- Belirli bir hedefe yönelikler – uygulamayı bir amacı gerçekleştirmek için kullanıcılar
- Uzun süre dikkatlerini toplayamazlar
- Yaparak öğrenirler
- Hatalar yaparlar
- Bir uygulamayı önceki deneyimlerine bağlı olarak kafalarında modellerler
- Zaman içinde değişirler
- Sosyaldirler
- Bireydirler





Öğrenme Çıktısı

3 Evrensel tasarım ve erişilebilirlik kavramlarını anlatabilme
Araştır 3
<p>Arayüz tasarımında evrensel veya erişilebilir tasarım nasıl olmalıdır?</p>
İlişkilendir
<p>Erişilebilir olmayan uygulama tasarımların engelli bireylerin hayatını nasıl zorlaştırabileceğini tartışınız.</p>
Anlat/Paylaş
<p>Çevrenizde olan engelli bir bireyin mobil uygulamalar ile ilgili tercih kriterlerini belirleyiniz.</p>

Mobil Uygulama Geliştirme

1

Kullanılabilirliğin temel bileşenlerini ve ölçüm yöntemlerini tanımlayabilme

Mobil Arayüzlerde Kullanılabilirlik

Kullanılabilirlik çalışmaları bugün kullanılan anlamı ile 1980'li yıllarda kullanılabiliirk mühendisliği alanı ile başlamıştır. 90'lı yıllarda bilgisayarların ve internetin yaygınlaşması ile daha fazla ilgi çekmeye başlayan insan-bilgisayar etkileşimi çalışmaları ile ulaşılan sonuçlar 2007'den itibaren akıllı telefonlarda uygulamaların yaygınlaşmasından sonra mobil alanda da uygulanmaya başlamıştır. Kullanıcı, araç/arayüz, görev ve bağlam kullanılabiliirliğin temel çerçevesini oluşturur. Uluslararası Standartlar Enstitüsü tarafından oluşturulan tanımda etkililik, verimlilik ve memnuniyet kullanılabiliirliğin temel ölçütleri olarak kabul edilir. Görev ve hedef tanımlı kullanılabiliirlik çalışmaları eğer kullanıcıların duyguları ve değerleri ile ilişkilendiriliyorsa bu tarz yaklaşım kullanıcı deneyimi olarak tanımlanır. Kullanılabilir bir tasarım için büyük firmalar tarafından hazırlanan rehberlere başvurulabileceği gibi bazı araştırmacılar gerçek kullanıcılar ile deneyel ortamlarda yapılan model temelli testleri varolan tasarımını değerlendirmek için tercih ederler. Rehberler ve testler dışında kullanılabiliirlik çalışmaları yapmak için uygulanabilecek diğer bir yöntem ise sezgisellerin kullanılmasıdır. Bu yaklaşımmda ise konu uzmanları tasarım ilkelerine göre arayüzü değerlendirip geri bildirim verirler.

2

Mobil uygulama tasarımının genel özelliklerini açıklayabilme

Mobil Uygulama Tasarımı

Mobil uygulamaların yaygınlaşması ile, web arayızları için geliştirilen kullanılabiliirlik ilkeleri mobil arayüzlerin tasarımında da uygulanmaya başlamıştır. Mobil uyumlu (responsive) web tasarım farklı cihazlarda kullanıcılar için kesintisiz kullanım kolaylığı sağlarken, uygulamalar ise offline kullanım veya kişiselleştirme gibi mobil sitelerin sunması zor olan hizmetlerin sunulmasını sağlamıştır. Tasarımcılar her kararın bir tercih olduğunu ve bu kararların aynı zamanda avantajları ve dezavantajları ile birlikte geldiğinin her zaman farkında olmalıdır. Tasarımla ilgili verilen bir kararın tamamen avantajlardan olduğunu düşünmemek gerekir. Örneğin mobil cihazlardaki ekranların teknik ve kullanım özellikleri, masaüstü sistemler ile olan etkileşim farkları, menüleri ve uygulamada gezinme yapısını tasarlarken mobil cihazların kullanım biçimleri ve ortamlarının dikkate alınmasını gerektirmektedir. Tasarımcılar tercihlerini uygulamanın amacıyla ve hedef kitlesine göre yapmalıdır. Bunun için benzer uygulamaların ve kullanıcıların alışkanlıklarının incelenmesi, yenilik ve geleneksel arasında dengenin korunması önemlidir.

3

Evrensel tasarım ve erişilebilirlik kavramlarını anlatabilme

Evrensel Tasarım ve Erişilebilirlik

Klişe olsa da tasarımcıların veya yazılımcıların, üzerinde çalıştıkları uygulamanın kullanıcı konumunda olmadıklarını akıl-da tutmalarının yararlı olacağını belirtmek gerekmek. Uygulamayı kullanacak, değerlendirecek ve geleceğine karar verecek olanlar geliştirilenler değil, dışarıdaki potansiyel kullanıcılardır. O halde tasarım sırasında kullanıcı hedefini geniş tutmak daha akılç bir tercihtir. Tasarım ile ilgili kararlar verilirken kullanıcıların fiziksel ve bilişsel çeşitliliğinin dikkate alınması, engelli bireylerin uygulamayı kullanabilmeleri için uygun yapının kurulması ticari hedefler yanında beşerî bir unsurdur. Evrensel tasarım ilkeleri, her kullanıcının bireysel özelliklerinden bağımsız olarak uygulamayı kullanabilmesine olanak verirken, erişilebilir tasarım ise kullanıcıların kendi ihtiyaçlarına göre uygulamada gerekli kişiselleştirmeleri yapmalarına izin verir. Tasarımcılar uygulamalarında evrensel veya erişilebilir tasarım ilkelerinden birini izleyerek en geniş hedef kitleşine erişmiş olurlar.

1 Aşağıdakilerden hangisi günümüzde kullanıldığı anlamda kullanılabilirlik kavramının öncüsüdür?

- A. Birinci Dünya Savaşı
- B. Kullanıcı deneyimi
- C. Biyometri
- D. Ergonomi
- E. Kişisel bilgisayarlar

2 Aşağıdakilerden hangisi kullanılabilirlik kavramının temel bileşenlerinden biri **değildir**?

- A. Kullanıcı
- B. Arayüz
- C. Laboratuvar
- D. Bağlam
- E. Görev

3 Uluslararası Standartlar Enstitüsü tarafından yapılan kullanılabilirlik tanımı için aşağıdaki ifadelerden hangisi doğrudur?

- A. Görevin verimli veya etkili yerine getirilmesi yeterlidir.
- B. Kullanıcının duygularına odaklanır.
- C. Görev bağlamdan bağımsız ele alınır.
- D. Yaygın olarak kullanılır.
- E. Kullanıcıların genel yeteneklerine odaklanır.

4 Aşağıdakilerden hangisi kullanılabilirliğin ölçülmesi için yararlanılan yöntemlerden **değildir**?

- A. Tasarım rehberleri
- B. Kullanılabilirlik testi
- C. Sezgiseller
- D. Model temelli yaklaşım
- E. Zeka testi

5 Nielsen'e göre iyi tasarlanmış testlerle kullanılabilirlik problemlerinin %75'ini ortaya çıkarmak için kaç denek çalışmaya katılmalıdır?

- A. 1
- B. 5
- C. 15
- D. 75
- E. Belirlenemez

6 Test esnasında denekler ile imzalanan ve deney süreçlerini açıklayan protokole ne ad verilir?

- A. Bilgilendirilmiş onam
- B. Pilot test
- C. A/B testi
- D. İşbirliği protokolü
- E. Onay formu

7 Görüntülendiği platformu tanıyararak kendisi o ekranın özelliklerine göre ayarlayabilen web tasarımına ne ad verilir?

- | | |
|-----------------------|-------------------|
| A. Minimalist tasarım | B. Akıllı tasarım |
| C. Klasik tasarım | D. Basit tasarım |
| E. Responsive tasarım | |

8 Aşağıdaki hangisi akıllı telefon kullanımı alışkanlıklarını incelediğinde ulaşılabilcek bir sonucutur?

- A. %99 tek elle kullanır.
- B. Başparmak ile kullanım yaygındır.
- C. İkinci el ile kullanım acemiler içindir.
- D. Ekran büyülüğu arttıkça tek elle işlem kolaylaşır.
- E. Telefon kullanımında tek el yerine iki el kullanımı özendirilmelidir.

9 Aşağıdaki hangisi Nielsen'in mobil cihazlar da menü tasarım önerilerinden biri olabilir?

- A. Görsel iletişimden yararlanılmalı
- B. Harika görünen özel efektler tercih edilmeli
- C. Menülerde sürpriz faktörü göz ardı edilmemeli
- D. İçerik önizlemesinden kaçınılmalı
- E. Büyük ekranlarda küçük simgeler kullanılmalı

10 Kullanıcının yapmasını istediği en önemli eylemi ekranda az bir yer kaplayarak önceliklendirmek isteyen bir tasarımcı aşağıdakilerden hangisini seçmelidir?

- A. Önceliklendirilmiş içerik
- B. Hamburger menü
- C. Yüzen düğme
- D. Sekme satırı
- E. Hiçbiri

1. D	Yanıtınız yanlış ise “Kısa Tarihçe” konusunu yeniden gözden geçiriniz.	6. A	Yanıtınız yanlış ise “Kullanılabilirlik Testleri (Deneysel Yaklaşım)” konusunu yeniden gözden geçiriniz.
2. C	Yanıtınız yanlış ise “Kullanılabilirliğin Temel Bileşenleri” konusunu yeniden gözden geçiriniz.	7. E	Yanıtınız yanlış ise “Mobil Uyumlu (Responsive) Tasarım” konusunu yeniden gözden geçiriniz.
3. D	Yanıtınız yanlış ise “Kullanılabilirliğin Tanımı” konusunu yeniden gözden geçiriniz.	8. B	Yanıtınız yanlış ise “Dikey Ekran - Yatay Ekran” konusunu yeniden gözden geçiriniz.
4. E	Yanıtınız yanlış ise “Kullanılabilirlik Nasıl Ölçülür?” konusunu yeniden gözden geçiriniz.	9. A	Yanıtınız yanlış ise “Mobil Uygulamalarda Menu Yapısı” konusunu yeniden gözden geçiriniz.
5. B	Yanıtınız yanlış ise “Kullanılabilirlik Testleri (Deneysel Yaklaşım)” konusunu yeniden gözden geçiriniz.	10. C	Yanıtınız yanlış ise “Navigasyon/Gezinme” konusunu yeniden gözden geçiriniz.

2

Araştır Yanıt Anahtarı

Araştır 1

Kullanılabilirlik, uluslararası standartlar enstitüsü tarafından bir görevin etkili, verimli ve memnuniyet verici olarak tanımlanması olarak tanımlanır. Neilsen ve Shneiderman için bu bileşenler biraz farklı terminoloji ile öğrenilebilirlik, verimlilik, akılda kalıcılık, hatadan kaçınma, ve memnuniyettir. Kullanıcı deneyimi (UX) ise daha geniş bir kavram olup, bir kullanıcının ürün ile etkileşim deneyimini tanımlar. Kişinin fayda, kullanım kolaylığı gibi algılarını içerir, aynı zamanda özneldir. Ancak, kullanıcı deneyimini oluşturan bileşenler nesneldir.

Araştır 2

Geleneksel olarak sahip olduğumuz duyu organlarının sayısı 5 olarak öğretilse de bu sayının bugün daha yüksek olduğu bilinmektedir. Örneğin ısı hissi, acı hissi, denge ve vücut farkındalığı yeni yaklaşımdaydı ayrı duyu olarak sınıflanmaktadır. Aynı zamanda farklı modalitelerdeki duyuların nasıl bağlandığı (birlikte veya ayrılmaları) halen çözümlenemeyen bir olgudur. Gelecekte arayüz tasarımda entegre edilen sanal ve artırılmış gerçeklik teknolojileri ile daha fazla duyudan beklenen yönde kullanıcı deneyimi oluşturulmak için yararlanılabilir.

Araştır 3

Metinde de belirtildiği gibi dünya nüfusunun yaklaşık %15'i engelli bireylerden oluşmaktadır. Bir milyara yakın nüfus, tipik kullanıcı için tasarlanan arayüzleri kullanırken ya yardımcı teknolojilere başvurmaktır ya da kullanmaktan vaz geçmektedir. Engelli bireyler için diğer bir çözüm ise uygulamanın ihtiyaçlara göre kişiselleştirilebilmesi olabilir. Kişiselleştirme olanağının sunulması ara bir çözüm olsa da bir tasarımın kaliteden ödün vermeden herkes tarafından fark edilmeden kullanılabilmesi, kesintisiz ve kullanımına yönelik dikkat gerektirmeyen deneyim sunması arayüz tasarımını daha iyi bir seviyeye taşıyacaktır.

Kaynakça

- Babich, N. (2020). Essential Patterns of Mobile Navigation <https://xd.adobe.com/ideas/principles/app-design/essential-patterns-mobile-navigation/> 2022 tarihinde erişildi
- Bailey, S. (1993) Iterative Methodology and Designer Training in Human Computer Interface Design. Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems, 198-205
- Bush, V. (1945). As We May Think. *The Atlantic Monthly*, 176, 101-108.
- Ceci, L (2021). App uninstall rate in selected markets worldwide in October 2020, www.statista.com/statistics/1278067/app-uninstall-rate-global/, 2022 tarihinde erişildi
- Çağiltay, K. (2018). *Teoriden Pratiğe İnsan Bilgisayar Etkileşimi ve Kullanılabilirlik Mühendisliği*. Seçkin Akademik ve Mesleki Yayıncılık, Ankara
- Evans, D.C. (2017). *Bottlenecks: Aligning UX Design With User Psychology* – Apress. Kenmore – Washington
- Greco, A. (2017) “Mainstream’ UX and Games UX” UXPA, UK <https://www.youtube.com/watch?v=XxZeFJpqIDY> 2022 tarihinde erişildi
- Hodent, C. (2018). *The Gamer’s Brain: How Neuroscience and UX Can Impact Video Game Design*. CRC Press
- Hooyer, S. (2013). How Do Users Really Hold Mobile Devices? <https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php> 2022 tarihinde erişildi
- Hooyer, S. (2021). “Touch Design for Mobile Interfaces” Smashing Media AG, Freiburg
- Kieras, D.E. (2005). Model Based Evaluation of user Interfaces, Seminar on Cognitive Modeling for UI design, <https://www.cs.cmu.edu/~bej/CognitiveModelingForUIDesign/>
- Medlock, M. C. (2018). An overview of gur methods. In: Drachen, A., Mirza-Babaei, P. and Nacke, L. E. eds., Games User Research. Oxford: Oxford University Press
- Morville, P. (2004). User Experience Honeycomb. http://semanticstudios.com/user_experience_design/ 2022 tarihinde erişildi
- Nielsen, J. (1993). *Usability Engineering*. Cambridge MA: Academic Press.
- Nielsen, J. (1994). 10 Usability Heuristics for User Interface Design, <https://www.nngroup.com/articles/ten-usability-heuristics/>, 2022 tarihinde erişildi
- Nielsen, J. (2006). F-Shaped Pattern For Reading Web Content (original study) <https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content-discovered/>, 2022 tarihinde erişildi
- Nielsen, J. (2012). Mobile Sites vs. Apps: The Coming Strategy Shift <https://www.nngroup.com/articles/mobile-sites-vs-apps-strategy-shift/>, 2022 tarihinde erişildi
- Nielsen, J. (2015). Menu Design: Checklist of 15 UX Guidelines to Help Users <https://www.nngroup.com/articles/menu-design/>, 2022 tarihinde erişildi
- Norman, D. (2013). *The Design of Everyday Things* (Revised and Expanded Edition). Basic Books, NY.
- Ocak, N. ve Cagiltay, K. (2017). Comparison of Cognitive Modeling and User Performance Analysis for Touch Screen Mobile Interface Design, International Journal of Human-Computer Interaction, 33:8, 633-641, DOI: 10.1080/10447318.2016.1274160
- Pernice, K. (2017) F-Shaped Pattern of Reading on the Web: Misunderstood, But Still Relevant (Even on Mobile) <https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/>, 2022 tarihinde erişildi
- van Welie, M., van der Veer, G.C., Elins, A. (1999). Breaking Down Usability. Proceedings of INTERACT 99 (Edinburgh) IOS Press, 613-620. <http://welie.com/papers/Interact99.pdf>
- Shneiderman, B. (1998). Designing the User Interface, Addison-Wesley Publishing Company
- Soegaard, M. (2012). The history of usability: From simplicity to complexity. Smashing Magazine. 2022 tarihinde erişildi
- Web Content Accessibility Guidelines (WCAG) <http://www.w3.org/TR/WCAG/>

■ Internet Kaynakları

<https://ocw.metu.edu.tr/course/view.php?id=257>

<https://dijitalakademi.bilgem.tubitak.gov.tr/kamis>

(İngilizce) usability.gov - digital.gov

(İngilizce) <https://www.nngroup.com/>

(İngilizce) design.google.com - developer.apple.com/design/

Bölüm 3

Mobil Uygulama Geliştirme Platformları

Öğrenme Çıktıları

Android Studio Kurulumu ve Arayüz Özellikleri

- 1 Android Studio'nun kurulumunu yapabilme
- 2 Android Studio'nun arayüz özelliklerini açıklayabilme

Flutter Eklentisinin Kurulumu

- 3 Flutter'ın özelliklerini açıklayabilme
- 4 Flutter'ın Android Studio'ya kurulumunu yapabilme

3

Emülatör Kurulumu

- 5 Android Studio'da emülatör (sanal cihaz) kurabilme

2

Örnek Uygulama Geliştirme

- 6 Örnek uygulama geliştirebilme

4

Anahtar Sözcükler: • Mobil Uygulama • Mobil Uygulama Geliştirme • Android Studio • Flutter



GİRİŞ

Uygulamalar, dijital araçlarda özel bir fonksiyonu yerine getirmek için geliştirilen bilgisayar yazılımlarıdır. Örneğin bu kitap bölümünün yazılımasına olanak sağlayan kelime işlemci bir **uygulamadır**. Bilgisayara kurulumu yapılarak yazı yazmayı, yazıları düzenlemeyi (font, büyülüklük, kalınlık, eğiklik, renk, vb.) sağlayan yazılımdır. Bilgisayar uygulamaları için dizüstü ya da masaüstü bilgisayara ihtiyaç duyulmaktadır. Ancak bilgi ve iletişim teknolojilerindeki gelişmelerle dijital araçlarda değişim gerçekleşmiş ve mobil cihazların kullanımı yaygınlaşmıştır. Türkiye İstatistik Kurumu (2021) verilerine göre mobil telefon (cep telefonu veya akıllı telefon) sahiplik oranı %99,3'tür. Bir diğer ifadeyle mobil telefon sahipliği olmayan birey yok denebilir.

Uygulama: Özel bir fonksiyonu yerine getirmek için geliştirilen bilgisayar yazılımıdır.

Dijital ortamlarda içerik üretme ve paylaşma Web 2.0'a geçiş ile kolaylaşmıştır. Kullanımında uzmanlık becerisi gerektirmeyen Web 2.0 araçları, mobil uygulamalar olarak geliştirilmekte ve mobil cihazlara yüklemesi yapılarak kullanıma hazır hale gelmektedir. Mobil uygulamaların mobil cihazlara kurulumu bilgisayar uygulamalarının bilgisayara kurulumundan daha kolaydır. Bireyler rahatlıkla uygulamaları dijital cihazlarına kurabilmekte ve kaldırıbmaktadır. Mobil uygulamalar yalnızca içerik üretme ve paylaşma noktasında hayatımıza girmemiştir. Alışveriş, banka, kamu işlemleri gibi birçok alanda kullanıma hazır mobil uygulamalar hayatı kolaylaştırmaktadır. Hayatın her alanına nüfuz eden mobil uygulamaların geliştirilmesi için farklı platformlara gereksinim duyulmaktadır. Android Studio, Visual Studio (Xamarin), Cordova (PhoneGap), Appcelerator Titanium, Qt, Eclipse, IntelliJ IDEA, NetBeans, Komodo ve AIDE mobil uygulama geliştirme platformlarından bazlıdır. Kitabın ilk ünitesinde (Mobil Uygulama Geliştirme Bileşenleri) bu platformlara yer verildiği için bu bölümde detaylandırılmayacaktır. Bu bölümde Android Studio platformunun kurulumuna, arayüz özelliklerine, Flutter ve Dart eklentilerinin ek-

lenmesine, uygulamaların test edilebileceği sanal cihaz (emülatör) eklemeye ve son olarak örnek bir proje oluşturmaya yer verilecektir. Android Studio platformuna Flutter ve Dart eklentilerinin eklenecek sürece devam edilmesinin sebebi geliştirilecek olan mobil uygulamaların hem Android hem de iOS işletim sistemine sahip mobil cihazlara yönelik derlenebilmesine olanak vermesidir.

ANDROID STUDIO KURULUMU VE ARAYÜZ ÖZELLİKLERİ

Android Studio, Google tarafından uygulama geliştiricilere kolaylık sunması amacıyla IntelliJ IDEA'ya dayalı olarak tasarlanmış bir platformdur. Android Studio'nun en önemli özelliği emülatör desteğinin sağlanmasıdır denebilir. Emülatör ile uygulama geliştiriciler, geliştirilen uygulamayı test etmek amacıyla herhangi bir Android işletim sistemi cihaza gereksinim duymazlar ve uygulamanın geliştirildiği cihaz üzerinden sanal bir cihaz yardımcı ile uygulamayı test etme imkanına sahiptirler. Mobil uygulama geliştirirken kullanılan diller Java, C, C++, Kotlin ve XML'dir. Ücretsiz lisans sahip olan Android Studio Windows, MacOS, Linux tabanlı işletim sistemi kullanan bilgisayarlarda kullanılabilir.



Android Studio'nun üstünlükleri:

- Google Cloud için (Google Cloud Platform, Google Cloud Messaging ve Google App Engine) yerleşik destek sunar.
- Tüm Android platformları (cihaz ve sürümleri) emülatör ile destekler niteliktedir.
- Android Studio, güçlü bir kod düzenleyici ve geliştirici araçları sunan IntelliJ IDEA'ya dayanmaktadır.
- Github entegrasyonu ve sık kullanılan uygulama özelliklerini oluşturmaya yardımcı hazır şablonlar barındırır.
- Vektör imajlar sağlar.

- C ++ ve NDK desteği içerir.
- Birleşik Modelleme Dili (UML), yazılım modelleri çizmenin, tasarımların taslağını oluşturmanın veya var olan tasarımları ve sistemleri belgelemenin standart bir yolu olan UML diyagramı desteği sağlar.
- Ücretsizdir.

Android Studio'nun sınırlılıkları:

- Java, C, C++, Kotlin ve XML gibi dil destekleri olmasına rağmen kısıtlanmış bir desteği sunmaktadır.
- Ram kullanımı diğer IDE'lere kıyasla yüksektir.
- Android işletim sistemli cihazlar için mobil uygulama geliştirilebilir.

Android Studio'nun üstünlük ve sınırlılıkları bu şekilde sıralanabilir. Android Studio ile mobil uygulama geliştirilebilmesi için bilgisayara kurulumu gerekmektedir. Windows, MacOS ve Linux ortamlarında kurulumu yapılarak kullanımı gerçekleştirilebilir. Ancak bazı sistem gereksinimlerinin karşılanması gerekmektedir. İşletim sistemlerine göre sistem gereksinimleri Tablo 3.1'de görüldüğü gibidir.

Tablo 3.1 Android Studio Kurulum Gereksinimleri

Windows	MacOS	Linux
64-bit Microsoft Windows 8/10	MacOS 10.14 (Mojave) veya daha üstü	Gnome, KDE'yi veya Unity DE'yi destekleyen herhangi 64-bit Linux dağıtıcı; GNU C Kütüphanesi (glibc) 2.31 veya daha sonrası
X86_64 CPU Mimarisi; Intel Core 2. Nesil veya daha yeni veya Windows Hypervisor için desteklenen AMD CPU	ARM tabanlı chipset veya 2. Nesil Intel Core veya Hypervisor. Framework için desteklenen daha yeni	X86_64 CPU Mimarisi; Intel Core 2. Nesil veya daha yeni veya AMD Virtualization (AMD-V) ve SSSE3 için desteklenen AMD işlemci
8 GB RAM veya daha fazlası	8 GB RAM veya daha fazlası	8 GB RAM veya daha fazlası
Minimum 8 GB boş disk (IDE + Android SDK + Android Emülatör)	Minimum 8 GB boş disk (IDE + Android SDK + Android Emülatör)	Minimum 8 GB boş disk (IDE + Android SDK + Android Emülatör)
Minimum 1280 x 800 ekran çözünürlüğü	Minimum 1280 x 800 ekran çözünürlüğü	Minimum 1280 x 800 ekran çözünürlüğü

Kaynak: <https://developer.android.com/studio>



dikkat

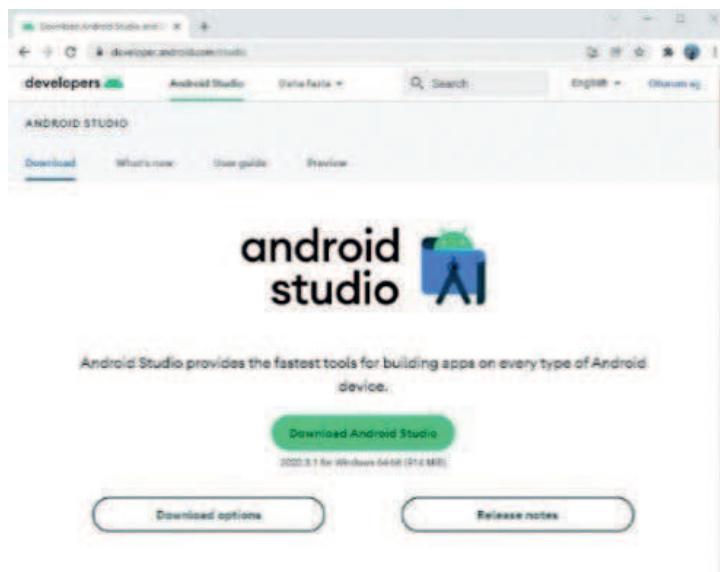
Android Studio platform bağımsız olarak (Windows, MacOS veya Linux) istenilen bilgisayara kurulabilir. Ancak işletim sistemi versiyonu, işlemci özellikleri, RAM kapasitesi, hard diskte boş alan ve ekran çözünürlüğü gibi sistem gereksinimlerinin karşılanması gerekmektedir.

Tablo 3.1 incelenerek sistem gereksinimlerini karşılayan bilgisayara Android Studio'yı kurma aşamasına geçilebilir.

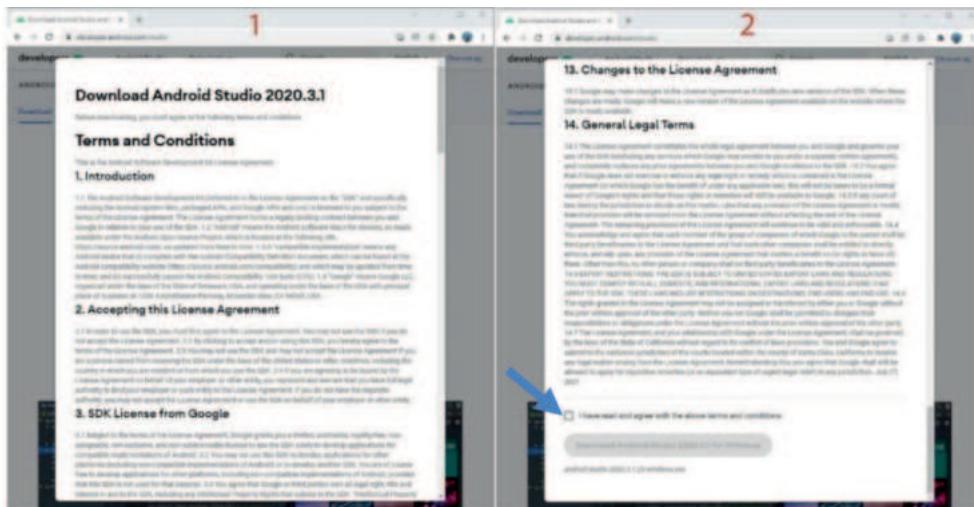
Android Studio'nun Bilgisayara Kurulumu

Android Studio'nun bilgisayara kurulumunun gerçekleştirilebilmesi için öncelikle kurulum dosyasının bilgisayara indirilmesi gerekmektedir. Bu kitap bölümünde Windows (11) işletim sistemli bir bilgisayara Android Studio kurulumu gerçekleştirilmektedir. Bu amaçla <https://developer.android.com/studio> adresinden gerekli dosya indirilmelidir (Şekil 3.1).

Şekil 3.1'de görülen "Download Android Studio" bağlantısına tıklanır. Açılan ekranda (Şekil 3.2) Android Studio'nun kullanımına ilişkin şartlar ve koşullar yer almaktadır.

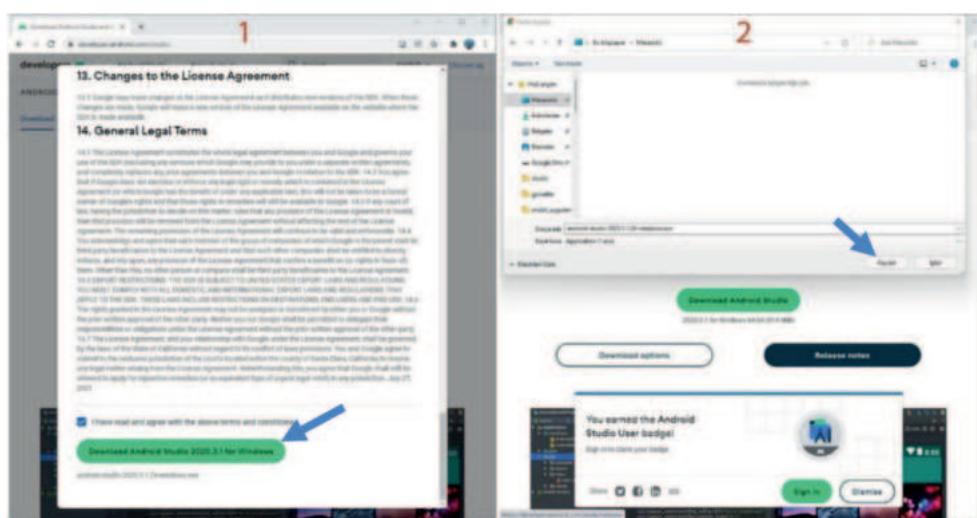


Şekil 3.1 Android Studio İndirme Ekranı



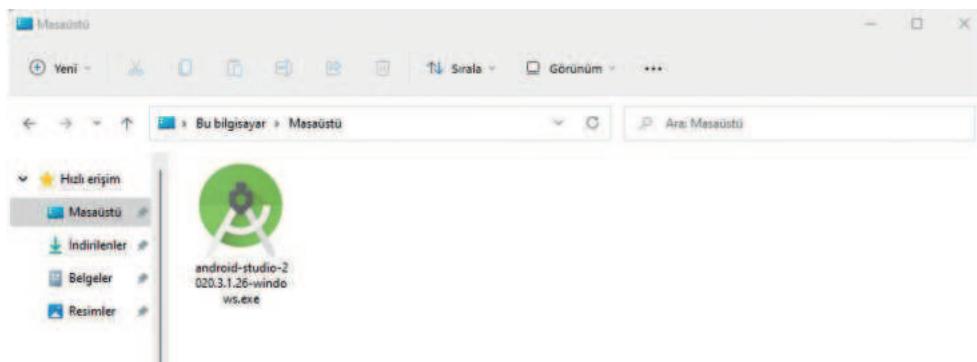
Şekil 3.2 Android Studio kullanım şartları ve koşulları

Android Studio kullanım şartlarını ve koşullarını kabul etmek için en alttaki kutucuk (Şekil 3.2) işaretlenerek onay verilir. Onay verildiğinde Şekil 3.3'teki gibi Windows için Android Studio İndir bağlantısı aktif olacaktır.



Şekil 3.3 Android Studio indirme işlemini başlatma

Şekil 3.3'te belirtilen bağlantıya tıkladığında .exe formatında dosyanın bilgisayarda hangi konuma indirileceğini soran ekran çıkmaktadır. Burada dosyanın indirilmesi istenen konum seçilerek “Kaydet” bağlantısına tıklanır. Dosya belirtilen konuma indirilerek kaydedilir.



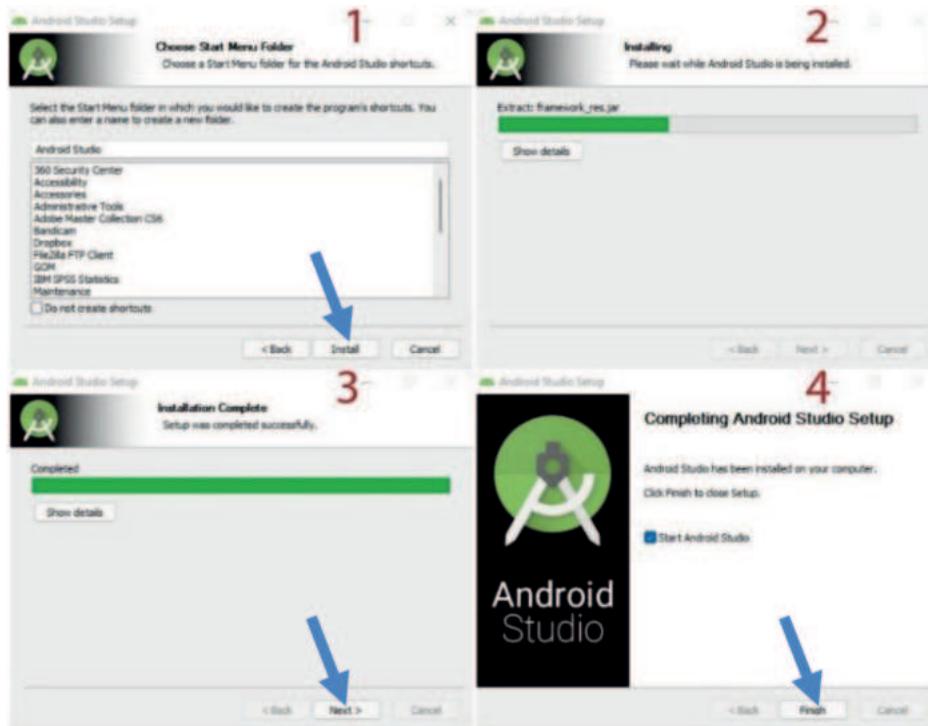
Şekil 3.4 Android Studio kurulum dosyası konumu

Şekil 3.4'te masaüstü konumuna indirilen Android Studio kurulum dosyasına çift tıklanarak kurulum başlanır. Kurulumu başlandığında ekrana gelen kurulum penceresinde “Next” butonu ile ilerlenir (Şekil 3.5).



Şekil 3.5 Android Studio kurulumu

Şekil 3.5'te görüldüğü gibi üç ekranda da “Next” butonuna tıklanarak devam edilmektedir. Üç numaralı ekranda dilenirse Android Studio'nun kurulum dosyasının konumu değiştirilebilir. Üçüncü ekranda da “Next” butonuna tıklandığında kurulumda devam edilir ve Şekil 3.6'daki ekrana geçilir.



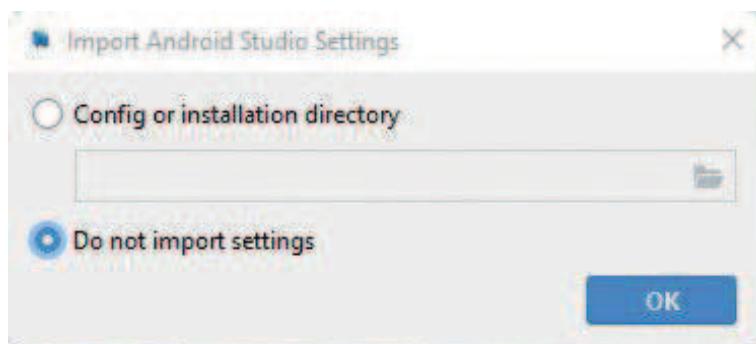
Şekil 3.6 Android Studio kurulumuna devam

Şekil 3.6'da görülen birinci ekranda “Install” butonuna tıklanır ve kurulum aşamasına geçilir. Kurulum tamamlandığında üçüncü ekranda görüldüğü gibi “Next” butonu aktif olur ve bu butona tıklanarak kurulumu devam edilir. Dördüncü ekranda görüldüğü gibi “Finish” butonuna tıklanarak kurulum tamamlanır. Kullanılan bilgisayara ilk defa Android Studio kurulumu yapılıyorsa Şekil 3.7'deki ekran çıkacaktır.



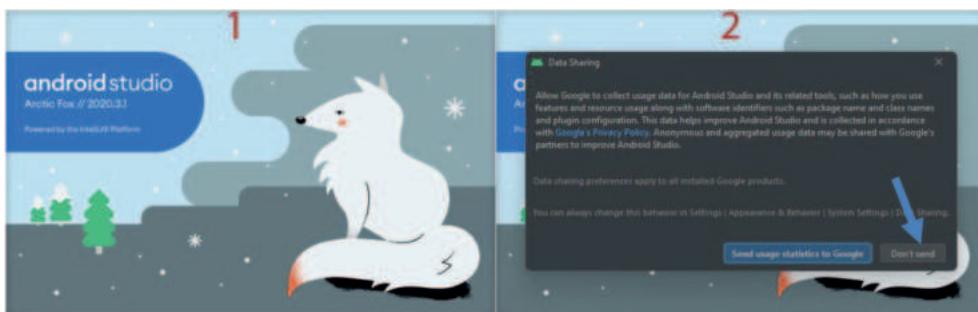
dikkat

Android Studio programı bilgisayara ilk kez kuruluyorsa önceden bilgisayarda Android Studio'ya ilişkin kayıtlı ayar olmadığı için “Do not import settings” ayarı seçilir. Ancak önceden kurulmuş Android Studio kaldırılıp tekrar kuruluyorsa eski ayarlara ihtiyaç olabileceği için eski ayarları yüklemek faydalı olabilir.



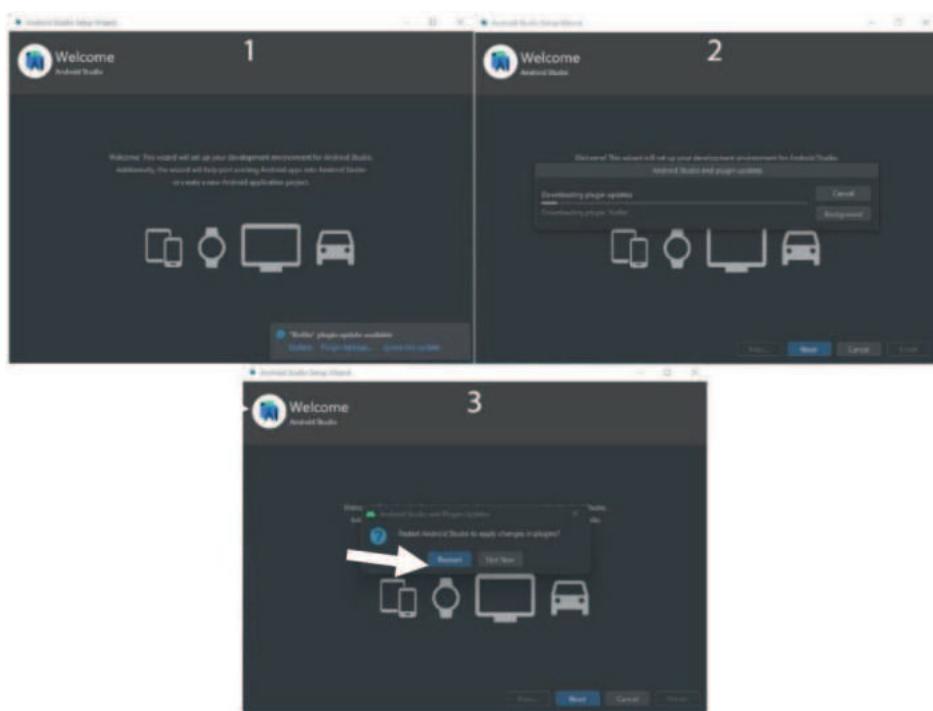
Şekil 3.7 Android Studio kurulum ayarları

Şekil 3.7 ekranında “Do not import settings” bağlantısını seçerek OK butonuna tıklanması gerekmektedir. “OK” butonuna tıklandığında Şekil 3.8'deki ekrana geçiş yapılacak ve Android Studio'nun açılışı gerçekleşecektir.



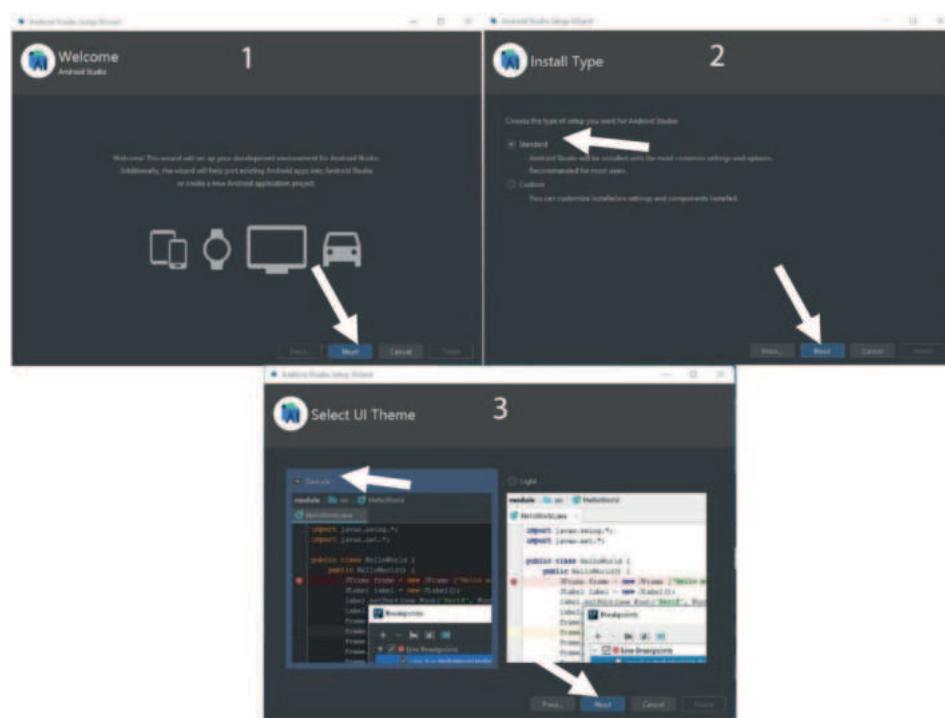
Şekil 3.8 Android Studio açılışı

Şekil 3.8'de görülen birinci açılış ekranı beklenir ve ikinci ekranda görüldüğü gibi ekrana Android Studio'yu kullanırken kullanıcı istatistiklerinin Google ile paylaşılıp paylaşılmayacağı bilgisi bulunmaktadır. Bu noktada karar kullanıcılara aittir. Bu bölümde kullanıcı istatistiklerinin Google ile paylaşılmaması için “Don't Send” butonuna tıklanarak açılışa devam edilmektedir.



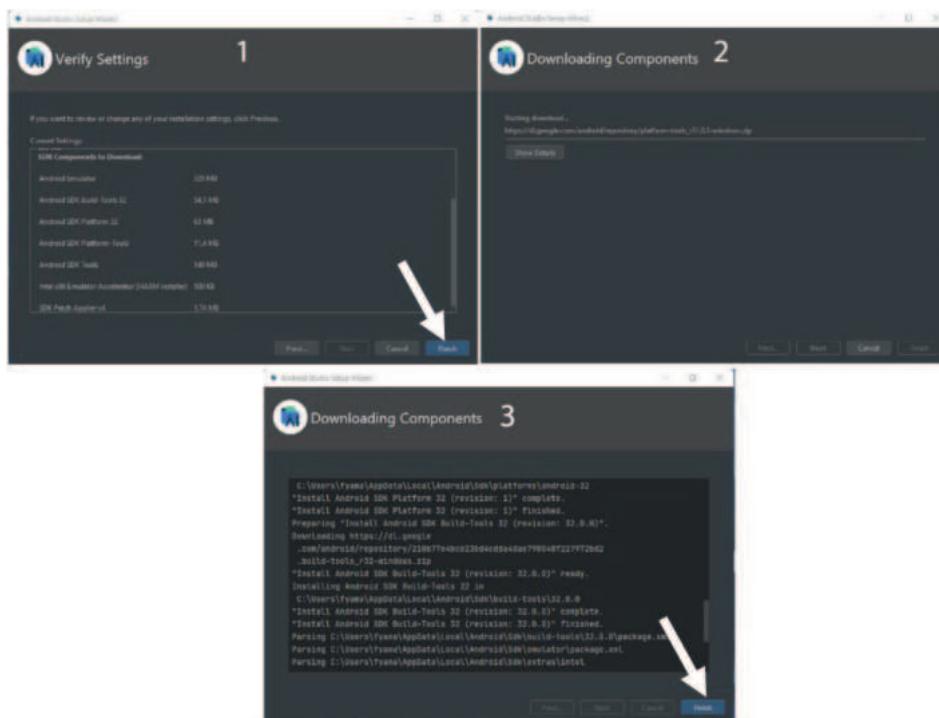
Şekil 3.9 Android Studio eklenti güncelleme

Android Studio kurulumu ilk defa gerçekleştirilmesine rağmen Şekil 3.9'da görüldüğü gibi eklenti güncellemesi gerekebilir. Örneğin bu aşamada Kotlin eklentisinin güncel versiyonunun bulunduğu ve güncellemek istenip istenmediği sorulmaktadır. Birinci ekranda görüldüğü gibi "Update" bağlantısına tiklanarak Kotlin eklentisinin güncellenmesi gerçekleştirilmektedir. İkinci ekranda güncelleme devam etmektedir. Güncelleme tamamlandığında üçüncü ekrandaki gibi ekrana bir uyarı çıkmaktadır. Güncellenen Kotlin eklentisinin sorunsuz çalışabilmesi için Android Studio'nun yeniden başlatılması gerekmektedir. Bu amaçla "Restart" butonuna tiklanarak Android Studio tekrar başlatılır.



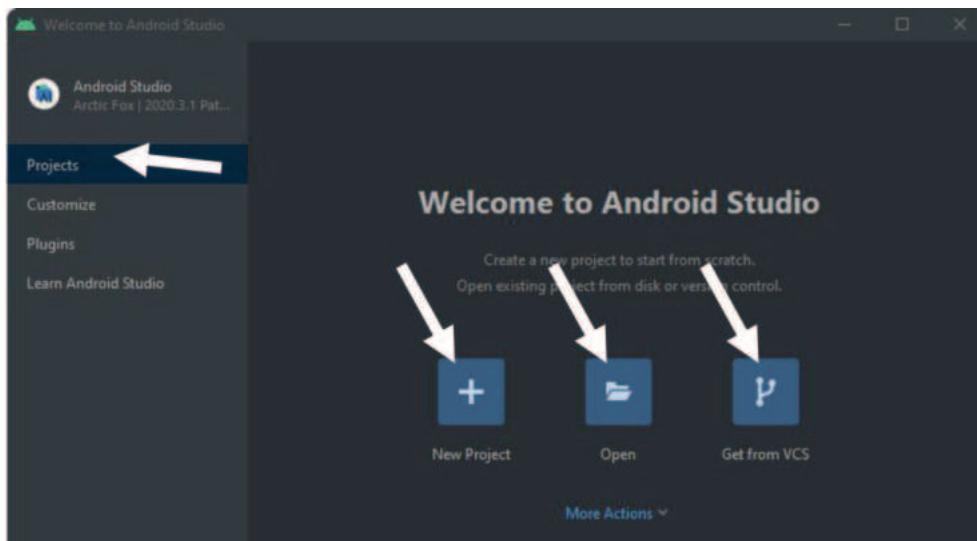
Şekil 3.10 Android Studio başlangıç ayarları

Android Studio tekrar başlatıldığında Şekil 3.10'daki birinci ekran çıkacaktır. Bu ekran “Next” butonu ile geçilir ve ikinci ekranda kurulum türü “Standart” olarak belirlenir. “Standart” seçeneği seçildikten sonra aynı ekranda “Next” butonu ile başlangıç ayarları yapılmaya devam edilir. Üçüncü ekranda Android Studio'yu nasıl bir ekranda (Dracula veya Light) kullanılacağı sorulmaktadır. Burada kullanıcılar, arka planı koyu-siyah (Dracula) ya da açık-beyaz (Light) olarak kullanım tercihini seçebilirler. Üçüncü ekranda “Dracula” tercihi yapılarak “Next” butonu ile devam edilmiştir.



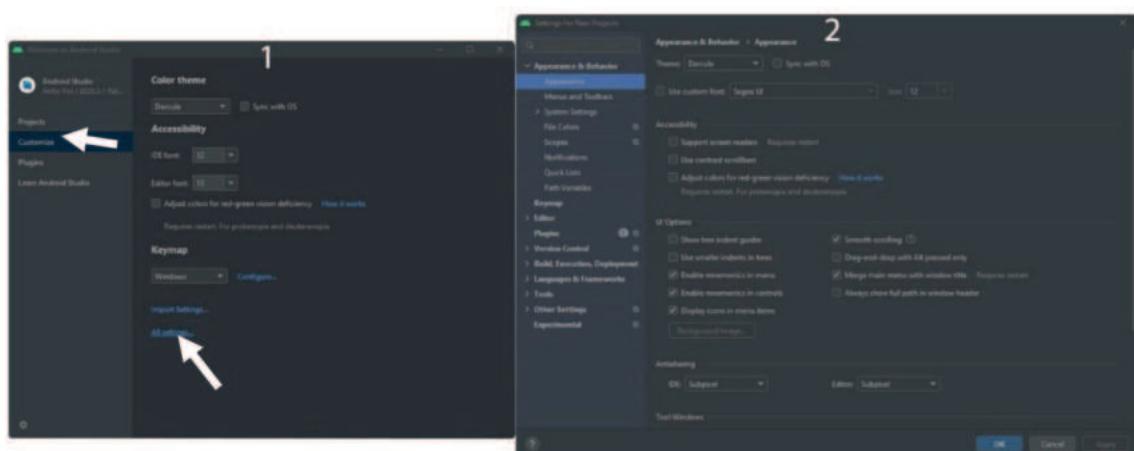
Şekil 3.11 Android Studio bileşenleri

Android Studio kullanıcı arayüz ayarlarından sonra gelen ekran Şekil 3.11'de görülen birinci ekrandır. Burada ayarların doğrulanması yapılıyor ve bu ekranda "Finish" butonu ile kurulum tamamlanıyor. Kurulumun tamamlanabilmesi için Android Studio bileşenlerinin de indirilmesi gerekmektedir. Bu amaçla birinci ekranda "Finish" butonuna tıklandığında ikinci ekran çıkıyor ve burada bileşenler indiriliyor. Ardından üçüncü ekran çıkıyor ve bu ekranda "Finish" bağlantısına tıklayarak Android Studio'nun kurulumu tamamlanmış oluyor.



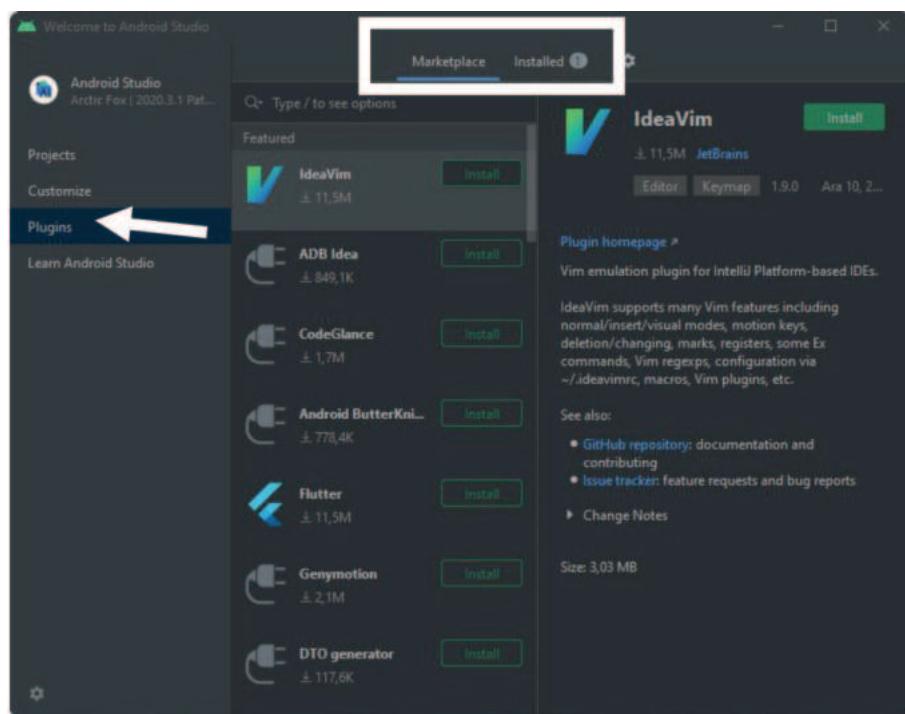
Şekil 3.12 Android Studio açılış ekranı - Projects

Şekil 3.12 incelendiğinde Android Studio'nun açılış ekranının basit bir arayüze sahip olduğu söylenebilir. Burada Projects bölümünde "New Projects" butonu ile yeni bir proje oluşturulabilir. "Open" butonu ile önceden var olan bir projenin açılması sağlanabilir. "Get from VCS" ile önceden hazırlanmış olan projelerin versiyon kontrol sistemi ile sisteme yüklenmesi sağlanır. Versiyon kontrol sistemi, önceden hazırlanan projelerin değişik sürümlerine erişmeyi ve bu sürümlerin korunmasını sağlamaktadır.



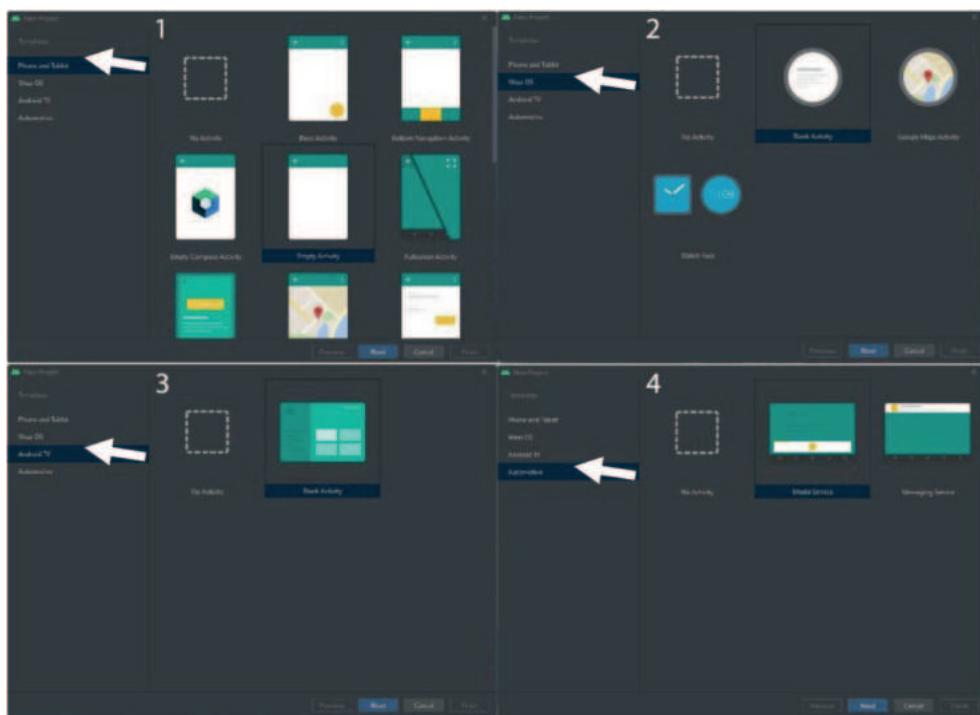
Şekil 3.13 Android Studio açılış ekranı - Customize

Şekil 3.13'te görülen Android Studio açılış ekranında sol menüde bulunan "Customize" bağlantısına tıklandığında birinci ekrandaki temel ayarlar görünecektir. Buradan Android Studio'nun özelleştirmesi sağlanabilmektedir. Birinci ekranın alt kısmında bulunan "All settings" bağlantısı ile ikinci ekranın açılması sağlanabilmekte ve bu ekranda Android Studio'ya ilişkin yapılabilecek bütün özelleştirme özellikleri yer almaktadır.



Şekil 3.14 Android Studio açılış ekranı - Plugins

Şekil 3.14'te görülen Android Studio açılış ekranında sol menüde bulunan “Plugins” bağlantısına tıklandığında Android Studio'ya eklenebilecek eklentiler bulunmaktadır. İstenilen eklenti yanında bulunan “Install” bağlantısı ile eklentiler Android Studio'ya ekleneilmektedir. Yüklenen eklentileri görmek için üst kısımda bulunan “Installed” bağlantısına tıklamak gerekmektedir.

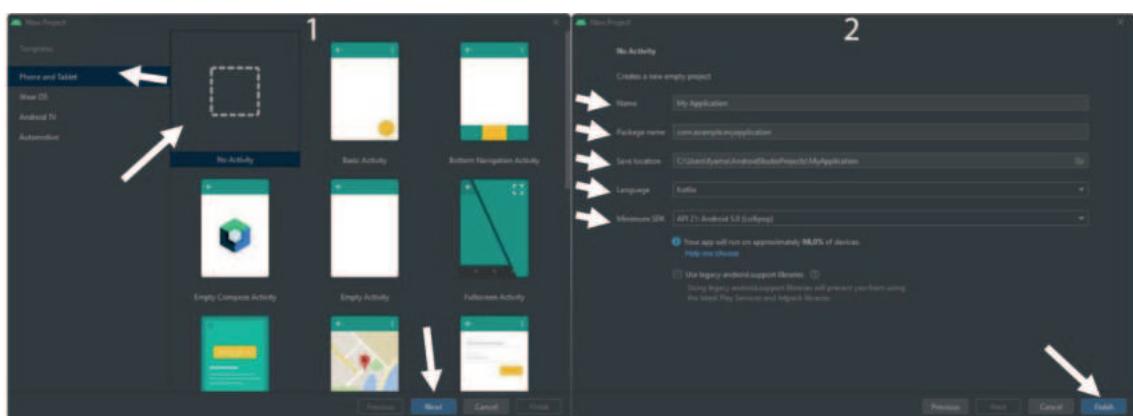


Şekil 3.15 Android Studio'da yeni proje türleri

Şekil 3.12'de görülen ekranda sol menüde “Projects” seçiliyken “New Project” butonuna tıklandığında öncelikle geliştirilecek olan mobil uygulamanın hangi platform için geliştirildiği seçilmelidir. Şekil 3.15'te birinci ekranда “Phone and Tablet” seçeneği bulunmaktadır. Burada akıllı telefonlar veya tabletler için geliştirilebilecek uygulama şablonları bulunmaktadır. Bu şablonlar seçilerek sürece devam edilebileceği gibi boş kutucuk seçilerek de sürece devam edilebilir. İkinci ekran da “Wear OS”, yani giyilebilir teknolojiler için uygulama geliştirilebilmekte; üçüncü ekran da “Android TV”ler için uygulama geliştirilebilmekte ve dördüncü ekran da “Automotive” yani araçların medya oynatıcıları için uygulama geliştirilebilmektedir. Dört ekran da görüldüğü üzere bir şablon seçilerek uygulama geliştirilebileceği gibi boş şablon seçilerek de sıfırdan bir uygulama geliştirilebilir.

dikkat

Mobil bir uygulama geliştirileceği zaman öncelikle hangi tür bir mobil cihazda kullanılacağı belirlenmelii ve bu bölümde doğru tercih yapılarak sürece devam edilmelidir.

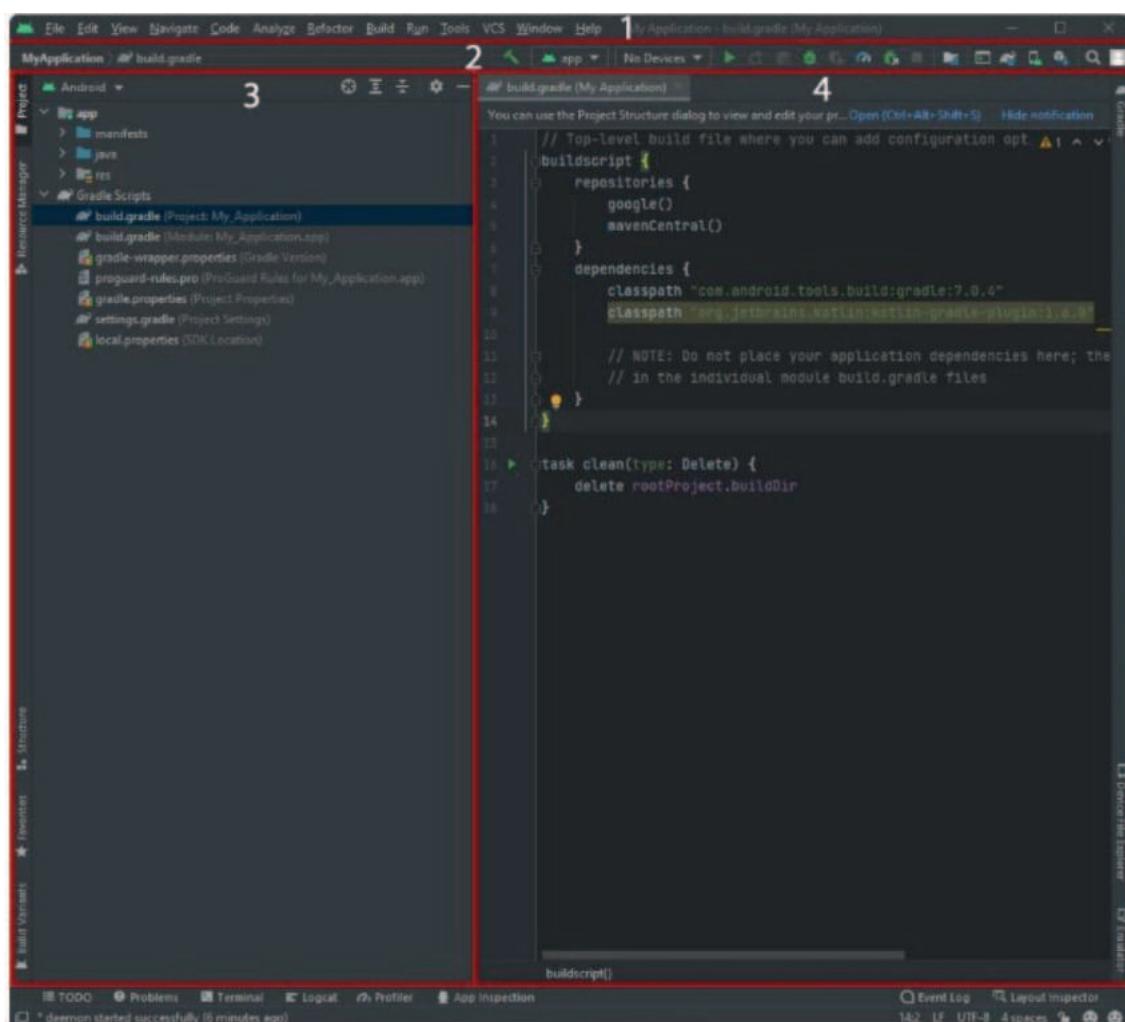


Şekil 3.16 Android Studio'da yeni proje oluşturma

Şekil 3.16'da görüldüğü gibi akıllı telefon veya tablet için bir uygulama geliştirilmesi seçilmiştir. Seçim yapıldıktan sonra “Next” butonu ile ilerleme sağlanmıştır. Gelen ekranda (ikinci ekran) geliştirilecek olan uygulamaya özgü bilgiler yer almaktadır. Öncelikle projenin bir adının olması gerekmektedir. Ardından ürünün derlenmesi sonucunda çıkacak paketin adının verilmesi gerekmektedir. Uygulamanın bilgisayarda kaydedileceği konum seçilmelidir. “Language” yani dil kısmı önemlidir. Çünkü uygulama geliştiriciler bu uygulamayı hangi dil ile yazacaktır. Bu kısımda Kotlin ve Java seçenekleri bulunmaktadır. Bunlardan biri seçilmelidir. Minimum SDK kısmı da dil seçimi kadar önemlidir. Çünkü geliştirilecek olan mobil uygulamanın Android sürümünün belirlenmesi gerekmektedir. Bir diğer ifadeyle Android sürümünün seçimi ile geliştirilecek olan uygulamanın hangi cihazlarda çalışabileceğine karar verilmektedir. Bazı akıllı telefonlar veya tabletler belirli bir süreden sonra üst sürümlerdeki Android sürümü ile desteklenmemekte ve kullanıcılar, geliştirilen uygulamayı kullanamama durumuyla karşılaşabileceklerdir. Bu ekranda ilgili alanlara bilgi girişi yapıldıktan sonra “Finish” butonu ile ilerleme sağlanmaktadır. “Finish” butonuna tıklandığında Şekil 3.17'deki ekran gelmektedir.

Android Studio Arayüz Özellikleri

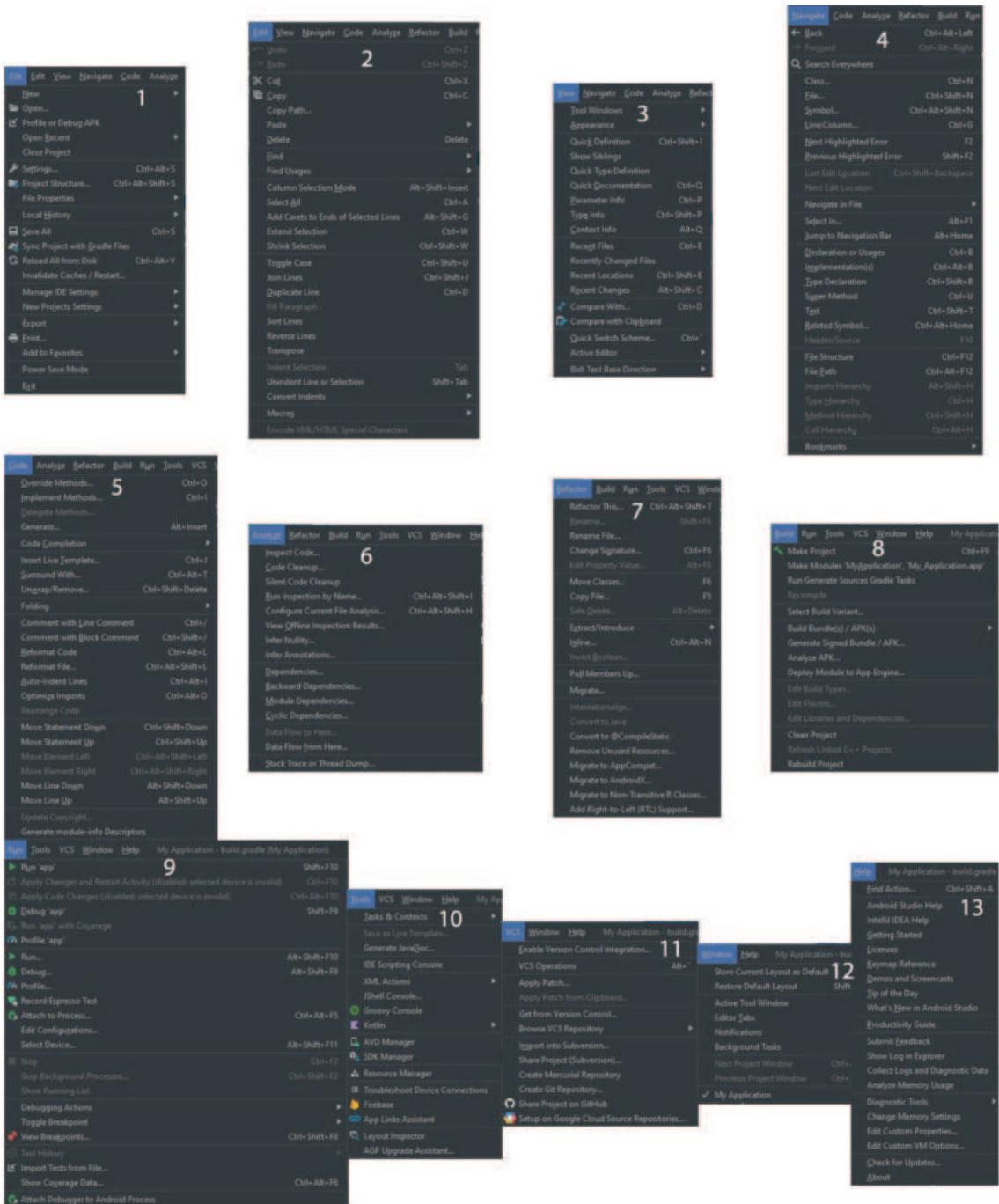
Android Studio'nun kurulumu tamamlandıktan sonra uygulama geliştirilmesi ekranı Şekil 3.17'de görüldüğü gibidir. Yazılım ile ilgilenenler için tanık bir arayüze sahip olduğu söylenebilir. Kitabın bu bölümünde Android Studio'nun arayüzü inceleneciktir.



Şekil 3.17 Android Studio kullanıcı arayüzü

Android Studio'nun ara yüzünü Şekil 3.17'de görüldüğü gibi dört bölümde incelemek mümkündür. Birinci bölümde menüler, ikinci bölümde araç çubuğu, üçüncü bölümde projelere ait panel ve dördüncü bölümde ise tasarım paneli bulunmaktadır. Birinci bölümde bulunan menülerin detayları Şekil 3.18'de görülmektedir.

Mobil Uygulama Geliştirme



Şekil 3.18 Android Studio menüleri

Şekil 3.18'deki menüler incelendiğinde Android Studio'da bu menülerle neler yapılabileceği anlaşılmaktadır. Menülerde yapılabilecek özellikleri özetlemek gerekirse;

- Yeni bir proje açılabilir.
 - Var olan proje bilgisayardaki konumunda bulunarak tekrar açılabilir.
 - Üzerinde çalışılan son proje açılabilir ya da üzerinde çalışılan proje kapatılabilir.
 - Ayarlar açılabilir ve Android Studio'nun ayarları değiştirilebilir.

- Menülerde proje yapısı ile projenin eklenti versiyonuna, SDK konumuna ve değişken gibi özelliklere ulaşmak ve değişiklikler yapmak mümkündür.
- Üzerinde çalışılan projeye ayarlar eklenebilir ya da projeye nihai hali verildiğinde dışa aktarılabilir.
- İşlemin geri-ileri alınması ya da seçilen bir kod parçasının kesilmesi, kopyalanması, yolunun kopulanması, istenilen yere yapıştırılması ya da silinmesi sağlanabilir. Bunun yanı sıra arama, kolonları seçme, satırları ayırma, birleştirme gibi projede gerekli düzenlemeler yapılabilir.
- Android Studio'da bir proje üzerinde çalışırken çalışma ekranında görünmesi istenilen özelliklerin görünmesi ya da kaldırılması sağlanabilir. Örneğin bir projedeki problemlerin nerede olduğunu görmek için problemler ekranının açık olması süreci kolaylaştırabilir.
- Üzerinde çalışılan projede gezinim yapılabilir. Örneğin proje dosyasında bulunan sınıflara, dosyalarra, sembollere veya aksiyonlara ulaşılabilirken; hata alınan noktalara de geçiş (önceki, sonraki hata) yapılabilir. Ya da proje dosyasında hazırlanmış *emmet kod* parçacıklarına veya son düzenlenen veya bir önceki düzenlenen kod konumuna da geçiş sağlanabilir. Bu ve benzeri gezinimler kolaylıkla yapılabilmektedir.
- Farklı metodların (override, implement, vb.) uygulanması sağlanırken; kod yazarken kod tamamlama da sağlanabilir.
- Kodun yer değiştirmesini (aşağı satıra, yukarı satıra) de menülerden yapmak mümkündür.
- Kodlar denetlenebilir, dipnotlardan anlam çıkarılabilir.
- Yeni proje yapılabilir.
- APK analiz edilebilir.
- Üzerinde çalışılan proje çalıştırılabilir.
- Üzerinde çalışılan proje için gerekli olan pencerelerin açılması-kapanması sağlanabilir.
- Help menüsü ile de genel olarak Android Studio'nun kullanımı hakkında bilgilere erişilebilir.

Bu işlemlerin bazılarını Şekil 3.17'deki ikinci bölümde yer alan araç çubuğundaki (Şekil 3.19) bağlanıtlarla da yapmak mümkündür.

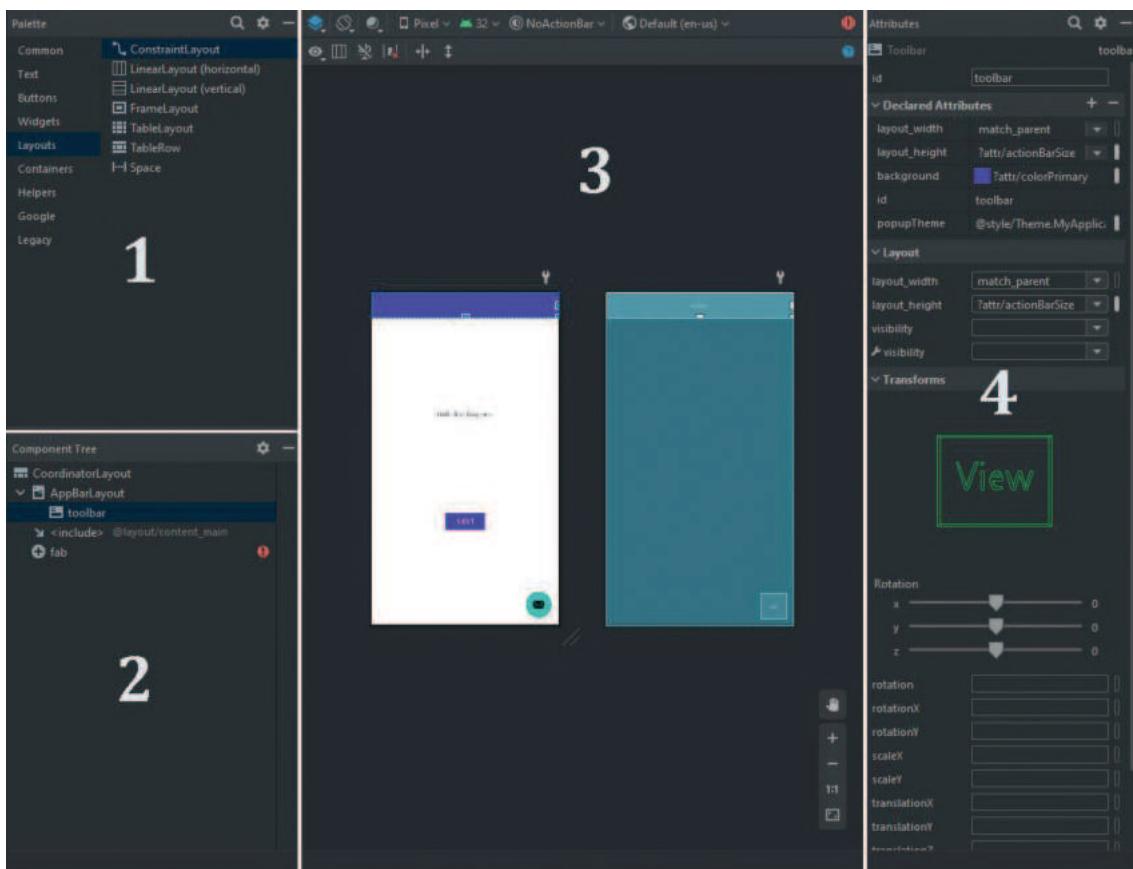


Şekil 3.19 Android Studio araç çubuğu

Şekil 3.19 incelediğinde ilk seçenek “Make Project”tir. Bu araç ile derleme işlemi gerçekleştirilir. İkinci seçenek “app” ile görünen seçenekdir. Burada “Build, Debug ve Run” işlemleri için gerekli olan ayarlamaların yapılmasına ve yapılan ayarlamaların seçilmesine olanak tanınır. “No Devices” olarak belirtilen bölüm sonraki kısımlarda incelenecektir. “Run App” ile üzerinde çalışılan projenin test edilmesi sağlanır. “Run App”ten sonraki iki seçenek şu an aktif olarak görünmüyor. Çünkü projenin çalıştırılacağı ortam seçilmemiş durumda. Bu iki seçenek ile koddaki değişiklikler uygulanır ve tekrar deneme yapılır. “Debug App” ile üzerinde çalışılan projedeki hatalar ayıklanır. “Run App with Coverage” ile uygulamaların çalıştırılmasında kaynak kodların test edilmesi sağlanır. “Profile app” ile hazırlanan projenin çalıştırıldığı ortamda testine bakılır. “Attach Debugger to Android Process” ile üzerinde çalışılan projeye hata ayıklama aracı olarak tanımlanan debuggerler eklenir. “Stop” ile çalıştırılan proje durdurulur. “Project Structure” ile bilgisayara kurulan Android SDK ve JDK bileşenlerinin kurulu olduğu konuma erişilmesi sağlanır. Bileşen konumları değiştirilmişse yeni dosya yolu oluşturulabilir. “Run Anything” ile projeden herhangi bir bölüm/görev seçilerek çalıştırılması yapılır. “Sync Project with Gradle Files” ile proje dosyalarının gradle dosyalarına uygun biçimde eşitlenmesini sağlar. “AVD Manager”, üzerinde çalışılan projelerin test edilebildiği sanal aygıtlardır. “SDK Manager” ile Android SDK'lardaki güncellemeler denetlenebilir ve bu SDK'ların kurulum konumlarına erişilebilir. Arama ve kullanıcı giriş araçları da bu bölümde yer almaktadır.

Araç çubuğundaki menülerle bu işlemler gerçekleştirilebilirken Şekil 3.17'deki proje panelinde, projeye ilişkin bütün dosyalar bulunmaktadır. Bu dosyalar hiyerarşik bir yapıdadır ve projeye eklenen dosyalar bu bölümde yer almaktadır. Bu bölümde proje adına, tüm kaynak dosyalara, Java paketlerine, projede kullanılan görsellere, simgelerre, ikonlara, proje arayüzü dosyalarına, projede kullanılan menülere ulaşmak da mümkündür. Bu bölümde erişilen ve projenin bilgilerini, kaynak kullanımı için gerekli olan izin durum bilgilerini saklayan *AndroidManifest.xml* dosyası önemli bir dosyadır.

Şekil 3.17'de görülen dördüncü bölümde yer alan tasarım paneli ise projenin görünen yapısıdır. Yani projenin ara yüzüdür. Tasarım paneline proje panelinde yer alan Android seçeneği seçiliyken layout klasöründeki *activity_main.xml* dosyasına çift tıklayarak ulaşmak mümkündür (Şekil 3.20).



Şekil 3.20 Android Studio tasarım paneli

Şekil 3.20'de bulunan birinci bölüm projenin arayüzünü tasarlama aşamasında gerekli olan buton, veri giriş alanı, listeme aracı gibi arayüz elemanlarını barındıran bölümdür. İkinci bölüm projenin görüntüleneceği ekrandaki arayüz elemanlarının ekran üzerinde hiyerarşik durumunun görüldüğü bölümdür. Üçüncü bölüm hazırlanan projenin ön izlemesinin yapılabildiği bölümdür. Dördüncü bölüm ise projenin görüntüleneceği ekrandaki ara yüz elemanlarının özelliklerinin düzenlenlenebildiği bölümdür.

Kitabın bu bölümüne kadar Android Studio'nun özelliklerine, arayüzüne ilişkin bilgilere degenilmiştir. Android Studio ile geliştirilen projelerin çıktıları Android işletim sistemli cihazlarda çalışabilmektedir. Geliştirilen uygulamaların farklı işletim sistemli cihazlarda da görüntülenebilmesi önem arz etmektedir. Bu nedenle de Flutter önemli bir konuma sahiptir. Çünkü Flutter ile geliştirilen uygulamaların çıktıları hem Android hem de iOS işletim sistemli cihazlar için alınabilmektedir.

Öğrenme Çıktısı



FLUTTER EKLENTİSİNİN KURULUMU



Flutter, Google tarafından geliştirilen açık kaynak kodlu uygulama geliştirme platformudur. Ücretsiz olan Flutter, Android veya iOS işletim sistemine sahip cihazlar için uygulama geliştirmeye olanak sağlamsaktadır. Bir diğer ifadeyle uygulama geliştiriciler tek bir ortamda uygulama geliştirdikten sonra bu uygulamaların çıktısını Android ve iOS işletim sistemine sahip

cihazlar için ayrı ayrı alabilmektedir. Bu durum uygulama geliştiriciler için zamanlı tasarruf sağlarken; uygulama geliştirenler için de maliyetten tasarruf sağlıyor denebilir. Flutter'in özellikleri şu şekilde sıralanabilir:

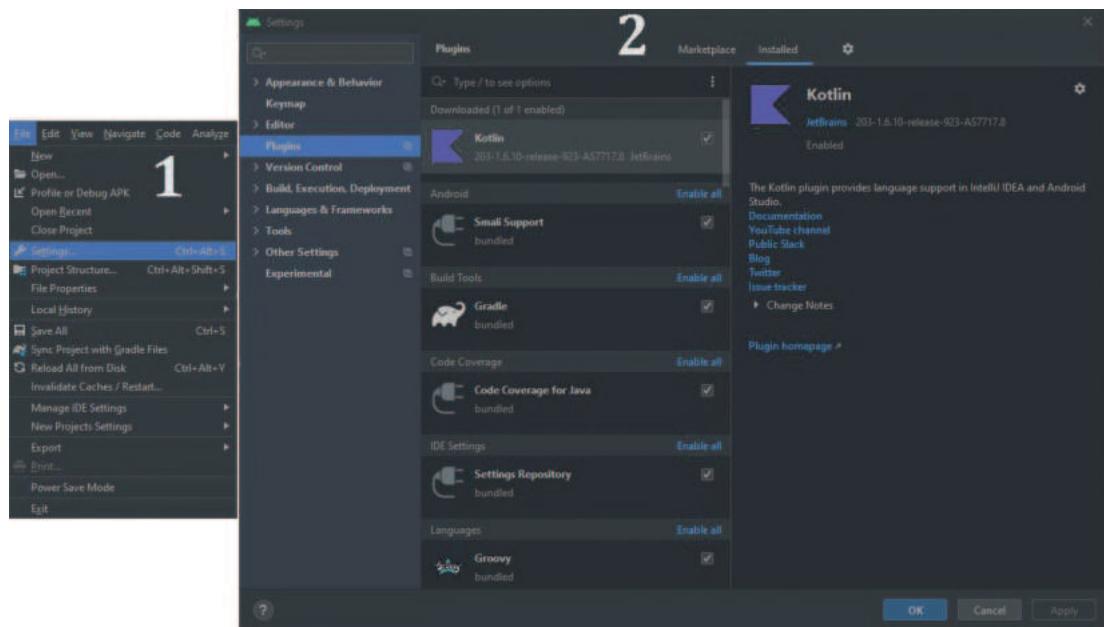
- Flutter, esnek yapıya sahip SDK'dır.
- Flutter, Windows, Linux ve MacOS gibi ortamlara rahatlıkla kurulabilir ve bu ortamlarda uygula-
ma geliştiriciler çalışabilir.
- Uygulamaları hızlı geliştirmeyi sağlar.
- Flutter hızlı bir şekilde yeniden yüklenebilir, hızlı ve kolay bir şekilde deneme yapılabilir, kullanıcı arayüzünün oluşturulması, yeni özelliklerin eklenmesi ve hataların da hızlı bir şekilde giderilmesi mümkünür.
- Güzel ve kullanışlı arayüzler geliştirmeyi sağlar.
- Flutter'a yerleşik olan Android Materyal Tasarımı ve Cupertino Widget'lerin varlığı, zengin ha-
reket API'leri, platform farkındalığı ile kullanıcılarla rahat bir kullanım sağlar.
- Flutter, render motoru olarak Mobile-first 2D render motorunu, framework olarak react-style framework'ü kullanmaktadır.
- Flutter, Android ve iOS işletim sistemli cihazlarda kullanılan Widget destegine sahiptir.
- Flutter, Hot Reload özelliğine sahiptir. Bu özellik ile gerçek zamanlı düzenleme yapılmaktadır. Yani uygulamalar geliştirilirken Hot Reload özelliği ile yapılan değişiklikler uygulamaya hemen yansımaktadır.

- Flutter ile uygulama geliştirildiğinde uygulamalar Android işletim sistemine sahip cihazlarda Android sürümünün en az Jelly Bean v16 üzeri ve daha sonraki sürümler olması beklenirken; iOS işletim sistemine sahip cihazlarda iOS sürümünün en az 8 ve daha sonraki sürümler olması beklenmektedir.



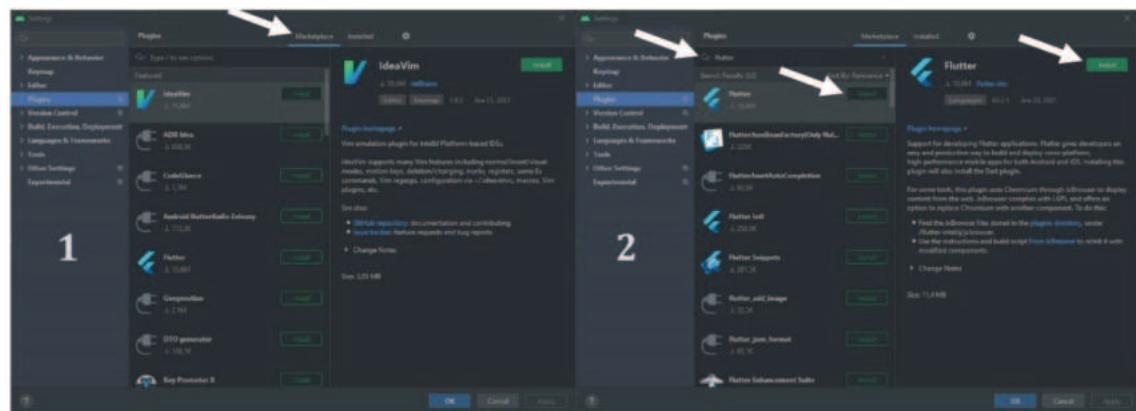
Flutter'ın Android Studio'ya Kurulumu

Flutter'ın Windows işletim sistemli bir bilgisayara kurulumu için işletim sisteminin Windows 7 SP1 ve daha sonrası (64-bit), x86-64 tabanlı, 1.64GB boş disk kapasitesi, Windows PowerShell 5.0 ve daha yenisi ve Windows 2.x için Git gereksinimlerine ihtiyaç duyulmaktadır. Bu gereksinimler diğer işletim sistemlerine sahip bilgisayarlar için değişmektedir. Bu gereksinimlere sahip olan bilgisayarda Android Studio'ya Flutter eklentisini eklemek için öncelikle File → Settings adımı takip edilmeli (Şekil 3.21).



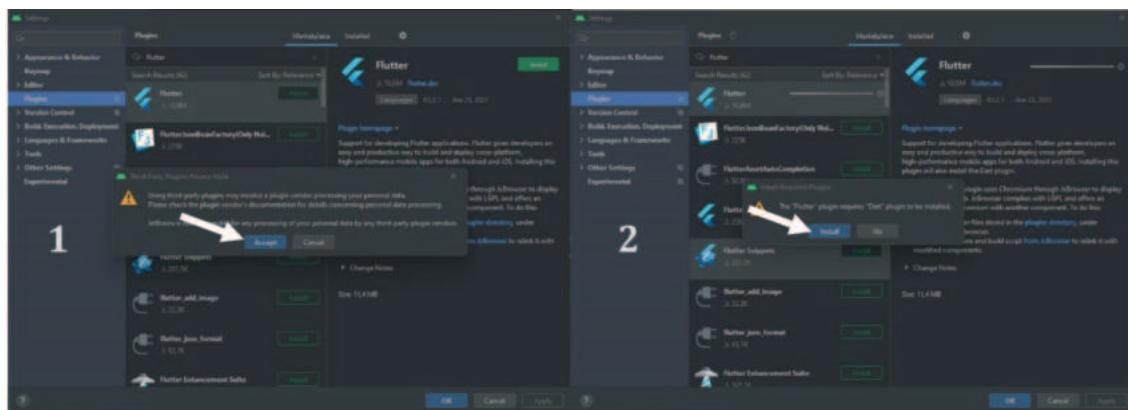
Şekil 3.21 Android Studio'ya Flutter kurulumu - Ayarlar

Şekil 3.21'de görüldüğü gibi “Settings” bağlantısına tıklandığında açılan ekranda “Plugins” bölümünde yer almaktadır. Burada Android Studio'daki eklentileri görüntülemek ve yenilerini eklemek mümkündür. Açılan pencerede öncelikle yüklü eklentiler görüntülenmektedir. Yeni eklenti kurmak için “Marketplace” sekmesine geçilmeli (Şekil 3.22).



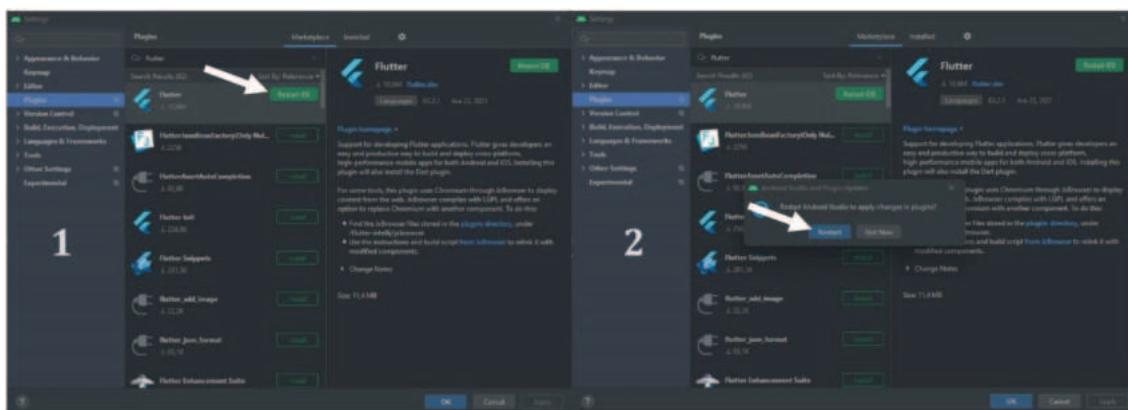
Şekil 3.22 Android Studio Flutter kurulumu - Eklenti arama

Şekil 3.22'de birinci ekranda “Marketplace”e geçiş yapılmış ve ikinci ekranda da arama çubuğuuna “Flutter” yazılmıştır. Çıkan sonuçta “Install” bağlantısı ile Flutter eklentisinin kurulumuna başlanır. “Install” bağlantısı ikinci ekranda görüldüğü gibi iki tarafta da bulunmaktadır. Herhangi biri ile işleme devam edilebilir. Kurulumu başlandığında Şekil 3.23'te görüldüğü gibi üçüncü parti uygulamalar için gerekli izinlere yönelik onay ekranı gelmektedir.



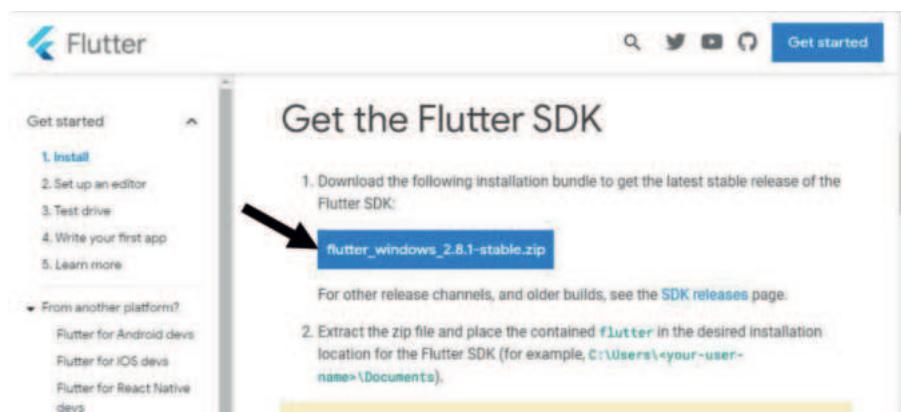
Şekil 3.23 Android Studio Flutter kurulumu - Onay ve Dart eklentisi

Şekil 3.23'te görülen birinci ekranda Android Studio'ya eklenen üçüncü parti eklentiler için gerekli izinler için onay ekranı bulunmaktadır. Gerekli izinler için “Accept” butonuna tıklanır ve ardından ikinci ekrandaki pencereye geçilir. İkinci ekranda Flutter eklentisinin Dart eklentisine ihtiyaç duyduğu ve bunun da Android Studio'ya yüklenmesi gereği bilgisi bulunmaktadır. Bu ekranda da “Install” butonuna tıklanır ve Dart eklentisi de yüklenir. Gerekli eklenti yüklemeleri tamamlandıktan sonra Şekil 3.24'teki ekrana gelir.



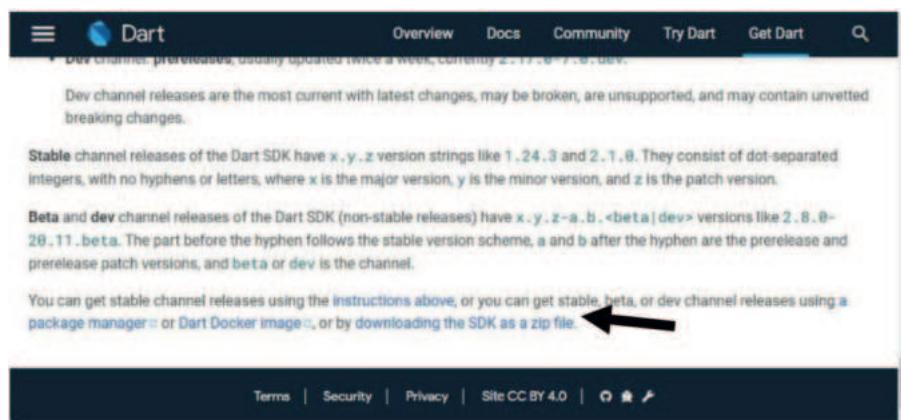
Şekil 3.24 Android Studio Flutter kurulumu tamamlama

Flutter eklentisi Android Studio'ya başarıyla eklendikten sonra Şekil 3.24'te birinci ekrana dönülmektedir. Burada yüklenen eklentinin sağlıklı çalışabilmesi için Android Studio'nun yeniden başlatılması gerekmektedir. Birinci ekrandaki “Restart IDE” butonuna tıklanır ve ikinci ekrandaki pencerenin açılması sağlanır. Bu ekranda eklentideki değişikliklerin Android Studio'ya uygulanması için yeniden başlatılmıştır. “Restart” butonuna tıklandığında Flutter eklentisinin kurulumunun tamamlanması ve değişikliklerin de Android Studio'ya yansıtılması sağlanmış olur. Bu aşamada Flutter ve Dart eklentileri eklenmiş oluyor. Ancak yapılması gereken bir işlem daha bulunmaktadır. Flutter ve Dart SDK'larının yolunun tanıtılması gerekmektedir. Bunun için öncelikle Flutter SDK'sı için <https://docs.flutter.dev/get-started/install/windows> adresinden gerekli .zip dosyası indirilir (Şekil 3.25).



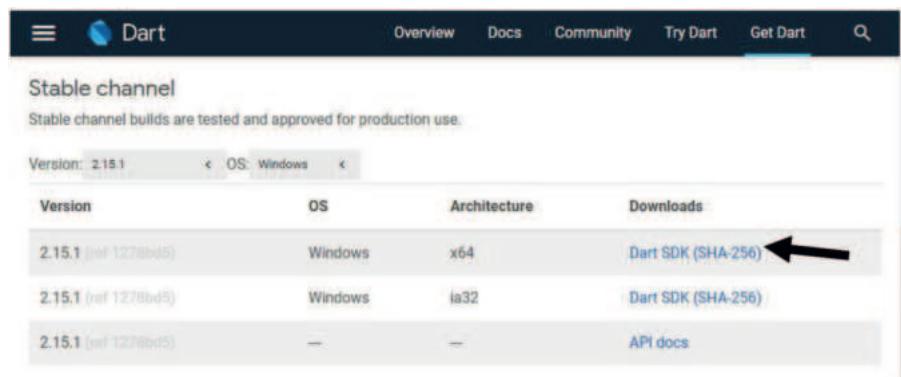
Şekil 3.25 Flutter SDK indirme

Şekil 3.25'te görülen bağlantıya tıklanarak Flutter SDK'sı indirilir. Ardından <https://dart.dev/get-dart#windows> adresinden Dart SDK'sı indirilir.



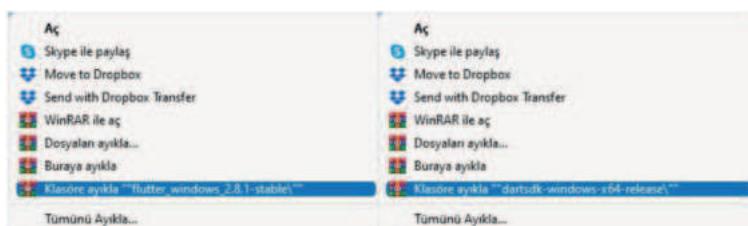
Şekil 3.26 Dart SDK indirme

Şekil 3.26'da görülen ekranda en alt kısımdaki bağlantıya tıklanır. Ardından gelen ekranda Windows mimarimize uygun olan Dart versiyonu seçilir (Şekil 3.27).



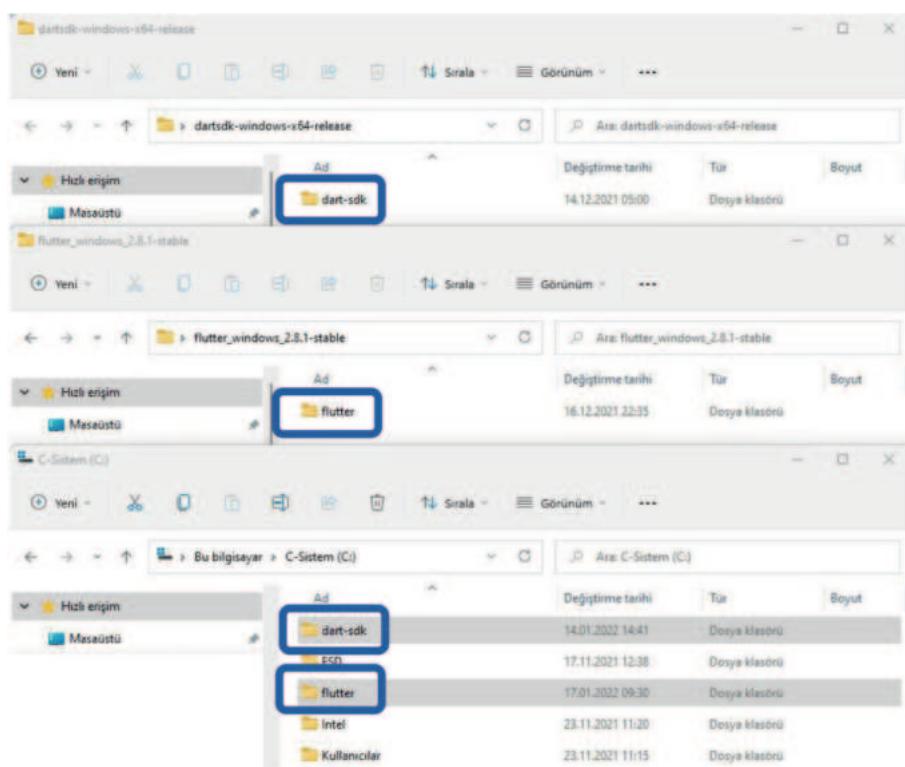
Şekil 3.27 Dart SDK indirme - Windows mimarisine göre

Şekil 3.27'de gerekli bağlantıya tıklandığında .zip dosyası bilgisayara indirilecektir. Flutter ve Dart SDK'ları .zip'li yapıda bilgisayara indirilmektedir. Bu iki dosya da klasöre çıkarılır (Şekil 3.28).



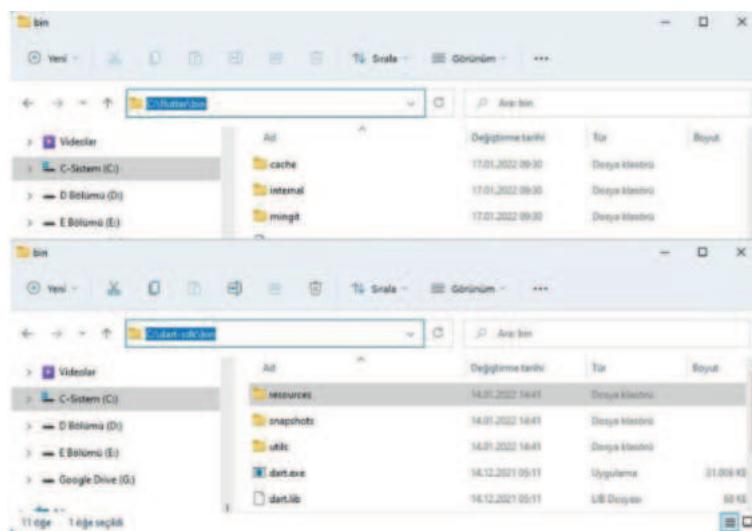
Şekil 3.28 Flutter ve Dart dosyalarını klasöre çıkarma

Şekil 3.28'de görülen işlem gerçekleştirildiğinde .zip'li dosya klasöre çıkarılmış olur. Klasöre çıkarılan dosyalar bilgisayarda C konumuna kopyalanır (Şekil 3.29).



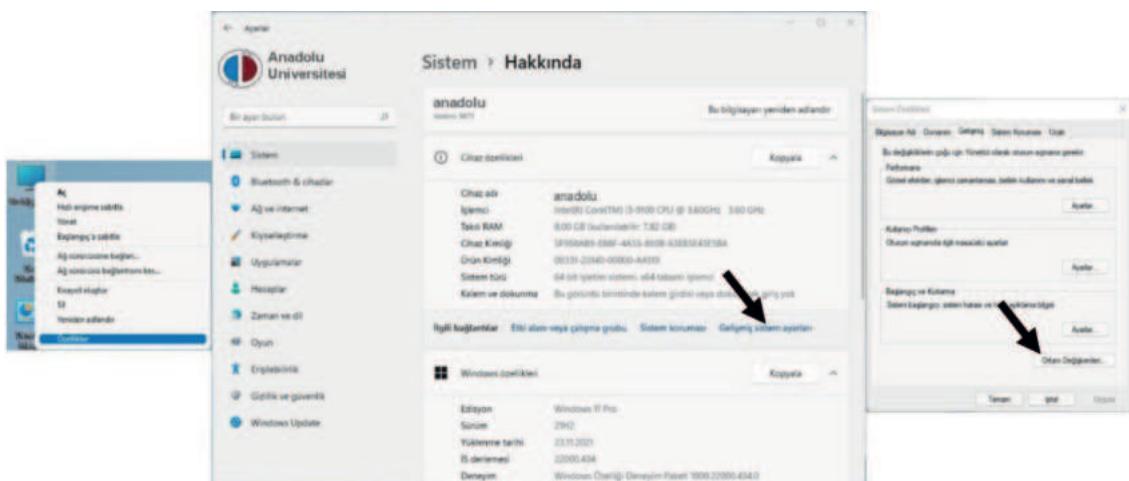
Şekil 3.29 Flutter ve Dart dosyalarının C klasörüne kopyalanması

Flutter ve Dart dosyaları C klasörüne taşındıktan sonra yollarının bilgisayara tanıtılması gerekmektedir. Bunun için C klasöründe bulunan Flutter ve Dart dosyalarına çift tıklanır ve “bin” klasörü de açılır (Şekil 3.30).



Şekil 3.30 Flutter ve Dart SDK yolunun seçilmesi

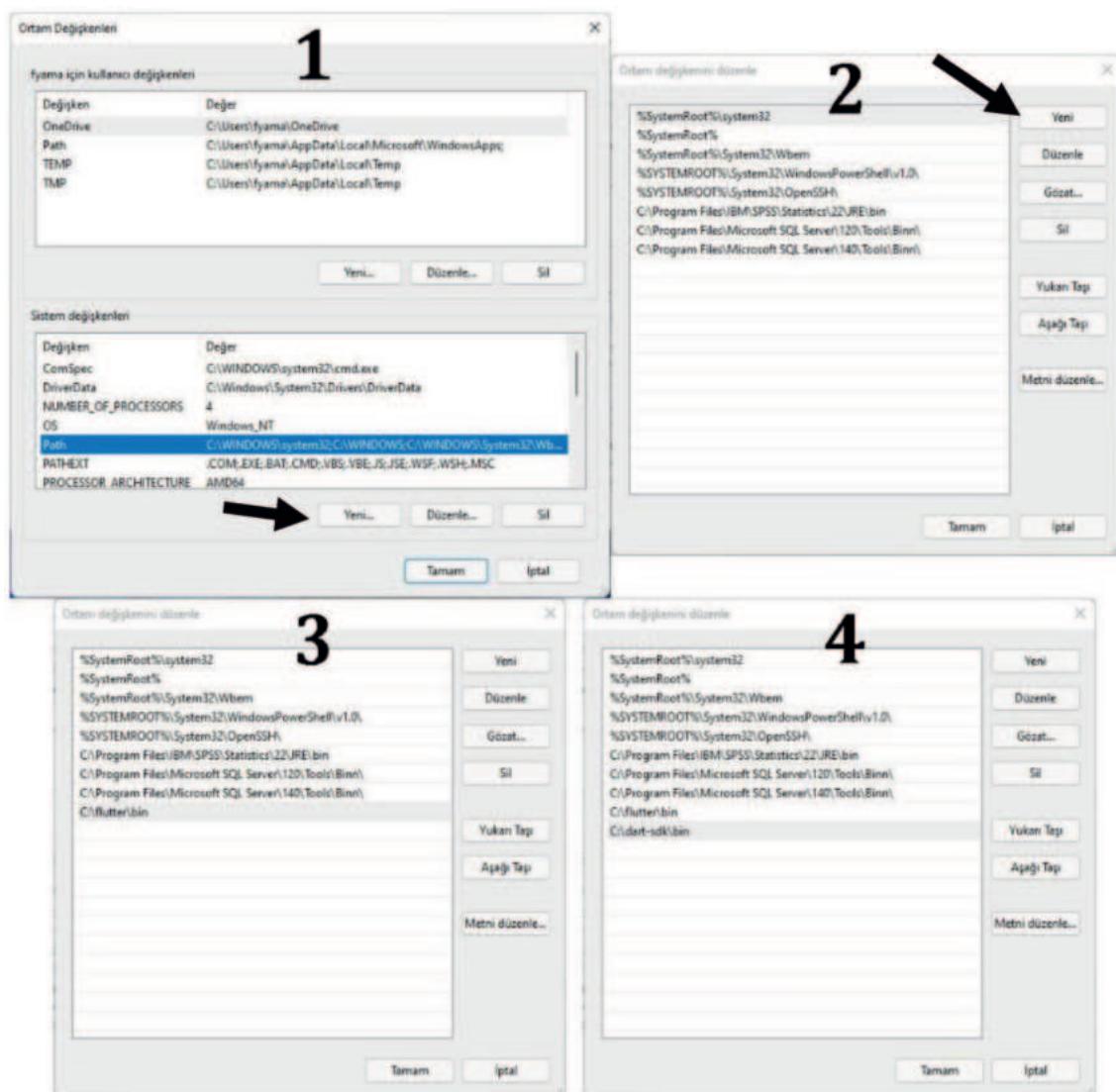
Şekil 3.30'da görüldüğü gibi SDK yolları ayrı ayrı seçilerek kopyalanır ve masaüstünde “Bu Bilgisayar” ikonuna sağ tıklanır ve özelliklere gidilir (Şekil 3.31).



Şekil 3.31 Gelişmiş sistem ayarları

Şekil 3.31'de görüldüğü gibi bilgisayarın özelliklerine gidildiğinde “Gelişmiş sistem ayarları” bağlantısına tıklanır. Bu adımdan sonra gelen ekranda “Ortam Değişkenleri” butonuna tıklanır ve ilgili ekrana geçiş yapılır (Şekil 3.32).





Şekil 3.32 Ortam değişkenlerine Flutter ve Dart SDK yolunun tanıtılması

Ortam değişkenleri için ilk açılan ekran Şekil 3.32'de görüldüğü gibidir. Birinci ekranda “Değişken” sütununda yer alan “Path” satırına tıklanır. Ardından alta bulunan “Yeni” butonuna tıklanır. Açılan ikinci ekranda da “Yeni” butonuna tıklanır ve giriş yapılabilecek satırı kopyalanan Flutter ya da Dart SDK'larının yolları ayrı ayrı yapıştırılır. Bu şekilde iki işlem yapılarak üçüncü ve dördüncü ekrandaki görüntüler elde edilmiş olunur. Bu aşamadan sonra Android Studio'ya Flutter eklentisinin sağlıklı yüklenip yüklenmediğini kontrol etmek için Windows + R (+ tuş bileşenlerine basılarak CMD ekranı açılır. Bu ekranda `flutter doctor` komutu yazılır, Enter'a basılır ve Şekil 3.33'teki sonuç alınır.

```
C:\Kütüphane>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.8.1, on Microsoft Windows [Version 10.0.22000.434], locale tr-TR)
[!] Android toolchain - develop for Android devices (Android SDK version 32.0.0)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses
[✓] Chrome - develop for the web
[✓] Android Studio (version 2020.3)
[✓] Connected device (2 available)

! Doctor found issues in 1 category.

C:\Kütüphane>
```

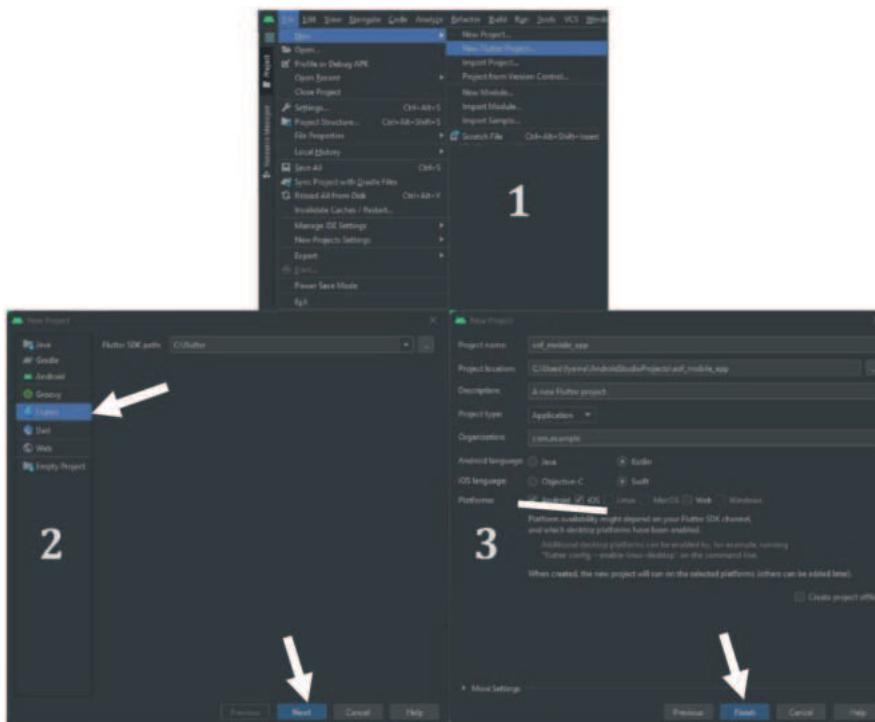
Şekil 3.33 Flutter Doctor komutu ile Flutter'ı test etme

CMD ekranında `flutter doctor` komutu yazarak ulaşılan ekranda Şekil 3.33'te görülen bilgilere ulaşılmaktadır. Burada Flutter'in Android SDK'sının versiyonlarına ulaşılabilir. Yanlarında “✓” işaretli olan bildirimler işlemlerde sorun olmadığını göstermektedir. Ancak bazı satırların yanında “!” işaretli olabilir. Bu durumda ne yapılması gerekiğine dair bilgi ilgili satırda verilmektedir. Örneğin ikinci satırda bazı Android lisanslarının kabul edilmediği belirtilmektedir. Bunun çözümü için CMD ekranında `flutter doctor --android-licenses` komutunun yapılması gerektiği belirtilmektedir. Bu kod CMD ekranında yazarak gerekli yönlendirmeler takip edilerek güncellemeler yapılabilir ve sorun çözülebilir.

dikkat
Flutter eklentisi yükledikten sonra `flutter doctor` komutu ile sorun olup olmadığı kontrol edilmelidir. Sorun varsa ilgili satırındaki bilgilendirme kullanılarak sorun çözülmelidir.

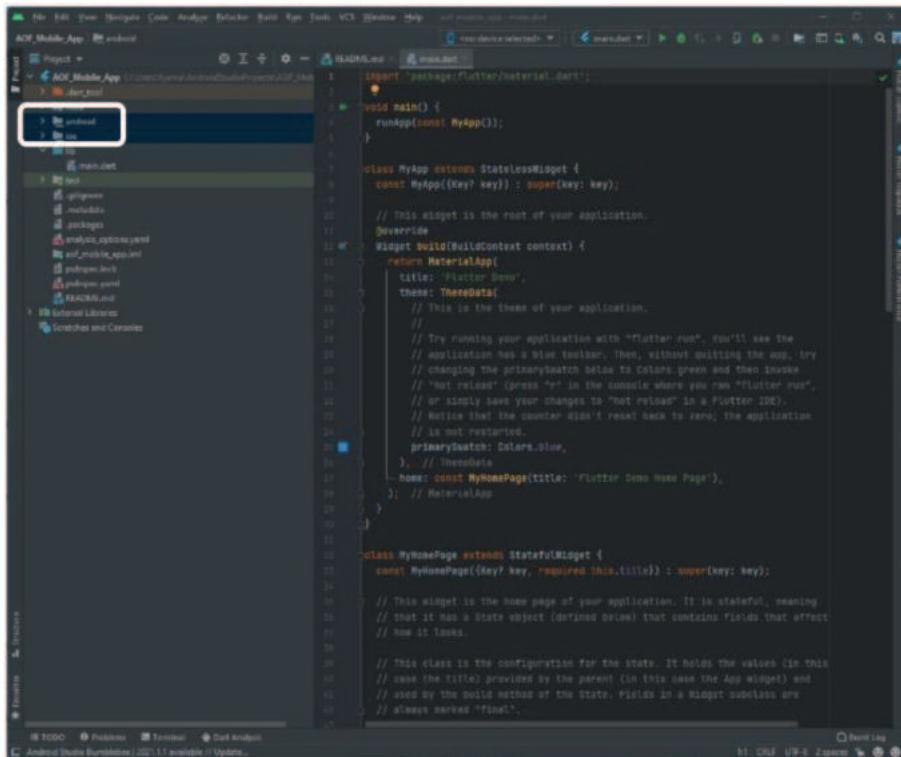
Flutter Projesi Oluşturma

Android Studio'ya Flutter ve Dart eklentileri ekledikten ve gerekli SDK ayarları yapıldıktan sonra Flutter projesi açma aşamasına geçilebilir. Öncelikle Android Studio'da “File → New → New Flutter Project” adımları takip edilir (Şekil 3.34).



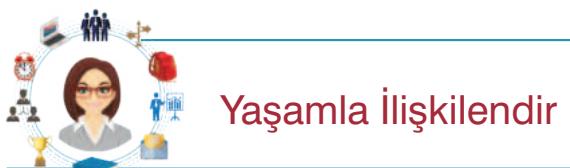
Şekil 3.34 Android Studio'da Flutter projesi oluşturma

Android Studio'da yeni bir Flutter projesi oluşturmak için Şekil 3.34'te görülen birinci ekrandaki adımlardan sonra ikinci ekrana geçilir. İkinci ekranda sol panelde bulunan Flutter sekmesi seçilir ve ilerlemek için “Next” butonuna tıklanır. “Next” butonuna tıklandığında üçüncü ekrana geçilir. Bu ekranda projenin adının girileceği ve konumunun seçileceği alanlar mevcuttur. Proje adı girildikten sonra isteğe bağlı olarak projenin konumu değiştirilebilir. Projenin açıklaması kısmına da bilgi girilebilir. Bilgi girilmemiş takdirde default olarak gelen “A new Flutter Project.” açıklaması kalacaktır. Bu bilgiler girildikten sonra “Finish” butonuna tıklanır. “Finish” butonuna tıklandığında Android Studio'da oluşturulan Flutter projesi çalışma ekranına ulaşacaktır (Şekil 3.35).



Şekil 3.35 Oluşturulan Flutter projesinin başlangıç ekranı

Şekil 3.35'te dikkat edilmesi gereken nokta proje dosyalarının bulunduğu kısım olmalıdır. Çünkü Flutter projesi ile tek ekranda yazılan mobil uygulama kodları ile Android ve iOS işletim sistemleri için uygulama çıktısı elde edilebileceği görülmektedir. Bu bilgiye proje dosyaları kısmında Android ve iOS dosyaları ile erişilebilmektedir. Proje yazımı sol panelde bulunan “Lib” klasörü içerisinde bulunan **main.dart** dosyası üzerinde gerçekleştirilmektedir. Flutter proje oluşturma ekranı, detaylı bir şekilde açıklanan Android Studio ara yüzüyle aynıdır. Bu nedenle bu bölümde tekrar incelenmeyecektir. Bu bölümde örnek proje oluşturma aşamasından önce hazırlanan projelerin kontrol edileceği emülatör (sanal cihaz) kurulumuna bakılacaktır.



Yaşamla İlişkilendir

Flutter'in Geleceği

Google tarafından geliştirilen ve Microsoft gibi dev yazılım şirketlerinden gelen destekle beraber Flutter geliştirilmeye devam ediliyor. Microsoft, çift ekranlı yeni Surface Duo cihazı için özel bir kütüphane yazarak bizzat Flutter ekibine bu konuda yardımcı olmuş. Flutter açık kaynaklı geliştirilen bir UI geliştirme aracı olduğu için birçok yazılım şirketi bu projeye katkı yaparak, kendi cihazlarına uygun olacak şekilde geliştirmeler ekleyebildiğinden Flutter hızlıca çok aranan bir teknoloji olabilir. Bu yazında Flutter ile geliştirilmiş popüler uygulamalara göz atabiliyorsunuz.

Amazon Web Services sunduğu hizmetlerin yazılım geliştirme kitelerini de Flutter'a uygun olarak dizayn ediyor. Bu doğrultuda iş ilanlarına Flutter geliştiricisi aradıklarını da çekinmeden belirtiyor. Ebay, yazdıkları blog yazısında Flutter kullanmanın uygulama geliştirme süresini 2 kat daha da hızlandırdıklarını ve Swift & Kotlin ile karşılaşlıklarını birçok problemi Flutter ile aşıklarını ifade ediyor.

Flutter ve Fuchsia OS

Fuchsia OS, Google tarafından geliştirilen ve yakın gelecekte Android işletim sisteminin yerini alması planlanan bir işletim sistemi. Android

cihazların yanında giyilebilir cihazların, IoT cihazların, ev içi akıllı cihazlara da güç verecek bir işletim sistemi olacak olan Fuchsia OS, Flutter ile entegre çalışacak. Yani Flutter geliştirme kiti ve Dart programlama dili Fuchsia OS'nin stabil çalışmasına katkı sağlayacak. Bu yeni işletim sisteminde de uygulama geliştirirken Kotlin veya Java değil, Dart programlama diliyle beraber Flutter kullanılacak.

Sonuç: Flutter öğrenmeye kesinlikle değer!

Flutter hızlıca gelişen bir araç ve devasa bir topluluğa sahip. Diğer cross platform uygulama geliştirme araçlarından çok daha hızlı çalışıyor. Hot reload gibi özelliklerle rakiplerinden katlarca üstün. Google tarafından desteklenen ve özellikle Microsoft, BMW, Ebay, Tencent, Alibaba gibi kendi pazarlarının ve sektörlerinin devlerinden olan şirketlerce kullanılıyor. Kısacası 2021 yılında yeni bir programlama dili veya framework öğrenmek istiyorsanız, Flutter tam da size göre!

Kaynak: Flutter'in Geleceği | Startup Dünyasını Nasıl Değiştirecek? <https://ahmethallac.com/flutterin-geleceği-startup-dunyasini-nasil-degistirecek> Erişim tarihi: 28/01/2022

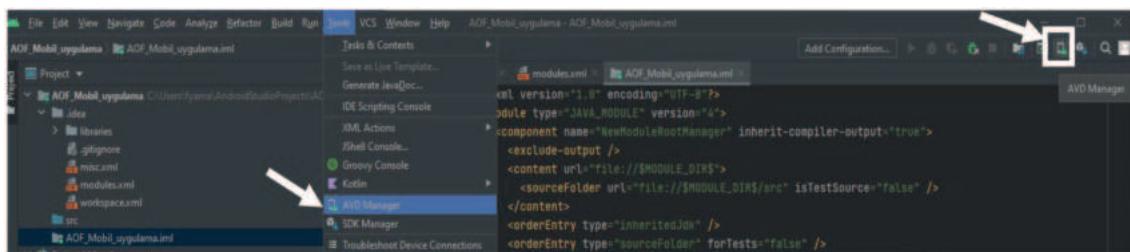


Öğrenme Çıktısı



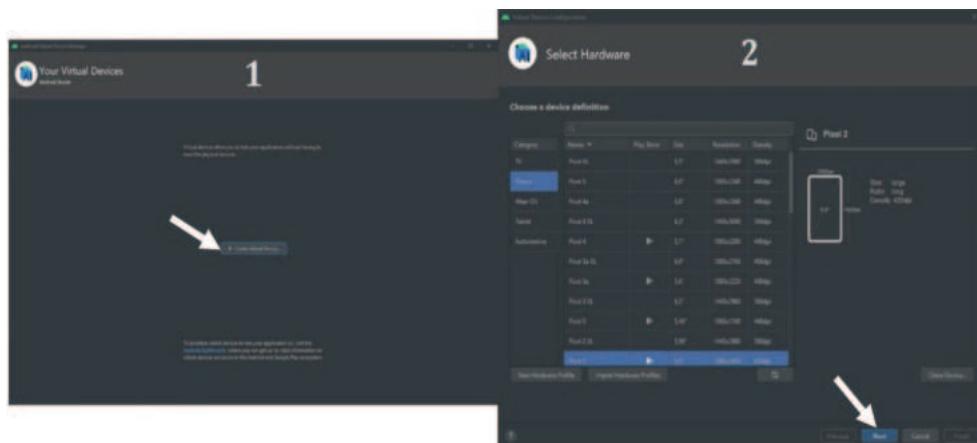
EMÜLATÖR KURULUMU

Emülatör veya sanal cihaz kurulumu yapılarak geliştirilen uygulama üzerinde anlık olarak mobil cihazlardaki görünümünün nasıl olacağı kontrol edilebilmektedir. Emülatör kurulumuna başlamak için iki yol vardır (Şekil 3.36).



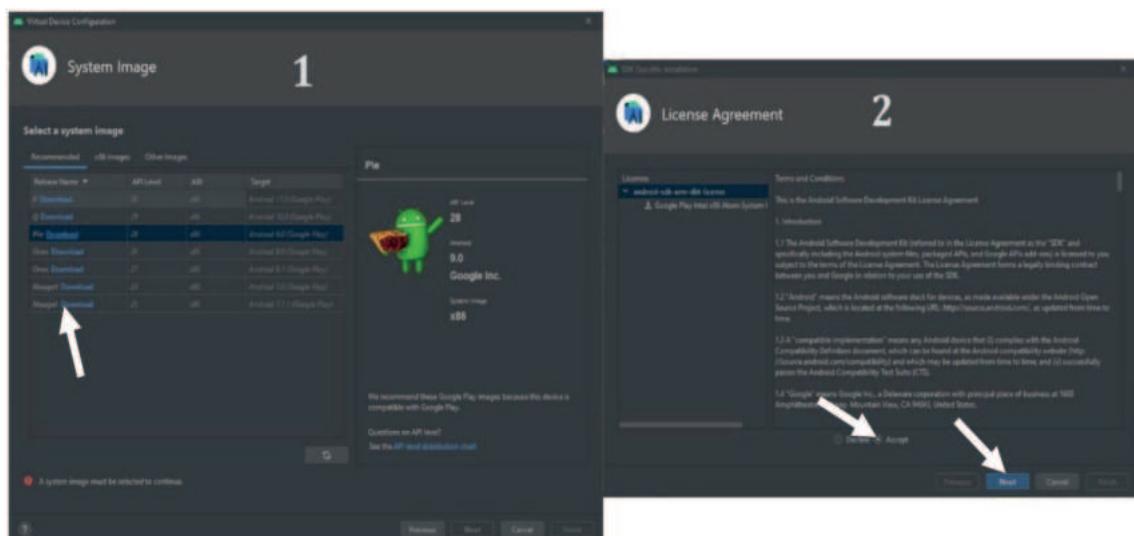
Şekil 3.36 Emülatör kurulumu - AVD Manager

Şekil 3.36'da görüldüğü üzere Android Studio'da “Tools → AVD Manager” adımları takip edilerek ya da araç çubuğunda bulunan “AVD Manager” bağlantısı ile emülatör kurulumuna başlanır. “AVD manager” bağlantısına tıklandığında Şekil 3.37'deki ekrana geçiş yapılır.



Şekil 3.37 Emülatör kurulumu - Cihaz seçimi

“AVD Manager” bağlantısından sonra gelinen ilk ekranda “Create Virtual Devices” butonuna tıklanır ve Şekil 3.37'deki ikinci ekrana geçiş yapılır. İkinci ekranda hazırlanan uygulamanın hangi platform (TV, akıllı telefon, giyilebilir teknoloji, tablet ve araç) için geliştirildiği kategori kısmından seçilmelidir. Kategori seçiminden sonra bu kategorideki cihazların asgari özellikleri (ekran boyutu, çözünürlük, yoğunluk) seçilir. Bu adımlardan sonra “Next” butonu ile kurulum devam edilir (Şekil 3.38).



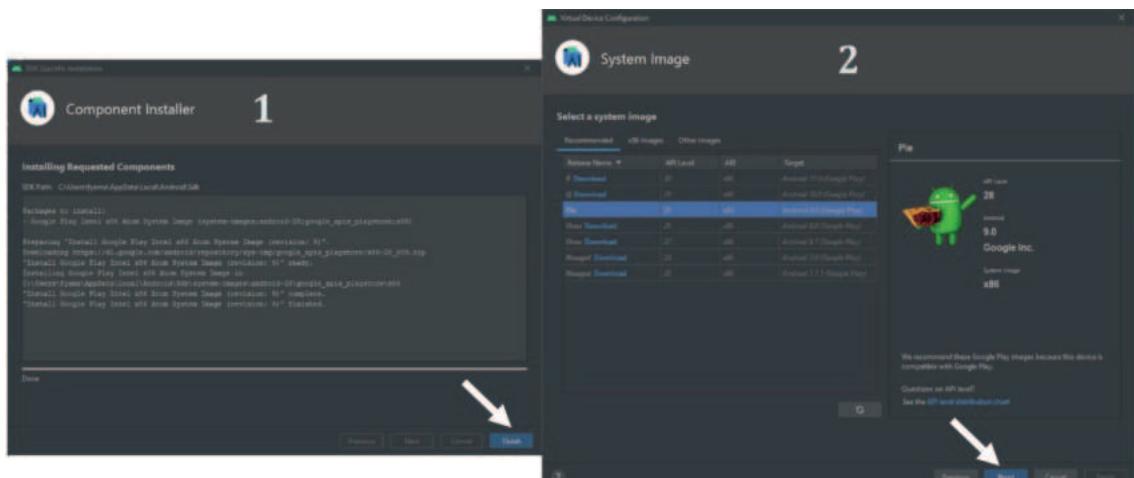
Şekil 3.38 Emülatör kurulumu - GörSEL seçimi

Uygulamanın kullanılacağı mobil cihaz özellikleri seçildikten sonra sistem görselinin seçilmesi ekranına geçirilir (Şekil 3.38). Burada devam edebilmek için sistem görselinin seçilmesi gerekmektedir. Seçim işlemi görsellerin yanında bulunan “Download” bağlantısına tıklanarak görselin indirilmesiyle gerçekleşmektedir. İlk ekranda “Download” bağlantısına tıklandığında ikinci ekrana geçilmektedir. Bu ekranda şartlar kabul edildikten (Accept) sonra “Next” butonu aktif olur ve bu butona tıklanır.



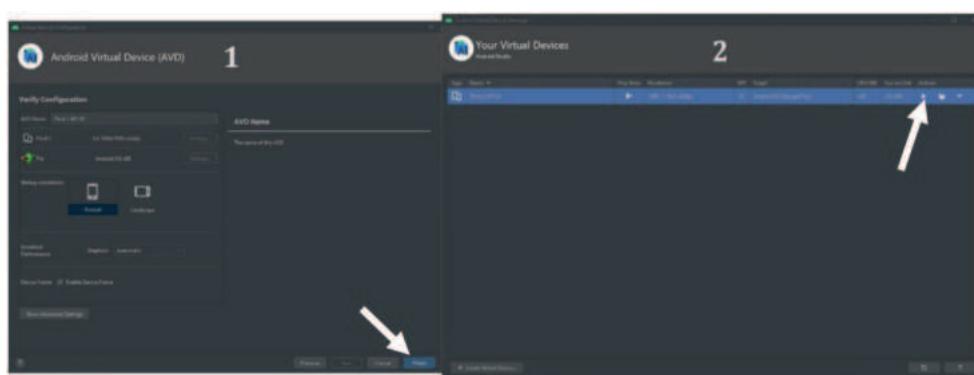
dikkat

Android Studio'da ilk defa sanal cihaz ekleniyorsa sistem görsellerinden birinin seçilerek indirilmesi gerekmektedir. Sonrasında sanal cihaz ekleme işlemi yapılırken önceden indirilmiş görseller kullanılabileceği gibi yeni görseller indirilerek sürece devam edilebilir.



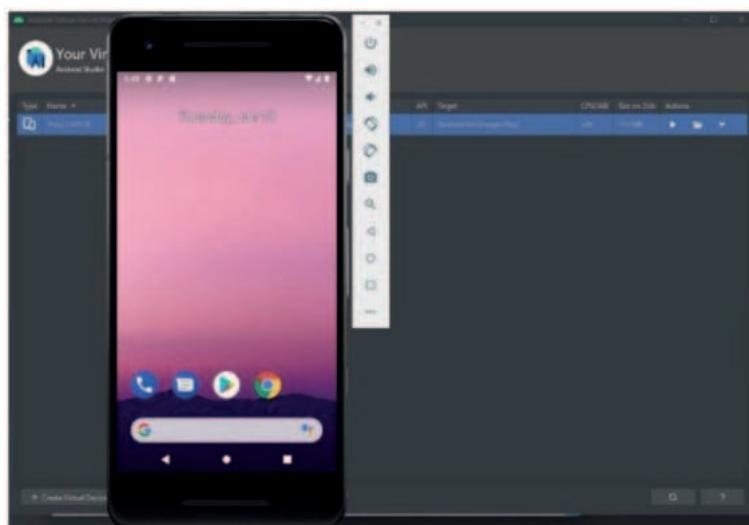
Şekil 3.39 Emülatör kurulumu - GörSEL bileşen yüklemesi

Görsel bileşen yüklemesi tamamlandığında Şekil 3.39'da birinci ekranda görüldüğü gibi “Finish” butonu aktif olmaktadır. “Finish” butonuna tıklandığında sistem görsel seçim ekranına geri dönülmektedir. Geri dönüldüğünde ilk açıldığında pasif durumda olan “Next” butonu aktif olarak görülmektedir. “Next” butonuna tıklanarak emülatör kurulumuna devam edilir (Şekil 3.40).



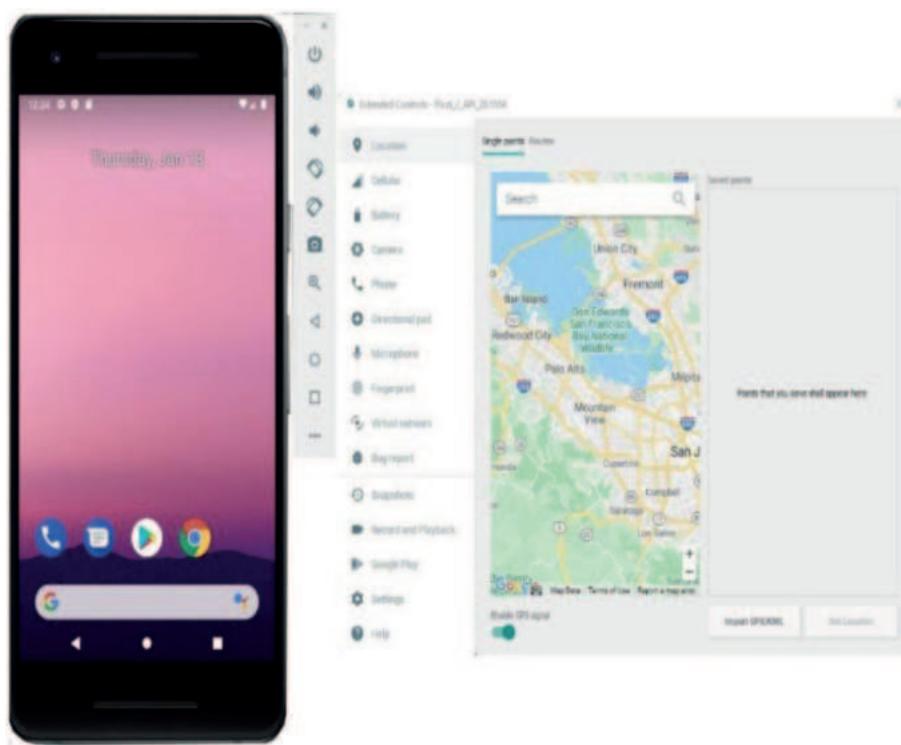
Şekil 3.40 Emülatör kurulum tamamlama

Emülatör kurulumunda sona gelinmiştir. Son kontrollerin yapılabacağı ekran Şekil 3.40'ta görülen birinci ekrandır. Birinci ekranın “Finish” butonuna tıklandığında ikinci ekran görünecektir. İkinci ekran Emülatör’ün sorunsuz yüklendiğini göstermektedir. İkinci ekranın “Play” butonuna tıklanarak sanal cihaza ulaşılabilmektedir (Şekil 3.41).



Şekil 3.41 Emülatör ekranı

Uygulamaların geliştirilmesi aşamasında uygulamanın kontrolü için kullanılacak sanal cihazın kurulumu tamamlandığında Şekil 3.41'de görüldüğü gibi seçilen cihaz yapısına benzer bir cihaz ekranında görünecektir. Uygulamaları kontrol ederken mobil cihazların sahip olduğu özelliklerin de denenebilmesi gerekmektedir.



Şekil 3.42 Emülatör özellikleri

Mobil cihazı açma-kapama, mobil cihazın sesini artırma-azaltma, mobil cihazı yan-dikey döndürme, mobil cihazda ekran görüntüsü alma, zoom yapma, geri alma, Home ve Overview butonlarıyla ana ekrana dönme ve açık uygulamaları görme gibi temel kontrol etme yapılabildiği gibi üç noktaya tıklandığında Şekil 3.42'de görüldüğü gibi ek özellikler de açılmaktadır. Bu özellikler ile uygulama kontrolü saflıklı yapılmaktadır.

Öğrenme Çıktısı



Araştır 3

Sanal cihazlarla çalışmanın üstünlükleri nelerdir?

5 Android Studio'da emülatör (sanal cihaz) kurabilme

İlişkilendir

Uygulama geliştirme için kullanılan sanal cihazlar gibi işletim sistemlerinin kurulabileceği sanal sürücüler de bulunmaktadır. Sanal sürücülerle sanal cihazların kullanım amaçları nelerdir?

Anlat/Paylaş

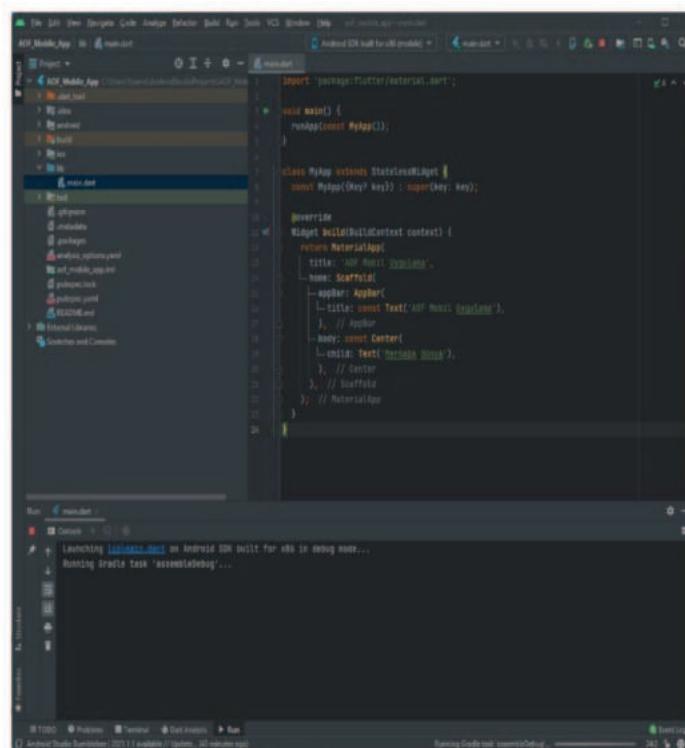
Uygulama geliştirirken sanal cihaz kullanma deneyimlerinizi paylaşınız.

ÖRNEK UYGULAMA GELİŞTİRME

Yazılımla ilgili bir proje yapıldığında ekrana öncelikle “Hello World” veya “Merhaba Dünya” yazısı yazdırılır. Kitabın ilerleyen bölümlerinde detaylı olarak uygulama oluşturma aşamalarına degeinilecektir. Bu nedenle bu bölümde ekrana “Merhaba Dünya” yazma uygulaması gerçekleştirilecektir.

```
import
'package:flutter/material.dart';
art';
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'AOF Mobil Uygulama',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('AOF Mobil Uygulama'),
        ),
        body: const Center(
          child:
Text('Merhaba Dünya'),
        ),
      );
    }
}
```



Sekil 3.43 “Merhaba Dünya” projesi

Merhaba Dünya projesini oluşturmak için Şekil 3.43'te sol tarafta bulunan kod Flutter projesinde **main.dart** dosyasındaki kodlar temizlenerek yapıştırılır ya da yazılır. Ardından araç çubuğundan öncelikle önceden oluşturululan emülatör seçilir (Şekil 3.44).



Sekil 3.44 Projeyi çalışma

Emülatör seçiminden sonra **main.dart** dosyasının aktif olduğu görülmeli altında da üçüncü adımda görünen play butonuna (Run) tıklanır. Proje dosyası çalıştırıldığında Şekil 3.45'te görüldüğü gibi ekran görünecektir.

“Merhaba Dünya” uygulaması ile bölümün sonuna gelinmiştir. Bölümde öncelikle Android Studio kurulumuna yer verilmiştir. Ardından Android Studio'nun arayüzü tanıtılmış ve eklenti olarak Flutter ve Dart eklentileri eklenmiştir. Bir Flutter projesinin nasıl oluşturulabileceği anlatıldıktan sonra geliştirilen uygulamanın mobil cihazlarda denenebilmesi için sanal cihaza gereksinim duyulmaktadır. Bu amaçla emülatör (AVD) kurulumu gerçekleştirilmiş ve en son olarak da “Merhaba Dünya” uygulaması geliştirilmiştir.



Şekil 3.45 “Merhaba Dünya” uygulaması ekran görüntüsü





Mobil uygulamalar, mobil cihazlarda özel fonksiyonları yerine getirmek için geliştirilen bilgisayar yazılımlarıdır. Mobil uygulama geliştirmek için birçok farklı program kullanılabilir. Android Studio bu programlardan yalnızca biridir. Android Studio'yu bilgisayara kurabilmek için kurulum dosyasının ihtiyaç duyulmaktadır. Bu amaçla <https://developer.android.com/studio> adresinden bilgisayarınızın işletim sistemine göre gerekli kurulum dosyası indirilerek kurulum sürecine geçilebilir. Ardından indirilen kurulum dosyasına çift tıklanarak kuruluma başlanır. Ekranda gelen uyarılarla göre işlemler yapılır ve Next butonu yardımıyla devam edilir ve en son Finish butonu ile kurulum tamamlanır. Kurulum tamamlandıktan sonra Android Studio programı açılır. İlk açılış ekranlarında Android Studio ile ilgili başlangıç ayarları değiştirilebilir. Gerekirse sonra da Android Studio menülerinden ayarlara ulaşarak değişiklikler yapılabilir. Android Studio arayüzüne ulaşabilmek için yeni bir proje oluşturmak gerekmektedir. Proje oluştururken geliştirilmesi planlanan mobil uygulamanın kullanılacağı cihaz (akıllı telefon, tablet, televizyon, giyilebilir cihaz veya araba) seçilir. Yeni proje oluşturulduktan sonra açılan ekran yazılım geliştirme ortamlarından aşina olunan arayüz ile kullanıcıları karşılamaktadır. En üstte menü çubuğu, altında araç çubuğu bulunmaktadır. Araç çubuğunun alt kısmında ise ekran ikiye bölünmüş yapıdadır. Sol panelde proje dosyalarına ulaşan bölüm ve sağ panelde ise tasarım için işlemlerin yapılabildiği bölüm yer almaktadır.



Flutter, Google tarafından geliştirilen açık kaynak kodlu uygulama geliştirme platformudur. Bu bölümde Flutter eklentisinin tercih edilme sebebi Flutter ile geliştirilen mobil uygulamaların çıktılarının hem Android hem de iOS işletim sistemli mobil cihazlara yönelik alınabilmesidir. Eklentinin Android Studio'ya eklenmesi çok kolaydır. İzlenmesi gereken adımlar şu şekilde sıralanabilir. File → Settings → Plugins. Bu adımlardan sonra Marketplace bölümünde Flutter eklentisi aratılır ve Install bağlantısı ile kurulum gerçekleştirilir. Flutter eklentisi Android Studio'ya ekleneceği zaman Dart eklentisine de ihtiyaç duyulmaktadır. Bu nedenle Dart eklentisinin kurulmasına da gelen bildirim ekranında onay verilerek kurulum gerçekleştirilir. Eklentilerin kurulumunda sonra <https://docs.flutter.dev/get-started/install/windows> ve <https://dart.dev/get-dart#windows> adreslerinden Flutter ve Dart SDK'ları indirilerek zip'li dosyalar açılır ve bilgisayarda C konumuna kopyalanırlar. C konumunda bulunan klasörlerin bin klasörlerinin adresleri Ortam Değişkeni olarak tanımlanmalıdır. Bu işlemlerden sonra Flutter'in sağlıklı çalışıp çalışmadığını kontrol etmek için CMD ekranında `flutter doctor` komutu ile kontrol sağlanır.

5

Android Studio'da emülatör (sanal cihaz) kurabilme

Emülatör Kurulumu

Emülatör veya sanal cihaz, mobil uygulama geliştirilirken uygulama geliştiricilerin işlerini kolaylaştıran bir özelliktir. Bu özellik ile uygulama geliştiriciler, mobil uygulama geliştirirken senkron olarak yaptıkları işlemlerin mobil cihazlarda nasıl görünüşünü görebilirler. Android Studio'da emülatör eklemek için Tools → AVD Manager adımları takip edilerek veya araç çubuğuunda bulunan AVD Manager bağlantısı ile emülatör kurulumuna başlanabilir. Yeni bir sanal cihaz ekleme adımıyla sürece başlanır ve geliştirilen uygulamanın hangi çözünürlükteki ekranada nasıl görüneceğini belirlemek gerekmektedir. Bu nedenle ekran çözünürlüğü, büyütülgü gibi işlemler seçilir ve Next butonu ile adımlar takip edilir. Sistem görseli seçme ekranında görsel seçimi yapılır ve adımlar takip edilerek en son Finish butonu ile süreç tamamlanır. Bu şekilde emülatör eklenmiş olur.

6

Örnek uygulama geliştirebilme

Örnek Uygulama Geliştirme

Kitabın bu bölümünde yazılım geliştirme uygulamalarında ilk olarak yapılan ekrana “Hello World” veya “Merhaba Dünya” ifadesi yazdırılır. Bu bölümde de ekrana “Merhaba Dünya” yazdırılmıştır. Bunun için File → New → New Flutter Project adımları takip edilmiştir. Lib klasöründe bulunan main.dart dosyasına ilgili kodlar yazılarak ekrana “Merhaba Dünya” yazısı yazdırılmıştır.

1 Aşağıdakilerden hangisi Android Studio'nun özelliklerinden biridir?

- A. Tüm iOS platformları için emülatör desteği sunma
- B. Vektör imajlar sağlama
- C. Asp.net desteği içerme
- D. Ücretli olma
- E. NDK içermeme

2 Android Studio kurulumu için gerekli olan özelliklerden ekran çözünürlüğü Windows, MacOS ve Linux işletim sistemleri için minimum kaçtır?

- A. 800 x 600
- B. 1024 x 768
- C. 1280 x 800
- D. 1400 x 1050
- E. 1920 x 1080

3 I. Dracula

- II. Mercan Yeşili
- III. Light

Yukarıdakilerden hangisi ya da hangileri Android Studio kullanıcı ara yüz renk seçeneklerindedir?

- A. Yalnız I
- B. Yalnız II
- C. Yalnız III
- D. I ve III
- E. II ve III

4 Android Studio'da bir mobil uygulama geliştirirken aşağıdakilerden hangisi için uygulama **geçitlenmez**?

- A. Tablet
- B. Akıllı Televizyon
- C. Giyilebilir Teknoloji
- D. Otomobil
- E. Analog Saat

5 Android Studio menüleri kullanılarak aşağıdakilerden hangisi **yapılamaz**?

- A. Var olan proje bilgisayardaki konumundan bulunarak tekrar açılabilir.
- B. Kodlar denetlenebilir, dipnotlardan anlam çıkarılabilir.
- C. Yazı fontu değiştirilebilir.
- D. APK analiz edilebilir.
- E. Üzerinde çalışılan proje çalıştırılabilir.

6 Android Studio ara yüzünde araç çubuğunda yer alan Debug app ile aşağıdakilerden hangisi yapılabılır?

- A. Üzerinde çalışılan projedeki hatalar ayıklanır.
- B. Uygulamaların çalıştırılmasında kaynak kodların test edilmesi sağlanır.
- C. Hazırlanan projenin çalıştırıldığı ortamdaki testine bakılır.
- D. Üzerinde çalışılan projeye hata ayıklama aracı olarak tanımlanan debuggerler eklenir.
- E. Bilgisayara kurulan Android SDK ve JDK bilenşenlerinin kurulu olduğu konuma erişilmesi sağlanır.

7 I. Esnek yapıya sahip SDK'dır.

- II. Android ve iOS işletim sistemli cihazlarda kullanılan widget destegine sahiptir.
- III. Hot reload özelliğine sahiptir.

Yukarıdakilerden hangisi ya da hangileri Flutter'ın özelliklerindendir?

- A. Yalnız I
- B. Yalnız II
- C. Yalnız III
- D. I ve II
- E. I-II ve III

8

Flutter eklentisinin Android Studio'ya eklenmesi için izlenmesi gereken adımlar aşağıdakilerden hangisinde sırasıyla doğru olarak verilmiştir?

- A. File → Settings → Marketplace → Plugins → “Flutter” araması yapma → Install
- B. File → Settings → Plugins → Marketplace → “Flutter” araması yapma → Install
- C. File → Plugins → Settings → Marketplace → “Flutter” araması yapma → Install
- D. File → Marketplace → Settings → Plugins → “Flutter” araması yapma → Install
- E. File → Plugins → Settings → Marketplace → “Flutter” araması yapma → Install

9

Flutter eklentisinin Android Studio'ya eklenmesi için gerekli olan eklenti aşağıdakilerden hangisidir?

- A. IdeaVim
- B. Scala
- C. Dart
- D. JPA Buddy
- E. Protobuf

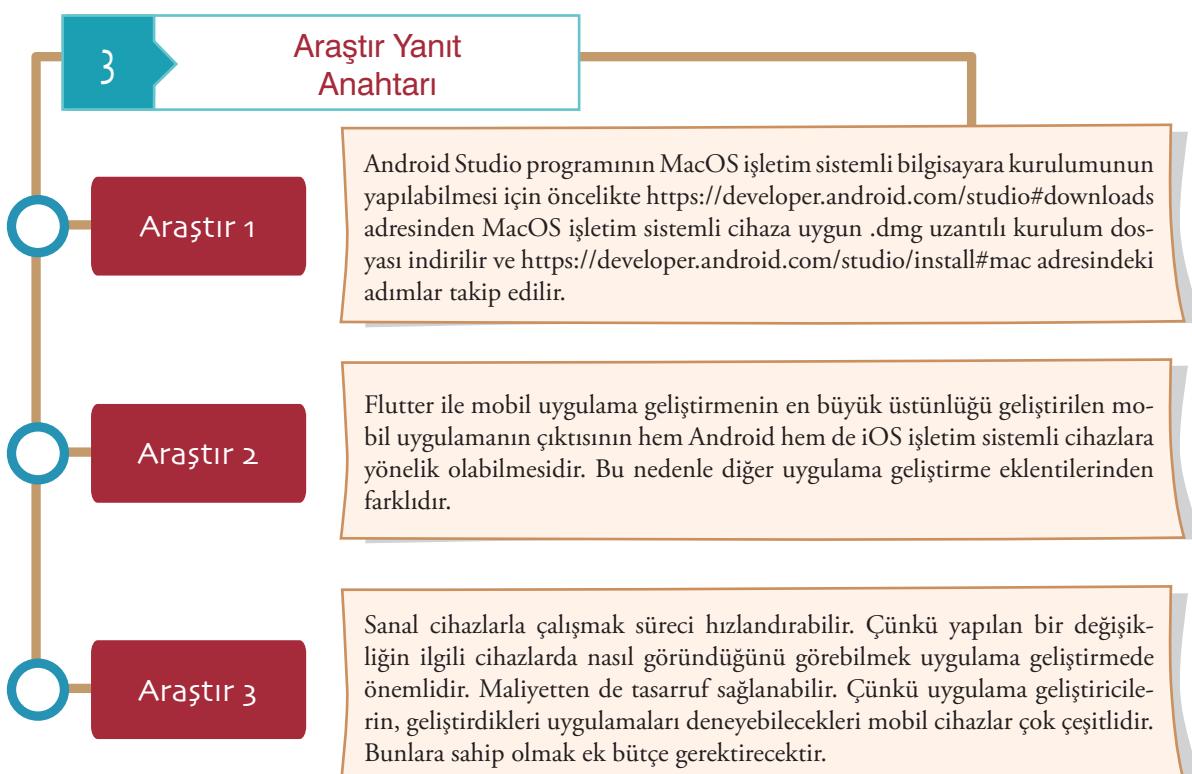
10

- I. Mobil cihazı yan-dikey döndürme
- II. Mobil cihazda ekran görüntüsü alma
- III. Açık uygulamaları görme

Yukarıdakilerden hangisi veya hangileri Android Studio'ya eklenen emülatör ile yapılmaktadır?

- A. Yalnız I
- B. Yalnız II
- C. Yalnız III
- D. I ve III
- E. I-II ve III

1. B	Yanıtınız yanlış ise “Android Studio Kurulumu ve Arayüz Özellikleri” konusunu yeniden gözden geçiriniz.	6. A	Yanıtınız yanlış ise “Android Studio Arayüz Özellikleri” konusunu yeniden gözden geçiriniz.
2. C	Yanıtınız yanlış ise “Android Studio Kurulumu ve Arayüz Özellikleri” konusunu yeniden gözden geçiriniz.	7. E	Yanıtınız yanlış ise “Flutter Eklentisinin Kurulumu” konusunu yeniden gözden geçiriniz.
3. D	Yanıtınız yanlış ise “Android Studio’nun Bilgisayara Kurulumu” konusunu yeniden gözden geçiriniz.	8. B	Yanıtınız yanlış ise “Flutter Eklentisinin Kurulumu” konusunu yeniden gözden geçiriniz.
4. E	Yanıtınız yanlış ise “Android Studio’nun Bilgisayara Kurulumu” konusunu yeniden gözden geçiriniz.	9. C	Yanıtınız yanlış ise “Flutter Eklentisinin Kurulumu” konusunu yeniden gözden geçiriniz.
5. C	Yanıtınız yanlış ise “Android Studio Arayüz Özellikleri” konusunu yeniden gözden geçiriniz.	10. E	Yanıtınız yanlış ise “Emülatör Kurulumu” konusunu yeniden gözden geçiriniz.





Öncelikle kullanılacak olan görsel bilgisayara indirilir ve proje dosyasının içerişine images, görseller veya resimler isminde bir klasör oluşturulabilir. Oluşturulan klasör içerişine indirilen görsel eklenir. Ardından Android Studio'da proje dosyalarının olduğu bölümde bulunan "pubspec.yaml" dosyasında eklenen görselin tanıtımı yapılır. Ardından

```
# assets:  
#   - images/a_dot_burr.jpeg  
bölümü  
assets:  
- images/dunya.jpg
```

olarak güncellenir. Bu güncellemenin işlenmesi için tasarım panelinde üst kısımda bulunan "Pub get" bağlantısına tıklanır. Ardından tasarım paneline aşağıdaki kod eklenir.

```
Container(  
    width: 222,  
  
    height: 222,  
    child: column(  
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
        children: <Widget> [  
            Image.asset("assets/images/dunya.jpg",width: 100,height: 100,),  
        ]  
    ),  
,
```

Kaynakça

Türkiye İstatistik Kurumu, (2021). *Hanehalkı Bilişim Teknolojileri (BT) Kullanım Araştırması, 2021*. [https://data.tuik.gov.tr/Bulten/Index?p=Hanehalki-Bilisim-Teknolojileri-\(BT\)-Kullanim-Arastirmasi-2021-37437](https://data.tuik.gov.tr/Bulten/Index?p=Hanehalki-Bilisim-Teknolojileri-(BT)-Kullanim-Arastirmasi-2021-37437) adresinden 15.11.2021 tarihinde erişilmiştir.

Bölüm 4

Mobil Uygulamalarda Arayüz Bileşenleri

Öğrenme Çıktıları

Diyalog Ekranları

- 1 Diyalog ekranlarını kullanabilme ve farklı amaçlar için kullanılan diyalog ekranlarını tanıyalabilme

Liste Kontrolleri

- 3 Listelerle işlem yapabilmeyi ve ekranda listelerin nasıl kontrol edileceğini öğrenebilme

İkon Kullanımı

- 5 İkonların nasıl kullanılacağını öğrenebilme

Buton Kontrolleri

- 2 Farklı şekillere sahip olan butonları tanıyalabilme ve özelliklerini öğrenebilme

Grafikler, Stiller ve Temalar

- 4 Grafikleri, stilleri ve temaları kullanabilme

Layout

- 6 Layout yaklaşımını uygulayabilme

Anahtar Sözcükler: • Diyalog Ekranları • Liste Kontrolleri • Temalar • İkon Kullanımı • Layout • Widget
• Flutter • Dart



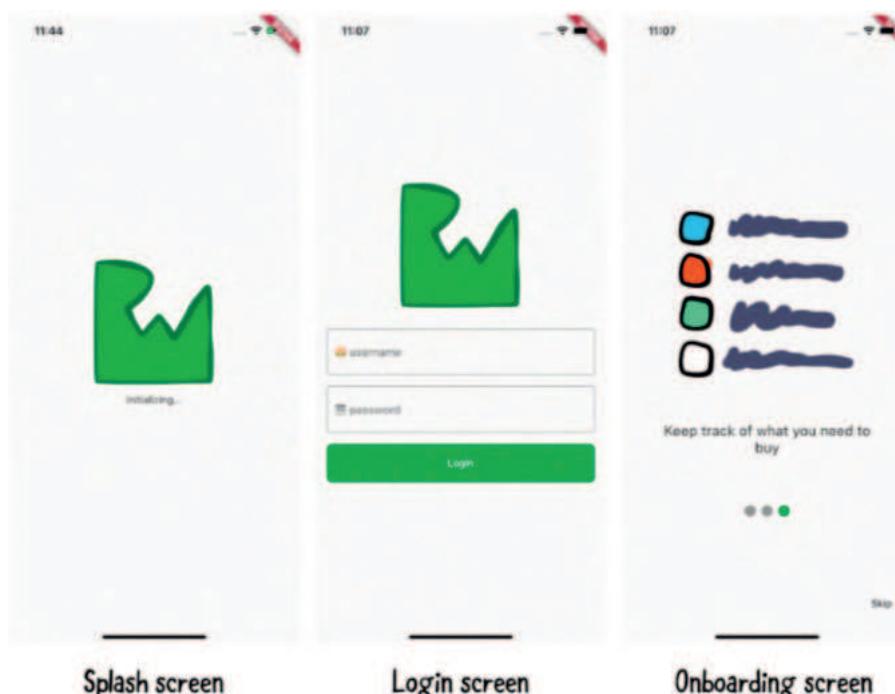
GİRİŞ

Flutter'da arayüz bileşenlerinin amacı kullanıcı etkileşimi sağlayarak kullanıcıların uygulamadaki bildirimleri, uygulamanın ürettiği çıktıları uygulama içinde farklı yerlerde kullanarak kullanıcıyla etkileşimi artırmaktır. Temel arayüz bileşenleri diyalog ekranları, buton kontrolleri, liste yapısı ve kontrolleri, stilleri ve temaları, ikon kullanımı ile programdaki ekran çıktılarının **kullanıcı dostu** bir arayüz tasarlamak için kullanılan araçlardır. Şekil 4.1 de görüldüğü üzere arayüz bileşenleriyle farklı ekranlar tasarlanabilir.

Kullanıcı Dostu: Kullanıcı dostu uygulama tasarımları kullanıcılarla en kısa yoldan ulaşmak istediği noktalara ulaşır ekran tasarım kurgusudur. Burada kullanıcı uygulama içerisinde dolaşmadan ve kısa sürede her istediği noktaya ulaşabilmesi çok önemlidir.

HotReload/HotRestart: Flutter'in bize sunduğu HotReload ve HotRestart özellikleri sayfada yaptığımız değişikleri uygulamayı derlemeden görmemizi sağlar. HotReload sayfadaki sadece isminden de anlaşıldığı gibi sadece sayfa yenileme yapar uygulamada değişkenler girişler sıfırlanmaz bu durumda. HotRestart ise uygulamayı tekrardan build eder bütün uygulamanın içerisindeki girişler ve değişkenler sıfırlanır.

dikkat
HotReload ve HotRestart yaparken uygulamada kodda hata olup olmadığına dikkat edin aksi halde reload ve restart'ı kullanırken bir hata gözmeyecek ve gözden kaçacaktır.



Şekil 4.1 Arayüz Bileşenleri ile Farklı Amaçlar İçin Sayfalar Tasarlayabiliriz.

Kaynak: raywenderlich Tutorial Team, Mike Katz, Kevin David Moore, Vincent Ngo - Flutter Apprentice (First Edition) _ Learn to Build Cross-Platform Apps-Razeware LLC (2021) page:275

DİYALOG EKRANLARI

Diyalog ekranları Flutter'da kullanıcılar kritik mesajları bildirmek için veya belirli durumlara karşı vermelerini sağlamak için kullanılır. Diyalog ekranları, ekranın kullanıcıya gösterildiğinde arkadaki sayfa açık kalmaya devam eder ve verilen aksiyona göre istenildiği gibi işlemler yapılır. AlertDialog, SimpleDialog, ShowDialog olmak üzere üç farklı diyalog ekranı türü vardır. Bununla birlikte diyalog ekranlarının tasarımları da Şekil 4.2 görüldüğü üzere Material Design ve Cupertino Design olarak iki farklı biçimde kullanılabilir.

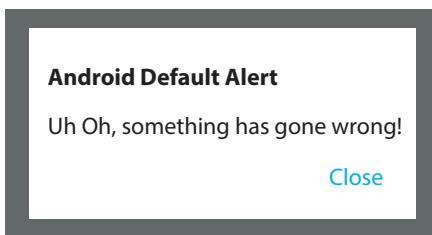


Diyalog ekranları show Dialog() fonksiyonuya kullanılması gerekmektedir. Çünkü ekrana farklı bir sayfa basılacaktır ve bununda context ağaçına bağlanması gereklidir. Buradaki fonksiyonda tam olarak bu işlemi yapmaktadır.



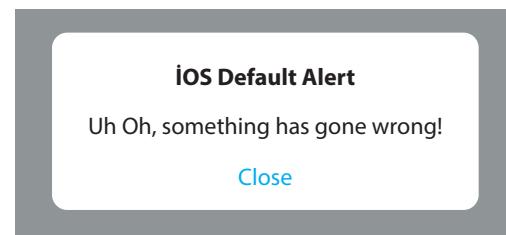
UI Style: Flutter kullanıcılarına Material ve Cupertino olmak üzere iki farklı arayüz sağlamaktadır. Material stili Android'de ki gibi bir tasarım çıktıları verir, Cupertino stili ise iOS'da ki gibi bir tasarım çıktıları verir. Şekil 4.2'de görüldüğü üzere.

Android



Material Design

iOS



Cupertino Design

Şekil 4.2 Bir alert uyarısı, material veya cupertino tasarımları olarak farklılaşır.

Kaynak: Simone Alessandria, Brian Kayfitz - Flutter Cookbook_ Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart-Packt Publishing (2021)

AlertDialog

Bir uyarı diyaloğu ekranı kullanıcıya herhangi bir koşul gerektiren durumda ekranın üzerinde uyarı hakkında bilgi verir ve isteğe bağlı seçenekler oluşturur. Kullanıcının hangi eylemi seçtiğine göre programda aksiyon alınabilir. Uyarı diyaloğlarının alabildiği dört adet özellik vardır bunlar Title, Action, Content ve Shape' dir.

Title başlığı belirleyen parametredir içerisindeki verilen Widget'ı başlık olarak diyaloğun en üstünde oluşturur. Content ise içerik kısmını şekillendirir. Kullanıcının hangi işlemi yaptığı hakkında bilgi vermek içindir. Content içerisinde Text Widget verip kullanılabilir.

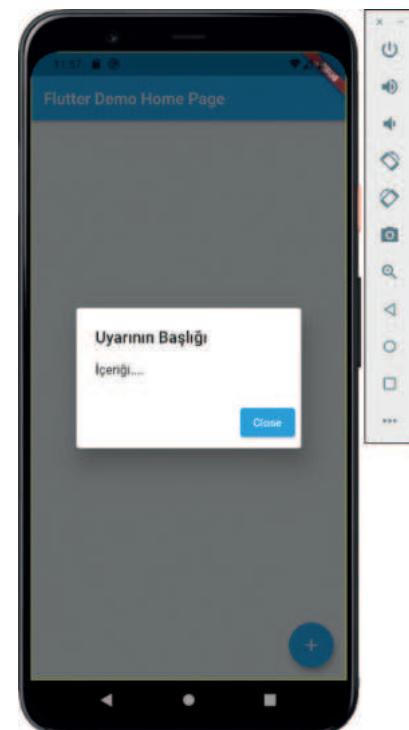
Action parametresinde ise kullanıcıya sunulan seçenekler belirlenir. Örnek olarak bir galeri uygulamasında fotoğraf silme uyarısı için kullanıcının silmekte emin olup olmadığını belirleneceği seçenekler burada verilir. Action parametresine liste içerisinde Widget'lar verilebilir. Bunlar buton olabilir. Shape parametresi ise diyaloğın ekranını şekillendirmek için kullanılır. Örneğin aşağıdaki kod parametre olarak girildiğinde diyaloğın kenarları ovalleşir.

```
RoundedRectangleBorder(borderRadius:BorderRadius.all(32.0))
```

```
void dialog() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: new Text("Uyarının Başlığı"),
        content: new Text("İçeriği...."),
        actions: [
          ElevatedButton(
            child: new Text("Close"),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
        ],
      );
    },
  );
}
```

Yukarıdaki kodda görüldüğü üzere Title parametresi olarak içerisinde bir Text Widget verilmiş ve onunla içerisinde String bir değer olan “Uyarının Başlığı” girilmiştir. Content parametresine ise yine bir Text Widget verilip içerisinde String bir değer girilmiştir. Actions parametresi olarak da liste içerisinde ileriki ünite anlatılacak olan bir buton olan ElevatedButton Widget’ı verilmiştir. Şekil 4.3’ de görüldüğü üzere dialog() fonksiyonu çağrıldığında ve bu kodlar çalıştırıldığında aşağıdaki görüntü görülecektir. Özellikleri aşağıdaki gibidir:

- actions
- actionsAlignment
- actionsOverflowButtonSpacing
- actionsOverflowDirection
- actionsPadding
- alignment
- backgroundColor
- buttonPadding
- clipBehavior
- content
- contentPadding
- contentTextStyle
- elevation
- hashCode
- insetPadding



Şekil 4.3 Örnek olarak verilen AlertDialog Kodunun Ekran Çıktısı.

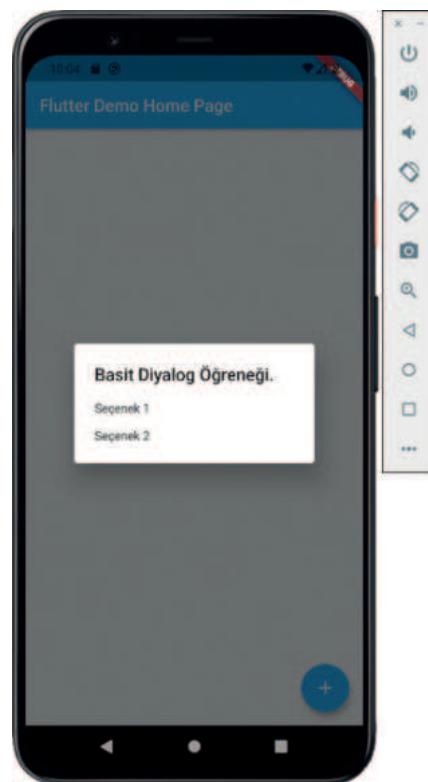
SimpleDialog

Bu basit diyalog ekranı kullanıcıya çoklu seçim işlemi sunulması gerekiğinde kullanılmalıdır. Bu Widget'in Title, Shape, backgroundcolor biçiminde üç temel özelliği bulunmaktadır. Title adından da anlaşılacağı üzere başlık olarak içerisinde verilen Widget'ı diyalog ekranının üstünde gösterir. Shape diyalog ekranını şekillendirmek için kullanılır. Backgroundcolor ise diyalog ekranının arka plan rengini belirlemeye yarar. Basit bir iletişim kutusu, kullanıcıya çeşitli seçenekler arasında bir seçim sunar. Basit bir iletişim kutusunda, seçeneklerin üzerinde görüntülenen isteğe bağlı bir başlık bulunur. Kullanıcıyı bir durum hakkında bilgilendiren diyaloglar için ise AlertDialog kullanılmalıdır. Basit diyaloglar genellikle, iletişim kutusunu görüntüleyen showDialog'a alt pencere ögesi olarak ilettilir.

```
void dialog() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return SimpleDialog(
        title: const Text('Basit Diyalog Örneği.'),
        children: <Widget>[
          SimpleDialogOption(
            onPressed: () {},
            child: const Text('Seçenek 1'),
          ),
          SimpleDialogOption(
            onPressed: () {},
            child: const Text('Seçenek 2'),
          ),
        ],
      );
    });
}
```

Yukarıdaki kodda bir SimpleDialog Widget'ı kullanılmıştır. Title parametresi olarak Text Widget verilmiş ve onun içerisinde String değer olarak "Basit Diyalog Örneği." Verilmiştir. Children olarak SimpleDialogOption Widget'ı verilmiştir. Bunlar seçeneklerin şekillendirildiği kısımdır. SimpleDialogOption Widget'ının içerisinde child olarak Text Widget'ına değerler girilmiştir. Bu kodlarla Şekil 4.4 gibi bir ekran çıktısı alınabilir. Özellikleri aşağıdaki gibidir:

- alignment
- backgroundColor
- children
- clipBehavior
- contentPadding
- elevation
- hashCode
- insetPadding
- key
- runtimeType
- semanticLabel
- shape
- title
- titlePadding
- titleTextStyle



Şekil 4.4 Örnek olarak verilen SimpleDialog Kodunun Ekran Çıktısı.

showDialog

Bu fonksiyon AlertDialog ve SimpleDialog Widget'larını ekrana bastırmak için gerekli bir fonksiyondur. Bu fonksiyon ekrana diyalog ekranlarını popup şeklindebastırır. ShowDialog'un builder fonksiyonuyla istenilen Widget ekrana bastırılabilir. Alert diyalog ve SimpleDialog örnek kodlarında görüldüğü üzere bu widgetlar showDialog builder fonksiyonunun içerisinde yazılır.



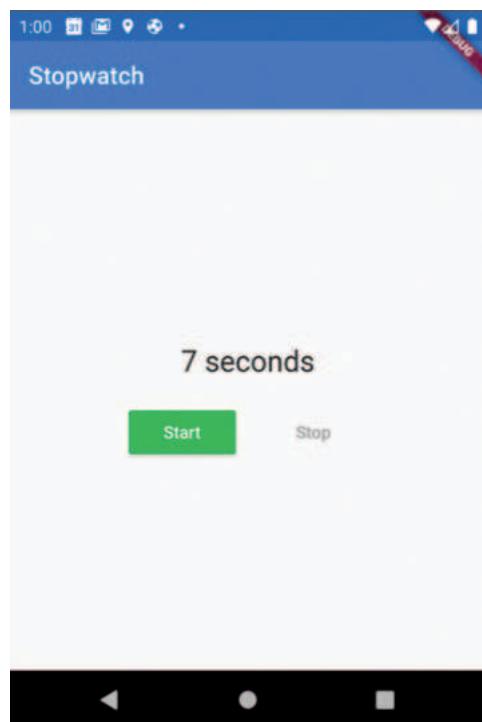
showDialog fonksiyonunu kullanmadan ekrana bir diyalog ekranı bastırılamaz.

Öğrenme Çıktısı



BUTON KONTROLLERİ

Buttonlar Flutter'da kullanıcıların etkileşimde bulunmalarını, seçme işlemlerini yaptırmak için kullanılır. Flutter SDK tarafından bir çok farklı şekilde butonlar sağlanır. Bunlar TextButton, ElevatedButton, OutlinedButton, IconButton, FloatingActionButton'dur. Butonlarda şekil haricinde büyük farklılıklar yoktur. Buton Widget'ının içerisinde child olarak istenilen Widget konulabilir.



Şekil 4.5 Basit bir kronometre uygulaması, butonlarla kronometre durdurulup başlatılabilir.

Kaynak: Simone Alessandria, Brian Kayfitz - *Flutter Cookbook_ Over 100 proven techniques and solutions for app development with Flutter 2.2* and Dart-Packt Publishing (2021)



Butonları eklediğimizde bir Click Event'i tanımlamak gereklidir. Aksi halde Flutter hata verecektir.

Column(

```

        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
            TextButton(
                onPressed: () {},
                child: Text('Text Butonu'),
            ),
            ElevatedButton(
                onPressed: () {},
                child: Text('Elevated Butonu'),
            ),
            OutlinedButton(
                onPressed: () {},
                child: Text('Outlined Butonu'),
            ),
            IconButton(
                icon: Icon(Icons.star),
                onPressed: () {},
            ),
            FloatingActionButton.extended(
                onPressed: () {},
                label: Text('Floating Action Butonu'),
            ),
        ],
    )
)

```

Yukarıdaki kodda farklı buton çeşitleri örnek olarak verilmiştir. Butonlarda görüldüğü üzere onPressed parametresine isimsiz fonksiyon verilmiş ve istenirse süslü parantez içerisine istenilen kodlar yazılabilir. Bu kodlar çalıştırıldığında ekranda şekil 4.6 deki gibi bir ekran çıktısı alınır.

```

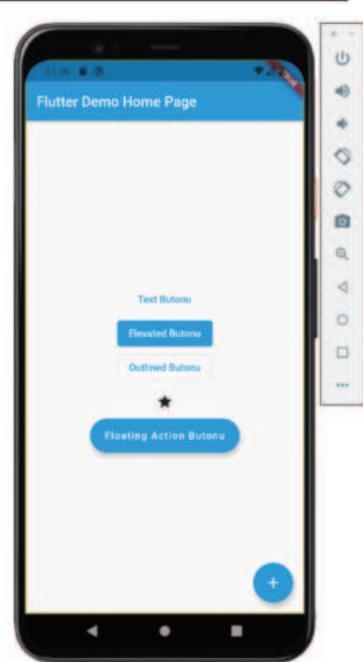
ElevatedButton(
    onPressed: () {print("Hello World.");},
    child: Text('Elevated Butonu'),
),

```

Bu örnekte butona basıldığında, consol çıktısında “Hello World.” yazacaktır.

Aşağıda ElevatedButton'un özellikleri verilmiştir:

- autofocus
- child
- clipBehavior
- enabled
- focusNode
- hashCode
- key
- onFocusChange
- onHover
- onLongPress
- onPressed
- runtimType
- style

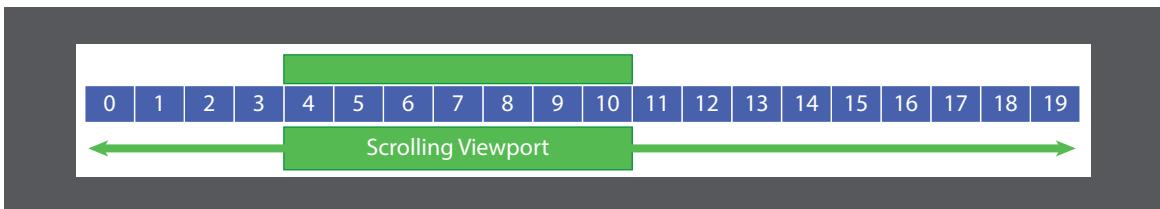


Şekil 4.6 Örnek olarak verilen buton kodunun Ekran Çıktısı.



LİSTE KONTROLLERİ

ListView'lar Flutter'da çoklu verileri görüntülemeye yarar. Ekrana sığmayacak kadar çok veri olduğunda bunlar liste içinde kaydırılabilir bir yapı halinde sunulabilir. Bu gibi durumlarda Flutterda ListView, ListView.builder diye iki adet kullanışlı Widget vardır. ListView daha çok sabit veri olduğunda kullanılır dinamik bir yapısı olmamasına rağmen ListView.builder dinamik bir yapıya sahiptir. Şekil 4.7 da bir kaydırılabilir(scroll) edilebilir bir bakış açısı verilmiştir.



Şekil 4.7 Scroll olabilir bir Widget'ın bakış açısı.

Kaynak: Simone Alessandria, Brian Kayfitz - *Flutter Cookbook_ Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart*-Packt Publishing (2021)

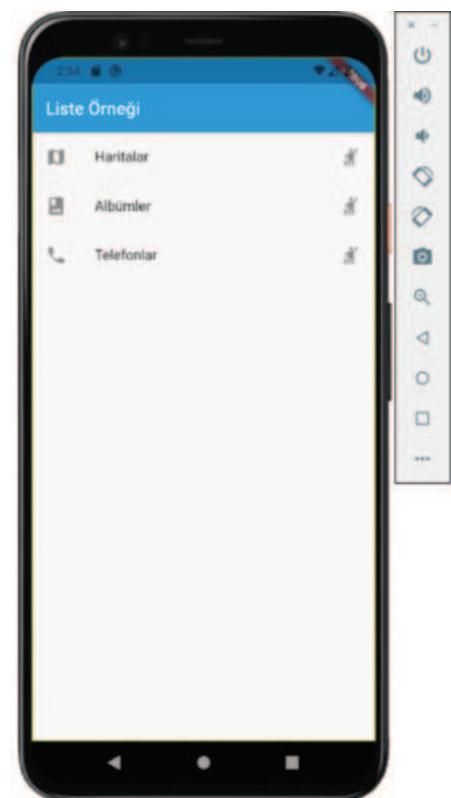
ListView

ListView Widget'i children olarak atanmış elemanları bir liste şeklinde göstermek için kullanılır. Bunlar resim, yazı vs. olabilir. Hazır olarak FlutterSdk tarafından oluşturulmuş ListTile Widget'ı da bu iş için kullanılabilir.

```
ListView(  
    physics:  
        BouncingScrollPhysics(parent: AlwaysScrollableScrollPhysics()),  
    children: [  
        ListTile(  
            trailing: Icon(Icons.hail),  
            leading: Icon(Icons.map),  
            title: Text('Haritalar'),  
        ),  
        ListTile(  
            trailing: Icon(Icons.hail),  
            leading: Icon(Icons.photo_album),  
            title: Text('Albümler'),  
        ),  
        ListTile(  
            trailing: Icon(Icons.hail),  
            leading: Icon(Icons.phone),  
            title: Text('Telefonlar'),  
        ),  
    ],  
),
```

Yukarıdaki kod parçasında görüldüğü üzere ListView children olarak liste alır. Bu listenin içerisinde hazır bir List Widget’ı olan ListTile Widget’ı kullanılmıştır. ListTile Widget’ı içerisinde burada trailing, leading, title verilmiştir. Trailing sağda, leading solda gözüken ikon, title ise başlıktır. Physics parametresi ile ListView’ın fiziksel özellikleri belirlenebilir içerisinde verilen parametre ile listenin aşağıya yukarıya kaydırıldığında liste elemanlarının ekranın içinde akışkan bir şekilde gelmesini sağlar. Şekil 4.8 de örnek kodun çıktısını görebiliriz. Özellikleri aşağıdaki gibidir:

- key
 - scrollDirection
 - reverse
 - controller
 - primary
 - physics
 - shrinkWrap
 - padding
 - itemExtent
 - prototypeItem
 - addAutomaticKeepAlives
 - addRepaintBoundraies
 - addSemanticIndexes
 - cacheExtent
 - children
 - semanticChildCount
 - dragStartBehavior
 - keyboardDismissBehavior
 - restorationId
 - clipBehavior



Şekil 4.8 Örnek olarak verilen ListView kodunun Ekran Çıktısı.

ListView.builder

`ListView.build` ise `ListView`'den daha kullanışlıdır. İçerisine index vererek o index'e göre `ListView`.
`builder`'ın içerisinde işlem yaptırılabilir.



dikkat

Bu Widget'i kullanırken `itemCount` özelliğinin yanlış girilmemesi
lazım. Aksi halde index hatası alınacaktır.

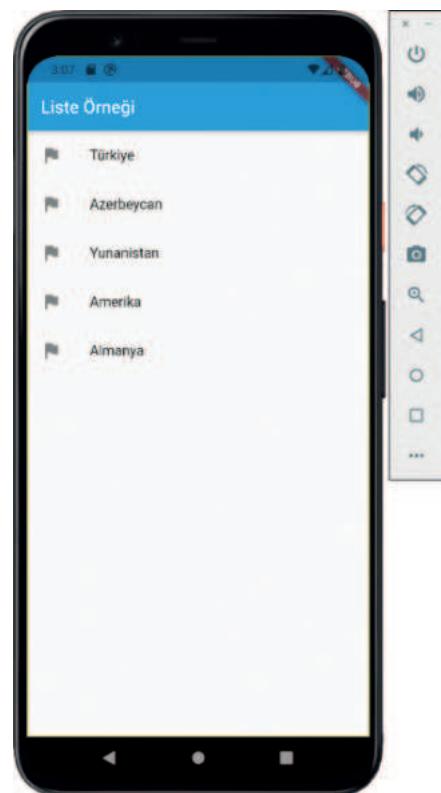
```
var children3 = ["Türkiye", "Azerbaycan", "Yunanistan", "Amerika", "Almanya"];
```

Yukarıdaki kodda `ListView.builder` Widget'ında kullanılmak için bir liste tanımlanmıştır.

```
ListView.builder(  
    itemCount: children3.length,  
    physics:  
        BouncingScrollPhysics(parent: AlwaysScrollableScrollPhysics()),  
    itemBuilder: (context, index) {  
        return ListTile(  
            leading: Icon(Icons.flag),  
            title: Text(children3[index]),  
        );  
    },  
,
```

`ListView.builder` fonksiyonu burada `ListView`'den farklı olarak içerişine `itemBuilder` parametresi alır. `itemBuilder` ise içerisinde `context` ve `index` değeri alır ve `index` değerine göre `ListView`'ın elemanlarını oluşturur. `itemCount` parametresi ise `ListView`'ın kaç elemanlı olduğunu belirler. Verilen örnek kod ile Şekil 4.8 de görüldüğü gibi bir çıktı alınabilir. Özellikleri aşağıdaki gibidir:

- key
- scrollDirection
- reverse
- controller
- primary
- physics
- shrinkWrap
- padding
- itemExtent
- prototypeItem
- itemBuilder
- itemCount
- addAutomaticKeepAlives
- addRepaintBoundaries
- addSemanticIndexes
- cacheExtent
- semanticChildCount
- dragStartBehavior
- keyboardDismissBehavior
- restorationId
- clipBehavior



Şekil 4.9 Örnek olarak verilen `ListView`.
`builder` kodunun Ekran Çıktısı.

Öğrenme Çıktısı



GRAFİKLER, STİLLER VE TEMALAR

Bu başlık altına grafikler, stiller ve temalar kavramlarına değinilecektir.

Grafikler

Grafikler ve istatistikler flutter'da verileri görselleştirmek için kullanılır. Verileri anlaşılır hale getirmek için çeşitli egriler / çizelgeler / grafikler / çizimleri ekrana çizilir. Bu kitapta grafikler için charts_flutter adlı paketi kullanılacaktır. Uygulamanın içine eklemek için bu paketi, projenin ana dizinindeyken “flutter pub add charts_flutter” komutu girilir. Ardından “import ‘package:charts_flutter/flutter.dart’ as charts;” komutu grafik kullanılmak istenilen flutter dosyasının en üstüne eklenir.

```

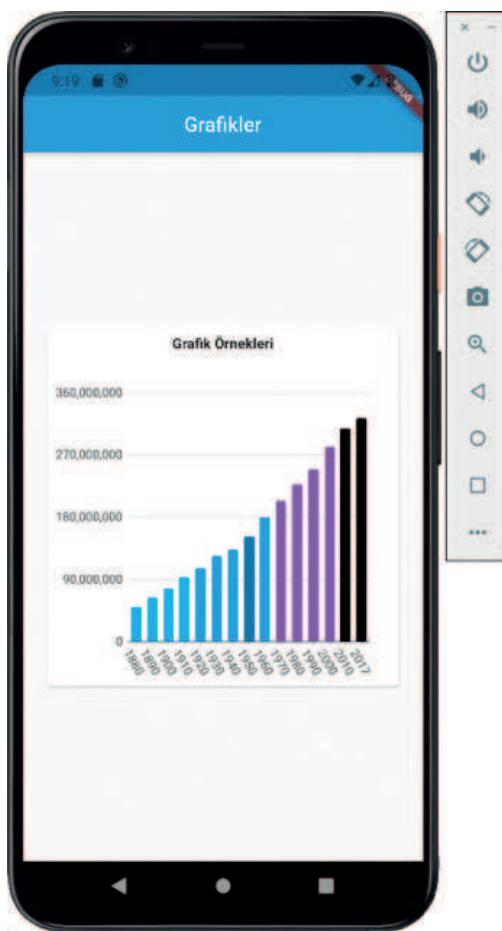
_getSeriesData() {
List<charts.Series<PopulationData, String>> series = [
charts.Series(
  id: "Population",
  data: data,
  domainFn: (PopulationData series, _) => series.year.toString(),
  measureFn: (PopulationData series, _) => series.population,
  colorFn: (PopulationData series, _) => series.barColor)
];
return series;
}
final List<PopulationData> data = [
PopulationData(
  year: 1880,
  population: 50189209,
  barColor: charts.ColorUtil.fromDartColor(Colors.lightBlue)),
PopulationData(
  year: 1890,
  population: 62979766,
  barColor: charts.ColorUtil.fromDartColor(Colors.lightBlue)),
PopulationData(
  year: 1900,
  population: 76212168,
  barColor: charts.ColorUtil.fromDartColor(Colors.lightBlue)),
PopulationData(
  year: 1910,
  population: 92228496,
  barColor: charts.ColorUtil.fromDartColor(Colors.lightBlue)),
];

```

```
class PopulationData {  
    int year;  
    int population;  
    charts.Color barColor;  
    PopulationData(  
        {@required this.year,  
         @required this.population,  
         @required this.barColor});  
}
```

```
Center(  
child: Container(  
height: 400,  
padding: EdgeInsets.all(20),  
child: Card(  
child: Padding(  
padding: const EdgeInsets.all(8.0),  
child: Column(  
children: <Widget>[  
    Text(  
        "Grafik Örnekleri",  
        style: TextStyle(fontWeight: FontWeight.bold),  
    ),  
    SizedBox(  
        height: 20,  
    ),  
    Expanded(  
        child: charts.BarChart(  
            _getSeriesData(),  
            animate: true,  
            domainAxis: charts.OrdinalAxisSpec(  
                renderSpec: charts.SmallTickRendererSpec(  
                    labelRotation: 60)),  
            ),  
            ),  
        ],  
    ),  
    ),  
    ),  
)
```

Yukarıdaki örnek kodda ilk olarak PopulationData isminde bir class oluşturulur ardından `_getSeriesData()` adında bir fonksiyon hazırlanır. Bu fonksiyon grafiklere bir fonksiyon olarak verilecek ve grafiğe girmek istenilen veriler buradan çektilirilir. Ardından data isminde PopulationData'lardan oluşan bir liste tanımlanır. Listenin içerisinde PopulationData sınıfından üretilen değerler girilir. Son olarak charts.BarChart(`_getSeriesData()`) komutuyla veriler grafiğe dökülmüş olur. Oluşan ekran çıktısı Şekil 4.10 da görülmektedir.



Şekil 4.10 Örnek olarak verilen Grafikler kodunun Ekran Çıktısı.

Stiller ve Temalar

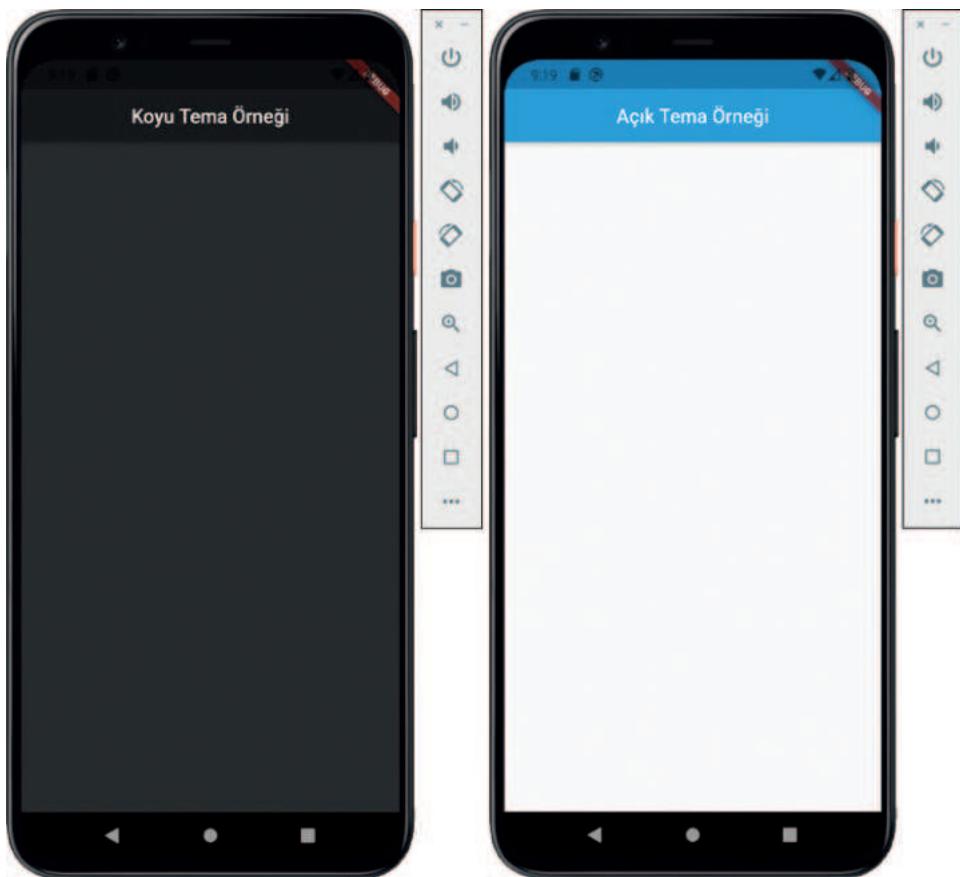
Flutter'da Temalar kolay bir şekilde ortak merkezden yönetilebilir. Basitçe anlatmak gerekirse MaterialApp Widget'ının özelliği olan theme parametresine "ThemeData.dark()" veya "ThemeData.light()" verilerek tüm sistem teması değiştirilebilir. Parametreye ThemeData.dark() girildiğinde sistem teması koyu, ThemeData.light() girilirse eğer sistem teması açık olur.

```
MaterialApp(  
    theme: ThemeData.light(),  
    home: Scaffold(  
        appBar: AppBar(  
            title: Text('Açık Tema Örneği'),  
            centerTitle: true,  
        ),  
        ),  
    );
```

Yukarıdaki kod parçasında açık tema örneği verilmiştir.

```
MaterialApp(  
    theme: ThemeData.dark(),  
    home: Scaffold(  
        appBar: AppBar(  
            title: Text('Koyu Tema Örneği'),  
            centerTitle: true,  
        ),  
        ),  
    );
```

Yukarıdaki kod parçasında ise koyu tema örneği verilmiştir.



Şekil 4.11 Örnek olarak verilen Tema kodunun Ekran Çıktısı.



Araştırmalarla İlişkilendir

İyi bir kullanıcı deneyimi için, kullanıcı ile sistem arasında tutarlı bir arayüzü tasarlanmalı, tasarım sürecinde olabildiğince çok sayıda kullanıcı sorunu düşünülmelidir. Kullanıcıların karşılaşabileceği olası karmaşık senaryolar geliştirme ve test süreçlerinde öngörülü, gerekli düzenlemeler bu olası senaryolara göre yapılabilir. Tasarımcı genel anlamda, kullanıcıların alışkanlıklarını, geleneksel olandan kolay kopardıklarını ve yeniliklere karşı direnç göstereceklerini göz ardı etmemelidir. Etkili deneyim yaşatacak bir kullanıcı arayüz tasarımı için tasarımcının sürekli tasarım denemeleri yapması ve ulaşılması amaçlanan hedef kitlenin kullanıcı bilgilerini alması gerekmektedir.

Kullanıcı deneyiminde etki eden bir diğer önemli faktör ise internet bağlantısının hızıdır. İyi tasarlanmış bir kullanıcı deneyimi ve grafiksel kullanıcı arayüzü, kullanıcının etkileşim sağlayacağı bir medyum olarak internet hızıyla da ilgilidir. Dünya nüfusunun ortalama sadece %63'ünün geniş bant internet erişimine sahip olduğu göz önüne alındığında, kullanıcı deneyimi ve bunun bunu sağlayan arayuzlerin internet bağlantı hızıyla bağlantılı olduğu söylenebilir.

Kaynak: Kaya, K. S. (2019). *Mobil uygulamalarda farklı ihtiyaç gruplarına yönelik arayüz çözümlemleri üzerine bir inceleme*. Yüksek Lisans Tezi, Güzel Sanatlar Enstitüsü, Anadolu Üniversitesi, Eskişehir.

Öğrenme Çıktısı



İKON KULLANIMI

İkonlar Flutter'da kullanılması gayet kolay ve kullanışlı simgelerdir. Flutter'ın bize sağladığı ikon paketinde hazır olarak verilen birçok ikon bulunmaktadır. Kullanımına gelirse eğer ikonları Icon Widget'ı içerisinde Icons. notasyonu kullanılarak işe uygun ikon seçili kullanılabılır.

```
Center(  
    child: Column(  
        children: [  
            Spacer(),  
            Icon(Icons.access_alarms_sharp),  
            Icon(Icons.adb),  
            Icon(Icons.dashboard),  
            Icon(Icons.zoom_out_map_outlined),  
            Icon(Icons.wysiwyg),  
            Icon(Icons.widgets_rounded),  
            Icon(Icons.hail),  
            Icon(Icons.face_outlined),  
            Spacer(),  
        ],  
    ))
```

Örnek olarak verilen kodda birçok ikon kullanılmıştır. Eğer kullanılan editörde otomatik tamamlama var ise Icons. notasyonundan sonra otomatik olarak hazır ikonlar ve isimleri gelecektir. Yukarıdaki kod çalıştırıldığında Şekil 4.11 de görüldüğü gibi bir ekran çıktısı alınır.



Şekil 4.12 Örnek olarak verilen İkon kodunun Ekran Çıktısı.



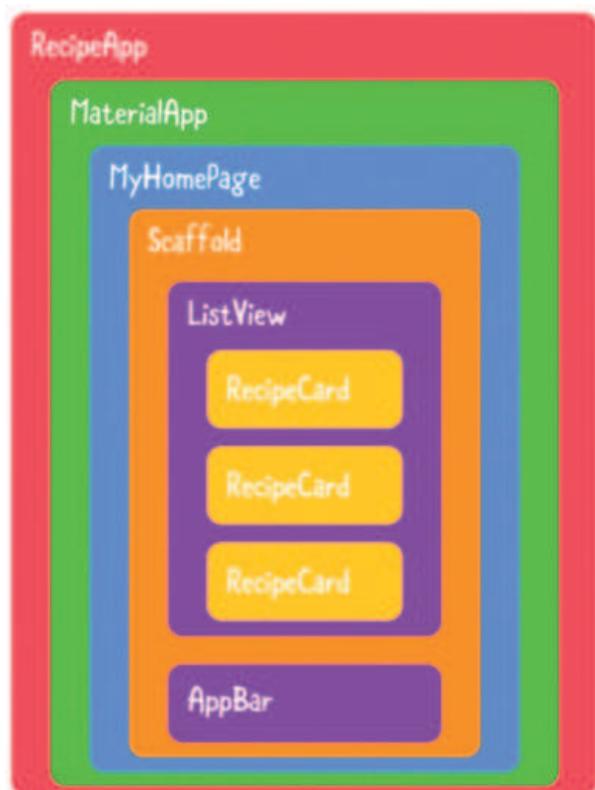
LAYOUT

Flutter'da en önemli konulardan birisi Layout kavramının mantığının anlaşılmasıdır. Eğer bu konu iyi anlaşılmazsa Responsive tasarım ve tasarımın kurgusu konularında eksiklikler çıkacaktır. Şekil 4.12 de bir layout'un görsel olarak şeması verilmiştir.

Center

Center Widget'i isminden de anlaşıldığı üzere yatayda ve dikeyde child Widget'ını ortalar. Bu widget, boyutları kısıtlıysa ve widthFactor ve heightFactor boşsa, mümkün olduğunda büyük olacaktır. Bir boyut sınırlandırılmamışsa ve karşılık gelen boyut faktörü boşsa, pencere ögesi o boyuttaki çocuğunun boyutuya eşleşir. Bir boyut faktörü boş değilse, bu parçacığın karşılık gelen boyutu, çocuğun boyutunun ve boyut faktörünün ürünü olacaktır. Örneğin, widthFactor 2.0 ise, bu parçacığın genişliği her zaman alt ögesinin genişliğinin iki katı olacaktır. Özellikleri ise aşağıdaki gibidir:

- Alignment
- Child
- hashCode
- heightFactor
- key
- runtimeType
- widthFactor



Şekil 4.13 Layout tasarıma örnek.

Kaynak: raywenderlich Tutorial Team, Mike Katz, Kevin David Moore, Vincent Ngo - Flutter Apprentice (First Edition) _ Learn to Build Cross-Platform Apps-Razeware LLC (2021) page:62

```
Center(child: Text("Center" Örneği)),
```

Row

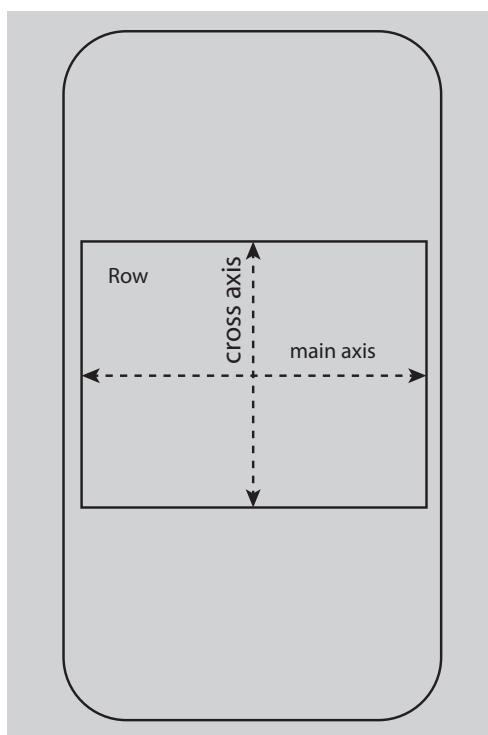
Row Widget'ı children olarak aldığı Widget'ları yatayda sıralar. En önemli özelliklerinden ikisi main axis ve cross axis alignment'tır. Şekil 4.13 görüldüğü gibi içerisinde bulunan children Widget'ları hizalama yarar. Bir alt ögenin mevcut yatay alanı dolduracak şekilde genişlemesi için SingleChildScrollView kullanılır.

Satır widget'ı kaydırma yapmaz. Ayrıca bir satırda mevcut alana sığacak olandan daha fazla children Widget olması bir hata olarak kabul edilir. Çok sayıda widget kullanılacak ve yetersiz yer varsa üstelik bunların kaydırılabilmesi isteniyorsa ListView kullanılmalıdır. Özellikleri ise aşağıdaki gibidir:

- Children
- clipBehavior
- crossAxisAlignment
- direction
- hashCode
- key
- mainAxisAlignment
- mainAxisSize
- runtimeType
- textBaseline
- textDirection
- verticalDirection



```
Row(children: [Text("Row Örneği"), Text("Row Örneği")]),
```



Şekil 4.14 Örnek olarak verilen İkon kodunun Ekran Çıktısı.

Kaynak: Barry Burd - Flutter® For Dummies® (2020) page:370

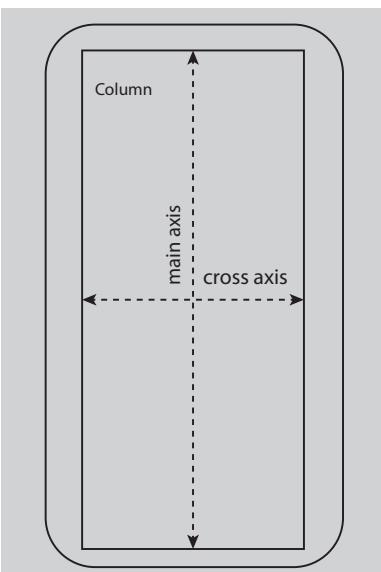
Column

Column Widget'ı children olarak aldığı Widget'ları dikey olarak sıralamak için kullanılır. En önemli özelliklerinden ikisi main axis ve cross axis alignment'tur.

```
Column(children: [Text("Column Örneği"), Text("Column Örneği")])
```

Bir alt ögenin kullanılabilir dikey alanı dolduracak şekilde genişlemesine neden olmak için SingleChildScrollView kullanılır. Özellikle ise aşağıdaki gibidir:

- Children
- clipBehavior
- crossAxisAlignment
- direction
- hashCode
- key
- mainAxisAlignment
- mainAxisSize
- runtimeType
- textBaseline
- textDirection
- verticalDirection



Şekil 4.15 Örnek olarak verilen İkon kodunun Ekran Çıktısı.

Kaynak: Barry Burd - Flutter® For Dummies® (2020) page:370

Spacer

Spacer Widget'ı Column veya Row Widget'ının içine child olarak eklendiğinde diğer child Widget'ları ötelebilir. Spacer, Satır veya Sütun gibi bir Flex kapsayıcısındaki widget'lar arasındaki aralığı ayarlamak için kullanılabilen, ayarlanabilir, boş bir aralayıcı oluşturur. Spacer widget'ı herhangi bir kullanılabilir alanı kaplayacaktır.

Bu nedenle Flex.mainAxisAlignment öğesini MainAxisAlignment.spaceAround, MainAxisAlignment.spaceBetween veya MainAxisAlignment.spaceEvenly için bir Spacer içeren esnek bir alanda kullanmanın görünür bir etkisi olmaz. Çünkü Spacer widget tüm alanı kaplamıştır. Özellikleri ise aşağıdaki gibidir:

- flex
- hashCode
- key
- runtimeType



```
Column(children: [Spacer(), Text("Spacer Örneği")])
```

Expanded

Expanded Widget'ı Column veya Row Widget'ının içine child olarak Eklendiğinde Expanded Widget'ının aldığı yeni childlarını dışta bulunan satır veya sütuna, flex'te aldığı değere göre oranlar. Genişletilmiş bir pencere aracının kullanılması, bir satır, sütun veya esnek ögesinin alt ögesinin, ana eksen boyunca kullanılabilir alanı dolduracak şekilde genişlemesini sağlar. Birden fazla child widget genişletilirse, kullanılabilir alan esnek faktöre göre aralarında bölünür.

Genişletilmiş bir pencere ögesi bir satır, sütun veya flex'in alt ögesi olmalıdır. Genişletilmiş pencere aracından onu çevreleyen satır, sütun veya flex'e giden yol yalnızca StatelessWidget veya StatefulWidget içermelidir. Özellikleri aşağıdaki gibidir:

- child
- debugTypicalAncestorWidgetClass
- fit
- flex
- hashCode
- key
- runtimeType



```
Column(children: [
    Expanded(flex: 2, child: Text("Expanded" Örneği")),
    Expanded(flex: 5, child: Text("Expanded" Örneği))
])
```

Container

Container Widget en temel Widget'tır içine özellik verilmediğinde boş bir kutu gibi düşünülebilir ancak içerisinde parametreler girildiğinde ona genişlik, yükseklik, renk vs. kazandırılabilir. Aşağıdaki kodda görüldüğü üzere width genişlik parametresidir. Height parametresi ise yükseklik parametresidir ve ekranın neresinde bulunuyorsa orada girilen yükseklik değerini kazanır.

```
Column(children: [
    Container(
        width: 200,
        height: 150,
        color: Colors.amber,
        child: Center(child: Text("Container" Örneği 1))),
    Container(
        width: 300,
        height: 360,
        color: Colors.blue,
        child: Center(child: Text("Container" Örneği 2)))
])
```

Color parametresi ise içerisinde Colors tipinde bir değer alır. Eğer editörünüz otomatik tamamlamayı destekliyorsa Colors. Notasyonu kullanıldığında birçok renk seçeneklerini görerek seçim yapılabilir. Container Widget'ın özellikleri aşağıdaki gibidir:

- alignment
- child
- clipBehavior
- color
- constraints
- decoration
- foregroundDecoration
- hashCode
- key
- margin
- padding
- runtimeType
- transform
- transfromAlignmen



Öğrenme Çıktısı

6 Layout yaklaşımını uygulayabilme

- Araştır 6**
- İlişkilendir**
- Anlat/Paylaş**

Daha farklı Layout için kullanılan Widget'ları araştırın.

Layout tasarımlı yaparken responsive tasarımla responsive olmayan tasarımları kıyaslayın.

Sizde nasıl bir Layout bakış açısı oluştu? Bir taslağını çizin.

1

Diyalog ekranlarını kullanabilme ve farklı amaçlar için kullanılan diyalog ekranlarını tanıyalabilme

Diyalog Ekranları

Diyalog ekranları Flutter'da kritik mesaj bildirmek veya belirli durumlara karar vermelerini sağlamak için kullanılır. Diyalog ekranları kullanıcıya gösterildiğinde arkadaki sayfa açık kalmaya devam eder, verilen aksiyona göre işlem gerçekleştirir. 3 farklı diyalog ekran türü vardır: AlertDialog, SimpleDialog, ShowDialog. Diyalog ekranlarının tasarımını Material Design ve Cupertino Design olarak 2 şekilde kullanılabılır. Diyalog ekranları ShowDialog fonksiyonuyla kullanılmalıdır ekran'a farklı bir sayfa basılacaktır ve bunun context ağacına bağlanmasıyla gerçekleşir. Material Design Android' deki, Cupertino ise IOS' daki gibi bir tasarım çıktısı verir.

2

Farklı şekillere sahip olan butonları tanıyalabilme ve özelliklerini öğrenebilme

Buton Kontrolleri

Flutter SDK tarafından birçok farklı şekillerde butonlar sağlanır. TextButton, ElevatedButton, OutlineButton, IconButton, FloatingActionButton'dur. Şekil haricinde büyük farklılıklar yoktur. Widget'in içerisinde child olarak istenilen Widget konulabilir. Buton eklediğimizde bir ClickEvent tanımlamak gerekecektir aksi halde Flutter hata verir.

3

Listelerle işlem yapabilmeyi ve ekranda listelerin nasıl kontrol edileceğini öğrenebilme

Liste Kontrolleri

Flutter'da çoklu verileri görüntülemeye yarar. Ekrana sığmayacak kadar çok veri olduğunda bu liste içinde kaydırılabilir bir yapı halinde sunabilir. 2 adet kullanışlı Widget vardır ListView daha çok sabit veri olduğunda kullanılabilir dinamik bir yapısı yoktur. ListViewBuilder dinamik yapıya sahip kaydırılabilir bir bakış açısı sunar. Bu Widget'ı kullanırken itemCount özelliği yanlış girilirse index hatası alınır.

4

Grafikleri, stilleri ve temaları kullanabilme

Grafikler, Stiller ve Temalar

Flutter'da verileri görselleştirmek için kullanılır. Bu kitapta grafikler için charts_flutter adlı paketi kullanılacaktır. Uygulamanın içine eklemek için bu paketi projenin ana dizininde “flutter pub add charts_flutter” komutu girilir. Ardından import “package:charts_flutter/flutter.dart as charts” komutu grafik kullanılmak istenilen flutter dosyasının en üstüne eklenir. Flutterda temalar ise kolay bir şekilde ortak merkezden yönetilebilir bir yapıya sahiptir.

5

İkonların nasıl kullanılacağını öğrenebilme

İkon Kullanımı

Flutter'da kullanılması gayet kolay ve kullanışlı simgeler vardır. Kullanımı ikonları Icon Widget'ı içe-risine Icons. notasyonu kullanılarak işe uygun ikon seçilip kullanılabilir. Kullanılan editörde otomatik tamamlama var ise Icons. notasyonundan sonra otomatik olarak hazır ikonlar ve isimleri gelecektir.

6

Layout yaklaşımını uygulayabilme

Layout

Center Widget'ı yatayda ve dikeyde child Widget'ını ortalar. Boyutları kısıtlıysa ve widthFactor ve heightFactor boşsa mümkün olduğunda büyük olacaktır. Bir boyut sınırlırmamış ve karşılık gelen boyut faktörü boşsa pencere ögesi o boyuttaki child'ının boyutuyla eşleşir. Bir boyut faktörü boş değilse bu parçacığın karşılık gelen boyutu, çocuğun boyutunun ve boyut faktörünün ürünü olacaktır. Row Widget'ı children olarak aldığı Widget'ları yatayda sıralar. En önemli özelliklerinden ikisi main axis ve cross axis alignmenttir. Satır Widget'ı kaydırma yapmaz ve bir satırda olması gerekenden fazla children Widget'ı olması olması durumunda taşmalar olacaktır. Çok sayıda Widget kullanılacak ve yetersiz yer varsa üstelik bunların kaydırılabilmesi isteniyorsa ListView kullanılmalıdır. Column Widget'ı children olarak aldığı Widget'ları dikey olarak sıralamak için kullanılır. En önemli özelliklerinden ikisi main axis ve cross axis alignmenttir. Bir alt ögenin kullanılabilir dikey alanı dolduracak şekilde genişlemesine neden olmak için SingleChildScrollView kullanılır. Spacer Widget'ı Column veya Row Widget'ının içine child olarak eklendiğinde diğer child Widget'ları öteles. Spacer Satır ve Sütun gibi bir Flex kapsayıcısındaki Widget'lar arasındaki aralığı ayarlamak için kullanılabilecek ayarlanabilir boş bir Spacer(aralayıcı) oluşturur. Spacer Widget'ı herhangi bir kullanılabilir boş bir Spacer(aralayıcı) oluşturur ve böylece Spacer Widget'ı herhangi bir kullanılabilir alanı kaplayacaktır.

Mobil Uygulama Geliştirme

- 1**
- I. AlertDialog
 - II. SimpleDialog
 - III. ClipBehavior
 - IV. OutlinedButton
 - V. Dialog

Yukarıdakilerden kaç tanesi ekrana diyalog bastırır?

- A. 1
- B. 2
- C. 3
- D. 4
- E. 5

- 2**
- I. Child
 - II. Children
 - III. OnHover
 - IV. OnTop
 - V. OpPan

Yukarıdakilerden kaç tanesi Elevated Button'unun özelliklerinden **değildir**?

- A. 1
- B. 2
- C. 3
- D. 4
- E. 5

3 Aşağıdakilerden hangisi Flutter'da çoklu verileri görüntülemeye yarar?

- A. Button
- B. ListView
- C. Physics
- D. ListTile
- E. Text

4 ListView.builder'ın ListView'den daha fazla kullanışlı olma sebebi aşağıdakilerden hangisidir?

- A. İçerisine index verilerek index'e göre içerisinde işlem yaptırılabilir.
- B. İçerisinde resim görüntüleyebildiği için
- C. Bir liste elemanı olduğu için
- D. Theme datasını içerisinde taşıdığı için
- E. İçerisinde ikon bulundurıldığı içim

- 5**
- I. Children
 - II. Reverse
 - III. ItemCount
 - IV. ItemBuilder
 - V. ShrinkWrap

Yukarıdakilerden kaç tanesi ListView.builder özeliklerindendir?

- A. 1
- B. 2
- C. 3
- D. 4
- E. 5

6 Aşağıdakilerden hangisi Flutter'da grafikleri çizdirmek için kullanılan paketlerden biridir?

- A. Charts_flutter
- B. Http
- C. Dio
- D. Google_fonts
- E. Provider

7 Tema ayarlarını karanlık moda geçirmek için aşağıdaki hangi fonksiyonu kullanılmalıdır?

- A. CenterTitle()
- B. Title()
- C. Home()
- D. Text()
- E. ThemeData.dark()

8 Aşağıdaki notasyonlardan hangisi ikonlara ulaşmayı sağlar?

- A. List.
- B. Icons.
- C. Int.
- D. Key.
- E. RuntimeType.

9 Aşağıdaki Widget'lardan hangisi içerisinde "child" olarak aldığı Widget'i ortalar?

- A. Row()
- B. Column()
- C. Center()
- D. Spacer()
- E. Expanded()

- 10**
- I. Alignment
 - II. Margin
 - III. RunTimeType
 - IV. Child
 - V. Padding

Yukarıdakilerden kaç tanesi Container özellikle rindendir?

- A. 1
- B. 2
- C. 3
- D. 4

1. B	Yanıtınız yanlış ise “Diyalog ekranları” konusunu yeniden gözden geçiriniz.	6. A	Yanıtınız yanlış ise “Grafikler” konusunu yeniden gözden geçiriniz.
2. C	Yanıtınız yanlış ise “Buton” konusunu yeniden gözden geçiriniz.	7. E	Yanıtınız yanlış ise “Stiller ve Temalar” konusunu yeniden gözden geçiriniz.
3. B	Yanıtınız yanlış ise “Liste kontrolleri” konusunu yeniden gözden geçiriniz.	8. B	Yanıtınız yanlış ise “İkon” konusunu yeniden gözden geçiriniz.
4. A	Yanıtınız yanlış ise “Liste kontrolleri” konusunu yeniden gözden geçiriniz.	9. C	Yanıtınız yanlış ise “Layoutlar Center” konusunu yeniden gözden geçiriniz.
5. D	Yanıtınız yanlış ise “Liste kontrolleri” konusunu yeniden gözden geçiriniz.	10. E	Yanıtınız yanlış ise “Layoutlar Container” konusunu yeniden gözden geçiriniz.

4

Araştır Yanıtları Anahtarı



Araştır 1

Android platformunda Alert diyalog ekranları bulunmaktadır. Bunu ekrana bastırmak için DialogFragment fonksiyonunun bir örneğini çıkarıp, show() fonksiyonuyla ekrana basılabilir.

IOS Swift 4.x platformunda ise UIAlertController nesnesi oluşturup bunların içerisinde gerekli parametreleri girildiğinde oluşturulan nesneye .addAction eklendiğinden “self.present(olusturulannesne,animated:true,completion:nil)” kodu yazarak ekrana diyalog bastırılabilir.



Araştır 2

Butonlarda style özelliği butonu şekillendirmek için kullanılır. Butona uygun style Widget’ı bulunur ve içerisinde parametreleri verilerek buton şekillendirilir. Örnek olarak ElevatedButton kodu “ElavatedButton.styleFrom(primary:Colors.teal)” girildiğinde butonun ana renk tonu belirlenmiş olur.



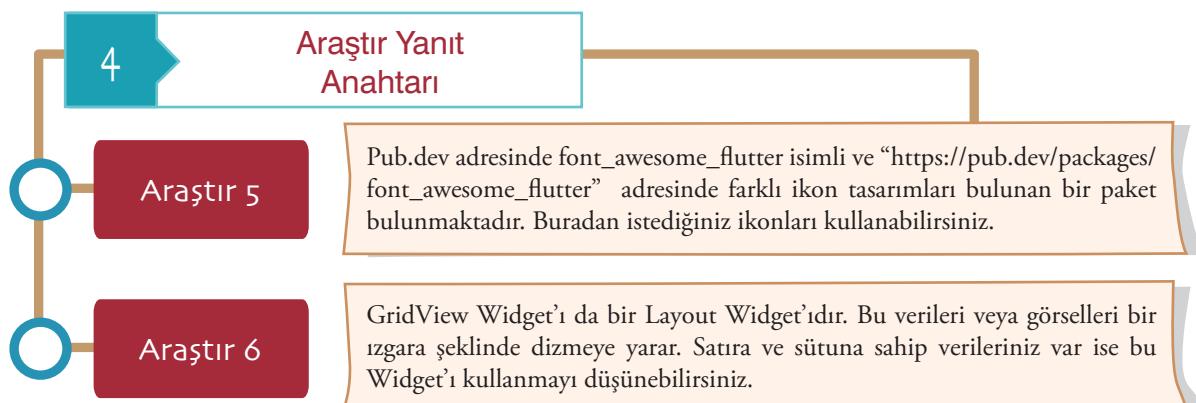
Araştır 3

SliverList bir üst sınıfı ve temel ancak kullanması zor bir listeleme Widget’ıdır. SliverList delegate fonksiyonu içerisinde SliverChildBuilderDelegate Widget’ı alır ve buda bir Widget return eder, ListView.builder’da kullanıldığı gibi.



Araştır 4

Pub.dev adresinde graphic isimli ve “<https://pub.dev/packages/graphic>” adresinde bir paket bulunmaktadır. Buradan verilerinizi görselleştirebilirsiniz.



Kaynakça

Raywenderlich Tutorial Team, Mike Katz, Kevin David Moore, Vincent Ngo - Flutter Apprentice (First Edition) _ Learn to Build Cross-Platform Apps-Razeware LLC (2021)

Barry Burd - Flutter® For Dummies® (2020)

Simone Alessandria, Brian Kayfitz - Flutter Cookbook_ Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart-Packt Publishing (2021)

Thomas Bailey, Alessandro Biessek - Flutter for Beginners_ An introductory guide to building cross-platform mobile applications with Flutter 2.5 and Dart (2021, Packt Publishing)

Bölüm 5

Dart ile Kodlama

Öğrenme Çıktıları

1 Dart Kavramı

- 1 Dart programlama dilinin temel özelliklerini açıklayabilme

2 Dart Uygulamaları

2 Dart Uygulamaları

- 2 Dart uygulamalarının temel çalışma mekanizmalarını açıklayabilme

3 Dart Dili

- 3 Dart programlama dilinde veri yapılarını kullanabilme

4 Akış Yöneticiler

4 Akış Yöneticiler

- 4 Dart programlama dilinde akış yöneticilerini ve döngü yapılarını kullanabilme

5 Dart ile Nesneye Yönelik Programlama

- 5 Dart programlama dilinde nesneye yönelik programlama kullanabilme

6 Çok Sayfalı Uygulamalar

6 Çok Sayfalı Uygulamalar

- 6 Çok sayfalı bir Flutter uygulaması oluşturabilme

Anahtar Sözcükler: • Dart • Veri Tipleri • Akış Yönetimi • Döngüler • Nesneye Yönelik Programlama
• Flutter Uygulamaları • Rotalar • Navigator



GİRİŞ

Bu bölümde Flutter uygulamalarının üretilmesinde kullanılan Dart programlama dilinin temel özelliklerinin tanıtılması ve çok sayfalı bir uygulamanın geliştirilmesi amaçlanmıştır. Dart programlama dili, çeşitli istemciler üzerinde çalıştırılacak uygulamaları oluşturmak için kullanılan oldukça esnek bir dildir. Dart, null güvenliği, nesneye yönelik programlama ve anında derleme gibi özellikleriyle programcılar işlerini kolaylaştıran bir programlama dilidir. Ünlüde Dart programlama dilinin temel özellikleri ve işlevleri anlatılacak; dilin temel mekanizmaları ve veri yapıları tanıtılarak ve nesneye yönelik programlama teknikleri aktarılacaktır. Ünite sonunda çok ekranlı bir Flutter uygulaması üretilecek, kullanıcı etkileşimiyle bu ekranlar arasında geçişlerin nasıl yönetilebileceği tanıtılacaktır.

DART KAVRAMI

Dart çeşitli platformda yürütülebilecek istemci uygulamalar üretебilen bir programlama dilidir (Google, t.y.). Dart dili ile, mobil uygulamalar, web uygulamaları, komut satırı betikleri (script) ya da sunucu tarafında çalışabilecek uygulamalar üretilebilir. Dart derleyicisi hazırlanan kodları hedeflenen platformlar için derleyerek sonuç üretir. Bu nedenle Dart dilinde üretilen uygulamalar her platforma özgüdür (native). Bunun yanında, anında derleme (Just In Time) ve yürütme öncesi derleme (Ahead of Time) gibi mekanizmaları kullanabildiğinden Dart uygulamaları rakip platformlara göre oldukça hızlı çalışabilmektedir.

 **Dart:** Çeşitli platformda yürütülebilecek istemci uygulamalar üretебilen bir programlama dilidir.



dikkat

Flutter uygulamaları Dart programlama dili kullanılarak hazırlanır.

Dart dilinin bazı kritik özellikleri şu şekilde sıralanabilir:

- Dart *nesne yönelimli bir dildir*. Dart dilinde kullanılan her yapının (değişkenler, fonksiyonlar) özünde bir sınıf yer alır.
- Dart tanımlama temelli (declarative) bir dildir. Dart programcılar arayüzlerin farklılaşan veri yapılarına göre nasıl davranışacağını belirler. Veri yapısının değişmesi halinde gerekli değişiklikler arayüze Dart tarafından yansıtılır. Yinelemeli (imperative) dillerde ise programcılar veri yapısındaki değişiklikleri dinleyerek gerekli durumlarda arayüze müdahale etmesi gereklidir. Örneğin, C# gibi dillerde TextBox gibi nesnelerde gösterilecek metinler, olay temelli bir sistem içinde (ör: Bir düğmeye basılması halinde gerçekleştirilen hesaplama) programcı tarafından değiştirilir. Dart programlarının bu tür düzenlemeleri yapmasına gerek yoktur.
- Dart değişkenlerinin veri tipleri *sıkı kontrol* (strongly typed) altındadır. Bu nedenle kod içinde değişkenlere aktarılabilen veri türleri (ör: *int, String*) değiştirilemez. Javascript ve PHP gibi dillerde sıkı veri tipi kontrolü olmadığından bir değişkene kod içinde *int* ya da *String* tipinde veriler aktarılması sorun yaratmaz. Fakat Dart kodlarında bir değişkene tanımlanan veri tipi dışında değer atanması halinde derleme sırasında bu durum yakalanır ve hata ayıklanması istenir.
- Dart, değişkenlere ilk değer ataması yapıldığı anda *tip çıkarımı* yapabilir. Bu mekanizma sayesinde programının değişkenin hangi tipte olması gerektiğine ilişkin bilgi vermesine gerek kalmaz. Değişkenler *var* anahtar sözcüğü ile tanımlanır. Dart gerçekleştirilen atamaya göre değişkenin veri tipine karar verebilir. Dilin geliştiricileri tip çıkarımı mekanizmasının kullanılmasını önermektedir.
- Dart null güvenliği (null-safety) mekanizmasını destekler. Bu yolla henüz değer atanmamış değişkenlere başvurulmasından kaynaklanan *null exception* hataları engellenir. Özellikle belirtildiği sürece değişkenler *null* değerini alamaz.
- Kodlama esnasında yazılan kodlar sıcak yükleme (hot reload) mekanizması ile yürütülen uygulamalara enjekte edilebilir. Bu sayede çalıştırılan uygulamaların baştan derlenmesi ile zaman kaybı yaşanmaz.



Öğrenme Çıktısı

1 Dart programlama dilinin temel özelliklerini açıklayabilme

Araştır 1

Null-safety kavramını araştırınız. Hangi programlama dillerinin bu mekanizmayı desteklediğini açıklayınız.

İlişkilendir

Dart dilinin yukarıda belirtilen özelliklerinin hangilerinin Javascript ve C# dillerinde desteklendiğini inceleyiniz.

Anlat/Paylaş

Sizce yukarıda belirtilen özelliklerden hangisi Dart dilinde kodlamayı kolaylaştırmaktadır?

DART UYGULAMALARI

Kod 1'de yarıçap bilgisi verilen bir çemberin çevresini hesaplayarak konsola yazan bir kod parçası bulunmaktadır. Bu kod parçasında Dart dilinin pek çok temel özelliği kullanılmaktadır.

```
import "dart:math";

void main(){
    var yaricap = 10;
    var cevre = cevresiniBul(yaricap);
    print(cuvre);
}

num cevresiniBul(int sayı)
{
    return(sayı * pi * 2);
}
```

Kod 1 Yarıçapı verilen çemberin çevresini hesaplamak

Dart farklı görevleri ve yetenekleri olan kütüphaneler kullanır. Temel veri tipleri ve bu verilerle ilgili işlemler için *dart:core* kütüphanesi kullanılır. Bu kütüphane uygulamalara otomatik olarak eklenildiğinden özellikle eklenmesine gerek yoktur. Fakat, asenkron veri çekme, http istekleri, veri tabanı erişimi ya da üçüncü parti görselleştirme araçları gibi işlevler için farklı kütüphanelerin projelere *import* anahtar sözcüğü ile eklenmesi gereklidir. Core kütüphanesi tarafından sağlanmayan matematiksel işlemler ve sabitler (ör: pi sayısı) için ilk satırda *dart:math* kütüphanesi projeye eklenmiştir.



internet
Dart çekirdek kütüphanelerini incelemek için <https://dart.dev/guides/libraries> adresini ziyaret edebilirsiniz.

dart:math kütüphanesi pi ve e gibi sabit sayıların yanında trigonometrik fonksiyonlar, üstlü ya da köklü işlemlere yönelik araçlar da sağlar.



internet
dart:math kütüphanesi hakkında daha fazla bilgi almak için <https://api.dart.dev/stable/2.14.4/dart-math/dart-math-library.html> adresini ziyaret ediniz.

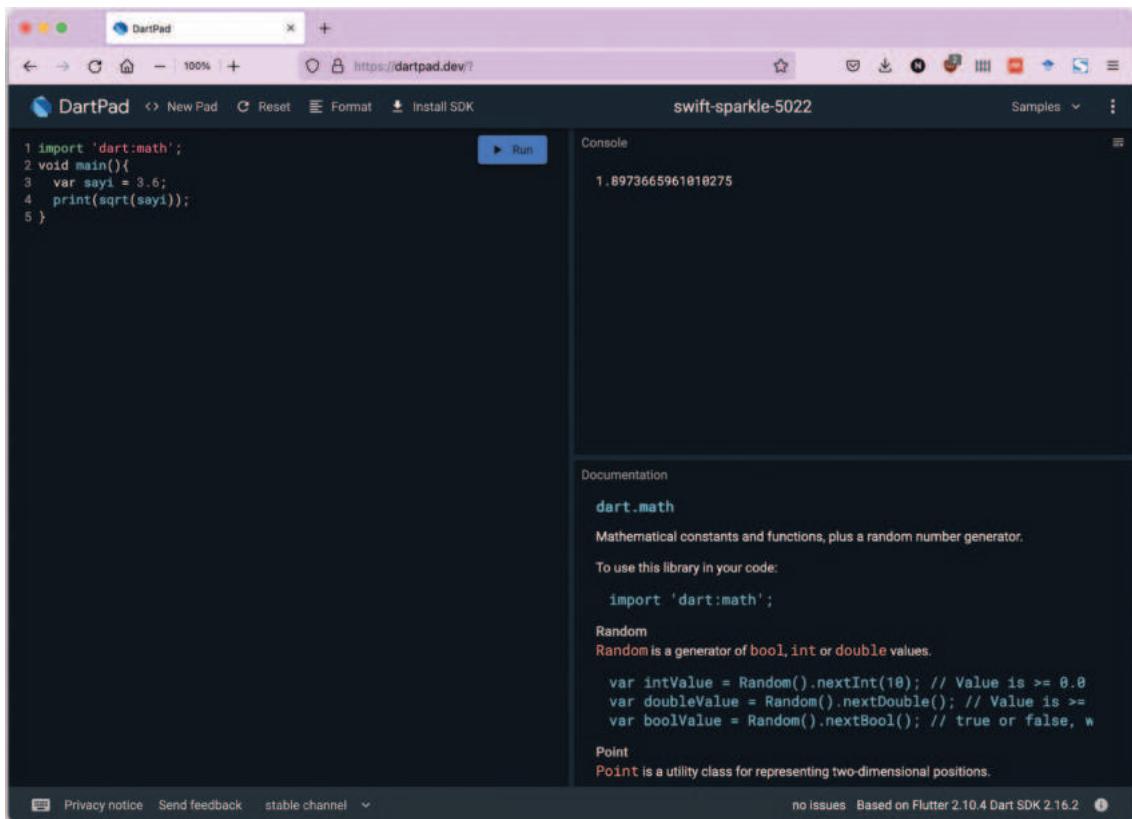


dikkat

Kişisel bilgisayarlarda Dart kodlarının derlenmesi ve sanal makinelerde çalıştırılması zaman alabilemektedir. Bu nedenle, Çok Sayfalı Uygulamalar başlığına kadar yazılacak kodların sonuçlarını daha hızlı inceleyebilmek için <https://dartpad.dev> internet sitesinin kullanılması önerilmektedir.



Bu sitede sol taraftaki panele yazılan Dart kodları Run düğmesine basılarak çalıştırılabilir. Bu kodların çıktıları sağ taraftaki konsol alanında görüntülenecektir. Sağ alt bölümde ise yazılan kodlarla ilgili bilgilendirme paneli yer almaktadır.



Görsel 5.1 Dartpad Sitesinin Ekran Görüntüsü

Dart uygulamalarının başlangıç noktası *main* fonksiyonudur. Dart sıkı tip denetimi yapan bir dil olduğundan fonksiyonlar çevirecekleri değerlerin veri tipleri ile tanımlanırlar. *main* fonksiyonu bir değer çevirmeceği için *void* anahtar sözcüğü ile tanımlanmıştır. *cevresiniBul* fonksiyonu ise sayısal değer çevirmesi beklenildiğinden *num* anahtar sözcüğü ile tanımlanmıştır.

Değişkenler tanımlanırken değer atanacaksız *String*, *num* gibi tip belirteçlerinin kullanılması yerine *var* anahtar sözcüğünün kullanılması önerilmektedir. Bu yolla değişkenini veri tipine, tip çıkarımı mekanizmasının karar vermesine izin verilmiş olur. *Yarıçap* ve çevre değişkenleri bu yolla tanımlanmıştır. *Yarıçap* değişkenine *10* değeri, *çevre* değişkenine ise *cevresiniBul* fonksiyonunun çevirdiği değer atanmıştır.

Print fonksiyonu kod içinde üretilen değerlerin konsol arayüzüne yazılması için kullanılan üst düzey fonksiyonlardan biridir. Özellikle hata ayıklama sürecinde (debug) kullanılması önerilir. Fakat son kullanıcıya yönelik uygulamalar üretilirken bu fonksiyonu içeren kodların ayıklanması önerilmektedir.

Fonksiyonlara parametreler yoluyla değer gönderilebilir. Fonksiyonlar da tanımlı veri tiplerindeki değerleri geri çevirebilir. *cevresiniBul* fonksiyonu *sayı* adında *int* tipinde



dikkat

Fonksiyonlar tanımlanırken geriye çevrilecek bir veri tipi belirlenmesi halinde fonksiyon gövdesinde *return* anahtar kelimesinin kullanılması zorunludur. Aksi halde hata üretilecektir (*The body might complete normally, causing 'null' to be returned, but the return type is a potentially non-nullble type.*).

bir değişkeni parametre olarak alacak şekilde tanımlanmıştır. Fonksiyon çağrılarında bu değişkene değer verilmesi zorunludur. Aksi halde hata üretilecektir (*1 positional argument(s) expected, but 0 found.*). Fonksiyon içinde `dart:math` kütüphanesi tarafından sağlanan π sabiti, parametre olarak gelen *sayı* değişkeni ve 2 değerleri ile çarpılarak (çemberin çevre formülü: $2\pi r$) verilen yarıçap için çember çevresi değeri üretilmektedir. `return` anahtar kelimesi kullanılarak oluşturulan sayısal değer geri çevrilmektedir.



DART DİLİ

İşleçler (Operatörler)

İşleçler programlama dili derleyicilerine verilerle ilgili atama, kontrol ve düzenleme işlemlerini yapma emri veren programlama yapılarıdır. Örneğin hesaplanan bir değerin bir değişkene atanması için atama işaretleri kullanılır. İşleçlerle birleştirilen değerlere ifade adı verilir. Derleyiciler ifadeleri otomatik olarak değerlendirerek gerekli görevlerin gerçekleştirilmesini sağlar.

Tablo 5. 1'de Dart dilinde kullanılabilen işaretlerin kısa bir özeti sunulmuştur.

- Son ek işaretleri bir ifadenin arkasında eklenen işaretleri ifade eder. Örneğin değişkenlerin değerini bir artırmak için `++` işaretini kullanılabılır.
- Ön ek işaretler ise ifadenin önünde kullanılan işaretlerdir. Örneğin Boolean türündeki bir değerin tersini almak için `!` işaretini kullanılır.
- Aritmetik işlemlerin gerçekleştirilmesi için çarpma/bölme ve toplama/çıkarma kümelerine ayrılmış işaretler kullanılır.
- Değerlerin birbirlerine göre durumlarını kontrol etmek için ilişki kontrolü işaretleri kullanılır. Bu işaretlerde büyülük/küçüklik ve eşitlik kontrolleri gerçekleştirilebilir.
- Son olarak ilişki kontrollerini birleştirmek için VE ve VEYA işaretleri kullanılabilir. Örneğin bir sayının 10 ile 30 arasında olma durumunu kontrol etmek için iki ilişki kontrolü `&&` işaretini ile birleştirilir (ör: `sayi>10 && sayi<30`).

Tablo 5.1 Dart İşleçleri

İşleç Ailesi	İşleçler
Son ek işaretçiler	ifade++ ifade-- () [] . ?.
Ön ek işaretçiler	-ifade !ifade ~ifade ++ifade --ifade
Çarpma / Bölme	* / % ~/
Toplama / Çıkarma	+ -
İlişki kontrolü	>= > <= <
Eşitlik kontrolü	== !=
Mantıksal VE	&&
Mantıksal VEYA	

Bir ifadede birden çok işlem kullanılabilir. Bu nedenle işaretlerin öncelik sırasının bilinmesi, üretilecek değerlerin bilinmesi açısından kritiktir. Tablo 1'de satırlarda verilen kümeler arkasından gelen kümelerden daha yüksek önceliklidir. Yani bir ifade içinde çarpma ve toplama işaretleri bir arada kullanılmışsa önce çarpma sonra toplama işlemi gerçekleştirilecektir. Bu nedenle $5+4*2$ ifadesinin sonucu 18 değil, 13 olur.

Aritmetik İşleçler

Aritmetik işaretler toplama, çıkarma, çarpma, bölme ve mod alma gibi işlemleri gerçekleştirmek için kullanılır. Kod 2'de Dart dilinde kullanılan aritmetik işaretler ve etkileri görülmektedir. Her satırın sonunda ilgili işlemin açıklaması ve sonucu yorum satırı halinde sunulmuştur.

```
print(3+2); // toplama 5
print(3-2); // çıkarma 1
var pi=3.14;
print(-pi); // sayının işaretini terse çevirme -3.14
print(2*pi); // çarpma 6.28
print(9/3); // bölme 3
print(10~/3); // kalansız bölme 3
print(10%3); // mod alma 1
```

Kod 2 Aritmetik işaretler

Ön Ek ve Son Ek ve Birleşik Atama İşleçleri

Ön ek ve son ek artırma/azaltma işaretleri uzun işlemlerin kısayoldan ifade edilmesi için kullanılır. Bir değişkenin değerini bir artırma ya da azaltma amacıyla bu işaretler kullanılabilir. Ön ek işaretleri kullanıldığında işlem sonucu ifade işletilmeden önce hesaplanarak değişkene aktarılır. Son ek işaretlerde ise, atama işlemi ifade işletildikten sonra gerçekleştirilir. Bu işaretlerin bir değişkene uygulanması sonucu oluşan değerler Kod 3'te yorum satırları halinde sunulmuştur.

```
var sayi1 = 5;
var sayi2 = 3;
var kalan = sayi1 % sayi2;
if (kalan == 0) {
    print("$sayi1, $sayi2'e kalansız bölünür");
} else {
    print("$sayi1'in $sayi2'e bölümünden kalan $kalan'dır"); // 5'in 3'e
    bölümünden kalan 2'dir
}
```

Kod 3 Ön ek / son ek ve birleşik atama işaretleri

Birleşik atama işleçleri ise değişken üzerinde bir sayı artırma ve azaltma dışında gerçekleştirilecek işlemlerin kısaca ifade edilmesi için kullanılır. Örneğin sayı değişkenini 3 ile çarpmak için açık bir şekilde `sayi=sayi*3;` yazmak yerine, kısaca `sayi*=3;` ifadesi ile bu işlem gerçekleştirilir.

İlişkisel İşleçler

İlişkisel işaretler verilen büyülüklük, küçüklük ya da eşitlik durumlarının kontrol edilmesi için kullanılır. Bu işaretler her zaman doğru (`true`) ya da yanlış (`false`) değerlerini üretir. Kod 4'te ilişki kontrolü işaretlerinin açıklamaları ve verilen durumlarda üretikleri değerler sunulmuştur.

```
var sayı = 3;
print(3==sayı); // eşitlik kontrolü true
print(3>sayı); // büyülüklük kontrolü false
print(3>=sayı); // büyülüklük ve eşitlik kontrolü true
print(3!=sayı); // eşit değildir kontrolü false
var isim = "onur";
print(isim == "onur"); // metin eşitliği kontrolü true
```

Kod 4 İlişki kontrolü işaretleri

Mantıksal İşleçler

Boolean veri tipindeki `true` ve `false` değerleri üzerinde işlem yapabilmek için mantıksal işaretler kullanılır. Kod 5'te kullanılabilen mantıksal işaretler ve Boolean tipindeki değerler ile işleme sokulduğunda üretilen sonuçlar açıklanmıştır.

```
print(!true); // mantıksal ifadenin tersini alma false
print(true && true); // mantıksal ifadeleri VE ile bağlama true
print(true && false); // mantıksal ifadeleri VE ile bağlama false
print(true || false); // mantıksal ifadeleri VEYA ile bağlama true
```

Kod 5 Mantıksal işaretler



dikkat

Gerçek yaşamda Kod 5'de verilen değerlerin ilişkisel sınamalarla üretilmesi (ör: `2==2` ya da `2 > 3`) ya da Boolean tipindeki bir değişkende tutulması (ör: `ciftmi = false`) gerekecektir.

✓ alıştırma

Tanımladığınız sayısal bir değişkende tutulan değerin 10 ile 30 arasında olma durumunu kontrol eden Dart kodunu yazınız.

✓ alıştırma yanıt

```
var sayı = 10;
if(sayı>=10 && sayı <
{
  print("sayı 10 i
```

Durumsal Atama

Dart tek satırlık durumsal atama işaretini destekler. Bu yapı basit bir `if-else` bloğuna benzer. Fakat kümeye parantezleri ve atama satırlarını gerektirmeden, çok daha kısa bir yoldur.

```
var sayı = 3;
var sonuc = (sayı % 2 == 0) ? "çift" : "tek";
print("$sayı $sonuc sayıdır"); // 3 tek sayıdır
```

Kod 6 Durumsal atama işlemi

Kod 6'da *sayı* değişkenine 3 değeri atanmıştır. İkinci satırda durumsal atama işlecinde parantezler içinde *sayı* değişkenin 2'ye bölümünden kalanın sıfır olma durumu kontrol edilmektedir. Bu kontrolün sonucu *true* çıkarsa *sonuc* değişkenine : karakterinden önceki değer; aksi halde sonraki değer atanır. Bu nedenle son satırda *print* işlemi sonucunda "3 tek sayıdır" ifadesi konsola yazılacaktır.



dikkat

Değişkenlerin String ifadeler içine enjekte edilmesi için önlerine \$ karakteri konulabilir.

Anahtar Sözcükler

Dart dilinin rezerve edilmiş, programcılar tarafından özel amaçlarla (ör: değişken isimlendirmek) kullanmayıcağı anahtar sözcükleri bulunmaktadır. Dart dilinin rezerve anahtar sözcüklerinin listesine <https://dart.dev/guides/language/language-tour#keywords> adresinden ulaşılabilir.

Değişkenler

Uygulamalar çalıştırıldığı bilgisayar sistemi üzerindeki bellek alanlarında bazı bilgileri saklayabilir. Bu amaçla programlama dillerindeki değişken yapıları kullanılır. Programcılar kodlarında değişkenler tanımlayarak istedikleri bellek alanlarının uygulamalar için rezerve edilmesini sağlar. Rezerve edilen bu alanlara değer göndermek için atama işlemleri kullanılır (ör: *var sayı = 3;*).

Dart dilinde değişken tanımlamak için *var* anahtar sözcüğü ya da değişkende tutulacak veri tipini belirten anahtar sözcükler (ör: *String, int*) kullanılır. Bu anahtar sözcük arkasında rezerve sözcükler arasında yer almayan bir ifade verilerek değişken tanımlanabilir. Kod 7'de çeşitli sayısal, metin ve mantıksal değişkenlerin tanımlama satırları görülebilmektedir. Bu satırlarda ilk 6 değişken hem tanımlanmış hem de değer ataması yapılmıştır. Son iki değişkene ise değer atanmadığına dikkat ediniz. Bu değişkenler üzerinde atama dışında bir işlem (ör: değerini artırma ya da uzunluğunu inceleme) denenmesi halinde sorun yaşanabilecektir.

```
var sayı = 4;
int sayı2 = 5;
double pi = 3.14;
String isim = "onur";
var isim2 = "onur";
bool ciftmi= true;
var degisken1;
String degisken2;
```

Kod 7 Çeşitli veri tiplerinde tanımlanan değişkenler



dikkat

Dart tip çıkarımı mekanizması sayesinde değişkene atanın ilk değere göre veri tipine karar verebilmektedir. Bu nedenle değişkenlerin int ya da String gibi tip belirten ifadeler yerine, var anahtar sözcüğü ile tanımlanması önerilmektedir.

```
final isim = "onur";
isim = "beril"; // The final variable 'isim' can only be set once.
const soyisim = "dönmez";
soyisim = "ceylan"; // Constant variables can't be assigned a value.
```

Kod 8 Final ve const kullanımı

Tanımlanan değişkenlerin içeriklerinin daha sonra düzenlenmesi engellenemektedir. Bu amaçla *final* ve *const* anahtar sözcükleri kullanılır. Bu anahtar sözcüklerle tanımlanmış değişkenlere kod içinde daha sonra atama işlemi yapılması halinde yorum satırları ile verilen hatalarla karşılaşılır.

Null Güvenliği (Null-Safety) Mekanizması

Null güvenliği mekanizmasının amacı tanımlanmış fakat henüz değer ataması yapılmamış değişkenlerin kullanılmasından kaynaklanan hataların önlenmesidir. Dart 2.1.2 versiyonuyla birlikte null güvenliği desteği sunmaya başlamıştır. Dart derleyicisi içeriği *null* olan değişkenleri derleme öncesinde yakalayarak çalışma zamanı (*run time*) hatalarının önüne geçer.

```
int sayı;
sayı *=6; // The non-nullable local variable 'sayı' must be assigned before
it can be used.
int? sayı2 = null;
```

Kod 9 Null Güvenliği mekanizması

Kod 9'da sayı isimli değişken tanımlanmış fakat bir değer ataması yapılmamıştır. Dolayısıyla bu değişken *null* değerini taşımaktadır. Bu değişken üzerinde çarpma işlemi gerçekleştirilmeye çalışıldığında Dart derleyicisi yorum satırı halinde gösterilen hatayı verecektir.

Programcılar bazı değişkenlerin özellikle *null* değerini alabilmelerini isteyebilir. Bu değişkenlere *nullable* adı verilir. Bir değişkenin *null* değeri alabilir biçimde tanımlanması için tip belirtecinin arkasına ? karakteri eklenir. Son satırda tanımlanan *sayı2* değişkeni bu şekilde tanımlanmıştır.



dikkat

Nullable değişkenler kullanılırken içeriklerinin null değerler için kontrol edilmesi gereklidir.

Öntanımlı Veri Tipleri

Sayılar

Dart kesirli ve tam sayılar için iki tür sayısal veri tipi sağlar. Tam sayılar için *int*, ondalıklı sayılar içinse *double* veri tipi kullanılır. Bu veri tiplerinin saklayabileceği sayıların büyülüüğü uygulamanın koşturulduğu ortama göre değişebilmektedir. Örneğin *int* veri tipi mobil cihazlarda -2^{63} ile $+2^{63}-1$ arasındaki değerleri tutabılırken; web tarayıcılarında bu sınırlar -2^{53} ile $+2^{53}-1$ arasındadır.

```
int sayı1=10;
double pi = 3.14;
var avagadro = 6.02e23;
```

Kod 10 Dart dilinin sayısal veri tipleri

Kod 10'da *sayı1* değişkeni tam sayı verileri tutacak şekilde tanımlanmıştır. *pi* değişkeni ise ondalıklı olacak şekilde tanımlanmıştır. Son satırdağı *avagadro* değişkeninde bir tip verilmemiş; değişkenin veri tipine Dartın karar vermesi beklenmiştir. *avagadro* değişkeninin karşısında üstel bir sayı ifadesi yer almaktadır (6.02×10^{23}).

Metinler

String nesneleri UTF-16 standardındaki karakter katarlarıdır. *String* nesnelerini üretmek için karakterler tek veya çift tırnaklar arasına alınır.

```
String isim = "onur";
var soyisim = 'önmez';
var tekTırnak = 'tek tırnak \' kaçırma';
print("Ad Soyad: $isim $soyisim"); // Ad Soyad: onur önmez
```

Kod 11 String nesneleri üretmek

Kod 11'de tanımlanan *isim* ve *soyisim* değişkenleri *String* tipindedir. Görüldüğü üzere tek tırnak ya da çift tırnak arasına alınan karakterler *String* tipinde nesneler olarak ele alınmaktadır. *String* nesneleri içinde tek tırnak karakterini kullanmak için önüne \ (backslash) karakteri eklenebilir. Ayrıca son satırda gösterildiği gibi tanımlı değişkenlerin önüne \$ karakteri eklenerek bu değişkenler *String* nesneleri içine gömülebilir.

Boolean

Mantıksal doğru ve yanlış değerlerinin saklanması için *bool* veri tipi desteklenir. Boolean tipindeki değişkenler yalnızca *true* ya da *false* değerleri alabilir. Kod 12'de *bool* tipinde iki adet değişken tanımlanarak bu değişkenlere değerler atanmıştır.

```
bool haftaSonu= false;
var isaretlendi = true;
```

Kod 12 Boolean tipindeki değişkenler

Listeler (List)

Bu noktaya kadar Dart dilinde kullanılabilen basit veri tipleri anlatılmıştır. Fakat Dart birden fazla değerin belirlenen düzenlerde saklanabileceğİ karmaşık veri tiplerini de destekler. En temel karmaşık veri tipi aynı veri tipindeki (ör: sayılar) nesnelerin saklandığı listelerdir.

```
var sayilar=[1, 20, 3.14, 6e10];
var haftaIci = ["pazartesi", "sali", "çarşamba", "perşembe", 'cuma'];
print(sayilar[1]); // 20
print(haftaIci.toString()); // [pazartesi, sali, çarşamba, perşembe, cuma]
sayilar[2]=0;
sayilar[4]=20; // Index out of range: index should be less than 4
sayilar.add(90);
sayilar.removeAt(2);
sayilar.insert(2, 3.14);
```

Kod 13 Liste tanımlama ve liste öğelerine erişme

Kod 13'te *sayilar* ve *haftaIci* isimli iki liste tanımlanmıştır. *sayilar* listesi sayısal nesneler tutmaktadır. *haftaIci* listesi ise *String* nesnelerden oluşmaktadır. Bu tipler dışında bir tipteki değerlerin listelere eklenmesi istendiğinde hata mesajı alınacaktır (Ör. *The argument type 'String' can't be assigned to the parameter type 'int'*).

Liste elementlerine erişmek için köşeli parantezler içinde sayısal element indis numarası kullanılır. Örneğin *sayilar* listesinin ilk elementine ulaşmak için *sayilar[0]* ifadesi kullanılır. Kodda da görüldüğü gibi 1 numaralı indisle çağrı yapıldığında ilk değil, ikinci değere erişilmektedir.

Listede tanımlı elementlerin değiştirilmesi için bu elemente indisyle erişerek atama işlemi gerçekleştirilir. Kodda 2 indisli elementin içeriği 0 olarak değiştirilmektedir. Fakat 4 indisli elemente erişim sağlamaya çalışıldığında yorum satırı olarak verilen hata gözlenecektir.

Listeler dinamik veri yapıları olduğundan *add* metodu yoluyla sonlarına element eklenebilir. Kodda *add* metodu kullanılarak *sayilar* listesinin sonuna 90 değeri eklenmektedir. Listelerden veri çıkarmak için *removeAt* metodu kullanılır. Bu metoda listeden çıkarılmak istenen elementin indis numarası verilir. Ör-

neğin, kodda 2 numaralı indisde yer alan `3.14` değeri listeden kaldırılmaktadır. Ardındaki satırda yine aynı indis numarası verilerek `3.14` değeri `insert` metodu yoluyla eski yerine yerleştirilmektedir.

```
sayilar.forEach((element) {print(++element);}); // 2 21 ...
print(sayilar.toString()); // [1, 20, 3.14, 60000000000, 90]
```

Kod 14 Listelerde `forEach` metodunun kullanımı

Listelerdeki elementlerle işlem yapmak için `forEach` metodu kullanılabilir. Kod 14'te sayılar listesinin `forEach` metodu kullanılarak listedeki her bir elementin 1 eklenmiş halleri konsola alt alta yazdırılmaktadır. `forEach` metodu parametre olarak anonim bir fonksiyon alır. Listedeki elementleri de bu anonim fonksiyona parametre olarak gönderir. Fonksiyon gövdesinde elementlere tanımlama bölümünde verilen değişken ismi ile erişilebilir. Dikkat edilmesi gereken bir nokta bu değişkenlerin etki alanının anonim fonksiyonun gövdesi ile sınırlı olduğunu. Koddaki ön ek arttırma işlemi orijinal listedeki sayılara etki etmez.



dikkat

Yukarıdaki kodda kullanılan anonim fonksiyonda tek bir ifade yer aldığından bu fonksiyon ok sentaksi (Ör: `sayilar.forEach((element) => print(++element))`) ile de yazılabilir. Bu yolla daha okunur kodlar üretilebilir. Fakat birden fazla ifadeden oluşan fonksiyon gövdeleri için ok sentaksi kullanılamaz.

```
print(sayilar.indexWhere((element) => element == 30)); // -1
```

Kod 15 `indexWhere` metodu ile listelerde arama yapma

Listelerde arama yapmak için `indexWhere` metodu kullanılabilir. `indexWhere` metodu `forEach` metodunda olduğu gibi parametre olarak anonim bir fonksiyon kabul eder. Bu anonim fonksiyon içine gönderilen değişkenin arama kontrolü yapılarak, bulunması halinde indis numarasını, aksi halde ise `-1` değerini cevirir. Kod 15'teki arama sonucu `-1` cevirdiğinden 30 değerinin sayılar dizisinde yer almadığı anlaşılacaktır.

```
var haftaIci = ["pazartesi", "sali", "çarşamba", "perşembe", "cuma"];
var günler = ["cumartesi", "pazar", ...haftaIci];
print(günler.toString()); // [cumartesi, pazar, pazartesi, sali, çarşamba,
perşembe, cuma]
```

Kod 16 Yayılım işlevi ile listeleri birleştirme

Yayılım işlevi kullanılarak listeler birleştirilebilir. Bu amaçla ikinci listenin oluşturulduğu satırda, ilk liste değişkeninin önüne ... işlevi eklenir. Kod 16'da tanımlanan `haftaIci` listesindeki `günler`, daha sonra tanımlanan `günler` listesine otomatik olarak eklenmektedir.

Haritalar (Map)

Haritalar, anahtar değer çiftlerinden oluşan kümelerdir. Haritalarda değerler birden fazla kez yer alabilirken, anahtarlar eşsiz olmak durumundadır. Ayrıca anahtarlar ve değerler farklı veri tiplerinde olabilir.

```
var baskentler = {
    "Türkiye" : "Ankara",
    "ABD" : "Washington",
    "İngiltere" : "Londra",
};

baskentler.forEach((anh, deg) {
    print("$anh'nin başkenti: $deg");
    // Türkiye'nin başkenti: Ankara
    // ABD'nin başkenti: Washington
    // İngiltere'nin başkenti: Londra
});
```

Kod 17 Harita tipinde kümeler üretme

Kod 17'de *baskentler* ve *sehirler* adında iki adet harita üretilmiştir. Ülkeler ve başkentlerinin ilişkilendirildiği *baskentler* haritasında anahtar ve değerler *String* tipindedir. *sehirler* haritasında ise anahtarlar *int*, değerler ise *String* tipindedir. Haritalardaki değerlere ulaşmak için değişken adı arkasında köşeli parantezler içinde anahtar değeri verilir. Örneğin, *sehirler* haritasındaki *İzmir* değerine ulaşmak için *sehirler[35]* çağrısı yapılmıştır.

```
var sehirler = Map<int, String>();
sehirler[35] = "İzmir";
```

Kod 18 Boş harita tanımlama ve değer ekleme

Kod 17'de tanımlanan haritalarda anahtar değer çiftleri verildiğinden tip çıkarımı mekanizması kullanılabilir miştir. Fakat verilerin internet üzerinden çekilmesi gibi durumlarda boş haritaların oluşturulması ve verilerin sonradan eklenmesi gerekebilir. Boş harita oluşturmak için *Map* anahtar sözcüğü kullanılır. Ardından <> karakterleri arasında anahtar, değer çiftlerinin veri tipleri verilir. Kod 18'de *sehirler* haritası boş olarak tanımlanmış, anahtarları *int* tipinde, değerleri *String* tipinde olacak şekilde belirlenmiştir. Tanımlı bir haritaya veri eklemek için harita adının ardından köşeli parantezler içinde (ör: *sehirler[34]*) anahtar değeri verilerek bu ifadeye atama yapılır. Bu yöntem haritada tanımlı bir anahtarın değerini güncellemek için de kullanılabilir. Haritaladaki bir değeri silmek için ise *remove* metodu kullanılır (ör: *sehirler.remove(34)*). *remove* metoduna silinmek istenen çiftin anahtar değeri verilir.

length özelliği bir haritaladaki anahtar değer çifti sayısını verir. Kod 19'da *sehirler* ve *sehirler2* haritaları tanımlanmaktadır. *sehirler* haritası, *sehirler2* haritasına ... işlevi ile eklenmektedir. Son satırda *sehirler2* haritasının içindeki element sayısı *length* özelliği kullanılarak elde edilmiştir.

```
var sehirler = Map<int, String>();
sehirler[45] = "Manisa";

var sehirler2 = {
    48 : "Muğla",
    ...sehirler
};
print(sehirler2.length); // 2
```

Kod 19 İki haritayı birleştirme ve haritadaki element sayısını belirleme



dikkat

Anahtar-değer çiftleri aynı veri tipinde olan iki haritanın birleştirilmesi için yayılım işlevi (...) kullanılabilir.

forEach metodu haritalardaki tüm değerlerin taranması için kullanılabilir. Bu metot parametre olarak anonim bir fonksiyon kabul eder. Bu anonim fonksiyona sırasıyla anahtar ve değer için iki adet parametre eklenir. Haritadaki çiftlere fonksiyon gövdesinde bu parametre değişkenleri yoluyla ulaşılabilir.

```
var baskentler = {
  "Türkiye" : "Ankara",
  "ABD" : "Washington",
  "İngiltere" : "Londra",
};

baskentler.forEach((anh, deg) {
  print("$anh'nin başkenti: $deg");
  // Türkiye'nin başkenti: Ankara
  // ABD'nin başkenti: Washington
  // İngiltere'nin başkenti: Londra
});
```

Kod 20 *forEach* metodu ile haritalardaki çiftleri tarama

Kod 20'de oluşturulan *baskentler* haritasındaki anahtarlar ve değerler *forEach* metodunda verilen anonim fonksiyon içinde konsola yazdırılmaktadır.

Haritaların *keys* ve *values* özellikleri yoluyla haritadaki anahtarlar ve değerlerin listelerine erişilebilir.

```
var sehirler = {
  1 : "Adana",
  6 : "Ankara",
  34 : "İstanbul",
  35 : "İzmir"
};

var plakalar = sehirler.keys;
var iller = sehirler.values;
print(plakalar.toString()); // (1, 6, 34, 35)
```

Kod 21 Haritalardaki anahtarlar ve değerlere ulaşma

Kod 21'de oluşturulan *sehirler* haritasının anahtarları *plakalar* listesine atanmaktadır. Benzer şekilde *iller* listesi de haritanın *values* özelliği yoluyla doldurulmaktadır. Son satırda *plakalar* listesi konsola yazdırılmaktadır.

Öğrenme Çıktısı



Araştır 3

Liste veri tiplerinin map metodunu ve yeteneklerini inceleyiniz.

İlişkilendir

Harita veri yapısını Javascript dilinde kullanılan JSON sistemi ile ilişkilendiriniz.

Anlat/Paylaş

Listeleri taramak için bir for döngüsü kullanılabilir. Bunun yanında liste veri tipinin *forEach* metodu da kullanılabilmektedir. Bu iki metodun kullanım performanslarını inceleyiniz

3 Dart programlama dilinde veri yapılarını kullanabilme

AKIŞ YÖNETİCİLER

if-else

Uygulamalar her zaman yazılan kodların alt alta işletilmesi ile doğru sonuçlara ulaşamaz. Kullanıcı girişi ya da üretilen ara değerlere göre uygulama akışının yönlendirilmesi gerekebilir. Bu amaçla kullanılan en temel yapı *if-else* bloklarıdır. *if-else* blokları verilen bir şart yapısının sağlanması halinde çalışacak bir blok (*if* bloğu) ve aksi halde çalışacak bir blok (*else* bloğu) tanımlamaya olanak verir. İki bloktan birinin işletilmesi bittiğinde uygulama akışı *else* bloğunun arkasındaki satırdan devam edecektir.

```
var sayıl = 5;
var sayı2 = 3;
var kalan = sayıl % sayı2;
if (kalan == 0) {
    print("$sayıl, $sayı2'e kalansız bölünür");
} else {
    print("$sayıl'in $sayı2'e bölümünden kalan $kalan'dır"); // 5'in 3'e
bölümünden kalan 2'dir
}
```

Kod 22 if-else yapısı ile kalan kontrolü yapma

Kod 22'de tanımlanan iki sayının kalansız bölünmesi incelenmektedir. Mod alma işlemi (%) kullanılarak bölme işleminden kalan değer, *kalan* değişkenine aktarılmaktadır. Bu senaryoda *sayı1* değişkenine 5, *sayı2* değişkenine 3 atandığından *kalan* değişkeninin değeri 2 olacaktır.

Kontrol ifadesinde kalan değişkenindeki değerin sıfıra eşitliği incelenmektedir. Bu kontrolden doğru (*true*) değeri çıkması halinde *if* bloğu çalışacak ve ekrana “5, 3'e kalansız bölünür” yazılacaktır. Fakat bu kontrol yanlış (*false*) değeri çıkaracağından (2, sıfıra eşit değildir) *else* bloğu çalışacak, ekrana “5'in 3'e bölümünden kalan 2'dir” yazılacaktır.

```
var mA = 90;
var mB = 60;

if (mA == mB) {
    print("A açısı B açısına eşittir");
} else if (mA > mB) {
    print("A açısı B açısından büyük");
} else {
    print("B açısı A açısından büyük");
}
```

Kod 23 Zincir if-else yapıları ile iki sayı arasındaki ilişkinin kontrolü

Kod 23'te iki sayının birbirine göre büyüklük ilişkisi incelenmektedir. A açısı B açısından büyük olduğundan ilk *if* kontrolü *else* bloğuna düşecektir. Burada yeni bir *if* kontrolü yapılmaktadır. Bu kontrol doğru sonucunu vereceğinden, *if* bloğu çalışacak ve ekrana “A açısı B açısından büyük” yazılacaktır.

switch-case Yapıları

switch-case yapıları bir değişkenin alabileceği farklı değerlere göre işlem yapabilir. Değişkenin alabileceği her değer için bir case alanı açılarak bu alan içinde gerekli düzenlemeler yapılır.



dikkat

if-else yapıları birbirine bağlanarak birden fazla durumun kontrol edildiği zincirler oluşturulabilir. Bu amaçla else ifadesinden sonra yine bir if kontrolü eklenir.

```

var sayil = 7;
var kalan = sayil % 3;

switch (kalan) {
  case 2:
  case 1:
    print("$sayil, 3'e kalansız bölünmez"); //7, 3'e kalansız bölünmez
    break;
  default:
    print("$sayil, 3'e tam bölünür");
    break;
}

```

Kod 24 switch-case yapısının kullanımı

Kod 24'te verilen bir sayının üçe kalansız bölünmesi kontrol edilmektedir. *sayil* değişkenine 7 değeri atanarak *kalan* değişkenine bölümün kalanı olan 1 değeri aktarılmıştır. *switch* bloğu kalan değişkenine göre çalışacak şekilde düzenlenmiştir. Bu işlemde kalan 0, 1 ya da 2 değerlerini alabileceğinden 1 ve 2 değerleri için *case* blokları tanımlanmıştır. 0 değeri için blok tanımlanmamış, diğer blokların çalışmaması halinde otomatik olarak çalıştırılacak olan *default* bloğu kullanılmıştır. *case* blokları içinde kullanılabilen ifadelein bir sınırı yoktur. Kodlar istenildiği kadar uzatılabilir. Her *case* bloğu *break* ifadesi ile sonlandırılır. *break* ifadesi ile sonlandırılmayan *case* blokları bir sonraki *case* bloğunun çalışmasını tetikler. Bu örnekte hem 1 hem de 2 kalanı için aynı çıktı verileceğinden ilk *case* bloğunda işlem yapılmamış ve bu blok özellikle sonlandırılmıştır. Tanımlı *case* bloklarının çalışmaması halinde çalıştırılan *default* bloğunun bulunması zorunlu değil, fakat önerilen bir uygulamadır.

for Döngüleri

Hazırlanan kodların belirli bir şart sağlandığı sürece tekrar tekrar çalıştırılması için döngü yapıları kullanılır. Dart standart *for* döngülerini destekler.

```

var sayı = 48;
var asalmi = true;
for (var i = 2; i <= sqrt(sayı); i++) {
  if (sayı % i == 0) {
    asalmi = false;
    break;
  }
}
if (asalmi) {
  print("$sayı asal bir sayıdır");
} else {
  print("$sayı asal bir sayı değildir");
}

```

Kod 25 for döngüsü ile asallık kontrolü

Asal sayılar tanımları gereği yalnızca 1 ve kendilerine bölünebilen sayılardır. Kod 25'te bir sayının asal olup olmadığını kontrol edilmektedir. *i* değişkeni döngü kontrol değişkeni olarak tanımlanmıştır. Asal sayılar tanımları gereği 1'e bölünebildiğinden değişken 2'den başlatılmıştır. Bir sayının en büyük böleni karekökü olabileceğiinden, döngü *i* değişkeninin değeri sayının karekökünden küçük ya da eşit olduğu sürece dönebilecek şekilde tanımlanmıştır. Son olarak döngünün her dönüşünde kontrol değişkeni *i*'ye 1 eklenmektedir.

Döngü blogunda ise *sayı* değişkeninin *i*'nin güncel değerine kalansız bölünme durumu kontrol edilmektedir. Bu durum gerçekleştiğinde *bool* tipindeki *asalmi* değişkeninin değeri *false* olacak şekilde güncellenmekte ve *break* ifadesi ile döngü kırılmaktadır. Döngünün arkasında *asalmi* değişkeninin değerine bakılarak sayının asal sayı olup olmaması konusunda bilgi verilmektedir.

while Döngüleri

while döngüleri belirlenen şartın sağlanması (ör: klavyeden bir tuşa basılana kadar) süresince dönen yapılardır. *for* döngüleri daha çok bir kontrol değişkeninin tanımlanması ve manipülasyonu yoluyla işletilir. Bu yönlemeyle *while* döngüleri *for* döngülerinden ayırlar. Fakat pek çok işlem her iki döngü türüyle de gerçekleştirilebilir.

```
var sehirler = ["adana", "ankara", "istanbul", "izmir", "trabzon"];
while (sehirler.isNotEmpty) {
    print(sehirler[0]);
    sehirler.removeAt(0);
}
```

Kod 26 while döngüsü ile bir listeyi boşaltmak

Kod 26'da verilen *while* döngüsü her dönüşünde *sehirler* listesindeki ilk elementi (*sehirler.removeAt(0)*) çiğnarmaktadır. Döngü kontrol blogunda *sehirler* dizisinde element olduğu sürece olarak ifade edilebilecek *sehirler.isEmpty* ifadesi kullanılmıştır. Bu yolla son element diziden çıkarıldıkten sonra döngü tekrar çalıştırılmaz.

💡 alıştırma

1 ile 200 arasındaki sayıları bir listeye atarak bu listedeki asal sayıları bulan kodu yazınız. Bu işlem için sayıların katlarını silme metodunu kullanabilirsiniz. Bu metodun kağıt üzerinde uygulanışını görmek için https://www.youtube.com/watch?v=B_hJcUAIOFA adresini ziyaret ediniz.

💡 alıştırma yanıt

```
List<int> sayilar = [];
var enbuyuk = 1000;
// sayilar listesini doldur
for (int i = 2; i < enbuyuk; i++) {
    sayilar.add(i);
}

var sayiIndeks = 0;
var indeks = 0;
var carpan = 2;
var deger = 0;
// en son sayıya kadar inceleme yap
while (sayiIndeks < sayilar.length) {
    var sayim = sayilar[sayiIndeks];
    carpan = 2;
    deger = sayim * carpan;
    // sayı ve çarpım değeri enbüyük değerinden küçük olduğu sürece çalış
    while (deger < enbuyuk) {
        // çarpım sonucu liste içinde varsa bunu listeden çıkar
        indeks = sayilar.indexOf(deger);
        if (indeks > 0) {
            sayilar.removeAt(indeks);
        }
        // carpanı arttırarak sonraki katı bul
        carpan++;
        deger = carpan * sayim;
    }
    // listedeki bir sonraki sayıya geç
    sayiIndeks++;
}
print(sayilar.toString());
```

Fonksiyonlar

Uygulamalar içinde sıkılıkla tekrarlanan kod blokları fonksiyonlar içine paketlenebilir. Bu kodlar fonksiyonun çağırılması yoluyla çalıştırılabilir. Kendi içlerinde çalışan kod blokları olan fonksiyonlar, parametreler yoluyla dışarıdan veri alabilir ve return ifadesi ile tanımlı veri tiplerinde değer döndürebilir. Kod 27'de kendisine gönderilen yarıçap değerini pi ve 2 ile çarparak (çemberin çevre formülü) sonucu geri çeviren *cevre* fonksiyonu tanımlanmıştır.

```
double cevre(double yaricap) {
  var cevrem = pi * 2 * yaricap;
  return cevrem;
}

double sonuc = cevre(3.6);
print(sonuc); // 22.61946710584651
```

Kod 27 Fonksiyon kullanımı

Dart dilinde fonksiyon tanımlamak oldukça basittir. Fonksiyonun adının ardından parantezler içinde fonksiyona veri göndermek için kullanılacak parametreler tanımlanır (ör: *cevre(double yaricap)*). Fonksiyon tanımlamak için özel bir anahtar kelime kullanılmaz (ör: *function*). Fonksiyonlara parametre gönderme zorunluluğu yoktur. Yine de tanımlama satırlarında parametrelerin gönderildiği alanı işaretleyen parantezlerin kullanılması zorunludur. Fonksiyon geriye değer döndürerekse bu değerin veri tipinin belirtilmesi (ör: *double*, *int*) gereklidir. Geriye değer döndürmeyen fonksiyonlar *void* veri tipinde tanımlanır. *void* dışında bir veri tipi tanımlanması halinde fonksiyon gövdesinde *return* ifadesinin kullanılması zorunludur. Aksi halde hata üretecektir (ör: *A non-null value must be returned since the return type 'double' doesn't allow null*). Fonksiyon gövdesi küme parantezleri arasına yerleştirilir.

Tanımlı bir fonksiyonu kullanmak için adının belirtilmesi yeterlidir (ör: *cevre(2)*). Bu çağrıda fonksiyon tanımlamasındaki parametreler uyulması gereklidir. Aksi halde hata üretecektir (ör: *Too few positional arguments: 1 required, 0 given*.). Değer çeviren fonksiyonlardan dönecek değerler bir değişkene atanabilir (ör: *sonuc = cevre(2);*) ya da doğrudan başka bir metoda (ör: *print(cevre(2));*) gönderilebilir.

Dart tek işlemlik fonksiyonlar tanımlanmasını sağlayan ok sentaksını destekler. Kod 28'de bir önceki örnekte tanımlanan *cevre* fonksiyonunun ok sentaksi ile tanımlanmış hali görülmektedir. Bu kullanımda fonksiyon tanımlama bloğu değişmez. Fakat fonksiyonun gövdesi ok (=>) ardından tek bir ifade olarak verilir. Bu ifadenin sonucu hesaplanarak otomatik olarak geri çevrilecektir. Bu nedenle *return* kullanılmasına gerek yoktur.

```
double cevre(double yaricap) => yaricap * 2 * pi;
print(cevre(3.6)); // 22.61946710584651
```

Kod 28 Ok sentaksi ile tek satırlık fonksiyonlar tanımlama



dikkat

Geriye değer çeviren fonksiyonlarda veri tipi tanımlaması zorunlu değildir. Tip çıkarımı mekanizması dönen değerin veri tipini çikarsayabilir. Fakat fonksiyon tanımlamasında dönecek değerin veri tipini belirtmek önerilen bir uygulamadır.



dikkat

Ok sentaksi fonksiyon gövdesinde ikinci bir ifade kullanılmasına izin vermez. Böyle durumlarda normal fonksiyon tanımlaması kullanılır.

Pozisyonel parametreler fonksiyonlara tanımladıkları sıra ile gönderilir. Fakat parametre sayısı arttığında bu sıranın takip etmesi zorlaşabilir. Bunun gibi durumlarda isimli parametreler kullanılabilir.

```
double alan(hipotenus, {double taban = 5, double yukseklik = 3}) {
    return (taban * yukseklik / 2);
}

print(alan(6, taban: 10, yukseklik: 3)); // 15
```

Kod 29 Pozisyonel ve isimli parametrelerin birlikte kullanımı

Kod 29'da tanımlı alan fonksiyonu aldığı taban ve yükseklik değerleri üzerinden üçgen alanı hesaplayarak geri çevirmektedir. Fonksiyonun tanımlama satırına bakıldığında *hipotenus* adında bir pozisyonel parametre ile *taban* ve *yukseklik* adında iki isimli parametre kullanıldığı görülmektedir. Fonksiyon tanımlamalarında isimli parametreler küme parantezleri içinde ve her zaman pozisyonel parametrelerden sonra gelir. Bu örnekte *taban* ve *yukseklik* parametrelerine varsayılan değer ataması yapılmıştır. Bu atamanın yapılmaması halinde bu değişkenlerin degersiz çağrıması ihtimali gündeme gelebilir. Bu durumu engellemek için değişkenlerin önüne *required* (ör: *required double taban*) anahtar sözcüğü eklenir.



dikkat

Fonksiyonlara parametre gönderilirken değişkenlerin kendileri değil, değerleri gönderilir. Dart referans yoluyla parametre gönderme işlemini desteklemez. Bu nedenle parametre olarak gönderilen değişkenler, fonksiyon gövdesindeki işlemlerden etkilenmez.

Değişkenlerin Etki Alanları

Küme parantezleri kod içinde özel etki alanları (*lexical scope*) tanımlar. Değişkenler yalnızca tanımlandıkları etki alanı içinde geçerlidir. Bu etki alanı dışından değişkenlere erişilmeye çalışıldığında bir hata üretilir (ör: *Error: Getter not found: degisen_adi*).

```
double alan({required double taban, required double yukseklik}) {
    var sonuc = taban * yukseklik / 2;
    return (sonuc);
}
print(sonuc); // Error: Undefined name 'sonuc'
```

Kod 30 Bir fonksiyon içinde tanımlı değişkene erişim hatası

En temel etki alanlarından biri fonksiyon gövdeleridir. Kod 30'da son satırda, *alan* fonksiyonunun etki alanında tanımlı *sonuc* değişkenine erişilmeye çalışılmaktadır. Bu değişken *alan* fonksiyonu etki alanında geçerli olduğu için bu çağrı hata verecektir (ör: *Error: Undefined name 'sonuc'*).

Değişkenler tanımlandıkları etki alanıındaki etki alanlarında kullanılabilir.

```
var taban = 10;
var yukseklik = 5;
double alan() {
    return (taban * yukseklik / 2);
}
print(alan()); // 25
```

Kod 31 Üst etki alanındaki değişkenlere erişim sağlama

Kod 31'de tanımlanan *alan* fonksiyonu içinde *taban* ve *yukseklik* değişkenlerine erişildiği görülmektedir. Fakat bu değişkenler fonksiyona parametre olarak gönderilmemiş ya da fonksiyon gövdesinde tanımlanmamıştır. Yine de bu kod hata üretmeden doğru sonucu verecektir. Çünkü bir etki alanı üstündeki etki alanında tanımlı değişkenlere erişebilmektedir.



Öğrenme Çıktısı

4 Dart programlama dilinde akış yöneticilerini ve döngü yapılarını kullanabilme

Araştır 4

Hazırlanan bir algoritmanın performansının en önemli göstergelerinden biri algoritmanın bitirilmesi için geçen sürenin hesaplanmasıdır. Bu amaçla algoritmanın başlangıcı ve bitişinde elde edilen Unix zamanlarının farkı incelenebilir. Dart dilinde Unix zamanını nasıl alacağınızı araştırınız.

İlişkilendir

Bu üitede yer alan “1 ile 200 arasındaki sayıları bir listeye atarak bu listedeki asal sayıları bulan kodu yazınız.” etkinliğini içeren alıştırma ile sayıların asal olma durumunu kontrol eden iki metot incelenmiş oldu. Bu iki metodun zamana göre performansını Unix zamanı farkı metoduyla inceleyiniz.

Anlat/Paylaş

İki asal sayı bulma yönteminden daha yüksek performans gösteren metodun neden daha hızlı olduğunu tartışınız.

DART İLE NESNEYE YÖNELİK PROGRAMLAMA

Dart nesne yönelimli bir dildir. Dart içinde kullanılan her yapı bir nesne üzerinden üretilmiştir. Nesneler (*Object*) birbirleri ile ilişkili kodlar ve değerleri bir arada tutan yapılardır. Nesnelerin temelinde sınıflar yer alır. Sınıflar, nesnelerin oluşturulması için kullanılan taslaklardır. Tanımlanmış bir sınıf üzerinden istenilen sayıda nesne üretilebilir. Tüm bu nesneler temellerinde yatan sınıfın özelliklerini taşıyacaktır.

```
class Ucgen {
  String tur = "dik üçgen";
  double taban = 10;
  double yukseklik = 20;

  double alan() {
    return (taban * yukseklik / 2);
}
```

Kod 32 Bir sınıf örneği

Kod 32'de *Ucgen* sınıfı tanımlanmıştır. Bu sınıf üçgenin türü, taban uzunluğu ve yüksekliği ile ilgili değerler tutmaktadır. Bu değişkenlere nesnenin özellikleri (*property*) adı verilir. Sınıf üzerinden üretilen tüm nesnelerde bu özellikler varsayılan olarak gelir. Bu örnekte *tur*, *taban* ve *yukseklik* özellikleri için değer atamaları yapılmıştır. Fakat bu zorunlu bir uygulama değildir. Sınıf üzerinden nesne üretilirken bu değerlerin verilmesi, ya da sonradan güncellenmesi sağlanabilir.

Ucgen sınıfı *alan* adında bir metot sağlamaktadır. Sınıflara bağlı fonksiyonlara metot adı verilmektedir. *alan* metodu *taban* ve *yukseklik* özelliklerini kullanarak üçgenin alanını çevirmektedir.

```
var ucgenim = Ucgen();
print(ucgenim.taban); // 10
ucgenim.yukseklik = 15;
print(ucgenim.alan()); // 75
```

Kod 33 Nesne üretme, özellik ve metodları kullanma

Kod 33'te tanımlanan *ucgenim* değişkeni *Ucgen* sınıfından üretilen bir nesnedir. Bu nesne *Ucgen* sınıfında tanımlı tüm özellik ve metodlara sahiptir. Bir nesnenin özelliklerine erişmek için nesnenin adından sonra *.ile* birlikte ilgili özelliğin adı verilir (ör: *ucgenim.taban*). Kodda ikinci satırda *taban* özelliğinin değeri okunurken, üçüncü satırda *yukseklik* özelliğinin değeri değiştirilmektedir. Son satırda ise *alan* metodu çağrılarak güncel değerlere göre *ucgenim* nesnesinin alanı konsola yazdırılmaktadır.

Nesnelerin özellikleri üretildikleri anda düzenlenebilir. Bu amaçla yapıçı metotlar (*constructor*) kullanılır.

```
class Ucgen {
    String tur;
    double taban;
    double yukseklik;
    Ucgen({required this.tur, required this.taban, required this.yukseklik

        Ucgen.dikUcgen(
            {this.tur = "dik üçgen", required this.taban, required
            this.yukseklik});

        double alan() {
            return (taban * yukseklik / 2);
        }
}
```

Kod 34 Yapıçı metot kullanımı

Kod 34'te tanımlanan *Ucgen* sınıfının *tur*, *taban* ve *yukseklik* değişkenlerine değer atanmamıştır. Bu değerlerin atanması için yapıçı metot işe koşulmuştur. Yapıçı metotlar sınıfın adı (ör: *Ucgen*) ile isimlendirilir. Bir sınıfta birden çok yapıçı metot tanımlanabilir. Bu kodda kullanılan *dikUcgen* metodu da aslında bir yapıçı metot olarak kullanılmaktadır. Bu metotta tür değişkeni varsayılan olarak dik üçgen olacak şekilde düzenlenerek kodda tekrar değer ataması gerekmemektedir.

```
var ucgenim = Ucgen(tur: "ikizkenar üçgen", yukseklik: 20, taban: 10);
var ucgenim2 = Ucgen.dikUcgen(taban: 30, yukseklik: 20);
print(ucgenim.alan()); // 75
print(ucgenim2.alan()); // 300
```

Kod 35 Yapıçı metotlar ile nesne üretme

Kod 34'te tanımlı *Ucgen* sınıfı yapıçı metodunda *tur*, *yukseklik* ve *taban* değişkenlerinin verilmesi zorunlu kılınmıştır. Bu nedenle *ucgenim* değişkenine atanan nesne üretilirken tüm değerler verilmek durumundadır. Kod 35'te tanımlanan *ucgenim2* değişkenine atanan nesne ise *dikUcgen* yapıçı metodu ile üretilmektedir. Bu metotta *taban* ve *yukseklik* değişkenlerinin verilmesi yeterlidir.

Kalıtım (*inheritance*) mekanizması yoluyla bir sınıf, daha önce yazılmış bir sınıfın kodlarını miras alabilir. Bu amaçla yeni sınıf adı verildikten sonra extends anahtar kelimesi kullanılarak hangi sınıfın kodlarının miras alınacağı belirtilir.

```
class EskenarUcgen extends Ucgen {
  EskenarUcgen({required double taban})
    : super(
      tur: "eskenar üçgen",
      taban: taban,
      yukseklik: taban * sqrt(3),
    );
}
```

Kod 36 Ucgen sınıfı üzerinden üretilen EskenarUcgen sınıfı

Eşkenar üçgenlerin bir kenar uzunluğunun bilinmesi yüksekliklerinin, dolayısıyla da alanlarının bilinmesi için yeterlidir. Bu nedenle bir eşkenar üçgen tanımı yapılırken yalnızca taban özelliğinin verilmesi yeterli olacaktır.

Kod 36'da EskenarUcgen sınıfının Ucgen sınıfından kalıtım yoluyla üretildiğini belirtmek için *extends Ucgen* ifadesi kullanılmıştır. Kalıtım yoluyla üretilen sınıflar üst sınıfların tüm özellik ve metodlarını miras alırlar. Sınıf içinde üretilen *EskenarUcgen* yapıcı metodunda yalnızca *taban* değişkeninin *required* olarak işaretlendiğine dikkat ediniz. *Ucgen* sınıfı yapıcı metodunda *tur* ve *yukseklik* değişkenleri *required* işaretlendiğinden bu değişkenlere değer ataması yapılması zorunludur. Bu amaçla : işlevi arkasında *super()* çağrıları ile kalıtım uygulanan sınıfın yapıcı metodu çağrılmaktadır. *super()* çağrıları içinde gerekli tüm değişkenlere değer ataması yapılmaktradır.

```
var ucgenim3 = EskenarUcgen(taban: 10);
print(ucgenim3.yukseklik); // 17.32050807568877
```

Kod 37 EskenarUcgen sınıfı üzerinden nesne üretme ve özelliklerini kullanma

Kod 37'de *EskenarUcgen* sınıfından bir nesne üretilerek *ucgenim3* değişkenine atanmaktadır. Bu üretim için yalnızca *taban* değişkeninin verildiğine, *yukseklik* değişkeninin otomatik olarak hesaplandığına dikkat ediniz.



dikkat

Bu amaçla *dikUcgen* örneğinde olduğu gibi *Ucgen* sınıfına bir *eskenarUcgen* yapıcı metodu eklenebilirdi.

Öğrenme Çıktısı

5 Dart programlama dilinde nesneye yönelik programlama kullanabilme		
Araştır 5 <p>Dart diğer dillerde bulunan <i>public</i> ve <i>private</i> gibi erişim denetleyicilerini kullanmaz. Yine de bir değişken ya da fonksiyonun kütüphane dışından erişimi engellenebilir. Bu amaçla hangi işaretinin kullanıldığını araştırınız.</p>	İlişkilendir <p>Dart ve Javascript dillerini nesneye yönelik programlama pratikleri açısından karşılaştırınız.</p>	Anlat/Paylaş <p>Dart ve Javascript dillerindeki nesneye yönelik programlama pratiklerinden hangisinin daha pratik olduğunu tartışınız. Bu pratiklik hangi avantaj ve dezavantajları getirmektedir?</p>

ÇOK SAYFALI UYGULAMALAR

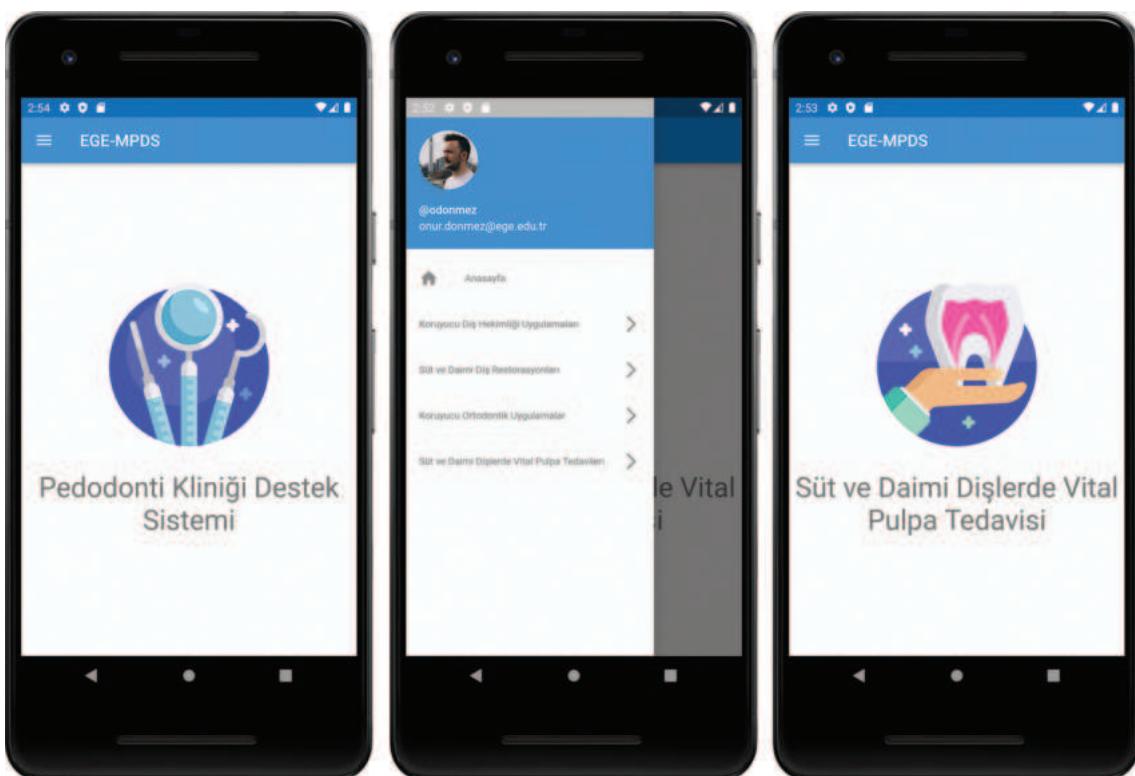
Mobil uygulamalar farklı işlevler için üretilmiş pek çok sayfadan oluşabilir. Örneğin bir sayfa sosyal ağ akışını sunabilirken, farklı bir sayfada yeni bir gönderi üretilebilir. Başka bir sayfada da uygulama ayarları güncellenebilir. Resim 1'de eğitim amaçlı hazırlanmış bir mobil uygulamanın farklı iki sayfası ve bu sayfalar arasında geçiş yapmak için kullanılabilecek çekmece (*drawer*) bileşeni gösterilmektedir.

Flutter uygulamaları farklı görev ve işlevleri olan bileşenlerin bir bileşen ağacı (Widget tree) bir araya getirilmesi ile oluşturulur. Flutter *StatelessWidget* ve *StatefulWidget* olmak üzere iki tür bileşen sağlar. Bunların arasındaki temel fark *StatelessWidget* türündeki bileşenlerin içeriklerinin kurgulandıktan sonra değişmemesidir. *StatefulWidget* türündeki bileşenler ise arkada çalışan veri yapısının güncellenmesi halinde kendini güncelleyebilir.



dikkat

Flutter uygulamalarının kullanıcı arayüzleri bileşenler (widget) ile kurgulanır. Tüm bir uygulama bileşen ağacı (widget tree) şeklinde tasarlanır. Flutter uygulamalarının kendileri dahi birer bileşendir.



Görsel 5.2 Çok sayfalı uygulama örneği

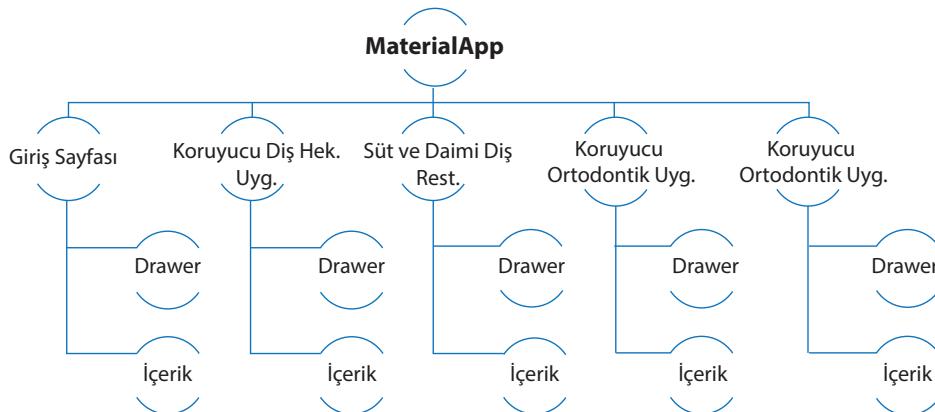
Bu uygulamanın geliştirilmesi için kullanılacak bileşen ağacının basitleştirilmiş bir temsili Şekil 1'de sunulmuştur. Ağacın kökünde uygulamanın kendisi olan bir *MaterialApp* bileşeni yer alır. Rota olarak adlandırılan her sayfa kendi içinde bir bileşen ağacıdır. Bu bileşenlerin kökünde sayfanın yerleşimini sayfaların yerleşimini düzenlemek için kullanılan *Scaffold* bileşenleri yer alır. *Scaffold* bileşenleri sayfa içeriği ve dikey menülerin yerleştirilmesi



dikkat

Scaffold bileşenleri sayfalardaki alt barlar, üst barlar ve menülerin yerleştirilmesi için kullanılabilecek slotlar sunar. Çoğu Flutter uygulamasının yerleşimi *Scaffold* bileşenleri ile düzenlenir.

icin gerekli slotları sağlar. *Drawer* bileşenleri resimlerde gösterilen dikey menülerin gösterilmesi için kullanılır. Dikey menü içinde hesap bilgileri alanı (*UserAccountsDrawerHeader*) ve bağlantılar için satırlar (*ListTile*) bileşenleri kullanılmıştır. Sayfalarda ise kullanılan sütun yapısı (*Column*) sayesinde alt alta gösterilen resim (*Image*) ve metin (*Text*) bileşenleri yer almaktadır.



Şekil 5.1 Çok sayfalı uygulamanın temel bileşen ağıacı

Başlangıç Sayfası

Flutter uygulamaları genellikle *lib* klasörü içindeki *main.dart* dosyası üzerinden başlatılır. Bu dosya içindeki *main* fonksiyonu uygulamanın başlangıç noktasıdır.

Kod 38 incelendiğinde uygulamanın bir dizi *import* ifadesi ile başladığı görülecektir. Bu *import* isteklerinden ilki Google tarafından Android uygulamalarının üretilmesi için oluşturulan Materyal Tasarım ilkelerine odaklı içeriklerin geliştirilmesini sağlayan *material.dart* dosyasını çağırmaktadır. Bu *import* sayesinde *MaterialApp*, *StatelessWidget* gibi bileşenler uygulama içinde kullanılabilir hale gelmektedir. Bu nedenle *material.dart* çağrıları zorunludur. Diğer *import* çağrıları ise üretilen sayfaların uygulamaya eklenmesi için kullanılmaktadır. Bu sayfaların anatomisi aşağıda açıklanacaktır.

main fonksiyonu içinde, *runApp* fonksiyonuna kodlar içinde tanımlanan *MyApp* sınıfının yapıcı metodу çağrılmaktadır. Bu yolla uygulama başlatılmış olur. *StatelessWidget* sınıfından türetilen *MyApp* sınıfı bir *MaterialApp* nesnesi üreterek geri çevirmektedir. Bu nesne uygulamanın köküdür.

Dikkat edilirse sınıftaki kodların tümünün *MaterialApp* nesnesinin yapıcı metoduna parametre gönderdiği anlaşılmaktadır. Bu parametreleri kısaca açıklayalım:

debugShowCheckedModeBanner: Uygulama başlık çubuğu DEBUG etiketinin gösterilmesini engeller ya da izin verir. Bu uygulamada değeri *false* olduğundan bu etiket gösterilmeyecektir.

title: Uygulama başlık çubuğu gösterilen ifadeyi belirler.

```

import 'package:flutter/material.dart';
import 'widgets/girisSayfasi.dart';
import 'widgets/koruyucuDisHekimligi.dart';
import 'widgets/koruyucuOrtodontik.dart';
import 'widgets/pulpaTedavisi.dart';
import 'widgets/restorasyon.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'EGE-MPDS',
      initialRoute: '/',
      routes: {
        '/koruyucu': (context) => const KoruyucuDisHekimligi(),
        '/restorasyon': (context) => const Restorasyon(),
        '/ortodonti': (context) => const KoruyucuOrtodontik(),
        '/pulpa': (context) => const PulpaTedavisi(),
        '/': (context) => const GirisSayfasi(),
      },
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
    );
  }
}

```

Kod 38 main.dart dosyasının kodları

routes: Uygulamada kullanılacak rotaların listesidir. '/koruyucu': (context) => const KoruyucuDisHekimligi() ifadesi rotalar listesine /koruyucu isimli bir rota kaydeder. /koruyucu rotası çağrıldığında KoruyucuDisHekimligi sınıfının bir nesnesi üretilerek gösterim alanına (context) eklenecektir.

initialRoute: Uygulamanın başlangıcında gösterilecek rotayı belirler. Buna göre uygulama / ile etiketlenmiş olan GirisSayfasi sınıfının bir nesnesinin gösterilmesi ile başlayacaktır.

theme: Uygulama teması ile ilgili genel düzenlemelerin yapılabacağı alandır. Burada uygulananın renk teması mavi olacak şekilde düzenlenmiştir.

Giriş Sayfası

Uygulama başladığı anda gösterilecek ilk rota olan *GirisSayfasi* özünde bir *Scaffold* bileşenidir. *Scaffold* bileşenleri üst bar (*appBar*), alt bar (*bottomNavigationBar*), çekmece (*drawer*), eylem düğmesi (*floatingActionButton*) ve sayfa gövdesi (*body*) gibi temel arayüz bileşenlerinin yerleştirilmesi için slotlar sağlar. Kod 39 dikkatle incelenirse tüm kodların bu isimli parametrelerle gönderilecek nesnelerin üretilmesi için kullanıldığı anlaşılacaktır.

```

import 'package:flutter/material.dart';
import 'cekmece.dart';

class GirisSayfasi extends StatelessWidget {
  const GirisSayfasi({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("EGE-MPDS")),
      drawer: Cekmece(),
      body: Center(
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              Image.asset(
                "img/dentist-tools.png",
                width: 200,
              ),
              SizedBox(height: 30),
              Text(
                "Pedodonti Kliniği Destek Sistemi",
                textAlign: TextAlign.center,
                style: Theme.of(context).textTheme.headline4,
              )
            ],
          ),
        ),
      ),
    );
  }
}

```

Kod 39 girisSayfasi.dart dosyasının kodları

Bir önceki örnekte olduğu gibi tüm parametreleri inceleyelim:

appBar: Uygulamanın üstünde gösterilecek menü barını göstermek için kullanılır. Kodda, bu parametreye yerleştirmek için bir *AppBar* nesnesi üretilmektedir. Bu nesnenin içinde metin içerik göstermek için de bir *Text* bileşeni kullanılmaktadır.

drawer: Çekmece bileşenleri normalde gizli duran ve üst bardaki bir kontrol ile gösterilen dikey menülerdir. Bu kodlarda farklı bir dosyada üretilen *Cekmece* sınıfının yapıçı kodları çağrılmaktadır.

body: Sayfa gövdesidir. Bu sayfada ekranda gösterilmek istenen tüm içerikler *body* parametresine atanmış bileşen ağacına eklenir. Bu örnekte içeriklerin alt alta gösterilmesi için bir *Column* bileşeni üretilmiştir. *Column* bileşenine eklenen üç farklı bileşen (*Image*, *SizedBox* ve *Text*) sayfada ortalanmış (*mainAxisAlignment* ve *crossAxisAlignment*) olarak gösterilecek şekilde tanımlanmıştır.

Çekmece Sayfası

Tüm sayfalarda gösterilen *Cekmece* sayfası bir *Drawer* nesnesidir. *child* parametresi tek bir bileşen nesni kabul ettiğinden dikey satırları üretmek için bir *ListView* nesnesi oluşturulmuştur. Her bir bileşen *ListView* nesnesinin *children* özelliğinde liste halinde oluşturulmuştur.

İlk satırda özelleşmiş bir içerik paneli olan *UserAccountsDrawerHeader* nesnesi yer almaktadır. Bu panel pek çok Flutter uygulamasında kullanıldığından tanındık gelebilir. Bu nesne kullanıcı adı, kullanıcı e-postası ve kullanıcı fotoğrafının gösterilmesi için özel alanlar sağlamaktadır. Kullanıcı fotoğrafı için yuvarlak bir avatar fotoğrafı oluşturan *CircleAvatar* nesnesi kullanılmıştır. Bu nesnenin artalan özelliğine de *AssetImage* nesnesi ile sağlanan görsel yerleştirilmiştir. Şu ana kadar bahsedilen tüm nesnelerin birer bileşen olduğuna dikkat ediniz.

Kullanıcıların dokunabileceği menü satırları ise *ListTile* bileşenleri ile oluşturulmaktadır. Bu bileşenler menü satırlarını üretmek için özel olanaklar sağlar. *title* parametresinde satırda gösterilecek ana metin yazılır. Bu örnekte kullanılmasa da bu metnin altında gösterilebilecek farklı bir satır için *subtitle* parametresi sağlanmaktadır. *ListTile* bileşenlerine iki adet simge eklenebilmektedir. Metnin solunda gösterilecek simgeler için *leading* parametresi kullanılır. Metnin sağında gösterilmek istenen simgeler içinse *trailing* parametresi kullanılabilir. Anasayfa satırı için *leading*, diğer satırlar için *trailing* parametrelerinin kullanıldığına dikkat ediniz.

Flutter çok zengin bir simge kütüphanesi sağlar. Bu simgelerin arayüze eklenmesi için *Icon* sınıfı kullanılır. *Icon* sınıfı yapıçı metoduna *Icons* sınıfı altında tanımlı özel değerlerden biri verildiğinde simge otomatik olarak oluşturulur. Bu değerleri ve oluşturulacak simgeleri incelemek için <https://api.flutter.dev/flutter/material/Icons-class.html> adresini ziyaret ediniz.



```
import 'package:flutter/material.dart';
class Cekmece extends StatelessWidget {
  const Cekmece({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        children: [
          UserAccountsDrawerHeader(
            accountName: Text("@odonmez"),
            accountEmail: Text("onur.donmez@ege.edu.tr"),
            currentAccountPicture: CircleAvatar(
              backgroundImage: AssetImage('img/account.jpg'),
            ),
          ),
          ListTile(
            title: Text(
              "Anasayfa",
              style: Theme.of(context).textTheme.caption,
            ),
            leading: Icon(Icons.home),
            onTap: () => Navigator.pushNamed(context, '/'),
          ),
          ListTile(
            title: Text(
              "Koruyucu Diş Hekimliği Uygulamaları",
              style: Theme.of(context).textTheme.caption,
            ),
            trailing: Icon(Icons.arrow_forward_ios),
            onTap: () => Navigator.pushNamed(context, '/koruyucu'),
          ),
          ListTile(
            title: Text(
              "Süt ve Daimi Diş Restorasyonları",
              style: Theme.of(context).textTheme.caption,
            ),
            trailing: Icon(Icons.arrow_forward_ios),
            onTap: () => Navigator.pushNamed(context, '/restorasyon'),
          ),
          ListTile(
            title: Text(
              "Koruyucu Ortodontik Uygulamalar",
              style: Theme.of(context).textTheme.caption,
            ),
            trailing: Icon(Icons.arrow_forward_ios),
            onTap: () => Navigator.pushNamed(context, '/ortodonti'),
          ),
          ListTile(
            title: Text(
              "Süt ve Daimi Dişlerde Vital Pulpa Tedavileri",
              style: Theme.of(context).textTheme.caption,
            ),
            trailing: Icon(Icons.arrow_forward_ios),
            onTap: () => Navigator.pushNamed(context, '/pulpa'),
          ),
        ],
      );
    );
  }
}
```

pubspec.yaml Dosyası

pubspec.yaml dosyası Flutter projelerine görseller, fontlar ya da ek kütüphaneler gibi dışarıdan eklenecek içeriğin tanıtılması için kullanılır. YAML formatında düzenlenen bu dosya güncellendiğinde projen çapında etkili olur. Bu örnekte kullanılan *Image.asset()* metodu, projeye varlık olarak eklenen görsellerin ekranda gösterilmesini sağlar. *Image.asset()* metodu yalnızca *pubspec.yaml* dosyasında tanımlı görselleri kullanabilir.



dikkat

pubspec.yaml dosyasında proje ile ilgili pek çok ayar bulunmaktadır. Kod 40'ta yalnızca görsellerin eklediği bölüm alınmıştır.

assets:

- img/account.jpg
- img/braces.png
- img/dentist-tools.png
- img/dentistry.png
- img/caries.png
- img/tooth.png

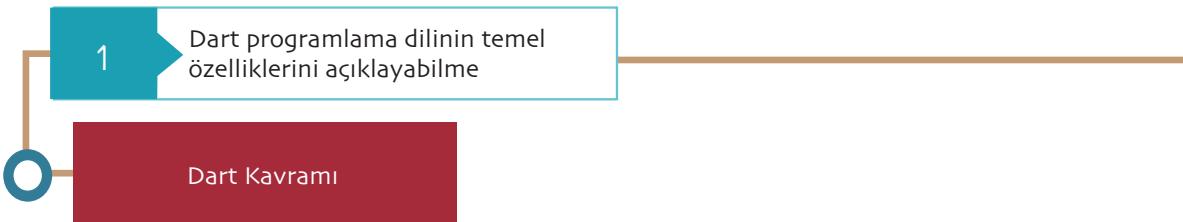
Kod 40 *pubspec.yaml* dosyası ile projeye görsel eklemek

Kod 40'ta projeye 6 adet görsel eklenmektedir. Burada belirtilen yollar proje kökünde yer alan *pubspec.yaml* dosyasına göre verilmektedir. Bu durumda görseller proje kökünde yer alan *img* klasöründe bulunmaktadır. Görsellerin projede gösterilmesi için buradaki yolun aynıının kullanılması gereklidir (ör: *Image.asset('img/account.jpg')*).

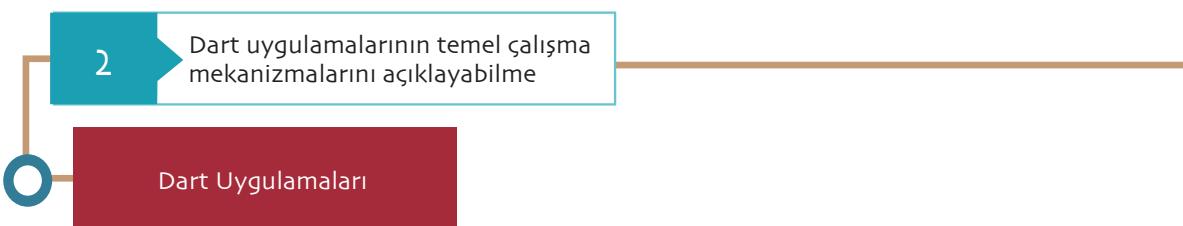
Öğrenme Çıktısı



Mobil Uygulama Geliştirme



Dart, sunucular, mobil, web tarayıcılar gibi farklı ortamlarda çalışabilen arayüzler üretmek için kullanılan tanımlama temelli (declarative) bir dildir. Dart nesne yönelimli programlama yaklaşımını esas alır. Dart dilinde kullanılan veri tipleri sıkı kontrol altındadır. Ayrıca Dart değişkenlere atanın değerlere göre kullanılacak veri tipine karar verebilir. Bunun yanında null güvenliği mekanizması ile uygulamalardaki basit hataların önüne geçilir. Son olarak sık sık söyleme mekanizması ile kodlarda yapılan değişiklikler anında ekranada görüntülenebilir.



Dart uygulamalarının başlangıç noktası main fonksiyonudur. Uygulamala bu noktadan başlayarak yazılan kodlara göre çeşitli kütüphaneler, sınıflar ve fonksiyonlara dallanabilmektedir. Dart çekirdek kütüphanesi yaygın kullanımındaki pek çok uygulamayı destekler. Bunun yanında matematiksel işlemler ya da asenkron işlemler için gerekli yetenekleri sağlayan kütüphaneler projelere eklenebilir. Dart'ın sıkı veri kontrolü nedeniyle fonksiyonlar da dahil olmak üzere her nesnenin veri tipinin belirtilmesi gereklidir. Fakat Dart tip çıkarımı mekanizması ile bu işlevi kendisi yürütebilir. Dart ile işlem yapabilmek için pek çok işlev kullanılır. Bu işlevlerin tam listesine erişmek için <https://dart.dev/guides/language/language-tour#operators> adresi ziyaret edilebilir.



Dart uygulamalarının gerçekleştirilebilmesi için basit ve birleşik veri yapıları sunar. Basit veri tipleri arasında sayılar (int, double), metinler (String) ve mantıksal veri tipi (bool) sayılabilir. Basit veri yapılarının birleştirildiği listeler (List) ve haritalar (Map) gibi veri yapıları da desteklenmektedir. Bunun yanında, sınıflar kullanılarak birbirleri ile ilişkili verilerden oluşan kompakt veri sistemleri oluşturulabilir.

Dart dilinde tanımlı veri tipi üzerinde çalışabilecek özelleşmiş metodlar sunar. Örneğin bir liste içinde bir verinin bulunup bulunmadığını anlamak için indexWhere metodunu kullanılabılır. Bunun yanında listeler ya da haritalara veri eklemek ve çıkarmak için add ve removeAt gibi metodlar sağlanmaktadır. Basit veri tipleri üzerinde işlem yapmak için çoğunlukla işlevler kullanılır. Örneğin bool tipindeki bir verinin tersini almak için ! işlevi kullanılır.

4

Dart programlama dilinde akış yöneticilerini ve döngü yapılarını kullanabilme

Akış Yöneticiler

Dart if-else ve switch gibi temel akış yöneticilerini destekler. if-else yapıları verilerin bir şart ifadesinin sağlanması ve sağlanmaması durumunda çalışacak kod bloklarını tanımlama şansı verir. Bunun yanında if-else yapıları zincirler halinde kullanılarak art arda şartların sağlanması / sağlanması durumunda çalışacak blok yapıları tanımlanabilir. Switch blokları ise bir değişkenin farklı değerlerine göre işletilecek bloklar tanımlama olağrı sağlar.

Belirli şartlar altında tekrar tekrar çalıştırılması istenen kodlar döngüler içinde kullanılabilir. En temel döngü yapıları for ve while döngüleridir. For döngülerinde, kontrol değişkenleri tanımlama satırında tanımlanır ve kontrol edilir. While döngülerinde ise böyle bir yapı yoktur. While döngülerinde kontrol değişkeni tanımlama bloğu öncesi tanımlanabilir. Döngü bloğu içinde değişkene müdahale edilmezse sonsuz döngüler üretilebileceği unutulmamalıdır. Bu temel döngülerin yanında listeler üzerinde çalışabilen forEach metodları da döngüler gibi düşünülebilir.

5

Dart programlama dilinde nesneye yönelik programlama kullanabilme

Dart ile Nesneye Yönelik Programlama

Sıklıkla kullanılan kod parçaları fonksiyonlar içinde paketlenebilir. Bu paketler fonksiyon isimleri yoluyla çağrılarak kodun istenen yerinden çağrılabılır. Fonksiyonlara veri göndermek için parametreler kullanılır. Parametreler pozisyonel ya da isimli olarak tanımlanabilir. Bunun yanında fonksiyonlar üzerindeki etki alanında tanımlı değişkenlere de erişebilmektedir. Fonksiyonlar geriye değer de çevirebilir. Geriye değer çevirmesi beklenen fonksiyonlar bu değerin veri tipi ile tanımlanır. Bir fonksiyon geriye değer çevirmeyecekle void anahtar sözcüğü ile tanımlanır.

Nesneler birbirleri ile ilişkili veri yapıları ve kodlardan oluşan paketlerdir. Nesneler, taslak olarak isimlendirebileceğimiz sınıflardan üretilir. Nesnelerin temelini oluşturan sınıflarda, nesnenin kullanılacağı veri tipleri (özellikler) ve bu veriler üzerinde işlem yapacak metodlar (fonksiyonlar) tanımlanır. Nesneleri üretmek için sınıfların yapıcı metodları (constructor) çalıştırılır. Dart bir sınıfta birden fazla yapıcı metod tanımlanmasına izin verir. Bunun yanında Dart sınıflarda kalıtım mekanizmasının kullanılmasına izin verir. Kalıtım mekanizmasını kullanmak için extends anahtar sözcüğü kullanılır. Kalıtım mekanizması ile üst sınıfta (super) tanımlanmış özellikler ve metodlar miras alınmış olur.

6

Çok sayfalı bir Flutter uygulaması oluşturabilme

Çok Sayfalı Uygulamalar

Flutter ile birden fazla sayfadan oluşan uygulamalar üretilebilir. Her bir sayfada uygulamanın özelleşmiş bir işlevi (ör: fotoğraf çekme, ayarları düzenlemeye, akışı izleme) gerçekleştirilebilir. Flutter uygulamalarının kullanıcı arayüzleri bileşenler (Widget) ile oluşturulur. Flutter uygulamalarının kendisi de olmak üzere ekranda gösterilen her şey bir bileşendir. Flutter uygulamaları bir bileşen ağaç şeklinde şematize edilebilir. Bu bileşen ağaçındaki nesnelerin ekranda düzenli bir şekilde gösterilmesi için slotlar sağlayan Scaffold bileşeni kullanılır. Bu bileşenin sağladığı parametrelere (ör: body, drawer, appBar) uygun bileşenler üretileerek kullanıcı arayüzü kurgulanır.

Flutter uygulamalarındaki sayfalara rota adı verilir. Navigator nesnesi sayfada tanımlı rotaların yığıldığı bir yapı sağlar. Her bir rota Navigator nesnesine eklenir ya da nesne üzerindeki yığından çıkarılır. Bu sayede rotalar arası gezinim sağlanmış olur.

1 Dart dilinde tanımlı değişkenlerin içeriklerinin boş olmasını engelleyen mekanizma aşağıdakilerden hangisidir?

- A. Null-safety
- B. Hot reload
- C. Kalıtım
- D. Parametre
- E. Void

2 Dart uygulamalarının başlangıç noktası aşağıdakilerden hangisidir?

- A. Void fonksiyonu
- B. Main fonksiyonu
- C. runApp metodu
- D. MaterialApp nesnesi
- E. Extends anahtar kelimesi

3 Dart uygulamalarına dışarıdan kod eklemek için aşağıdaki hangi anahtar sözcük kullanılır?

- A. Import
- B. Extend
- C. Void
- D. Material.dart
- E. Get

4 Flutter uygulamalarında kullanılacak temel veri yapılarını sağlayan kütüphane aşağıdakilerden hangisidir?

- A. Cupertino.dart
- B. Core.dart
- C. Math.dart
- D. Material.dart
- E. Pubspec.dart

5 Aşağıdaki değişken tanımlarından hangisi hatalıdır?

- A. Var sayı = 3;
- B. İnt sayı = 3;
- C. Var sayı = 3.6;
- D. Var sayı = “üç”;
- E. İnt sayı = 3.6;

6 Sayısal bir değişkenin işaretini değiştirmek için aşağıdaki işlemlerden hangisi kullanılır?

- A. -
- B. +
- C. ^
- D. ~
- E. !

7 Aşağıdakilerden hangisi sonsuz döngü üretmez?

- A. While(true)
- B. While(2=2)
- C. For(var i=0; i<2; i--)
- D. While(false)
- E. While(2>=2)

8 *Hesapla(int say1, say2) => say1 % say2;*

Yukarıda tanımlanan hesapla fonksiyonu hesapla(6, 4) şeklinde çağrılırsa nasıl davranıştır?

- A. Ekrana 2 yazar
- B. Geriye 2 çevirir
- C. Ekrana 1,5 yazar
- D. Ekrana 24 yazar
- E. Geriye 24 çevirir

9 Dart programlama dilinde kalıtım mekanizmasını çalıştırmak için aşağıdakilerdeki anahtar sözcüklerden hangisi kullanılır?

- A. Extends
- B. Coid
- C. Null
- D. Private
- E. Public

10 Flutter uygulamalarında ekranda içerik göstermek için kullanılan yapılara ne ad verilir?

- A. Bileşen (Widget)
- B. Yapıçı (Constructor)
- C. Döngü
- D. Değişken
- E. Rota

1. A	Yanıtınız yanlış ise “Dart Kavramı” konusunu yeniden gözden geçiriniz.	6. A	Yanıtınız yanlış ise “Aritmatik İşleçler” konusunu yeniden gözden geçiriniz.
2. B	Yanıtınız yanlış ise “Dart Uygulamaları” konusunu yeniden gözden geçiriniz.	7. D	Yanıtınız yanlış ise “While Döngüleri” konusunu yeniden gözden geçiriniz.
3. A	Yanıtınız yanlış ise “Dart Uygulamaları” konusunu yeniden gözden geçiriniz.	8. B	Yanıtınız yanlış ise “Fonksiyonlar” konusunu yeniden gözden geçiriniz.
4. D	Yanıtınız yanlış ise “Başlangıç Sayfası” konusunu yeniden gözden geçiriniz.	9. A	Yanıtınız yanlış ise “Dart ile Nesneye Yönelik Programlama” konusunu yeniden gözden geçiriniz.
5. E	Yanıtınız yanlış ise “Sayılar” konusunu yeniden gözden geçiriniz.	10. A	Yanıtınız yanlış ise “Çok Sayfalı Uygulamalar” konusunu yeniden gözden geçiriniz.

5

Araştır Yanıt
Anahtarı

Araştır 1

Null güvenliği mekanizmasının amacı tanımlanmış fakat henüz değer ataması yapılmamış değişkenlerin kullanılmasından kaynaklanan hataların önlenmesidir. Tanımlanmış fakat değer atanmamış String bir değişkenin değeri null'dur. null değeri Null sınıfının bir üyesi olduğundan length özelliğini desteklemez. Fakat programcı String tipinde tanımladığı değişkenin length özelliğinin olmasını beklediğinden bu özelliği kullanmaktadır. Bu tür durumlarda uygulama çalışma zamanı hataları üretir. Çalışma zamanı hataları kullanıcıların cihazlarında üretildiğinden burlar hakkında veri toplamak güçtür. Bunun yanında bu hatalar uygulamaya yönelik güveni de düşürmektedir. Null güvenliği mekanizması değişkenlerin değerlerinin null kalmasına izin vermez. Böylece çalışma zamanı hataları azaltılır. Dart dışında, Kotlin, Rust ve Swift dilleri null güvenliği desteği sunmaktadır.

Araştır 2

Dart web uygulamalarının geliştirilmesi için pek çok kütüphane sağlar. HTML etiketlerinin yorumlanması için dart:html, web tarayıcıdaki veritabanı işlemleri için dart:indexed_db, javascript kodlarının yorumlanması için js paketi, SVG dosyalarının gösterilmesi için dart:svg kütüphanesi, web seslerinin işlenmesi için dart:web_audio ve tarayıcıda 3D programlama özelliklerinin kullanılması için dart:web_gl kütüphaneleri sunmaktadır.

5

Araştır Yanıtları Anahtarı

Araştır 3

Tıpkı forEach metodunda olduğu gibi map metodu da parametre olarak bir anonim fonksiyon alır. forEach metodunda olduğu gibi map metodu da listedeki her bir element üzerinde işlem yapabilir. map metodunun en önemli farkı elementler üzerinde gerçekleştirdiği işlem sonucunda geriye liste çevirmesidir. Aşağıdaki kodda tanımlanan şehirler listesi üzerinde map metodu kullanılmıştır. Bu metod içindeki anonim fonksiyonda her elementin bir Text bileşeni oluşturulması sağlanmıştır. map metodu arayüzde metinlerin gösterilmesi için kullanılabilen üç adet Text bileşeni oluşturarak bunları etiketler listesine aktarmaktadır.

```
var sehirler = ["ankara", "izmir", "istanbul"];
var etiketler = sehirler.map((e) => Text(e));
print(etiketler); // (Text("ankara"), Text("izmir"), Text("istanbul"))
```

Araştır 4

Unix zamanı 1 Ocak 1970 tarihinden itibaren geçen saniye sayısını ile ifade edilir. Dart dilinde zaman işlemleri için DateTime sınıfının metodları kullanılır. 1 Ocak 1970'ten beri geçen zamanı milisaniyeler cinsinden almak için DateTime.now().millisecondsSinceEpoch ifadesi kullanılabilir.

Araştır 5

Dart public ve private gibi erişim denetleyicilerini kullanmaz. Yine de bir kütüphane dışından erişilmesi istenmeyen değişkenler ve fonksiyonların önüne _ işaretçisi eklenerek (ör: _sayı, _alanHesapla()) erişim kısıtlaması uygulanabilir.

Araştır 6

Navigator nesnesi Flutter uygulamalarında rotaların üst üste yığıldığı bir listedir. Bu listenin pop() ve add() metodları kullanılarak ekrana rota eklenebilir ya da ekrandaki rotalar kaldırılabilir.

Internet Kaynakları

Google. (t.y.). Dart Language Tour. <https://dart.dev/guides/language/language-tour> adresinden 04.04.2022 tarihinde erişilmiştir.

Bölüm 6

Mobil Uygulamalarda Veri Tabanı Uygulamaları

Öğrenme Çıktıları

Asenkron (Async) İşlemler, Future ve Await

- Flutter'da asenkron işlemleri yapabilme, future, async, await anahtar kelimelerini doğru bir şekilde kullanabilme

Flutter ile Sqlite İşlemleri

- Flutter'da Sqlite veri tabanı oluşturabilme, Select, Insert, Update, Delete işlemlerini gerçekleştirebilme ve bu işlemlerde modelleri kullanabilme

Flutter ile API İşlemleri

- API'ye bağlanabilme ve API ile veri ekleme, getirme, güncelleme ve silme işlemlerini gerçekleştirebilme

3

Anahtar Sözcükler: • Flutter • Sqlite • API



GİRİŞ

Programlamanın amaçlarından biri veri işlemedir. Veri işlemede, veriyi alma/yazma, anlamlı bir şekilde saklama, getirme, raporlama, güncelleme, silme işlemleri önem arz eder. Bu işlemler gerçekleştirilirken verileri saklamak için farklı araçlar kullanılabilir. En yaygın kullanılan araçlar veri tabanlarıdır. Veri tabanı olarak birçok şirket farklı veri tabanı sistemleri geliştirmiştir. Mobil uygulamalarda en sık kullanılan veritabanı Sqlite'tir. Bu ünitede Sqlite ile ilgili veri getirme, yazma, silme ve güncelleme işlemlerinin nasıl gerçekleşeceğini göreceğiz.

Programlar arası veri alışverişi için kullanılan teknolojilerden biri de API(Application Programming Interface) teknolojisidir. İnternet üzerinden veri çekme ve gönderme işlemlerinin yapılması için geliştirilen API teknolojisi GET ve POST işlemleri yaparak karşı taraftan veri çekme ve veri gönderme işlemlerine imkan verir. Bu bölümde SQLite işlemlerinden sonra API'ler ile de veri alma ve gönderme işlemlerinin nasıl yapılacağını göreceğiz.

Hem SQLite işlemleri hem de API işlemlerinin yapılması kullanılan cihazın donanımsal kapasitesi, SQLite için veritabanını büyülüğu, çalıştırılacak sorgunun karmaşıklığı, çağrılabilecek verinin boyutu; API kullanılyorsa internet hızı, karşısındaki serverin cevap verme hızı, çağrılabilecek verinin boyutuna bağlı olarak zaman alabilemektedir. İşlemenin yapılması için zamanın uzaması demek zaman zaman donmalara, kasımlaraya, dahi hatalara neden olabilmektedir. Bu yüzden asenkron programlama tekniğinin kullanılması gerekmektedir. Bu bölümde asenkron işlemler ile başlanacaktır.

ASENKRON (ASYNC) İŞLEMLER, FUTURE VE AWAIT

Dart dilinde kod satırları sıralı olarak çalışır ve bir satır çalışıp tamamlandıktan sonra bir diğer satır geçer. Fakat bazen o satırda gerçekleşen eylem uzun süre alabilir. Örneğin internetten veri çekmek, veritabanından veri çekmek/yazmak veya yüksek boyutlu bir dosya okumak gibi. Böyle bir durumda süre uzayacağı için beklemeden bir sonraki satır geçer ve işlem tamamlanmadan diğer satırlar işlemeye başlar. Dart programlama dilinde de bu gibi uzun süre alacağı düşünülen işlemler asenkron olarak gerçekleştiriliyor ve await anahtar kelimesi ile o işlemin tamamlanmasının beklenmesi sağlanabilir.

Örneğin aşağıdaki kod bloğunda sıra değişkenine 018 değeri atanacaktır. Fakat Future.delayed ile bu atama bilerek 2 saniye geciktirilmiştir (Bu işlemin internetten veri getirme, veritabanından sorgu çekme, yahut bir dosya getirme işlemi gibi uzun süre alacak bir işlem olduğunu düşünebilirsiniz).



```

1. void sıraAl() {
2.     print('Kullanıcı için sıra alınıyor...');
3.     var _sira = Future.delayed((seconds: 2), () => '018');
4.     print('Sıra numaranız: $_sira');
5. }
```

Bu işlem gecittiği için program 2 sn beklemeden 4. satır atlayacaktır ve Console ekranına:

```
I/flutter (15106): Kullanıcı için sıra alınıyor...
I/flutter (15106): Sıra numaranız: Instance of 'Future<String>'
```

İfadeleri yazılacaktır. Burada Instance of 'Future<String>' ifadesi sıra değişkenine henüz atama yapılmadığını gösteriyor.

Burada siraAl metodunun asenkron bir metot olduğu ifade edilirse ve 3. satırda _sira değişkenine atama işleminin sonlamadan diğer bir satır geçilemeyeceği belirtilirse problem ortadan kalkacaktır. Bunun için 3. satırda işlem beklenecesinden başına **await** anahtar kelimesinin yazılması gereklidir. Bu yüzden ilgili satır şu şekilde güncellenir:

```
3. var sira = await Future.delayed(...)
```

await anahtar kelimesi o işlemin bekleneceği anlamına gelir. Fakat eğer bir metotta **await** anahtar kelimesi kullanılarak belirli işlem(ler)in sonucunun beklenmesi gerekiyorsa içinde olduğu metotun da asenkron bir metot olduğunu belirtmesi gereklidir. Bunun içinde aşağıda belirtildiği gibi metodun tanımlandığı satırda parametre için kullanılan parantez kapatıldıkten sonra **async** ifadesi konulur.

```
1. void siraAl() async {
2.   print('Kullanıcı için sıra alınıyor...');
3.   var sira = await Future.delayed(Duration(seconds: 2), () => '018');
4.   print('Sıra numaranız: $sira');
5. }
```

Async ifadesi ile birlikte siraAl() metodu asenkron bir şekilde çalışacak, dolayısıyla içerisinde **await** anahtar kelimesi kullanılan yerlerde cevabın gelmesini bekleyecek ve cevap geldikten sonra işleme devam edecektir.

Future burada devreye girmektedir. Future asenkron çalışmaların sonucunu temsil eder. Eğer bir metot içerisinde **await** ile bir işlem beklenenekse bu metot Future ile tanımlanır ve böylece metot işlemlerinin tamamlanmasından sonra yapılacak işlemlerde belirlenebilir. Şöyle ki yukarıda siraAl metodu void olarak tanımlanmıştır. siraAl metodundan sonra başka metot/lar çağrılması gerekebilir. Eğer siraAl asenkron olarak çalışacaksa metodun bitmesi beklenmeden diğer metoda geçilecektir.

```
void calistir() {
  siraAl();
  kaydet();
}
```

Yukarıda siraAl() metodunun çalışması bitmeden kaydet metodu çalışmaya başlayacak, belki de sonlanacaktır. Eğer kaydet metodu siraAl metodunun sonucuna bağlı ise yani o bitmeden çalışmaması gerekiyorsa siraAl metodu Future olarak tanımlanmalıdır ve kaydet metodu then ifadesi ile çağrılmalıdır.

```
siraAl().then((value) => kaydet());
```

Yukarıdaki kod satırında then ifadesi ile kaydet metodunun siraAl metodu bittiğinden sonra çalışacağı ifade ediliyor. Yukarıda bahsedildiği gibi then'in kullanılması için siraAl Future olarak tanımlanmalıdır.

```
Future<void> siraAl() async { ... }
```

Future ifadesi kullanıldığındaysa yukarıda olduğu gibi **then**, işlemin bitmesinden sonra kullanılabilir. Bunun yanı sıra whenComplete hata olsun olmasın işlemin bitmesi durumunda, onError, catchError hata ile karşılaşılması durumunda, timeout zaman aşımı olması durumunda kullanılabilir.



FLUTTER İLE SQLITE İŞLEMLERİ

Mobil uygulamalarda veri saklama için kullanılan araçlardan biri SQLite veritabanıdır. Burada iki seçenek kullanılabilir. Birinci olarak siz Flutter uygulaması içerisinde bir veri tabanı oluşturup onun üzerinden işlemler yapabilirsiniz. İkinci olarak Flutter'dan bağımsız olarak bir SQLite veritabanı oluşturup projenize dahil edebilirsiniz. Eğer kullanacağınız veritabanı ön verileri çok büyük veriler değilse Flutter içerisinde uygulamanın ilk açılması ile beraber oluşturabilirsiniz. Şayet kullanacağınız veritabanı içerisinde bir çok farklı veri olacaksa önceden hazırlayıp projenize o şekilde dahil edebilirsiniz.

Sqlite Kurulumu

Sqlite herhangi bir sunucuya ihtiyaç olmadan, platformdan bağımsız olarak çalışan, özel bir konfigürasyon gerektirmeyen özellikle mobil uygulamalar için popüler olan, C tabanlı bir veritabanıdır. Bireysel veya ticari her türlü geliştirmeler için ücretsizdir.



Diğer veritabanlarında olduğu gibi SQL dilini destekler ve sütun türlerinde tanımlamalar da farklılık gösterebilir. Sqlite

- Ücretsiz,
- Açık kaynaklı,
- Konfigürasyon gerektirmeyen kullanımı kolay bir veritabanıdır.

Flutter'da SQLite kullanmak için projeye kütüphane eklemek gerekmektedir. <https://pub.dev/> adresinden SQLite kütüphaneleri arattığınızda bir çok farklı kütüphane gelmektedir. Burada popülerliğe baktığınızda sqflite kütüphanesi en popüler kütüphane olduğu için bu bölümde de bu kütüphane üzerinden anlatılacaktır. Bu kütüphaneye <https://pub.dev/packages/sqflite> adresinden ulaşabilirsiniz (Ulaşma tarihi: 10.01.2022).

Kütüphaneyi projenize dahil etmek için terminalden

```
flutter pub add sqflite
```

komutunu çalıştırabilir ya da pubspec.yaml dosyanızda dependencies: altına

```
sqflite: ^2.0.1
```

ifadesini ekleyebilirsiniz. (Bu satırların yazıldığı gün sqflite'in son sürümü ^2.0.1'dir. Sizin bu bölümü okuduğunuz tarihte daha ileri sürümleri olabilir. 10.01.2022) Bu ayarlamadan sonra projenizde SQLite'ı istediğiniz yerde kullanabilmek için sayfanın başında kütüphaneyi şu şekilde import etmeniz gerekmektedir.

```
import 'package:sqflite/sqflite.dart';
```

Artık veri tabanı işlemlerine başlayabiliriz. Yukarıdaki işlemleri veri tabanı oluşturma videosunda uygulamalı olarak izleyebilirsiniz.

Veri Tabanı Oluşturma

İlk olarak sqflite kütüphanesini import ettikten sonra, veri tabanını oluşturmak için asenkron olarak yeni bir metot tanımlanır ve bu metot içerisinde Database türünde bir nesne tanımlanır. Bu nesne openDatabase metodu ile aşağıdaki şekilde gibi tanımlanır.

```
Future<void> dbOlustur() async {
    Future<Database> db = openDatabase("dbAOF");
}
```

Bu satırın bir kez çalıştırılması ile veritabanı oluşturulmuş olur. Veri tabanı oluşturulduktan sonra bu kod ile artık veritabanına erişim sağlanabilir. Kodun ilk çalıştırılmasından veri tabanı oluşturulurken eğer tablolarında oluşturulması isteniyorsa openDatabase altında onCreate metodu çağrılır.

```
Future<Database> db = openDatabase(
    "dbAOF",
    onCreate: (db, version) {
        print("oncreate çalıştı");
    },
    onUpgrade: (db, oldVersion, newVersion) {
        print("upgrade çalıştı");
    },
    onDowngrade: (db, oldVersion, newVersion) {
        print("downgrade çalıştı");
    },
);
```

Burada veritabanı ilk oluşturulduğunda onCreate metodu çalışır ve ilk çalıştığında gerekli bütün tablolar db.Execute ile oluşturulabilir. Hatta gerekli ise bu kısımda Insert ile tablolara veriler de eklenebilir. Eğer veri tabanı versiyonu yükseltilirse onUpgrade eğer versiyon küçültülürse onDowngrade çalışacaktır.



Tablo Oluşturma

Tablo oluşturmak için SQL ile tablo oluşturma kodlarını kullanılır. “Create table” ile başlayıp oluşturulacak tablo adı yazılır ve ardından alanlar belirlenir.

```
CREATE TABLE TabloAdi (AlanAdi AlanTuru, AlanAdi AlanTuru, AlanAdi AlanTuru)
```

Örneğin bir kullanıcı tablosu oluşturalım.

```
db.execute("CREATE TABLE Kullanici (kullaniciID INTEGER PRIMARY KEY, adi TEXT, oyadi INTEGER);")
```

Tablolar genel olarak veri tabanı oluşturulduğu zaman onCreate metodu içerisinde oluşturulur. Eğer daha sonra herhangi bir tablo eklenecek, silinecek veya güncellenecekse veri tabanının execute metodu yine kullanılarak bu işlem yapılabilir.

Yukarıda verilen sorguyu PRIMARY KEY ifadesi o alanın birincil anahtar olduğunu ifade eder ve her yeni kayıt için bir artarak otomatik olarak eklenir. Veritabanında her tabloda birincil anahtar kullanılması önemlidir. Bu yüzden oluşturacağınız her tablo için tabloadi+Id olarak birincil anahtar tanımlamanız tavsiye edilir.

SQLite Veri Tipleri

Sqlite diğer veri tabanlarına göre daha az veri tipine sahiptir. Bunlar:

INTEGER	: Sadece tam sayıları
REAL	: Ondalıklı sayılar
TEXT	: Metin ifadeleri
BLOB	: Binary

Boolean olarak SQLite herhangi bir veri tipine sahip değildir. True/False (1/0) şeklinde kayıtlar için INTEGER kullanılabilir. Tarih/Saat işlemleri içinde özel bir veri tipi yoktur. Bunlar için Text, Real ve Integer veri tipleri kullanılabilir. Fakat genel olarak tarih işlemleri için Text kullanılır ve yyyy-MM-dd HH:mm:ss formatında kaydedilir. Sqlite veri tipleri ile ilgili detaylı bilgiyi <https://www.sqlite.org/datatype3.html> adresinden alabilirsiniz.

Kayıt Ekleme (Insert)

Insert veritabanına yeni kayıt eklemek için kullanılır. Insert ardından tablo adı yazılır ve parantez içinde alan isimleri yazılır. Daha sonra values ile alanlara gelecek değerler yazılır. Örneğin



```
Insert Into TabloAdi (alan1,alan2,alan3,...) values (deger1,deger2,deger3,...)
```

Peki bu işlem Flutter ile nasıl yapılır? Flutter'da veri tabanı işlemleri farklı yollarla yapılmaktadır. Hem SQL komutu ile yapılabildiği gibi hem de Flutter'ın hazır metotları ile ekleme işlemlerini yapabilirsiniz.

```
Future<void> kayitEkle() async {
    Database db = await openDatabase("dbAOF");
    db.rawQuery("insert into Kullanici (Adi,Soyadi) values
('Alp','Onur')");
    db.close();
}
```

Yukarıdaki kodda kayitEkle isimli metot ile yeni bir kayıt eklenmektedir. Bu işlem için önce database nesnesi oluşturulmaktadır ve daha önce oluşturulan dbAOF veritabanı bağlanmaktadır. Daha sonra db.rawQuery ile insert SQL cümlesi çalıştırılmaktadır. Bu işlem sonucu yeni kayıt eklenmiş olur. Burada eklenecek veriler parametre olarak da gönderilebilir. Parametre olarak gönderilirken değerlere soru işaretü (?) ve SQL cümlesi bittikten sonra virgülden sonra parametreler bir köşeli parantezler [] içinde dizi şeklinde gönderilebilir.

```
db.rawQuery("insert into Kullanici (Adi,Soyadi) values (?,?)",
["Alp", "Onur"]);
```

rawQuery metodu genel SQL çalışma metodudur. Bunun yerine insert yaparken rawInsert kullanılır. Burada yapacağınız rawQuery yerine rawInsert yazmak olacaktır. rawInsert size eklenen kaydın birincil anahtar değerini verir.

```
int KullaniciId = await db.rawQuery("insert into Kullanici...
```

Yukarıdaki satırda KullaniciId isimli değişken Kullanici tablosuna yeni kayıt eklendikten otomatik oluşturulan KullaniciId'ye eşitlenir.

insert yapmak için SQLite kütüphanesinin insert metodu vardır. Bu metotta tablo ismi ve kaydedilecek değerleri Map ile almaktadır. Burada Map klasik json formatında verilir. Örneğin

```
db.insert("Kullanici", {'adi': 'ata','soyadi': 'eren'});
```

Burada mapping olayını dışında yapıp bir nesne olarak da insert'e gönderebilirsiniz.

```
var Kullanici = {'adi': 'ata', 'soyadi': 'eren'};
db.insert("Kullanici", Kullanici);
```

Insert işleminin bir çok farklı şekilde yapıldığını gördünüz. Bir de model kullanarak yapılan insert işlemleri var ki bölümün ilerleyen kısmında görebilirsiniz.

Kayıt Getirme (Select)

Select ile veritabanından veri getirme işlemleri için kullanılır. Burada (1) veritabanından getirme işlemi, (2) gelen verinin gösterilme işlemi vardır. Veritabanından veri getirilirken yine SQL dili kullanılarak veriler getirilir.



```
Select * from TabloAdı where Şart
```

Flutter kısmında ise async bir metot tanımlanarak aşağıdaki gibi veriler getirilir.

```
Future<void> kayıtGetir() async {
  Database db = await openDatabase("dbAOF");
  List lKullanici = await db.rawQuery("select * from Kullanici");
  print(lKullanici);
  db.close();
}
```

Yukarıda kayıtGetir metodunda List nesnesine db.rawQuery ile erişilmek istenen soru sonucu doldurulur. List nesnesi json formatında kayıtları getirir. Burada console'a veriler şu şekilde yazılacaktır.

```
[{kullaniciId: 1, adı: Alp, soyadı: Onur}, {kullaniciId: 2, adı: veli, soyadı: ali}, {kullaniciId: 3, adı: ata, soyadı: eren}]
```

Yukarıda bahsedilen kod ile değerler Console ekranına yazdırıldı. Şimdi bu değerleri ekranда nasıl göstereceğimize gelelim. Burada örnek olarak tablodan bir soru yapalım ve o soruya gelen değerleri ekranда bulunan Text'lere yazalım.

```

1. String _adi = "", _soyadi = "";
2. Future<void> kayitGetir() async {
3.     Database db = await openDatabase("dbAOF");
4.
5.     List lKullanici =
6.         await db.rawQuery("select * from Kullanici where
KullaniciId=5");
7.     setState(() {
8.         _adi = lKullanici.elementAt(0)["adi"].toString();
9.         _soyadi = lKullanici.elementAt(0)["soyadi"].toString();
10.    });
11. }

12. body: Column(
13.     children: [
14.         TextButton(onPressed: kayitGetir, child: const Text("Kayıt
Getir")),
15.         Text(_adi),
16.         Text(_soyadi)
17.     ],
18. ),

```

Yukarıda getirme ve ekranda gösterme işlemleri yapılır. Bunun için ekranada göstereceğimiz her bir alan için bir değişken tanımlandı (_adi,_soyadi) (s.1). Bu değişkenler body içerisinde Text'lerle ilişkilendirildi (s. 15,16). "Kayıt Getir" butonuna (s. 14) basıldığında veritabanına bağlanarak rawquery ile istenilen sorgu yazıldı (s. 6). Yukarıda KullaniciId'si 5 olan kayıt getirilir. Bu kayıt List nesnesi olan lKullanici listesine atandı. 8. ve 9. satırlarda adı, soyadı değişkenlerine lKullanici listesinde elemenAt ile 0. satırın adı ve soyadı alanları getirildi. Burada işlemler setState içerisinde yapılır. setState kullanmadığınızda atamaları yapılır ve ekrana yansımaz. setState burada yapılan işlemlerin kullanıcı arayüzüne (ekrana) yansımıası için kullanılmaktadır. Burada eğer sorgu birden fazla kayıt döndürüyorsa ve 3. satırdaki veriyi çekeceksiniz elementAt(2) olarak çağrılabilsiniz.

Kayıt getirme işlemleri için Sqflite kütüphanesinin bir başka metodu olan query metodu da kullanılabilir.

```
List lKullanici = await db.query("Kullanici", columns: ["Adi",
"Soyadi"]);
```

query metodu parametre olarak String table, { bool? distinct, List<String>? columns, String? where, List<Object?>? whereArgs, String? groupBy, String? having, String? orderBy, int? limit, int? offset} değerlerini gönderebilirsınız.

```
List l = await db.query("Kullanici",
    columns: ["Adi", "Soyadi"], where: "KullaniciId=?",
    whereArgs: [3]);
```

Örneğin yukarıdaki kod satırında Kullanici tablosundan Adı ve Soyadı alanları gelecek ve şart olarak KullaniciId = 3 olan satır(lar) gelecektir. Eğer sadece tablo adı gönderilirse metoda o tablodaki bütün kayıtlar tüm sütunlar ile birlikte gelecektir.

ListView

Mobil uygulamalarda tek kayıtın yanı sıra birçok yerde çoklu kayıt getirilmesi gerekebilir. Bunun için farklı widgetlar kullanılır. Bu widgetlardan birisi de ListView'dır. ListView kayıtların liste şeklinde istenildiği formatta görüntülemeye izin veren, kaydırma olayını destekleyen bir Widgettir.

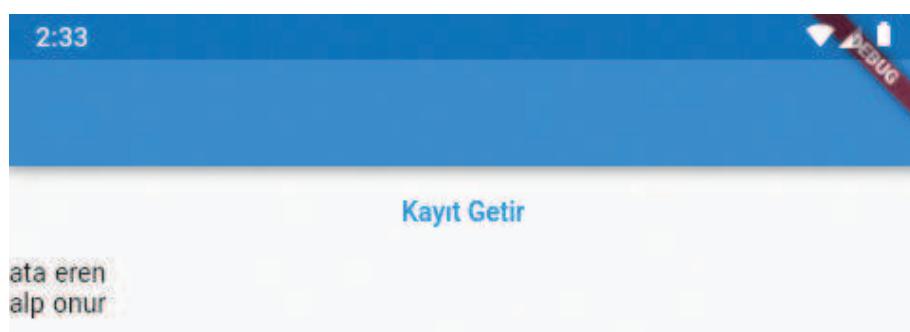
ListView kullanımına aşağıda örnek verilmiştir.

```

1. List lKullanici = [];
2.
3. Future<void> kayitGetir() async {
4.   Database db = openDatabase("dbAOF");
5.
6.   lKullanici = await.rawQuery("select * from Kullanici");
7.   setState(() {});
8. }
9.
10. @override
11. Widget build(BuildContext context) {
12.   return Scaffold(
13.     appBar: AppBar(),
14.     body: Column(
15.       children: [
16.         TextButton(
17.           child: const Text("Kayıt Getir"),
18.           onPressed: kayitGetir,
19.         ),
20.         Expanded(
21.           child: ListView.builder(
22.             itemCount: lKullanici.length,
23.             itemBuilder: (context, index) {
24.               return Row(
25.                 children: [
26.                   Text(lKullanici.elementAt(index)["adi"]),
27.                   const Text(" "),
28.                   Text(lKullanici.elementAt(index)["soyadi"]),
29.                 ],
30.               );
31.             },
32.           ),
33.         ),
34.       ],
35.     ),
36.   );
37. }

```

Yukarıdaki kod çalıştığında şu şekilde sonuç verecektir.



Burada yapılan işlemleri sırasıyla görelim. İlk olarak Kayıt Getir butonu ile kayıtGetir metodu çağrılmıyor. kayıtGetir metodunda daha önce gördüğümüz gibi rawQuery işlemi ile Kullanici tablosundaki bütün veriler getiriliyor (s. 6). Burada iKullanici nesnesi metodun dışında tanımlanmıştır. Bunun sebebi diğer metodlarda ya da body içerisinde de kullanılmaktır.

```
2. List iKullanici = [];
```

Eğer iKullanici nesnesine yukarıdaki gibi atama yapmazsanız ([]) null olarak kalacak program derlenmeyecektir. Bundan dolayı [] ile nesnemize ilk atamayı boş olarak yapıyoruz.

Gelelim ListView.builder Widget'ına. Burada ilk olarak itemCount ile ListView'in eleman sayısını belirliyoruz. Burada da mevcut iKullanici nesnesini eleman sayısına eşitliyoruz (s.22). itemBuilder ile de ListView'in nasıl görüntüleneceğini belirliyoruz. itemBuilder context ve index iki parametre olarak geliyor. itemBuilder içerisinde return ile istenilen tasarımi yaparak daha önce kullanıldığı gibi iKullanici.elementAt(index)[“alanadi”] ile verileri gösteriyoruz.

```
return Row(
  children: [
    Text(iKullanici.elementAt(index)[“adi”]),
    const Text(" "),
    Text(iKullanici.elementAt(index)[“soyadi”]),
  ],
);
```

Burada return ile önceki gördüğünüz bölümlerdeki container, column, row gibi widgetları kullanarak istediğiniz gibi görselleştirebilirsiniz.

Not: ListView widgetini builder ile kullandığınıza başka widgetlar ile kullanmak için column, row gibi widgetların içerisine almanız gerekmektedir. Fakat direkt olarak ListView'i bu bahsedilen widgetlar içerisinde alamazsınız. Bunun için Expanded (s.20) widgeti içine almanız gerekmektedir.

Güncelme – Update

Güncelme yeni kayıt eklemeye olduğu gibi hem SQL komutları ile hem de sqlite'ın hazır metodları ile yapılabilmektedir. İlk olarak SQL komutunu hatırlayalım. Update ile başlayıp tablo adı yazılır ve set dedikten sonra alan=değer olarak devam eder, ardından where ile güncellenecek satırlar için şartlar yazılır.



```
Update TabloAdı set alan1=@değer1,alan2=@değer2,alan3=@değer3,... where şart
```

Flutter'da güncelleme işlemleri için daha önce gördüğümüz rawQuery kullanılabileceği gibi güncelleme sorguları çalıştırma yönelik rawUpdate metoduda kullanılabilir. rawUpdate'in rawQuery'den farkı kaç satır güncellendiğinin sayısının geri döndürür.

```
Future<void> kayıtGuncelle() async {
  Database db = await openDatabase("dbAOF");
  await db.rawQuery(
    "update Kullanici set adi=?, soyadi=? where KullaniciId=?",
    ["Ali", "Yılmaz","1"]);
}
```

Yukarıda rawQuery ile update sorgusu çalıştırılmış KullanıcıId 1 olan kaydın adı ve soyadı Ali Yılmaz olarak güncellenmiştir.

rawQuery ile yapılan işlem rawUpdate ile yapıldığında güncellenen satırların sayısı aşağıdaki gibi bir değişkene atanabilir.

```
int count = await db.rawQuery("update Kullanici set adi=?, soyadi=? where KullaniciId=?",
    ["Ali", "Veli", "1"]);
```

Diğer bir yöntem olarak db.Update ile güncelleme işlemi yapılabilir.

```
Future<void> kayitGuncelle() async {
    Database db = await openDatabase("dbAOF");
    await db.update("Kullanici", {"Adi": "Ataa", "Soyadi": "Erenn"}, where: "KullaniciId=1");
}
```

Yukarıda db.Update metoduna tablo adı, Map olarak güncellenecek alanlar ve where şartı gönderilerek de güncelleme işlemi yapılabilir. Burada map içerisindeki hangi alanlar gönderilirse o alanlar güncellenecektir.



Silme – Delete

Silme yine diğer işlemlerde olduğu gibi hem SQL ile hem de sqlite'in hazır metotları ile yapılmaktadır. İlk olarak SQL komutunu hatırlayalım. Delete ile başlayıp tablo adı yazılır ve daha sonra silinecek satırlar ile ilgili şartlar yazılır.

```
Delete from TabloAdi where şart
```

Gelelim Flutter'da silme işleminin nasıl yapıldığına.

```
Future<void> kayitSil() async {
    Database db = await openDatabase("dbAOF");
    await db.rawQuery("Delete from Kullanici where KullaniciId=?",
    ["1"]);
}
```

Yukarıda silme işleminde rawQuery ile SQL komutu çalıştırılmış ve kayıt silinmiştir. rawQuery yerine rawDelete kullanılırsa kaç kaydın silindiği bilgisi alınabilir.

```
int count = await db.rawQuery("Delete from Kullanici where KullaniciId=?",
    ["1"]);
```

Silme işlemi aynı şekilde db.Delete() ile de yapılabilir.

```
Future<void> kayitSil() async {
    Database db = await openDatabase("dbAOF");
    db.delete("Kullanici", where: "KullaniciId=1");
}
```

db.Delete() parametre olarak tablo ismi ve where şartı gönderilir ve silme işlemi gerçekleştirilir.

Modellerle Sqlite İşlemleri

Bu bölümde gördüğümüz ekleme, güncelleme, silme işlemleri modeller ile kolay bir şekilde yapılabilmektedir. Modellerle SQL komutları kullanılmadan ekleme, güncelleme ve silme işlemleri yapılr. Bu modeller veriyi kolay bir şekilde json'a, ya da json formatında bir veriyi model dönüştürülmesini sağlar.



Bunun için her bir tabloya ait modellerin oluşturulması gerekmektedir. Otomatik model oluşturmak için çeşitli araçlar kullanabilirsiniz. İlk olarak yapılması gereken örnek bir json verisi oluşturmaktır. Bu veri tablonuzu temsil edecek şekilde aşağıdaki gibi oluşturulmalıdır. Text alanları için tırnak, sayısal alanlar için 0 ya da başka bir sayı kullanabilirsiniz.

```
{
    "kullaniciId":0,
    "adi":"",
    "soyadi":""
}
```

Bu örnek bir veriyi bir modele dönüştürmek en kolay yöntemlerden biri <https://app.quicktype.io/> adresinde Name(1) kısmına modelinizin adını yazdıktan sonra, Source Type olarak JSON(2) ve Language olarak Dart(4)'ı seçmek. Daha sonra (3) kısmına json formatındaki metni yapıştırmak.

The screenshot shows the quicktype.io interface. In the 'Name' field (1), 'KullaniciCls' is entered. In the 'Source type' dropdown (2), 'JSON' is selected. The JSON input (3) contains the following code:

```
{
    "kullaniciId":0,
    "adi":"",
    "soyadi":""
}
```

The generated Dart code (4) is as follows:

```
// To parse this JSON data, do
// final KullaniciCls = KullaniciClsFromJson(jsonString);
//
// Implementation
import 'dart:convert';

KullaniciCls KullaniciClsFromJson(String str) => KullaniciCls.fromJson(json.decode(str));

String KullaniciClsToJson(KullaniciCls data) => json.encode(data.toJson());

class KullaniciCls {
    KullaniciCls({
        this.kullaniciId,
        this.adi,
        this.soyadi,
    });

    int kullaniciId;
    String adi;
    String soyadi;

    factory KullaniciCls.fromJson(Map<String, dynamic> json) => KullaniciCls(
        kullaniciId: json["kullaniciId"],
        adi: json["adi"],
        soyadi: json["soyadi"],
    );

    Map<String, dynamic> toJson() => {
        "kullaniciId": kullaniciId,
        "adi": adi,
        "soyadi": soyadi,
    }
}
```

Bu işlem yapıldıktan sonra sizin projenize yeni bir dart dosyası açıp modelinizi projenize ekleyebilirsiniz. Projenize Kullanici.dart ile metni yapıştırıldıktan sonra yapıçı metotta değişkenlerini altını çizecektir. Dart dili 2.12 sürümü ile birlikte tanımlanan değişkenler null değer alamamaktadır. Bunun için aşağıdaki gibi değişkenler tanımlarken yanlarına soru işaretri (?) ekliyoruz ve böylece değişkenlerin null değer alabileceğini belirtmiş oluyoruz.

```
import 'dart:convert';

KullaniciCls kullaniciClsFromJson(String str) =>
KullaniciCls.fromJson(json.decode(str));

String kullaniciClsToJson(KullaniciCls data) =>
json.encode(data.toJson());

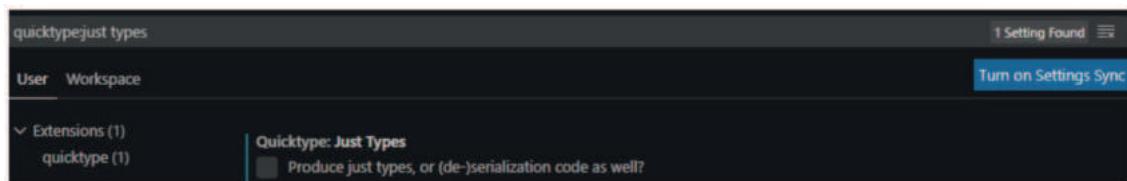
class KullaniciCls {
    KullaniciCls({
        this.kullaniciId,
        this.adi,
        this.soyadi,
    });

    int? kullaniciId;
    String? adi;
    String? soyadi;

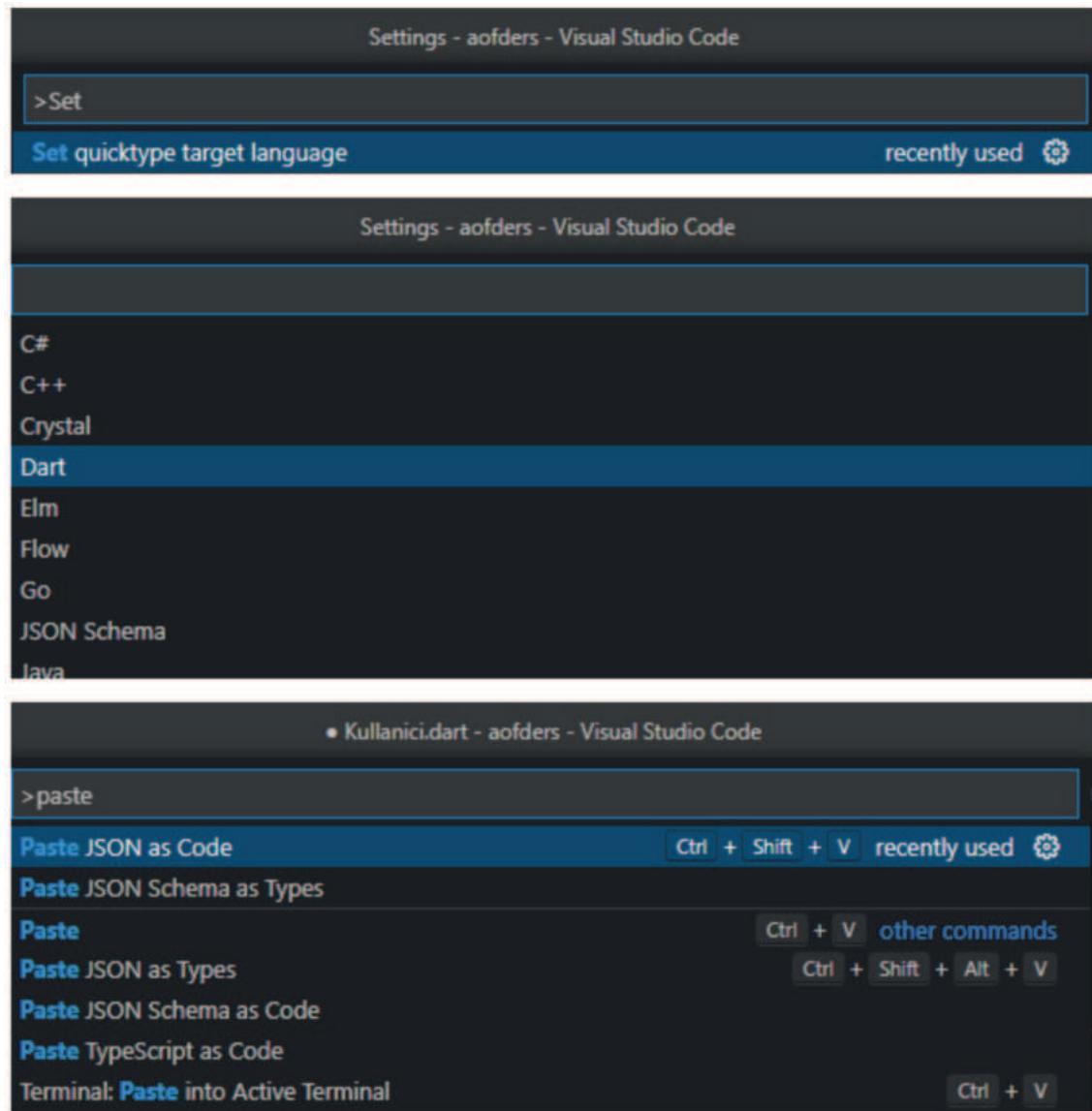
    factory KullaniciCls.fromJson(Map<String, dynamic> json) =>
KullaniciCls(
        kullaniciId: json["kullaniciId"],
        adi: json["adi"],
        soyadi: json["soyadi"],
    );

    Map<String, dynamic> toJson() => {
        "kullaniciId": kullaniciId,
        "adi": adi,
        "soyadi": soyadi,
    };
}
```

Yukarıda <https://app.quicktype.io/> ile yaptığınız işlemi eğer her tablo için ve proje boyunca tekrar tekrar yapacaksanız Visual Code'da “Paste JSON as Code” ile bu işlemi yapabilirsiniz. Bunun için json formatında verinizi kopyalamanız daha sonra F1 tuşuna basarak Paste JSON as Code seçeneği ile dosyanıza yapıştırdığınızda yukarıdaki model aynı şekilde gelecektir. Fakat bundan önce iki önemli değişiklik yapmanız gerekmektedir. İlk olarak File Prefrence'den Settings(Ayarlar)'ı açtıktan sonra arama çubuğuna QuickType:Just Types yazınız. Burada QuicKType:Just Types başlığında kutucuğu boşaltınız.



İkinci olarak QuickType eklentisinin dilini ayarlamak için aynı menüden Set quicktype target language seçeneği ile dili Dart olarak seçiniz.



Bu işlemleri ilgili videoda ayrıntılı olarak izleyebilirsiniz.

Şimdi ekleme işleminin model ile nasıl yapılacağını görelim.

```
Future<void> kayitEkle() async {
    Database db = await openDatabase("dbAOF");
    KullaniciCls Kullanici = KullaniciCls(adi:"Alp",soyadi: "Eren");
    db.insert("Kullanici", Kullanici.toJson());
}
```

Yukarıda kayıt işlemi sınıfından türettiğimiz Kullanici ile gerçekleşmektedir. Burda KullaniciCls içerisinde adı ve soyadı alanlarına değer girilmiştir. Eğer sadece adı alanı girelse de soyadı alanı null olarak eklenecekti. Siz de tablolarınızda hangi alanlara veri girecekseniz o alanlara parantez içinde tanımlama yapabilirsiniz.

kayitGetir işlemini de model ile yapabilirsiniz.

```
Future<void> kayitGetir() async {
    Database db = await openDatabase("dbAOF");
    iKullanici = await db.rawQuery("select * from Kullanici");
    list = List<KullaniciCls>.from(
        iKullanici.map((e) => KullaniciCls.fromJson(e)));
    setState(() {});
}
```

Burada amaç kayıtların bir model olarak alınıp model olarak işlenmesidir. Kayıtlar bu şekilde alındığında gösterimi aşağıdaki gibi olacaktır.

```
Expanded(
    child: ListView.builder(
        itemCount: iKullanici.length,
        itemBuilder: (context, index) {
            return Row(
                children: [
                    Text(list[index].adi.toString()),
                    const Text(" "),
                    Text(list[index].soyadi.toString()),
                ],
            );
        },
    )
)
```

Daha önce kayıt getir için ListView içerisinde

```
Text(lKullanici.elementAt(index)["adi"])),
```

olarak kayıtları getiriyorduk. Fakat model kullanıldığı için

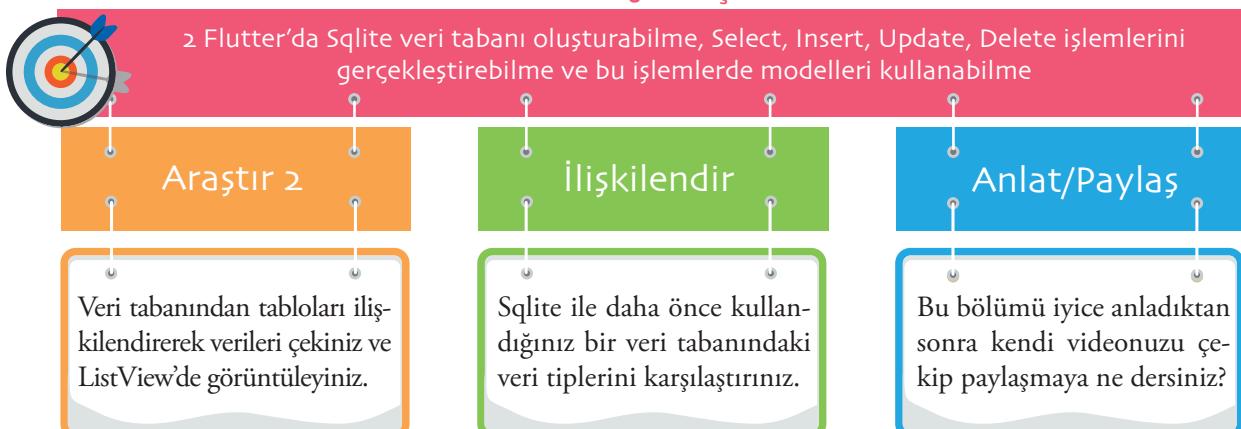
```
Text(list[index].adi.toString()),
```

şeklinde kayıt çağrılr. Eğer büyük ölçekli bir proje yapıyorsanız modellerden faydalamanız gerekmektedir. Tabloları modellediğinizde işlemleri daha hızlı ve doğru bir biçimde yaparsınız.

```
final _kullaniciId = TextEditingController();  
final _adi = TextEditingController();  
final _soyadi = TextEditingController();  
KullaniciCls _kullanici = KullaniciCls();  
Future<Database> database = openDatabase("dbAOF");  
Future<void> kayitGetir() async {  
    final db = await database;  
    List l = await db.rawQuery(  
        "select * from Kullanici where KullaniciId=?",
[_kullaniciId.text]);  
    _kullanici = KullaniciCls.fromJson(l[0]);  
    // lKullanici =  
    // List<KullaniciCls>.from(l.map((e) => KullaniciCls.fromJson(e)));  
    setState(() {  
        _adi.text = _kullanici.adi.toString();  
        _soyadi.text = _kullanici.soyadi.toString();  
    });  
}  
  
Future<void> kayitGuncelle() async {  
    final db = await database;  
    _kullanici.adi = _adi.text;  
    _kullanici.soyadi = _soyadi.text;  
    db.update("Kullanici", _kullanici.toJson(),  
        where: "KullaniciId=?", whereArgs: [_kullanici.kullaniciId]);  
    List l = await db.rawQuery(  
        "select * from Kullanici where KullaniciId=?",
[_kullaniciId.text]);  
    print(l);  
}  
  
@override  
Widget build(BuildContext context) {  
    return Scaffold(  
        appBar: AppBar(),  
        body: Column(  
            children: [  
                TextFormField(  
                    controller: _kullaniciId,  
                    decoration: InputDecoration(hintText: "KullanıcıId"),  
                ),  
                TextButton(onPressed: kayitGetir, child: const Text("Getir")),  
                TextFormField(  
                    controller: _adi,  
                    decoration: InputDecoration(hintText: "Adı"),  
                ),  
                TextFormField(  
                    controller: _soyadi,  
                    decoration: InputDecoration(hintText: "Soyadı"),  
                ),  
                TextButton(onPressed: kayitGuncelle, child: const  
Text("Güncelle")),  
            ],  
        ),  
    );  
}
```

Burada model ile güncelleme işlemini nasıl yaptığımızı özetleyelim. "Getir" butonu ile tek bir kayıt KullaniciCls sınıfından türetilen _kullanici ile ilişkilendirilmektedir. "Kaydet" butonu ile de _kullanici nesnesine adı ve soyadı güncellenmekte ve db.Update() metoduna direkt olarak _kullanici nesnesi ve Where şartı ile de _kullanici nesnesinin KullaniciId değişkeni gönderilmektedir. Bu şekilde herhangi bir SQL komutu yazmadan güncelleme yapılmış olur.

Öğrenme Çıktısı



FLUTTER İLE API İŞLEMLERİ

Flutter ile API'leri kullanmak için pub.dev'den http (<https://pub.dev/packages/http>) kütüphanesini projemize dahil ediyoruz. Bunun için console'a



```
flutter pub add http
```

yazarak, ya da pubspec.yaml dosyasına dependences altına

```
http: ^0.13.4
```

satırını ekliyoruz(Bu satırların yazıldığı gün sqflite'in son sürümü ^0.13.4'dür. Sizin bu bölümü okuduğunuz tarihte daha ileri sürümleri olabilir. 10.01.2022).

Kayıt Getirme (Get)

API ile get işlemleri yapılırken aşağıdaki gibi http.get() metodu kullanılır. Bu metot ile API'den veri alınır ve bir değişkene atanır.

```
List ogrenci = [];

void getir() async {
  var url = Uri.parse('https://[adres]/ogrencis');
  var response = await http.get(url);
  if (response.statusCode == 200) {
    ogrenci = jsonDecode(response.body);
    print(ogrenci);
  } else
    print("hata");
}
```

Yukarıda ilk olarak url değişkenine API'nin adresi atanmıştır. http.get() metodu url ile veriyi alır ve response isimli değişkene atar. Burada if içerisinde statusCode sorgulanır. Eğer statusCode 200 ise cevap alınmış anlamına gelir ve işlem devam eder. Değilse hata döndürür. Burada response body bizim API'den çektiğimiz veridir. Bu veriyi print ile Console'a yazdırıp gelen veriyi görebilirsiniz.

```
[{KullaniciId: 1, Adi: Alp, Soyadi: Onur}, {KullaniciId: 2, Adi: Ata, Soyadi: Eren}]
```

Şimdi API'den çektiğimiz veriyi ListView'de göstermek için metodumuzu değiştirelim.

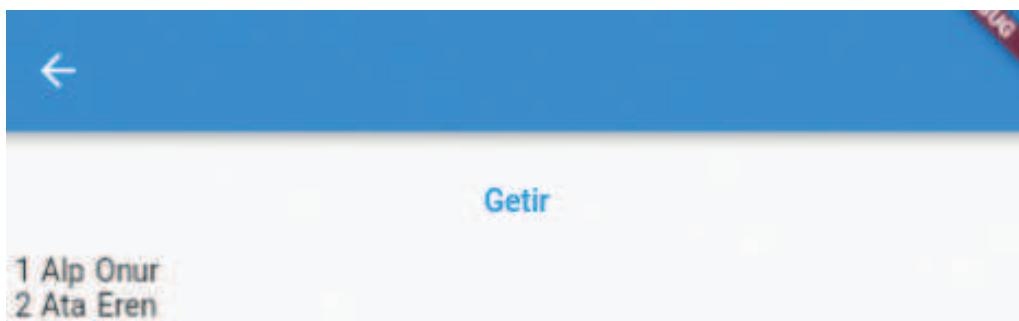
```
List<Kullanicicls> kullanici = [];

void getir() async {
    var url = Uri.parse('https://[adres]/ogrencis');
    var response = await http.get(url);
    if (response.statusCode == 200) {
        setState(() {
            kullanici = List<Kullanicicls>.from(
                jsonDecode(response.body).map((e) =>
Kullanicicls.fromJson(e)));
        });
        print(data);
    } else
        print("hata");
}
```

Yukarıdaki işlemde http.get() ile veri getirildikten sonra, yukarıda kullanici isimli oluşturulan List'e mapping ile atanır. Bu işlem setState() içerisinde yapıldığı için veri getirildiğinde ListView'e dolacaktır.

```
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        body: Column(
            children: [
                TextButton(onPressed: getir, child: const Text("Getir")),
                Expanded(
                    child: ListView.builder(
                        itemCount: kullanici.length,
                        itemBuilder: (context, index) {
                            return Row(
                                children: [
                                    Text(kullanici[index].ogrenciId.toString()),
                                    Text(" "),
                                    Text(kullanici[index].adi.toString()),
                                    Text(" "),
                                    Text(kullanici[index].soyadi.toString())
                                ],
                            );
                        })),
                ],
            )));
}
```

Burada SQLite ile daha önce doldurduğumuz gibi ListView oluşturulur. itemCount yine List'in length'ı atanır ve itemBuilder içerisinde ListView'in tasarımları yapılır.



Kayıt Ekleme (Post)

API'lerde ekleme için post, güncelleme için put ve silme işlemleri için delete методu kullanılır. Tabi bunun için bağlantı kuracağınız API bu işlemlere yönelik yapılandırılmış olması gerekmektedir.

```
void kayitekle() async {
    var url = Uri.parse('https://[adres]/ogrencis');
    Kullanicicls kul = Kullanicicls(
        KullaniciId: 0, adi: _adicontroller.text, soyadi:
        soyadicontroller.text);

    var response = await http.post(url,
        headers: {"Content-Type": "application/json"},
        body: jsonEncode(kullanici.toJson()));
}
```

Yeni bir kayıt eklerken http.post() metodları kullanılabilir. Bu tamamen API'de hangi özelliklerin var olduğu ile ilgilidir. Yukarıdaki kod ile http.post() yapılmış ve 3 parametre girilmiştir. İlk parametre http.get() de olduğu gibi url. İkinci parametre verinin hangi formatta gideceğinin belirlendiği headers. Headersda gidecek verinin json formatında olduğu ifade ediliyor. Üçüncü parametre ise body. Burada daha önce oluşturduğumuz KullaniciCls isimli sınıfın türetilen ve değerleri TextFormField'lerden alan bir nesne json formatına çevriliyor.

Kayıt Güncelleme (PUT)

API ile güncelleme yapmak için http.put() методu kullanılır. Ayrıca güncellenecek kaydın birincil anahtar olan alanı API adresinde belirtilir.

```
void guncelle() async {
    var url = Uri.parse('https://[adres]/ogrencis/[ ]');
    Kullanicicls kullanici = Kullanicicls (
        kullaniciId: 1, adi: "Ali", soyadi: "Yılmaz");

    var response = await http.put(url,
        headers: {"Content-Type": "application/json"},
        body: jsonEncode(kullanici.toJson()));
    print(response.body);
}
```

Burada API adresinden sonra ilgili kaydın birincil anahtarı eklendi. Yukarıda KullanıcıId 1 olan kayıt güncelleniyor. Yine kullanıcı isimli bir nesne oluşturuldu. Bu sefer http.put() ile url, headers ve body gönderildi ve bu şekilde kayıt güncellenmiş oldu. Tabi burada kullanıcılıd, adı ve soyadı alanları dinamik olarak gelebilir, seçilen kayıt güncellenebilir. Burada örnek olmasından 1 nolu kayıt güncellenmiştir.

Kayıt Silme (Delete)

Kayıt silme işlemi yine API'nin desteklemesi koşulu ile aşağıdaki gibi yapılır.

```
void kayitsil() async {
  var url = Uri.parse('https://[adres]/ogrencis/1');
  var response = await http.delete(url);
}
```

Silme işleminde http.delete() metodu kullanılır. http.delete() metoduna API adresi yukarıdaki gibi birincil anahtarı gönderilerek silme işlemi gerçekleştirilir.

Bu bölümde anlatılan tüm içeriklerin uygulamalı bir şekilde anlatımına ulaşmak için aşağıdaki karekodu okutunuz.



Öğrenme Çıktısı

API'ye bağlanabilme ve API ile veri ekleme, getirme, güncelleme ve silme işlemlerini gerçekleştirebilme		
 Araştır	İlişkilendir	Anlat/Paylaş
Internet üzerinden herkese açık API'lerden worldtimeapi.org'u kullanarak İstanbul'a ait anlık zaman bilgisini gösteriniz.	Internet üzerinden farklı API'ler bularak Flutter projelerinde kullanınız ve ilişkilendiriniz.	Bu bölümü iyice anladıkten sonra kendi videonuzu çekip paylaşmaya ne dersiniz?

1

Flutter'da asenkron işlemleri yapabilme,
future, async, await anahtar kelimelerini
doğru bir şekilde kullanabilme

Asenkron (Async) İşlemler,
Future ve Await

SQLite mobil uygulamalarda veri saklama amaçlı olarak kullanılan en popüler veritabanıdır. Fakat veri tabanı ile yapılan işlemler eş zamansız olarak gerçekleştirilmektedir. Flutter'da dart ile kodlarken normalde satırlar arası ardına çalışır. Fakat bazen bazı satırlardaki veritabanından veri çekmek, yazmak, güncellemek ya da API'ler ile veri almak/göndermek gibi işlemler zaman alabilmektedir. Bu işlemler gerçekleştirken sonuç hemen dönmeden hatalar/eksiklikler neden olabilir. Bu yüzden zamana ihtiyaç olan kodlar için asenkron işlemleri kullanılır. Uygulama bu tür işlemler bitmeden aynı metotta bir sonraki satıra geçmez. Bunun için asenkron metodlar oluşturulur. Asenkron metod oluştururken metod tanımlandığı satırda parametre parantezleri kapatıldıktan sonra "async" anahtar kelimesi yazılır. Ayrıca asenkron olarak gerçekleştirilecek işlemlerin başına da "await" anahtar kelimesi yazılır. Böylece ilgili satırın gerçekleşmesi zaman gerektirirse bile o işlem beklenir ardından diğer satıra geçilir. Böyle işlemler bir değişken yada bir nesneye atandığında o nesne Future anahtar kelimesi ile tanımlanırsa, nesnenin işlemlerinin zaman alacağı belirtilmiş olur. Böyle bir durumda işlemin bitmesi, hata döndürmesi, zaman aşımı olması gibi durumlarda müdahale etme imkanı verir.

2

Flutter'da asenkron işlemleri yapabilme, future, async, await anahtar kelimelerini doğru bir şekilde kullanabilme

Flutter ile Sqlite İşlemleri

Sqlite işlemlerini yapabilmek için Sqflite kütüphanesinin projeye dahil edilmesi gerekmektedir. Terminalden “flutter pub add sqflite” komutu ile dahil edebildiğiniz gibi, pubspec.yaml dosyasında dependencies’ın altında “sqflite: ^2.0.2” satırını ekleyerek de projenize Sqlite’ı dahil edebilirsiniz. Bu noktadan sonra veritabanını oluşturmak ve açmak için openDatabase(yol) komutunu kullanmanız gerekmektedir. openDatabase ilk çalıştığında veritabanını oluşturur, eğer veritabanı varsa veritabanını açar. Database türünden bir nesne tanımlayarak işlemler o nesne üzerinden yapılır.

Database db = await openDatabase(yol);

Veritabanı bağlantısı yapıldıktan sonra execute, rawQuery, rawInsert, rawUpdate, rawDelete, Query, Insert, Update, Delete metodları ile işlemler yapılabilir.

Execute: Parametre olarak girilen SQL komutunu çalıştırır.

rawQuery: Parametre olarak girilen Select cümlesini çalıştırır ve dönen veriyi alır. Bu veri List türünden nesneler atanıp kullanılabilir.

rawInsert: Insert cümlesi ile tabloya veri eklendikten sonra, eklenen satırda ait birincil anahtarı geri döndürür. Insert işlemleri execute ve rawQuery ile de yapılabilir fakat bu metodlar ilgili birincil anahtarı geri döndürmez.

rawUpdate: Update cümlesi ile tabloda veri güncelledikten sonra güncellenen satır sayısını geri döndürür. Update işlemleri execute ve rawQuery ile de yapılabilir fakat bu metodlar güncellenen satır sayısını geri döndürmez.

rawDelete: Delete cümlesi ile tabloda veri silindikten sonra, silinen satır sayısını geri döndürür. Update işlemleri execute ve rawQuery ile de yapılabilir fakat bu metodlar silinen satır sayısını geri döndürmez.

query: Select işlemi yaparken SQL cümlesi kullanılmaz. Parametre olarak tablo adı ve varsa şartlar, sıralamalar, gruplamalar için alan adları ve değerler girilir.

Insert: Insert’ta SQL cümlesi kullanılmaz. Tablo adı ve parametreler girilir. Metot ekleme işlemini yapar.

Update: Update’ta de SQL cümlesi kullanılmaz. Tablo adı ve parametreler girilir. Metot güncelleme işlemini yapar.

Delete: Delete’ta de SQL cümlesi kullanılmaz. Tablo adı ve parametreler girilir. Metot silme işlemini yapar.

Veritabanı işlemlerinde genel olarak SQL cümleleri kullanılır. Fakat SQLite kütüphanesine ait query, insert, update, delete metodları SQL cümleleri kullanmadan işlemlerin yapılmasını sağlar. Bu işlemler yapılırken parametreler JSON formatında gönderilir. Her tablo ile ilgili işlem yapılırken manuel olarak JSON oluşturmaktansa tablolara sınıflara dönüştürüldükten sonra modeller üzerinden işlemler yapılabilir. Bunun için ilk olarak işlem yapılacak tablo herhangi bir araç ile dart diline uygun sınıflara dönüştürülür. Daha sonra ekleme, güncelleme yapılacak metodlarda bu sınıftan nesneler türetilir ve bu nesneler JSON formatında insert , update metodlarına parametre olarak gönderip işlemler gerçekleştirilebilir. Aynı şekilde rawQuery ile alınan veriler Map ile o tabloyu temsil eden sınıflardan türetilen List'lere atılabilir ve görüntüleme işlemleri bu List'ler ile kolay bir şekilde yapılabilir.

3

API'ye bağlanabilme ve API ile veri ekleme, getirme, güncelleme ve silme işlemlerini gerçekleştirebilme

Flutter ile API İşlemleri

Flutter ile internet üzerinden veri alışverişi yapmanın yollarından biri de API'leri kullanmaktadır. API'ler “http” kütüphanesi projeye dahil edilerek kullanılabilir. Terminalden “flutter pub add http” komutu ile dahil edebildiğiniz gibi, pubspec.yaml dosyasında dependencies'in altında “http: ^0.13.4” satırını ekleyerek de projenize http kütüphanesini dahil edebilirsiniz (Burdaki sürüm bölümün yazıldığı günlerde mevcut olan son sürümüdür). http kütüphanesi ile yaptığınız işlemleri SQLite'da olduğu gibi asenkron olarak yapmanız gerekmektedir. Veri çekme işlemini gerçekleştirirken http.get() metodunu kullanarak gelen cevabı Json formatında alıp işlemlerinizi yapabilirsiniz. API'lere veri göndermek için yeni bir kayıt eklenerek http.post(), güncelleme yapılırken http.put() ve silme işleme yapılırken http.delete() metodlarını kullanabilirsiniz.

1 Dart dilinde “Future” kavramının tanımı aşağıdakilerden hangisiidir?

- A. Future asenkron çalışmaların sonucunu temsil eder.
- B. Future senkronize çalışmaların sonucunu temsil eder.
- C. Future tamamlanmış çalışmaların sonucunu temsil eder.
- D. Future başarı ile tamamlanmış çalışmaların sonucunu temsil eder.
- E. Future taslak halinde olan çalışmaların sonucunu temsil eder.

2 I. `Async`
II. `Await`
III. `Future`

Yukarıdaki ifadelerden hangisi veya hangileri asenkron metodlarda kullanılması zorunlu **değildir**?

- A. Yalnız I
- B. Yalnız II
- C. Yalnız III
- D. I ve II
- E. I ve III

3 Aşağıdakiler metodlardan hangisi Sqflite kütüphanesi ile veri çekmek için kullanılır?

- A. `Execute`
- B. `RawQuery`
- C. `RawSql`
- D. `RawSelect`
- E. `RawInsert`

4 Aşağıdaki metodlardan hangi Sqflite kütüphanesi ile veri eklemek için **kullanılmaz**?

- A. `Execute`
- B. `RawQuery`
- C. `RawInsert`
- D. `Insert`
- E. `Update`

5 Aşağıdaki metodlardan hangi Sqflite kütüphanesi ile veri güncellemek için **kullanılmaz**?

- A. `Execute`
- B. `RawQuery`
- C. `Insert`
- D. `RawUpdate`
- E. `Update`

6 Dart dilinde değişken tanımlarken değişkenin null değer alabileceğini belirtmek için aşağıdakilerden hangisi kullanılır?

- A. ! (Ünlem işaret)
- B. % (Yüzde işaret)
- C. ? (Soru işaret)
- D. # (Diyez işaret)
- E. \$ (Dolar işaret)

7 I. Veriler Json formatına dönüştürülerek işlenir.
II. Veritabanından gelen veriler Map ile modellere dönüştürülür.
III. Veriler XML formatına dönüştürülerek işlenir.

Yukarıdakilerden hangisi veya hangileri model kullanımı ile ilgili doğrudur?

- A. Yalnız I
- B. Yalnız II
- C. Yalnız III
- D. I ve II
- E. I ve III

8 Aşağıdaki metodlardan hangisi API’ler ile kayıt eklerken kullanılır?

- A. `Http.get`
- B. `Http.put`
- C. `Http.post`
- D. `Http.delete`
- E. `Http.copy`

9 Aşağıdaki metodlardan hangisi API’ler ile kayıt güncellerken kullanılır?

- A. `Http.get`
- B. `Http.put`
- C. `Http.post`
- D. `Http.delete`
- E. `Http.copy`

10 Aşağıdaki metodlardan hangisi API’ler ile kayıt silerken kullanılır?

- A. `Http.get`
- B. `Http.put`
- C. `Http.post`
- D. `Http.delete`
- E. `Http.copy`

1. A

Yanıtınız yanlış ise “Asenkron (Async) İşlemler, Future ve Await” konusunu yeniden gözden geçiriniz.

2. C

Yanıtınız yanlış ise “Asenkron (Async) İşlemler, Future ve Await” konusunu yeniden gözden geçiriniz.

3. B

Yanıtınız yanlış ise “Flutter ile Sqlite İşlemleri” konusunu yeniden gözden geçiriniz.

4. E

Yanıtınız yanlış ise “Flutter ile Sqlite İşlemleri” konusunu yeniden gözden geçiriniz.

5. C

Yanıtınız yanlış ise “Flutter ile Sqlite İşlemleri” konusunu yeniden gözden geçiriniz.

6. C

Yanıtınız yanlış ise “Flutter ile Sqlite İşlemleri” konusunu yeniden gözden geçiriniz.

7. D

Yanıtınız yanlış ise “Modellerle Sqlite İşlemleri” konusunu yeniden gözden geçiriniz.

8. C

Yanıtınız yanlış ise “Flutter ile Api İşlemleri” konusunu yeniden gözden geçiriniz.

9. B

Yanıtınız yanlış ise “Flutter ile Api İşlemleri” konusunu yeniden gözden geçiriniz.

10. D

Yanıtınız yanlış ise “Flutter ile Api İşlemleri” konusunu yeniden gözden geçiriniz.

6

Araştır Yanıt
Anahtarı

Araştır 1

Flutterda dosya işlemlerinin de asenkron olduğundan bahsetmiştik. Burada AOFDeneme.txt adında bir dosya açıp içerisinde Merhaba dünya.. yazıyoruz. 2. satırda uygulamanın ana dizini directory isimli değişkene atarken await'i kullanıyoruz. Sonra bu dizinin yolunu path isimli değişkene atıyoruz (s.3). 4. satırda File sınıfından f isimli bir nesne üretip AOFDeneme.txt dosyasını oluşturuyoruz. 5. satırda ise dosyaya "Merhaba dünya..." yazıyoruz. 6. satırda f nesnesinde readAsString ile içerisindeki metni okuyoruz. Burada işlem asenkron olacağı için await anahtar sözcüğünü

```
1. void yaz() async {  
2.   final directory = await getApplicationDocumentsDirectory();  
3.   var path = directory.path;  
4.   File f = File('$path/AOFDeneme.txt');  
5.   f.writeAsString("Merhaba dünya...");  
6.   var metin = await f.readAsString();  
7.   print(metin);  
8. }
```

Araştır 2

Burada yapmanız gereken SQL komutu ile iki tabloyu birleştirmek. Örneğin kullanıcı tablosunda doğum yeri olarak şehir kodu(SehirKodu) bulunsun. Şehir tablosunda ise şehir kodu(SehirKodu) ve Şehir(Sehir) bulunsun. Gelin bu 2 tabloyu birlestirelim.

Select k.KullaniciId, k.Adi, k.Soyadi,s.Sehir from Kullanici k inner join Sehir s on s.SehirKodu = k.SehirKodu

Yukarıdaki sorguyu artık rawQuery ile çalıştırabilirsiniz. Ama ilk olarak Sehir tablosunu oluşturmayı unutmayın.

Araştır Yanıt
Anahtarı

6

Araştır 3

```
var zaman = {};  
  
void getir() async {  
    var url =  
Uri.parse('http://worldtimeapi.org/api/timezone/Europe/Istanbul');  
    var response = await http.get(url);  
    if (response.statusCode == 200) {  
        zaman = jsonDecode(response.body);  
    } else  
        print("hata");  
    setState(() {});  
    print(zaman);  
}
```

```
return Scaffold(  
    appBar: AppBar(),  
    body: Column(  
        children: [  
            TextButton(  
                child: const Text("Getir"),  
                onPressed: getir,  
            ),  
            Row(  
                children: [  
                    Text("TimeZone: "),  
                    Text(zaman["timezone"].toString()),  
                ],  
            ),  
            Row(  
                children: [  
                    Text("DateTime: "),  
                    Text(zaman["datetime"].toString()),  
                ],  
            ),  
            Row(  
                children: [  
                    Text("Hafatının kaçinci günü: "),  
                    Text(zaman["day_of_week"].toString()),  
                ],  
            ),  
            Row(  
                children: [  
                    Text("Yılınca kaçinci günü: "),  
                    Text(zaman["day_of_year"].toString()),  
                ],  
            ),  
        ],  
    ),
```

Kaynakça

<https://dart.dev/codelabs/async-await>,
<https://api.flutter.dev/flutter/dart-async/Future-class.html>
<https://pub.dev/packages/sqlite>
<https://pub.dev/packages/http>
<https://www.youtube.com/watch?v=SmTCmDMi4BY>

Bölüm 7

İleri Düzey Uygulama Seçenekleri

Öğrenme Çıktıları

Sensör Tabanlı Uygulamalar

- 1 Mobil cihazlardaki sensörlerle etkileşime girerek Flutter tabanlı uygulamalar geliştirebilme

Artırılmış Gerçeklik Uygulamaları

- 3 Artırılmış Gerçeklik teknolojisi kullanan Flutter tabanlı uygulamalar geliştirebilme

Kamera Kullanan Uygulamalar

- 2 Mobil cihaz kamera kullanımını içeren Flutter tabanlı uygulamalar geliştirebilme

Yapay Zeka Uygulamaları

- 4 Yapay zeka içeren Flutter tabanlı uygulamalar geliştirebilme

Anahtar Sözcükler: • Sensörler • İvmeölçer • Manyetometre • Mobil Kamera • Artırılmış Gerçeklik
• Yapay Zeka • Makine Öğrenmesi • Dart



GİRİŞ

Mobil cihazlarda gerçekleştirebileceğimiz uygulamalar donanım ve yazılım tabanlı birçok gelişmeye yakından ilgilidir. Özellikle cep telefonu formatında sadece aramaları ve mesajlaşmaları destekleyen mobil cihazlar, zamanla telefon rolünün ötesine geçmiş ve her türlü bilgi oluşturma, değiştirme ve paylaşma çözümlerini de kapsamları içeresine almıştır. Söz konusu değişimde özellikle *Internet* ve *Web* teknolojilerinin önemli etkileri vardır. İnternet sayesinde artan bireyler arası etkileşim düzeyi, farklı uygulamaların geliştirilmesine ve mobil cihazlara entegre edilmesine sebep olmuştur. Örneğin *sosyal medya* platformları, iletişim uygulamalarındaki önemli dönüm noktaları arasında yer almaktadır. Bununla birlikte gerek sosyal medya platformları gerekse alternatif e-ticaret, mesajlaşma (eposta servisleri ve özel mesajlaşma uygulamaları) ve işbirlikçi çözüm ortamlarının oluşturmaya başladığı, *Büyük Veri* olarak anılan bilgi hengamesi, mobil cihazların hızlı çözüm üretebilme kapasitelerinin yenilenmesine sebep olmuştur. Bu yenilemeler donanım tarafında olduğu kadar yazılım tarafında da gerçekleşmiştir. En basitinden dokunmatik ekranlarla, daha yüksek bellekler, işlemciler ve ek kameralarla birlikte yenilikçi sensörler sayesinde *akıllı (smart)* niteliği kazanan mobil cihazlar yazılım taraflı kabiliyetlerini de artırmıştır. Artık sadece akıllı telefon adı verilen cihazları değil; tabletleri ya da taşınabilir dizüstü bilgisayarları, hatta giyilebilir cihazları da kapsamı içeresine alan *mobil teknolojiler* dünyası, daha fazla kullanıcı etkileşimi ni içeren yenilikçi yazılımların tasarılanıp geliştirilmesini de içermektedir.

Web: İnternet teknolojisinin kullanıcılarla açılan, yazılım tabanlı uygulamalarını içeren, bir tür sanal yüzüdür.

Büyük Veri: Bilişim araçlarının yoğun kullanımıyla birlikte hızlı bir ivmeye oluşan, miktarı oldukça fazla, içeriği karışık, değerli veri bütünüleridir.

Mobil teknolojiler 21. yüzyılın sadece ilk çeyreğinde bile hayatımızın ayrılmaz birer parçası haline gelmiştir. Artık günlük hayatı gerçekleştirdiğimiz birçok faaliyet, sayısal ortamlardaki bilgiler (*veriler*) üzerinde çalışan özel mobil uygulamalar sayesinde yerine getirilebilmektedir. Mobil teknolojiler kullanıcılar arası anlık iletişim çözümlerinden, eğitim uygulamalarına, bankacılık işlemlerinden sağlık uygulamalarına kadar hayatın her alanında yoğun bir şekilde kullanılmaktadır (Agarwal, & Bhattacharyya, 2018; Keengwe, & Bhargava, 2014; Meder, & Wegner, 2015; Shaikh, & Karjaluo, 2015). Ancak bu kadar farklı kullanım senaryosu, kullanıcı etkileşimine daha fazla önem veren, veri işleme süreçleri sayesinde otomatik çözümler üreten uygulamaları gerekliliği kılmalıdır. Vazgeçilmez parçamız olan mobil cihazlardan etkileşimsel bağlamda faydalananın en pratik yolu, donanımsal bileşenlerden elde edilen verilerle kullanım tecrübesini artıran uygulamardan geçmektedir.

Mobil uygulamalarda, kullanım tecrübesini etkileşim boyutunda artırmak için şu hususları dikkate almak gerekmektedir:

1. *Kolay kullanım sağlamak:* Hayat koşturmacası içerisinde en hızlı şekilde kullanılabilen ve kullanıcıların en çok sevdiği mobil uygulamalar kolay kullanım sağlayanlardır. Söz konusu kolay kullanım sade, anlaşılır arayızlerle olabileceği gibi, az sayıda kullanıcı dönütü içerebilecek senaryolarla da sağlanabilmektedir.
2. *Donanım-yazılım etkileşiminden faydalananmak:* Mobil cihazlar, etkileşim ihtiyacının artışı nedeniyle alternatif kameralardan sensörlere kadar kullanıcı ile gerçek dünyayı bağlantılıyan ve kullanıcı eylemlerini yakalayan birçok donanımla desteklenmiş durumdadır. Dolayısıyla bu donanımlardan faydalanan yazılımları (uygulamaları) geliştirmek kullanım tecrübesinde yenilikçi ürünlerin elde edilmesini mümkün kılmaktadır.
3. *Gerçek dünyayı mobil ortama taşımak:* Kullanıcıların mobil uygulamaları kullanırken gerçek dünyadan destek alabilmeleri, etkileşim düzeyinde büyük avantajlar sunmaktadır.

4. İleri düzey veri işleme süreçleri işletmek: Mobil uygulamalar aldıkları verileri işleyip anlamlı hale getirerek kullanıcılarla zekice çözümler üretebilmekte, onları *asiste* ederek destekleyebilmektedir.



Şekil 7.1 Günümüz Mobil Uygulamalarında Birçok Farklı İletişim ve Etkileşim Seçeneği Yer almaktadır.



Mobil uygulamalarda özellikle etkileşimi artıran mekanizmalar tasarlarken kullanım amacı ve hedef kitleyi iyi analiz etmek gerekmektedir. Bu da mobil uygulama geliştirme süreçlerinin dikkatli planlanmasıının ve kullanıcı deneyimlerinin önemini işaret etmektedir.

Dünya çapında açık ara yaygın kullanılan Android ve iOS tabanlı mobil cihazlar dikkate alındığında, cihaz üzerindeki kameraların, cihaz içeresine eklenecek kullanıcı / cihaz hareketlerini yakalama amacı taşıyan, **sensör** adı verilen bileşenlerin kullanımını içeren ileri düzey uygulamalar kodlamak oldukça kolaydır. **Flutter** geliştirme ortamı ve **Dart** programlama dili bu konuda etkin kodlama yaklaşımını sunmakta, böylelikle *iletişimin ötesinde, etkileşime de hitap eden*, etkileyici uygulamalar kodlanabilmektedir. Etkileşimin artırılması noktasında, **Artırılmış Gerçeklik** teknolojileri ile gerçek ortam içerisinde sanal unsurlar eklenebilmekte, kullanıcılarla benzeriz kullanımları sunulabilmektedir. Ayrıca, **Yapay Zeka** tabanlı çözümler sayesinde otomatik bir biçimde verileri işleyen ve kullanıcıya *karar destek* süreçlerinde dönütler bulunabilen *zeki uygulamalar* da Flutter ile birlikte kullanılabilen destekleyici API ve eklentilerle geliştirilebilmektedir.

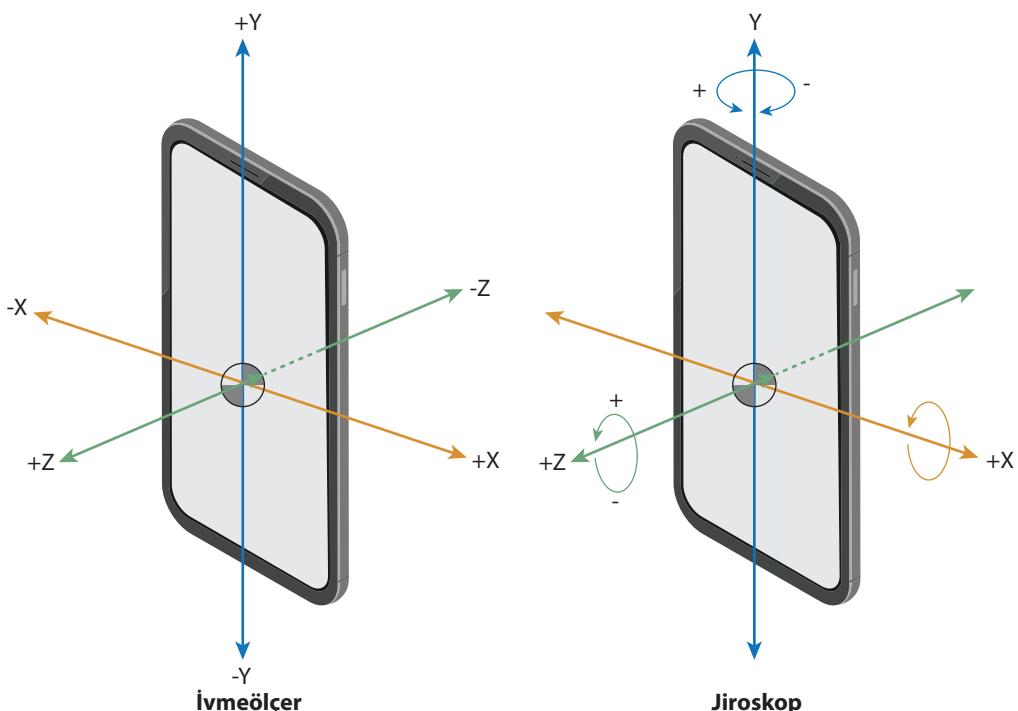
SENSÖR TABANLI UYGULAMALAR

Sensörler, güncel mobil teknolojilerin vazgeçilmez bileşenleri arasında yer almaktadır. Aslında mobil cihazlarımızdaki mikrofonlar bile sensör niteliğindedir. Ancak çevre ortamı algılama aşamasındaki farklı ihtiyaçlar birçok sensörün geliştirilmesine sebep olmuştur. *Mobil taşınırlığı sağlamak* ya da cihazı *giyilebilir* bir biçimde kullanmak adına önce küçük geliştirilen mobil cihazlarda yazılım kabiliyetlerinin artırılması sensörler sayesinde mümkün olmaktadır. Sensörler, çevredeki ışık, ses, basınç gibi unsurları algılayan birçok farklı makinede / cihazda kullanılıyor olsa da, mobil teknolojiler sayesinde seviye atlamp ve yenilikçi mobil uygulamaların doğmasını kolaylaştırmıştır.

✓ **Sensör:** Çevre ortamındaki fiziksel değişimleri / olayları algılayıp sayısal olarak ölçebilen donanımsal bileşenlerdir.

Mobil cihazlarda üretici ve model bazlı olmak üzere birçok farklı sensörün kullanımı söz konusudur. Ancak diğerlerine göre daha çok dikkat çeken, uygulamalara daha kritik mekanizmalar kazandıran en önemli sensörleri, görevleriyle birlikte şöyle sıralayabiliriz:

- İvmeölçer:** İngilizce *Accelerometer* olarak da anılan bu sensör, bağlı olduğu cihaz kapsamındaki ivmeyi algılayarak, yerçekimi düzeyini ya da hızlanma, durma gibi olayları ölçebilmektedir.
- Jiroskop:** Üzerinde bulunduğu cihazın X-Y-Z eksenlerindeki açısal konumunu ölçerek, açısal hızı tespit edebilen bir sensördür. Bu yönyle jiroskop, ivmeölçerin sadece ivme ölçmeye dayanan kabiliyetini üç eksenli düzeye taşımakta ve açısal hız tes-
- Manyetometre:** Adından da tahmin edileceği üzere, çevredeki manyetik alan yoğunluğunu ve yönünü ölçebilen bir sensördür. İngilizce *Magnetometer* olarak ifade edilmektedir.
- Yönlendirme Sensörü:** Mobil cihazın yatay-dik keyfli kullanım değişikliklerinin tespitini sağlayan sensördür. Yönlendirme sensörü İngilizce *orientation sensor* olarak da bilinmektedir.



Şekil 7.2 Mobil Cihazlardaki İvmeölçer ve Jiroskop Sensörlerinin Algılama Mekanizmaları.

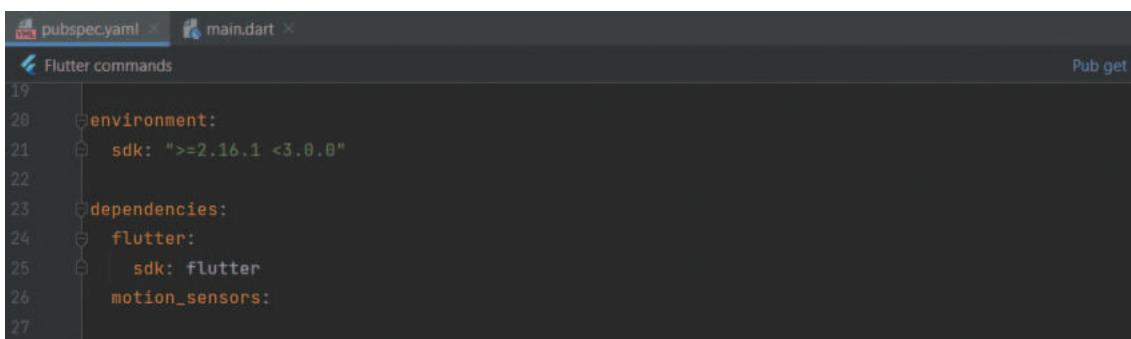
Kaynak: https://www.researchgate.net/publication/331320409_Physical_Activity_Recognition_by_Utilising_Smartphone_Sensor_Signals

Android Studio ortamında, Flutter'dan destek alarak ve Dart programlama dilini kullanarak, ilgili sensörleri içeren uygulamaları şu şekilde geliştirebiliriz:



Flutter Ortamında Sensör Kullanımı

Flutter ortamında farklı sensörler için uygulama kodlamayı sağlayan birçok farklı eklenti bulunmaktadır. Bunlar arasında İvmeölçer, Jiroskop, Manyetometre ve Yönlendirme Sensörünün dördünü de destekleyen eklenti **motion_sensors** eklentisidir. İlgili sensörleri içeren uygulamalarda öncelikli olarak Flutter projesindeki **pubspec.yaml** dosyasına sensör kullanımını işaret eden **dependencies** meta bilgisini eklemek gerekmektedir. Bunu sağlamak için **dependencies**: satırı altına **motion_sensors**: ifadesi yazılır (Şekil 7.3).



Şekil 7.3 Sensörler İçin motion_sensors Eklentisinin Proje Ortamına Eklenmesi

Dependencies eklemesi sonrası **terminal** ekranında aşağıdaki komutu kullanarak, ya da Android Studio ortamında **pubspec.yaml** penceresinin sağ üst alanında beliren **pub get** linkine tıklayarak (Şekil 7.3) güncel eklenti paketlerinin (buradaki senaryoda motions_sensors eklentisinin) **flutter pub get** komutuyla kurulması sağlanır. Eklenti kurulumu sonrası projenin **flutter run** komutu ile rebuild edilmesi sağlanır. Söz konusu işlem sonrasında **dart** kod dosyası içerisinde motions_sensors eklentisinin *import* edilmesi suretiyle ilgili sensör **event** kod bütünlelerinin çağrıları yapılabilmektedir:

```
import 'package:motion_sensors/motion_sensors.dart';
```

Eklenti sayesinde **listen** fonksiyonları üzerinden farklı sensörlerle yönelik okumalar yapılmaktadır:

1. *Accelerometer.Listen*: Yer çekimini (*gravity*) dikkate almak suretiyle İvmeölçer sensörünün ölçüdüğü değerlerin okunmasını sağlar:

```
motionSensors.accelerometer.listen((AccelerometerEvent event) {
  print(event);
});
```

2. *UserAccelerometer.Listen*: Yerçekimini dikkate almadan, İvmeölçer sensörünün ölçümlerinin okunmasını sağlar. Bir bakıma *sadece kullanıcının cihaz üzerindeki etkilerini* okumak için kullanılmaktadır:

```
motionSensors.userAccelerometer.listen((UserAccelerometerEvent event) {
  print(event);
});
```

3. *Gyroscope.Listen*: Jiroskop sensörünün ölçüdüğü değerleri; yani cihazın dönüş durumunu okumak için kullanılır:

```
motionSensors.gyroscope.listen((GyroscopeEvent event) {
  print(event);
});
```

4. *Magnetometer.Listen*: Manyetometre sensörünün manyetik alanı algılamasına yönelik değerleri okumak için kullanılır:

```
motionSensors.magnetometer.listen((MagnetometerEvent event) {
  print(event);
});
```

5. *ScreenOrientation.Listen*: Yönlendirme Sensörü üzerinden cihazın açısal konumlanmasını (yatay-dik) okumak için kullanılır:

```
motionSensors.screenOrientation.listen((ScreenOrientationEvent event) {
  print(event);
});
```

Yönlendirme sensörü ile ilgili olarak hareketin başladığı noktaya göre **görece** ölçüm ya da kuzey yönünü dikkate alarak, **absolute (mutlak) yönlendirme** ölçümü de yapılmaktadır.

Örneğin, her bir sensörden okunan değerleri ekranda görüntüleyen aşağıdaki uygulama kodlarını **main.dart** dosyası içerisine yazarak çalıştırabilirsiniz. Buna göre öncelikli olarak **import** çağrıları, ana program çağrısı, **widget** çağrıları ve sensör değerlerinin tutulacağı değişkenlerin tanımlanması yapılır:

```
import 'package:flutter/material.dart';
import 'package:vector_math/vector_math_64.dart' hide Colors;
import 'package:motion_sensors/motion_sensors.dart';

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

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  Vector3 _accelerometer = Vector3.zero();
  Vector3 _gyroscope = Vector3.zero();
  Vector3 _magnetometer = Vector3.zero();
  Vector3 _userAccelerometer = Vector3.zero();
  Vector3 _orientation = Vector3.zero();
  Vector3 _absoluteOrientation = Vector3.zero();
  Vector3 _absoluteOrientation2 = Vector3.zero();
  double? _screenOrientation = 0;

  int? _groupValue = 0;
```

Ardından her sensörün **listen** event blokları üzerinden okunmalarını sağlayan kodlar yazılır:

```
@override
void initState() {
  super.initState();
  motionSensors.gyroscope.listen((GyroscopeEvent event) {
    setState(() {
      _gyroscope.setValues(event.x, event.y, event.z);
    });
  });
  motionSensors.accelerometer.listen((AccelerometerEvent event) {
    setState(() {
      _accelerometer.setValues(event.x, event.y, event.z);
    });
  });
  motionSensors.userAccelerometer.listen((UserAccelerometerEvent event) {
    setState(() {
      _userAccelerometer.setValues(event.x, event.y, event.z);
    });
  });
  motionSensors.magnetometer.listen((MagnetometerEvent event) {
    setState(() {
      _magnetometer.setValues(event.x, event.y, event.z);
      var matrix = motionSensors.getRotationMatrix(_accelerometer, _magnetometer);
    });
  });
}
```

```
    _absoluteOrientation2.setFrom(motionSensors.getOrientation(matrix));
});
});

motionSensors.isOrientationAvailable().then((available) {
  if (available) {
    motionSensors.orientation.listen((OrientationEvent event) {
      setState(() {
        _orientation.setValues(event.yaw, event.pitch, event.roll);
      });
    });
  }
});
motionSensors.absoluteOrientation.listen((AbsoluteOrientationEvent event) {
  setState(() {
    absoluteOrientation.setValues(event.yaw, event.pitch, event.roll);
  });
});
};

motionSensors.screenOrientation.listen((ScreenOrientationEvent event) {
  setState(() {
    _screenOrientation = event.angle;
  });
});
});
```

Bu uygulama örneğinde değerlerin güncellenmesi için farklı saniye bazlı seçenekler (Frame Per Second: FPS cinsinden) sunulmaktadır. Bunun için değerlerin güncellemelerini sağlayan kodlar eklenir:

```
void setUpdateInterval(int? groupValue, int interval) {
    motionSensors.accelerometerUpdateInterval = interval;
    motionSensors.userAccelerometerUpdateInterval = interval;
    motionSensors.gyroscopeUpdateInterval = interval;
    motionSensors.magnetometerUpdateInterval = interval;
    motionSensors.orientationUpdateInterval = interval;
    motionSensors.absoluteOrientationUpdateInterval = interval;
    setState(0) {
        _groupValue = groupValue;
    });
}
```

Son olarak çeşitli **widget** bileşenleri yardımıyla sensörlerin anlık değer okumaları arayüzü yansıtılır:

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: const Text('Sensörlerden Değer Okuma'),
      ),
      body: SingleChildScrollView(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('Güncelleme (FPS)'),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Radio(
                  value: 1,
                  groupValue: groupValue,
```

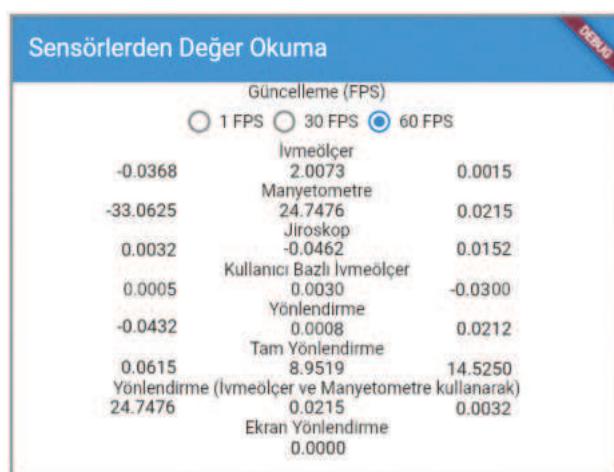
```
onChanged: (dynamic value) => setUpdateInterval(value, Duration.microsecondsPerSecond ~/ 1),
),
Text("1 FPS"),
Radio(
  value: 2,
  groupValue: _groupValue,
  onChanged: (dynamic value) => setUpdateInterval(value, Duration.microsecondsPerSecond ~/ 30),
),
Text("30 FPS"),
Radio(
  value: 3,
  groupValue: _groupValue,
  onChanged: (dynamic value) => setUpdateInterval(value, Duration.microsecondsPerSecond ~/ 60),
),
Text("60 FPS"),
],
),
Text('İvmeölçer'),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Text('${_accelerometer.x.toStringAsFixed(4)}'),
    Text('${_accelerometer.y.toStringAsFixed(4)}'),
    Text('${_accelerometer.z.toStringAsFixed(4)}'),
  ],
),
Text('Manyetometre'),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Text('${_magnetometer.x.toStringAsFixed(4)}'),
    Text('${_magnetometer.y.toStringAsFixed(4)}'),
    Text('${_magnetometer.z.toStringAsFixed(4)}'),
  ],
),
Text('Jiroskop'),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Text('${_gyroscope.x.toStringAsFixed(4)}'),
    Text('${_gyroscope.y.toStringAsFixed(4)}'),
    Text('${_gyroscope.z.toStringAsFixed(4)}'),
  ],
),
Text('Kullanıcı Bazlı İvmeölçer'),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Text('${_userAccelerometer.x.toStringAsFixed(4)}'),
    Text('${_userAccelerometer.y.toStringAsFixed(4)}'),
    Text('${_userAccelerometer.z.toStringAsFixed(4)}'),
  ],
),
Text('Yönlendirme'),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Text('${degrees(_orientation.x).toStringAsFixed(4)})'),
    Text('${degrees(_orientation.y).toStringAsFixed(4)})'),
    Text('${degrees(_orientation.z).toStringAsFixed(4)})'),
  ],
)
```

```

    ],
),
Text('Tam Yönlendirme'),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Text('${degrees(_absoluteOrientation.x).toStringAsFixed(4)}'),
    Text('${degrees(_absoluteOrientation.y).toStringAsFixed(4)}'),
    Text('${degrees(_absoluteOrientation.z).toStringAsFixed(4)}'),
  ],
),
Text('Yönlendirme (İvmeölçer ve Manyetometre kullanarak)'),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Text('${degrees(_absoluteOrientation2.x).toStringAsFixed(4)}'),
    Text('${degrees(_absoluteOrientation2.y).toStringAsFixed(4)}'),
    Text('${degrees(_absoluteOrientation2.z).toStringAsFixed(4)}'),
  ],
),
Text('Ekran Yönlendirme'),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Text('${_screenOrientation!.toStringAsFixed(4)}'),
  ],
),
],
),
),
),
),
);
}
}

```

Uygulamanın çalıştırılmasıyla beraber farklı sensörlerden gelen ölçüm-tespit değerleri, seçilen yenileme seçenekine göre arayüze yansıtılmaktadır (Şekil 7.4).



Şekil 7.4 Sensör Uygulamasının Arayüz Görüntüsü.



dikkat

Sensörlerden elde edilen değerler üzerinde farklı değerlendirmeler (örneğin, şart-koşul kontrolleri, döngüler kullanmak, ek matematiksel hesaplamalar yapmak) yaparak, yalın ölçülmüş değerleri uygulama amacına yönlendirmek mümkündür. Bununla birlikte alternatif widget bileşenler de uygulama arayüzüne renk katabilecektir. Bu konuda kitabınızın önceki bölümle-rinde anlatılanları hatırlamanız önemlidir.



Araştırmalarla İlişkilendir

Mobil Uygulamaların Geleceği: Süper Uygulama

Akıllı telefonlar, özellikle yaşadığımız pandemi sürecinde hayatımızın vazgeçilmez bir parçası haline geldi. Milyarlarca insanın evde kaldığı bir yılda uygulama kullanım alışkanlıklarını da oldukça güçlendi. Son araştırmalara göre mobil telefon uygulamalarında harcanan ortalama süre, geçen iki yıla göre, yüzde 30 ile yüzde 45 arasında artmış. Bir başka verİYE göre, dünya genelinde günde ortalama 4 saatten fazla zamanımızı akıllı telefonlardaki uygulamalar ile geçiriyoruz.

Bu benzeri görülmemiş değişim, milyonlarca yeni kullanıcının dijital araçları kullanarak kamu hizmetlerinden, bankacılık işlemlerine kadar çeşitli işlemlere mobil cihazlardan eriştiği anlamına geliyor. Akıllı telefonlarımızda her gün onlarca farklı uygulamaya göz atıyoruz. Bazen sosyal medyada dakikalarca sayfaları dolaşıyor, bazen bankadan para gönderiyor, bazen de arkadaşlarımıza yazışıyoruz. Anlık veya dakikalar sürebilen çok farklı ihtiyaçlar ve etkileşimler için akıllı telefonlarımızda kullandığımız bu hizmetler artık tek bir uygulamada sunulmaya başlandı. Birçok işlevi bir çatı altında toplayan bu tür uygulamalara süper uygulama yani SuperApp adı verilmektedir.

Süper uygulamaların mantığı birbiriley bağlı ve bazen de bağlantısız ürün ve servislerin tek uygulama üzerinden kullanıcılar sunulmasına dayanıyor. Böylece aynı uygulama içinden farklı uygulamalara tek bir kimlik giriş ile giriş yapıyor, farklı ürün ve hizmetleri kullanıyor ve tek cüzdana ödemelerini gerçekleştirebiliyoruz. Bir diğer deyişle, mobil ekranlardaki deneyimimizi hızlı, kolay ve basit kılıyoruz.

Tüm ihtiyaçları karşılayan tek bir uygulama

Süper uygulamalar, ödeme ve kimlik yönetimi dahil olmak üzere birçok işlemin tek bir uygulama üzerinden entegrasyonunu sağlar, daha da önemlisi bu hizmetleri ihtiyaçımıza göre özelleştirilebilir. Süper uygulamaların kullanıcıya sağladığı birçok avantajı bulunmaktadır. Bunlardan bazıları:

- Günlük kullanılan hizmetlere (toplu taşıma, para transferi, ödemeler, mesajlaşma vs.) tek bir ekran üzerinden ulaşabilmek

- Farklı uygulamalardaki değişen kullanıcı deneyimi yerine aynı ekranlar üzerinden ulaşmanın getirdiği kullanım kolaylığı
- Farklı uygulamalara giriş için gerekli olan kayıt (onboarding) süreci yerine tek bir uygulamada sadece bir kere kayıt olduktan sonra platformdaki tüm uygulamalara anında erişim
- Finansal işlem yada ödeme gerektiren durumlarda yeni bir uygulama yerine süper uygulama bünyesinde tanımlanmış ödeme yöntemi ile ödeme deneyimi
- Birden fazla mobil uygulama kullanımı ile tamamlanabilecek seyahat organizasyonu gibi bazı ilişkili işlerin tek aşamada, bir kaç gerçekleştirilebilmesi

Süper uygulamaların giderek daha popüler olmalarının ardından, kullanıcı deneyiminin artan önemi yatiyor. Bu dalda yer almak isteyen bir oyuncunun değeri finansal varlıklarının yanı sıra sahip olduğu ekosistemin değeri ile ölçülüyor. Kurdukları ekosistemi kullanıcı deneyimi içerisinde konumlamayı başaran Süper Uygulamalar geleceğin parlayan yıldızları olacaklar. Bu dalgayı yakalamak isteyen kurumların iş modellerini reize etmeye, yatırımlarını artırmaya ve yeniliklerin peşine düşmeye ihtiyacı var.

Süper uygulamalarda lider Asya

Bu alanda en bilinen uygulama, Çin'de belki de tüm akıllı telefonlara yüklenmiş olan WeChat. İlk olarak 2011 yılında bir mesajlaşma uygulaması olarak tüketicilere sunulan WeChat, bugün 1 milyarı aşan kullanıcı sayısıyla mesajlaşmanın yanı sıra kapsamlı bir sosyal paylaşım, iletişim, oyun uygulaması ve ödeme aracı olarak öne çıkmıyor. Bu geniş kapsamlı özelliklerine ek olarak diğer sosyal medya içeriklerini de sunarak tam anlamlı bir süper uygulama özelliği taşıyor.

WeChat'in başarı hikâyесini örnek alan ve benzer deneyimleri sunan süper uygulamalar birbiri ardına kullanıma sunuldu. Alipay, Grab ve Gojek bu kategorideki en bilinen diğer uygulamalar. Bugün batı ülkeleri hala süper uygulamaları tartışırken, Asyalı kullanıcılar mobil hayatını süper uygulamalar ile düzenliyor. Bu uygulama-

ların en yoğun kullanıldığı coğrafya hala Asya olmakla birlikte, dünya genelinde yükselen trend olduklarını söylemek yanlış olmaz.

Türkiye'de rekabet artıyor

Ülkemizde kullanıcılar her zaman yeni teknolojileri denemeye eğilimlidir. Yeni bir bilgisayar modeli, akıllı telefon satışa sunulduğunda en çok talep gören ülkelerde arasında yer alır. Geçmiş tecrübelerimizden biliyorum ki yeni teknoloji ve ödeme yöntemlerine karşı adaptasyonumuz her zaman diğer ülkelerden daha yüksek olmuştur.

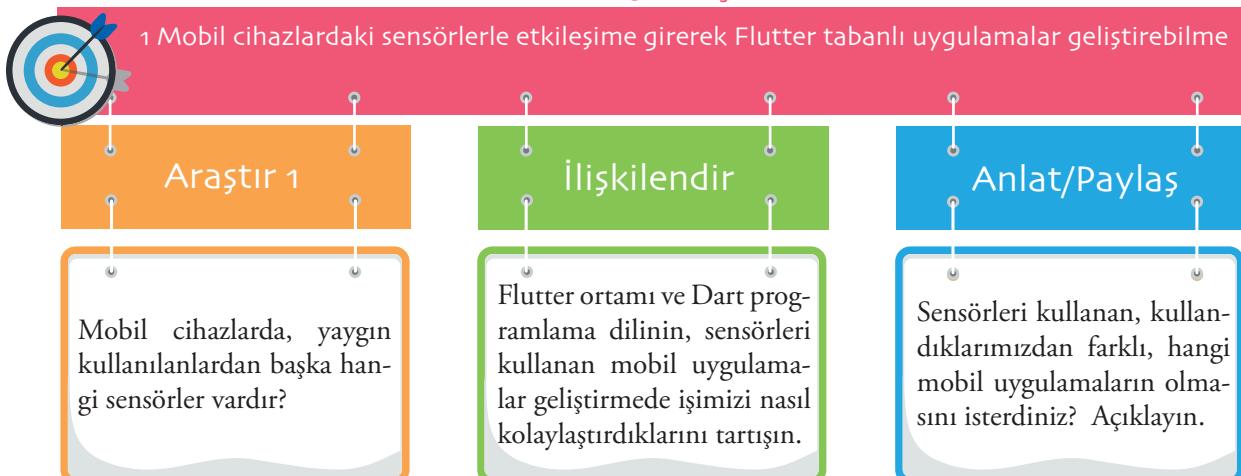
Ülkemizde finans, telekom ve perakende sektörleri yeni teknolojileri kullanmak ve sektörde rekabette öne geçmek adına her zaman yeniliklere açık olmuştur. Gelişmiş ülkelerdeki benzerlerinden çok daha erken teknolojik yenilikleri

sunan bu sektörler tüketiciden gelen ilgi ve kabul karşısında her zaman hazırlıklıdır.

Şimdilik kendi içinde ek özelliklerle mobil uygulamalarının kapsama alanını genişleten oyuncularının kullanıcı beklentilerini karşılamak üzere süper uygulama olma yolunda ayak seslerini duyuyoruz. Her sektörden köklü kurumların, yeni nesil dijital oyuncular ile rekabeti sayesinde süper uygulamalar ülkemizde kısa sürede hızla büyüyen bir iş modeli haline gelecektir. Lakin başarı en hızlı, en büyük, en ünlü markaların değil, kullanıcıya hızlı, kolay ve basit bir tecrübeyi en güvenli şekilde sunanların olacaktır.

Kaynak: Canko, S. (2021). Mobil Uygulamaların Geleceği: Süper Uygulama. Bloomberg.com. Yorum Yazları (2021).

Öğrenme Çıktısı

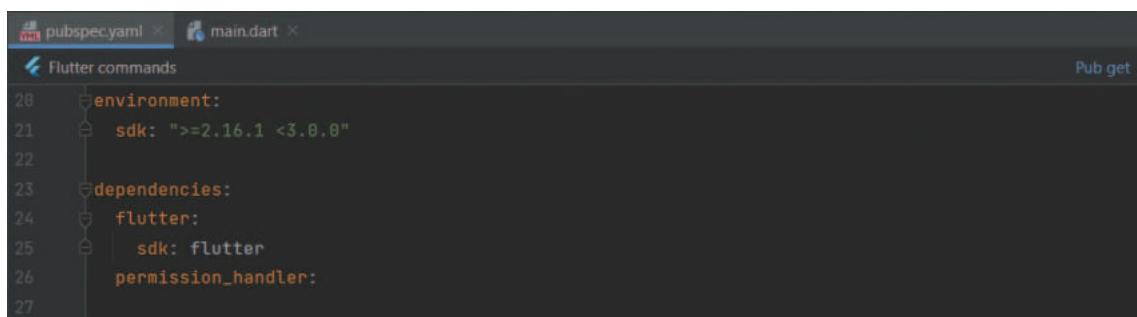


KAMERA KULLANAN UYGULAMALAR

Mobil uygulamalarda gerçek dünya ile görsel etkileşime girmenin en kolay yolu, cihaz üzerinde yer alan kameraları kullanmaktır. Kameralar sayesinde yine ilerleyen başlıklar altında deagineceğimiz Artırılmış Gerçeklik tabanlı uygulamalar da geliştirilebilmektedir. Ancak öncelikli olarak kameraların temel kullanım şekillerine ilişkin bazı ön bilgilere hakim olmak gerekmektedir.

Kamera Kullanım İzni

Flutter ortamında Dart programlama diliyle geliştirilen uygulamalarda kamera kullanım izinlerinin denetlenmesi için **permission_handler** eklentisi kullanılabilir. Eklentiyi kullanabilmek adına projedeki **pubspec.yaml** dosyasına **dependencies**: satırı altına **permission_handler**: çağrıları yapılmaktadır (Şekil 7.5).



Şekil 7.5 Kamera Kullanım İzni Eklentisinin Proje Ortamına Eklenmesi

Tıpkı diğer eklentilerin eklenme sürecinde olduğu gibi, **yaml** dosyası altındaki çağrı sonrasında **terminal** ekranında **flutter pub get** komutunu kullanarak, ya da pencere sağ üst alanında beliren **pub get** linke tıklayarak (Şekil 7.5) güncel eklenti paketlerinin kurulması sağlanıp, **flutter run** komutu ile rebuild işlemi gerçekleştirilmektedir. Ardından **dart** dosyası içerisinde **permission_handler** eklentisinin import edilmesi suretiyle kamera izinleri kullanıma açılmaktadır:

```
import 'package:permission_handler/permission_handler.dart';
```

Uygulama içerisinde herhangi bir anda kamera kullanım izni durumunu kontrol etmek için **permission.camera.status** metodu kullanılabilir. Metod sayesinde elde edilen döntüler **undetermined**: *izin belirlenmemiş*, **granted**: *izin elde edilmiş*, **denied**: *izin reddedilmiş*, **restricted**: *izin kısıtlı elde edilmiş* veya **permanentlyDenied**: *izin kalıcı olarak reddedilmiş* durumlarından herhangi birisine tekabül etmektedir.

Kullanıcıdan izin talebinde bulunmak için **permission.camera.request()** komut çağrıları yapılmalıdır. Ardından kullanıcı dönütünü beklemek için **await Permission.camera.request().isGranted** komut bütünü kullanılabilir. Eğer kullanıcı kullanım iznini onaylarsa uygulamada kullanım izni alınmış kamera üzerinden işlemlere devam edilebilmektedir.

Söz konusu senaryoyu basit bir arayüz üzerinde uygulayabiliriz. Öncelikli olarak proje içerisindeki **AndroidManifest.xml** dosyası içerisinde *kamera kullanımı* yönünde şu etkiket eklenir:

```
<uses-permission android:name="android.permission.CAMERA" />
```

İlgili düzenleme sonrasında proje **dart** dosyası içerisinde uygulama MyApp widget altyapısı ve kamera izninin ön tespitine yönelik metodlar kodlanır:

```

import 'package:flutter/material.dart';
import 'package:permission_handler/permission_handler.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

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

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;
  String _status = 'Permission not granted';

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  Future<void> _requestPermission() async {
    var status = await Permission.camera.request();
    if (status.isGranted) {
      _status = 'Permission granted';
    } else {
      _status = 'Permission denied';
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('You have pushed the button this many times:'),
            Text(_counter.toString()),
            Text(_status),
            TextButton(
              onPressed: _requestPermission,
              child: Text('Request Permission'),
            ),
          ],
        ),
      ),
    );
  }
}
  
```

```

title: 'Kamera Izni Uygulaması',
theme: ThemeData(
  primarySwatch: Colors.blue,
  visualDensity: VisualDensity.adaptivePlatformDensity,
),
home: AppHome(),
);
}

@Override
void initState() {
  super.initState();
  WidgetsBinding.instance.addPostFrameCallback(onLayoutDone);
}

void onLayoutDone(Duration timeStamp) async {
  _permissionStatus = await Permission.camera.status;
  setState(() {});
}

```

Ardından arayüzde mevcut kullanım iznini belirten bileşenleri göstermek için şu komutlar yazılır:

```

@Override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Kamera Izni'),
    ),
    body: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: <Widget>[
          Text(
            '_permissionStatus',
            style: Theme.of(context).textTheme.headline5,
          ),
        ],
      ),
    );
}

```

Kullanıcının kamera izni verebilmesi için **RaisedButton** ile kullanım izni talebi tetiklenebilmektedir:

```

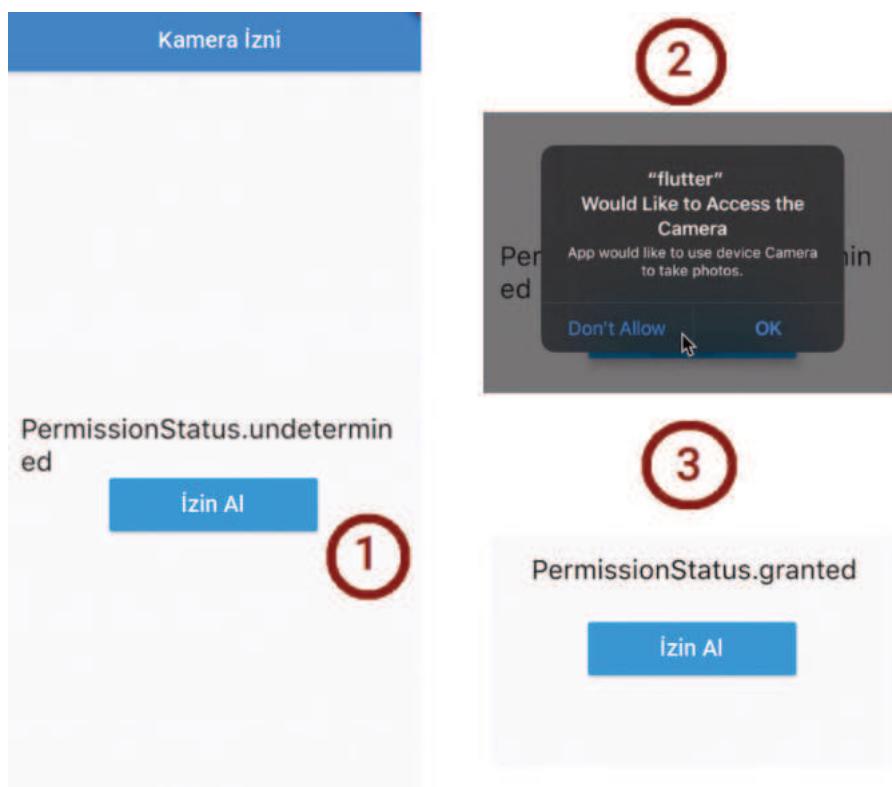
RaisedButton(
  child: Padding(
    padding: const EdgeInsets.symmetric(
      vertical: 14.0,
    ),
    child: Text(
      'İzin Al',
      style: TextStyle(
        color: Colors.white,
        fontSize: 16.0,
      ),
    ),
    color: Colors.lightBlue,
    onPressed: () {
      _askCameraPermission();
    },
)

```

RaisedButton tıklandıktan sonra kullanıcı izni alınması ve durumun güncellenmesi süreçleri kodlarak uygulama tamamlanmaktadır:

```
void _askCameraPermission() async {
  if (await Permission.camera.request().isGranted) {
    _permissionStatus = await Permission.camera.status;
    setState(() {});
  }
}
```

Uygulama çalıştırıldığında kameralın mevcut izin durumu **undetermined** durumunda ekranda görüntülenecektir. **İzin Al** düğmesine tıklandığında kullanıcıdan izin talep edilecektir. Kullanıcının izin vermesi durumunda kameralım izni **granted** olarak ekrana yansıtılacaktır (Şekil 7.6).

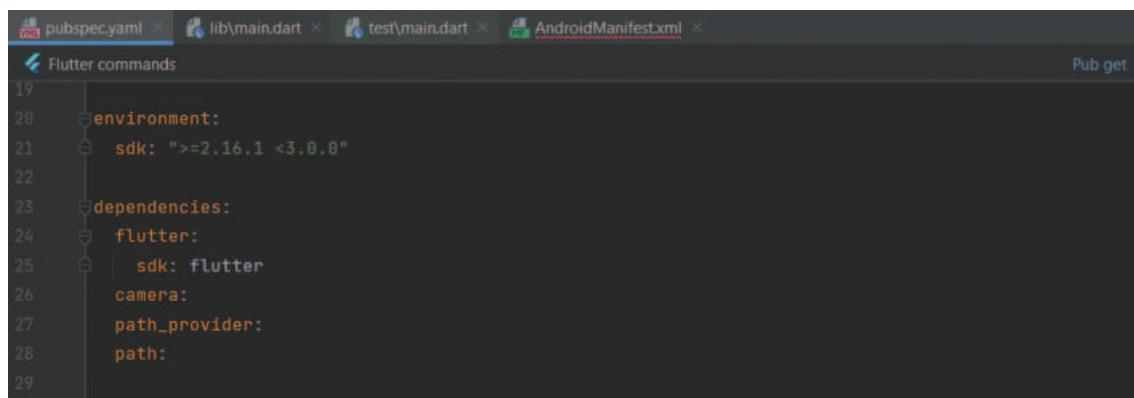


Şekil 7.6 Kamera Kullanım Izni Örnek Uygulaması.

Kamera ile Fotoğraf Çekimi

Flutter ve Dart işbirliği içerisinde geliştirilen uygulamanızda kamera üzerinden fotoğraf çekimi sağlayabilmek için **camera** eklentisine ihtiyacınız bulunmaktadır. Bu eklentiyi öncelikli olarak **pubspec.yaml** dosyasına **dependencies**: satırı altında ekleyip, önceki sayfalarda açıklanan güncelleme süreçlerini gerçekleştirmeniz gerekmektedir. Yine çekilen fotoğrafların mobil cihaz galerisine kaydedilebilmesi için **path_provider** ve **path** eklentileri de **yaml** dosyasına eklenmelidir (Şekil 7.7).





Şekil 7.7 Kamera ile Fotoğraf Çekimi İçin Gerekli Eklentilerin Tanımlanması.

Mobil uygulamamıza kamera fonksiyonlarını kullanma işlevi sağlayan **camera** eklentisi şu komut ile **dart** dosyası içerisinde *import* edilmelidir:

```
import 'package:camera/camera.dart';
```

Uygulamanın ilk kodları, kamera kontrolünü sağlayacak, **CameraController** sınıfından değişkenin ve diğer destekleyici değişkenlerin tanımlımıyla birlikte, kamera arayüzünün arka plana atılıp tekrar çağrılmaları durumunda çalışmaya devam etmesini sağlayacak metodların çağrılmaması üzerine yazılır:

```

CameraController _controller;
Future<void> _initializeControllerFuture;
isCameraReady = false;
showCapturedPhoto = false;
var ImagePath;
@Override
void initState() {
  super.initState();
  _initializeCamera();
}
Future<void> _initializeCamera() async {
  final cameras = await availableCameras();
  final firstCamera = cameras.first;
  _controller = CameraController(firstCamera, ResolutionPreset.high);
  _initializeControllerFuture = _controller.initialize();
  if (!mounted) {
    return;
  }
  setState(() {
    isCameraReady = true;
  });
}
_controller = CameraController(firstCamera, ResolutionPreset.high);
_initializeControllerFuture = _controller.initialize();

@Override

```

```
void devamEttir(AppLifecycleState state) {
    if (state == AppLifecycleState.resumed) {
        _controller != null
            ? _initializeControllerFuture = _controller.initialize()
            : null;
    } }
```

Kamera görüntüsünün uygulamaya yansması için **camera** eklentisinden gelen **CameraPreview** bileşeni kullanılarak ilgili metodlar kodlanır:

```
final size = MediaQuery.of(context).size;
final deviceRatio = size.width / size.height
FutureBuilder<void>(
future: _initializeControllerFuture,
builder: (context, snapshot) {
if (snapshot.connectionState == ConnectionState.done) {
return Transform.scale(
scale: _controller.value.aspectRatio / deviceRatio,
child: Center(
child: AspectRatio(
aspectRatio: _controller.value.aspectRatio,
child: CameraPreview(_controller), //kamera görüntüsü
),
));
} else {
return Center(
child:
CircularProgressIndicator());
} }, ),
```

Kamera görüntüsü aktif hale geldikten sonra çekim yapabilmek için bir **button** üzerinden **Controller** yardımıyla görüntü alma (**takePicture** metodu) ve görüntünün **png** formatında kaydedilmesi süreçleri işletilir:

```
void onCaptureButtonPressed() async {
try {

final path = join(
(await getTemporaryDirectory()).path,
'$pageStatus${DateTime.now().png',
);
goruntu = path;
await _controller.takePicture(path);

setState(() {
showCapturedPhoto = true;
});
} catch (e) {
print(e);
}
}
```

Çekilen görüntünün (fotoğrafın) ekranda sabit bir şekilde görüntülenmesi için **Image** widget bileşeni kullanılabilmektedir. Ayrıca ekranda görüntüleme sonrasında **controller** değişkeninin (nesnesinin) bellekten silinmesi için **dispose** metodu kullanılır:

```
Center(child: Image.file(File(goruntu)));
@override
void dispose() {
  _controller?.dispose();
  super.dispose();
}
```



ARTIRILMIŞ GERÇEKLIK UYGULAMALARI

Kullanıcıların mobil uygulamalar üzerinden sanal ve gerçek dünyayı aynı anda görüntüleyebilmeleri ve böylelikle etkileşim düzeylerini artırmaları, son yıllarda uygulama geliştirme süreçlerinde sıkılıkla tercih edilen bir yaklaşım olmuştur. Böylelikle *gerçek dünyayı mobil ortama taşıma* ve *sanal unsurlarla destekleme* mekanizmaları sayesinde kullanıcıların **daha anlamlı kullanım süreçleri** tecrübe etmeleri sağlanmaktadır.

Esasında kullanıcılar olarak *sanal dünya* ile etkileşim içerisinde girmemiz, **Sanal Gerçeklik (VR: Virtual Reality)** adı altında 20. Yüzyıl'daki gelişmelere kadar uzun bir geçmiş sahiptir (Schroeder, 1993). Ancak kullanıcı etkileşim fırsatlarını çeşitlendiren teknolojik gelişmeler, **Karma Gerçeklik (Mixed Reality)** adı verilen; sanal ve gerçek dünyanın aynı anda varolabildiği çözümlerinin de geliştirilmesini sağlamıştır (Bennford, & Giannachi, 2011). Sanal Gerçeklik özellikle son birkaç yılda, **Metaverse** oluşumuyla birlikte toplumlari ve hayatı dönüştüren bir karaktere bürünmüştür (Sparkes, 2021), Karma Gerçeklik uygulamaları da mobil cihazların donanım ve yazılım tabanlı kabiliyetleri sayesinde popüleritesini artırmaya devam etmiştir. Sadece mobil uygulamalarla sınırlı kalmayıp; TV ortamlarında bilgi aktarımı ve reklam sunumu süreçlerine kadar yayılan Karma Gerçeklik, temelde iki başlık altında incelenmektedir:

1. *Artırılmış Gerçeklik (AR: Augmented Reality)*: Gerçek dünyadaki görüntüye sanal unsurların dahil edildiği Karma Gerçeklik türüdür (Craig, 2013). Örneğin, TV ekranlarında spor karşılaşmaları izlerken, gerçek görüntü üzerine eklenmiş izlenimi veren sanal gösterimler Artırılmış Gerçeklik ile ilgilidir. Yine insanların gerçek dünyada *Pokemon* adı verilen *anime* karakterleri arayıp yakalaması üzerine kurulu olan *Pokemon Go* mobil oyunu da bu teknolojiyi kullanmaktadır.
2. *Artırılmış Sanallık (AV: Augmented Virtuality)*: Sanal görüntülenen ortama gerçek dünyadan unsurların dahil edildiği Karma Gerçeklik türüdür (Ternier vd., 2012). Artırılmış Gerçeklik'e göre daha ender olduğunu yorumlayabileceğimiz bu Artırılmış Sanallık'ın en iyi örnekleri, gerçek kişilerin çizgi karakterlerin dünyasına dahil olduğu, çizgi film / animasyon filmleridir.

Mobil uygulamalar düşünüldüğünde, özellikle Artırılmış Gerçeklik kullanımının daha çok rağbet gördüğünü ifade edebiliriz. Artırılmış Gerçeklik içeren uygulamalar sayesinde eğitimden turizme, ticari uygulamalardan video oyun dünyasına, hatta sağlığa kadar hayatın farklı alanlarında etkili uygulamalar geliştirilebilmektedir (Avila-Garzon vd., 2021; Aytekin vd., 2019; Eckert vd., 2019; Wetzel vd., 2011; Yung, & Khoo-Lattimore, 2019).

Özellikle bilgiyi kullanıcıya aktarma ve kullanıcıyı gerçek dünya ile sanal dünyadan birleştigi noktada daha yüksek etkileşim düzeyine ulaşırma amacıyla güven Artırılmış Gerçeklik ile mobil uygulamalar içerisinde kullanımını kolay ve bir o kadar etkili mekanizmalar eklenebilmektedir.



Şekil 7.8 Artırılmış Gerçeklik Uygulamaları Gerçek Dünyayı Sanal Süreçlerle Destekleyerek, Bilgiyi Anlamlandırma ve Kullanıcıları Etkileşime Dahil Etmede Yenilikçi Çözümler Ortaya Koymaktadır.

Yaşamla İlişkilendir

Gerçekliğin tanımını değiştiren teknoloji: Artırılmış Gerçeklik

Teknoloji dünyayı değiştirmekle kalmayıp, alternatif dünyalar da üretiyor. Bu, dünyaların alt yapısı ise artırılmış gerçeklikle sağlanıyor. Peki bu teknoloji nasıl ortaya çıktı, nereye gelindi ve nasıl bir geleceği olacak?

Henüz üretikleri, yaygın olarak kullanımında olmasa da Artırılmış Gerçekliğin (AR) var olduğu alanlar hızla artıyor. Metaverse evreninden oyun dünyasına, modadan askeri alana kadar pek çok sektör, bu teknolojinin nimetlerinden yararlan-



maya başladı. Peki artırılmış gerçeklik tam olarak nedir ve nasıl ortaya çıktı?

Artırılmış Gerçeklik en basit tanımlıyla, gerçek dünyadaki görüntülerin bilgisayar aracılığıyla üretilen; grafik, video, ses, GPS ile aynı ekranda birleştirilen teknoloji. 1990 yılında Boeing şirketi, montaj işçilerinin işlerini kolaylaştırmak için bir arayışa geçti ve bu arayış, Artırılmış Gerçekliğin ilk adının atılmasını sağladı.

Uçak kablolardan sokaklardaki Pokemon'lara...

Boeing'deki bilgisayar sistemlerindeki araştırmacı Thomas Caudell uçak üreten çalışanlar

icin bir gösterici sistemi üretti. Bu göstericileri takanlar, yapım aşamasındaki uçağın yüzeyleri üzerinde, kabloların nelerden geçeceğini sanal diyagramlar halinde görebildi.

1999'da Hirokazu Kato ve Mark Billinghurs açık kaynak kodlu yazılım ARToolKit'i kullanma sundu. Basılı işaretçilerin kullanımıyla bu yazılım artırılmış gerçekliğin kişisel bilgisayarlar üzerinden görüntülenmesini sağladı. Günümüzde web üzerinden sunulan Flash tabanlı birçok artırılmış gerçeklik uygulaması da ARToolKit kullanılarak yapılmıyor.

2000'lerde teknolojideki büyük gelişmeler artırılmış gerçeklik sistemlerinin kullanımını için önemli bir zemin hazırladı. Telefonların GPS ve kamera gibi özellikleri üzerinde barındırması, kablosuz ağların geliştirilmesi ve sonuç olarak akıllı telefonların ortaya çıkışları, bu teknolojinin günlük hayatı dahil olmasını sağladı.

Böylece uçak kablolarının yerini tespit etmek için başlatılan teknoloji, ünlü çizgi dizi Pokemon'ların sokakta yakalanabilmesine kadar uzandı. Microsoft, Google, Sony, Epson gibi teknoloji devleri de bu alanda yatırımlarını hızlandırmaya başladı.

Artırılmış Gerçeklik ve Sanal Gerçeklik

Artırılmış gerçeklik teknolojisinin en sık karşılaştırıldığı ve karıştırıldığı teknoloji sanal gerçeklik teknolojisi. Ancak iki teknoloji arasında derin farklar bulunuyor. Sanal gerçeklik, sanal bir dünyanın içerisinde gerçekten bulunuyormuş gibi hissetmemizi sağlayan dijital ve fiziksel elementlerin bütününden oluşuyor. Kullanıcılar çeşitli araç ve gereçlerle sanal ortamlara dahil olabiliyor. Sanal Gerçeklik gerçek ve hayalin birleştilmesini hedeflerken, artırılmış gerçeklik ise sanal nesnelerin gerçek görüntülerle harmanlanmasını amaçlıyor.

Artırılmış Gerçeklik ile neler değişiyor?

Daha önce bahsettiğimiz gibi Artırılmış Gerçeklik teknolojisi, neredeyse tüm hayatımızda değişiklikler yaratmaya başladı. Peki Artırılmış Gerçekliğin dokunduğu sektörler nasıl bir değişim geçirmeye başladı? İşte örnekler...

Eğitim: Artırılmış Gerçeklikle öğrenciler eğitimi daha etkileşimli ve interaktif şekilde öğrenebiliyor. Örneğin tarih dersleri sanal turlarla gerçekleştiriliyor ya da Fen Bilgisinde gezegenler gerçek zamanlı izlenebiliyor, hücrenin oluşumu anbean gözlemlenebiliyor.

Sağlık: Doktorlara ameliyatlarında en çok yardımcı olan teknolojilerden biri Artırılmış Gerçeklik. Sağlık çalışanları, kanserli hücreleri, tümörleri ve iç kanamayı bulabiliyor, komplikasyonlara bu yolla müdahale ediyor, hatta kilometrelerce öteden ameliyatlar gerçekleştirilebiliyor.

Emlak: Artık ev arayan birçok kişi, oturukları yerden kalkmadan kiralık ve satılık daireleri görebiliyor, evleri gezebiliyor. Kendisine uygun daireyi bulan kullanıcı, evini boyamayı düşündüğü rengi, boyayı satın dahi almadan duvarında görüp test edebiliyor. Evine uygun mobilya almak isteyen kullanıcılar, herhangi bir satın alma gerçekleştirmeden sadece telefonlarını kullanarak nerede nasıl duracağına bakabiliyor. Bu evdeki tüm eşyalar için de geçerli bir uygulama.

Moda: Alışveriş yapmak ve yeni kiyafetler almak mutluluk verici olsa da kiyafet denemek her zaman keyif vermiyor. Bazı markalar mağazalarında kiyafetleri giymeden kendi başımıza deneyimlememize izin veriyor. Bunun için bardak uygulamaya okutup aynaya bakmak yeterli oluyor.

Turizm: Hepimiz daha çok gezmek ve görmek istiyoruz. Ancak ne yazık ki bu hem bütçe hem de zaman açısından mümkün olmuyor. Ancak Artırılmış Gerçeklik uygulamaları, gerçekinin yerini alamasa da meraklımız giderecek kadar gezip görme imkanı sunuyor. Ayrıca bu yolla yapılan gezilerde gezilen yerler hakkında bilgi almak, önceki hallerini de görmek mümkün.

Yazarlar: Ali Burak Biber, & Naz İrem Üstündağ

Kaynak: TRT Haber Sitesi (21 Ocak 2022)
<https://www.trthaber.com/haber/dunya/gercekligin-tanimini-degistiren-teknoloji-artirilmiş-gercelik-647344.html>. **Erişim tarihi:** 25/02/2022

Flutter Ortamında Artırılmış Gerçeklik

Mobil uygulamalarınıza Artırılmış Gerçeklik entegre etmek için **arcore_flutter_plugin** eklentisi kullanılabilir. Daha önce anlatılan eklentilere benzer şekilde, eklentinin **pubspec.yaml** dosyasına **dependencies**: altında eklenmesi ve güncelleme süreçlerinin gerçekleştirilmesi önemlidir.

Söz konusu eklenti, **Google** firmasının sunduğu **ARCore** teknolojisinin kullanımını sağlamaktadır. Bu nedenle mobil uygulamada ARCore eklentisinin yüklü olup olmadığı kontrolü için **AndroidManifest.xml** dosyasına çeşitli etiketlerin eklenmesi gerekmektedir. Bu noktada iki farklı yaklaşımma göre: **AR Required** ya da **AR Optional** altında etiketler yazılmaktadır. AR Required etiket bütünü, mobil uygulamanın AR Core desteği olan cihazlara görünmesini sağlamaktadır:

```
<uses-permission android:name="android.permission.CAMERA"/>
<!-- Limits app visibility in the Google Play Store to ARCore supported
devices
(https://developers.google.com/ar/devices). -->
<uses-feature android:name="android.hardware.camera.ar" android:required="true"/>
<application ...>
    <!-- "AR Required" app, requires "Google Play Services for AR" (AR-
Core)
        to be installed, as the app does not include any non-AR fea-
tures. -->
    <meta-data android:name="com.google.ar.core" android:value="required" />
</application>
```

AR Optional ise AR Core desteği olmayan cihazlarda da seçimlik bir şekilde Artırılmış Gerçeklik süreçlerinin işletilebilmesini mümkün kılmaktadır:

```
<uses-permission android:name="android.permission.CAMERA" />
<application ...>
    <!-- "AR Optional" app, contains non-AR features that can be used when
        "Google Play Services for AR" (ARCore) is not available. -->
    <meta-data android:name="com.google.ar.core" android:value="optional" />
</application>
```

Uygulamanın **dart** dosyası içerisinde AR Core eklentisi *import* edilmek suretiyle Artırılmış Gerçeklik özellikleri çağrılmaktadır:

```
import 'package:arcore_flutter_plugin/arcore_flutter_plugin.dart';
```

AR Core sayesinde uygulama ortamına çeşitli üç boyutlu (3D / 3B) şekillerin dahil edilmesi oldukça kolaydır. Bunun için önce **arcore_flutter_plugin** ile birlikte ihtiyaç duyulan diğer eklentiler *import* edildikten sonra Artırılmış Gerçeklik süreçlerini uygulamada hayatı geçirmemizi sağlayan **ArCoreController** sınıfını çağırmamız gerekmektedir:



```

import 'package:flutter/material.dart';
import 'package:arcore_flutter_plugin/arcore_flutter_plugin.dart';
import 'package:vector_math/vector_math_64.dart' as vector;
class ARCore extends StatefulWidget {
  ARCore({Key key, this.title}) : super(key: key);
  final String title;
  @override
  _ARCoreState createState() => _ARCoreState();
}
class _ARCoreState extends State<ARCore> {
  ArCoreController arCoreController;

```

ArCoreController sınıfı `_onArCoreViewCreated` metodu altında çağrılmakta ve ardından farklı görsel unsurların eklenmesi bağımsız şekilde ekleme metodları altından gerçekleştirilebilmektedir:

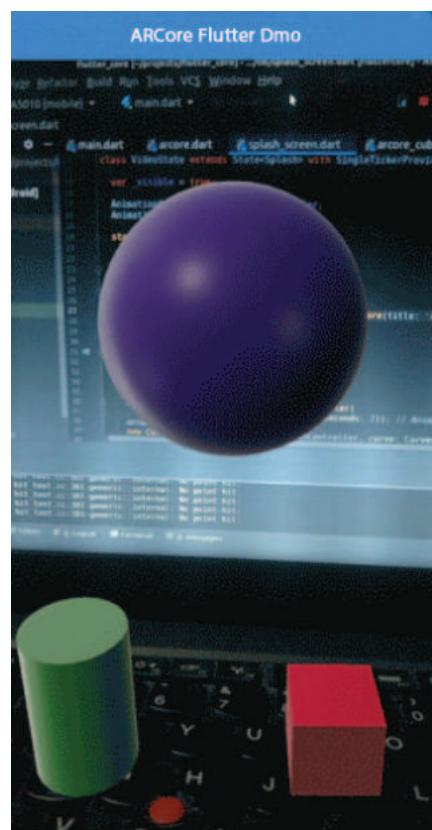
```

_onArCoreViewCreated(ArCoreController _arCoreController){
  arCoreController = _arCoreController;
  _addSphere(arCoreController);
  _addCube(arCoreController);
  _addCylinder(arCoreController);
}
_addSphere(ArCoreController _arCoreController) {
  final material = ArCoreMaterial(color: Colors.deepPurple, );
  final sphere = ArCoreSphere(materials: [material], radius: 0.2,);
  final node = ArCoreNode(
    shape: sphere,
    position: vector.Vector3(
      0, -0.3, -1
    ),
    );
  _arCoreController.addArCoreNode(node);
}
_addCylinder(ArCoreController _arCoreController) {
  final material = ArCoreMaterial(color: Colors.green, reflectance: 1);
  final cylinder =
  ArCoreCylinder(materials: [material], radius: 0.3,
  height: 0.1,);
  final node = ArCoreNode(
    shape: cylinder,
    position: vector.Vector3(
      -0.3, -1, -1.0
    ),
    );
  _arCoreController.addArCoreNode(node);
}
_addCube(ArCoreController _arCoreController) {
  final material = ArCoreMaterial(color: Colors.pink, metallic: 1);
  final cube = ArCoreCube(materials: [material], size: vector.Vector3(0.5, 0.5, 0.5));
  final node = ArCoreNode(
    shape: cube,
    position: vector.Vector3(
      0.5, -3, -3
    ),
    );
}
```

Her şekil çağrısında **ArCoreMaterial** kısmı şeklin renk ve materyal yapısını (örneğin, metalik) belirlemekte, **ArCoreNode** altında ise nihai şekil ve şeklin *konum* değerleri tanımlanabilmektedir. **ArCoreNode** kamera ile görüntülemede sanal nesnelerin yerleştirileceği konumları işaret etmektedir.

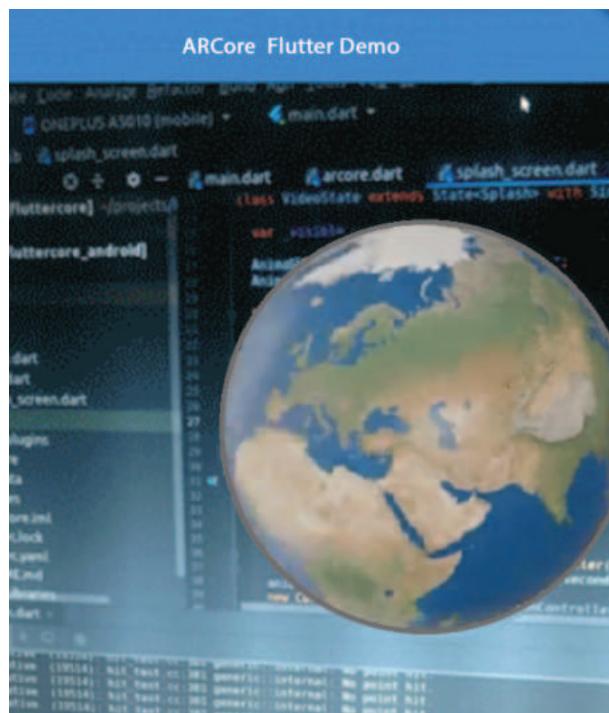
Şekil çağrılarının da eklenmesi akabinde **widget** çağrıları ve **body** altında **ArCoreView** çağrıları yapılmak suretiyle, görüntüleme ortamının dahil edildiği uygulama arayüzü tamamlanmaktadır:

```
    arCoreController.addArCoreNode(node);
}
@Override
void dispose() {
    arCoreController.dispose();
    super.dispose();
}
@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text(widget.title),
        ),
        body: ArCoreView(
            onArCoreViewCreated: _onArCoreViewCreated,
        ), ); }
```



Şekil 7.9 AR Core Tabanlı Uygulamanın Çalıştırılması

Uygulamada **ArCoreNode** altında imaj dosyalarının şekillere eklenmesi sayesinde spesifik nesneler de oluşturulabilmektedir. Bunun için **ArCoreMaterial** altında ikinci bir parameter ile imaj dosyasını tanımlamak yeterlidir. Örneğin, proje dosyaları içerisine eklenen bir dünya imaj dosyasını, mevcut **sphere** şekli kodlarında güncelleme yapmak suretiyle, Artırılmış Gerçeklik ortamına ekleyerek sanal bir yer küre nesnesi elde edebilirsiniz:



Şekil 7.10 Küre Şekli Üzerine Dünya İmaj Dosyasının Eklentimesi ile Yer Küre Nesnesinin Elde Edilmesi

```
_addSphere(ArCoreController _arCoreController) {
    final ByteData textureBytes = await rootBundle.load('image_dosyalar/earth.jpg');
    final material = ArCoreMaterial(
        color: Colors.deepPurple,
        textureBytes: textureBytes.buffer.asUint8List());
    final sphere = ArCoreSphere(materials: [material], radius: 0.2,);
    final node = ArCoreNode(
        shape: sphere,
        position: vector.Vector3(
            0, -0.3, -1
        ),
    );
    arCoreController.addArCoreNode(node);
}
```



Öğrenme Çıktısı



3 Artırılmış Gerçeklik teknolojisi kullanan Flutter tabanlı uygulamalar geliştirebilme

Araştır 3

Sanal Gerçeklik ve hatta Artırılmış Gerçeklik ile anılan Metaverse (Metaevren) kavramı / teknolojisi nedir?

İlişkilendir

Artırılmış Gerçeklik teknolojisini kullanmak suretiyle, eğlendirirken öğretme görevini de yerine getiren mevcut mobil uygulamaların neler olduğunu tartışın.

Anlat/Paylaş

Flutter ortamını ve Dart programlama dilini kullanmak suretiyle, geliştirebileceğinizi düşündüğünüz mobil uygulama fikirlerini açıklayın.

YAPAY ZEKA UYGULAMALARI

Yapay Zeka, hedef problemleri irdeleyip esnek çözümler üretebilen, gelişmiş yazılımlar geliştirilmesini mümkün kıyan, önemli bir bilim ve teknoloji alanıdır. Özellikle verilerle tanımlanmış, bilinen problem örüntülerini algılayıp, olası yeni durumlara tanımlayıcı (*descriptive*) ve tahmin edici (*predictive*) çözümler üretebilen Yapay Zeka algoritmaları (Hunt, 2014; Kelleher vd., 2020), gerek günümüzün gereksi geleceğin en önemli teknoloji araçları olarak bilinmektedir. Yapay Zeka tabanlı sistemler, çözülmesi zor ya da geleneksel yöntemlerle çözülememiş problemlerin üstesinden gelinmesi noktasında başarılı sonuçlar ortaya koymaktadır. Bu nedenle günümüz teknolojik ilerlemeleri, insanlardan bağımsız bir şekilde bağımsız muhakemelerde bulunan ve çözüm üreten *zeka araçlarının* tasarılanması-geliştirilmesi yönünde ilerlemektedir.



Şekil 7.11 Yapay Zeka destekli Yazılım Teknolojileri Sayesinde, Gündümüz Günlük Hayattaki Problemler İçin Zeki ve Esnek Çözümler Elde Edilebilmektedir.

Yapay Zekanın temelinde yer alan matematiksel ve mantıksal mekanizmalar, döngülerle desteklenmiş fonksiyonlar ve hedef verilere uyum sağlayarak ve esnek çıktılar üretmesini sağlayan değişken yapıları, Yapay Zeka tabanlı yazılımları standart yazılımlardan ayırmaktadır. *Bu açıdan yazılım teknolojisi Yapay Zeka sayesinde üst düzey atılımlara sahne olmuştur.* Yapay Zeka tabanlı yazılımlar esasında farklı özel algoritmaların amaca uygun bir şekilde kullanımını içermektedir. Bu noktada, en etkin ve yetenekli Yapay Zeka algoritmalarının **Makine Öğrenmesi** adı altında toplanan algoritmalar olduğunu söyleyebiliriz (Köse, 2021). Ancak gerek Kural tabanlı, gerekse modeller üzerinden zeki optimizasyona dayanan algoritmalar da güclü Yapay Zeka alanını şekillendiren diğer algoritmalar arasında yer almaktadır (Köse, 2017; Nabihev, 2005).

Makine Öğrenmesi: Çözülmlesi istenen problemin, problem ile alakalı verileri (örnekleri ya da bilinen senaryoları) dikkate almak suretiyle öğrenilmesini sağlayan Yapay Zeka yaklaşımıdır. Bu yaklaşım kapsamında, farklı öğrenme yöntemlerini takip eden birçok algoritma-teknik yer almaktadır.

Makine Öğrenmesi, Yapay Zekanın çeşitli örnekler ve senaryolar üzerinden (*eğitim verilerinden*) problem kapsamını öğrendikten sonra, aynı probleme ilgili yeni karşılaşılan durumlara etkin cevaplar üretme çabasını içermektedir. Makine Öğrenmesi, 2000'li yıllarda ortaya konulan daha gelişmiş teknikler sayesinde **Derin Öğrenme** adı altındaki bir alt-alanı da içermeye başlamıştır. Makine Öğrenmesi ile geliştirilen Yapay Zeka yazılımlarının böylelikle çözüm üretebildiği problem durumlarını şöyle örnekleleyebiliriz:

- Görüntülerden nesne tanımlama, tanımlanan nesnelerden durum analizi yapma
- Yüz tanıma, kişi tanıma, yüz tanıma üzerinden duyu tanıma
- Nedenlere göre sonuç tahmin etme (örneğin hastalık teşhis, makine arıza tespiti)
- Yeni veri türetme (örneğin anlamlı metinler, yeni görüntüler türetme, ilaç keşfetme)
- İnsanlara karar destek sağlama (örneğin tedavi planlama, işletmeler için stratejiler belirleme)

Derin Öğrenme: Hedef problem ile bağlantılı verileri daha detaylı (derinlemesine) irdeleyip, özellikle karmaşık düzeydeki, zorlu problemlerin daha başarılı bir biçimde çözülmесini sağlayan, ileri düzey algoritmaları teknikleri Makine Öğrenmesi alt-alanıdır.

Yapay Zekanın çözüm üretme süreci her zaman için ilgili algoritmaların kodlanması, kodlanan algoritmaların yazılım sistemlerinin oluşturulması (çok sayıda alt programın birleştirilmesi -tipki Flutter ortamında metodlarım / fonksiyonların tanımlanması gibi-) ve bu sistemlerin hedef donanımlarda (örneğin robotik sistemler, bilgisayarlar, mobil cihazlar) çalıştırılması yönünde organize edilmelidir.



Flutter Ortamında Yapay Zeka Çözümleri

Flutter ortamında, Dart programlama dili desteğiyle Yapay Zeka uygulamaları geliştirmek, -tipki diğer ileri düzey uygulamalarda olduğu gibi- çeşitli eklentilerle mümkün olmaktadır. Bunlar arasında en dikkat çekeni Google tarafından sunulan **Firebase ML Kit** (*Machine Learning Kit*) eklentisidir. Bu eklenti Yapay Zeka çözümlerini bir tür **API** (*Application Programming Interface: Uygulama Programlama Arayüzü*) üzerinden sunmaktadır. Eklentinin kullanılabilmesi için *Google Firebase* Web platformu üzerinden proje oluşturmak gerekmektedir. Bunun için *create project* seçeneğine tıkladıktan sonra *proje adı* ve *Google Analytics* seçeneklerinin (kapalı seçilmesi durumunda iki adımda) belirlenmesi suretiyle Flutter uygulamasının iletişim kuracağı API için proje açılışı yapılmalıdır (Şekil 7.12).

Proje platformu açıldıktan sonra Android uygulama seçeneği seçili, uygulama isminin *kayıt altına alınması* (*register*) gerekmektedir. Kayıt sonrası Flutter ortamındaki projeye sunulan *JSON* dosyasının ve *build* kodlarının (*build.gradle* dosyasına) eklenmesi gerekmektedir (Şekil 7.13).

Firebase ML Kit taraflı düzenlemeler sonrasında Flutter ortamında projenin **pubspec.yaml** dosyası altındaki **dependencies** kapsamında **mlkit**: ibaresi eklenir ve uygulama **dart** dosyasında **import 'package:mlkit/mlkit'**; kodu ile eklentinin çağrıları yapılır.



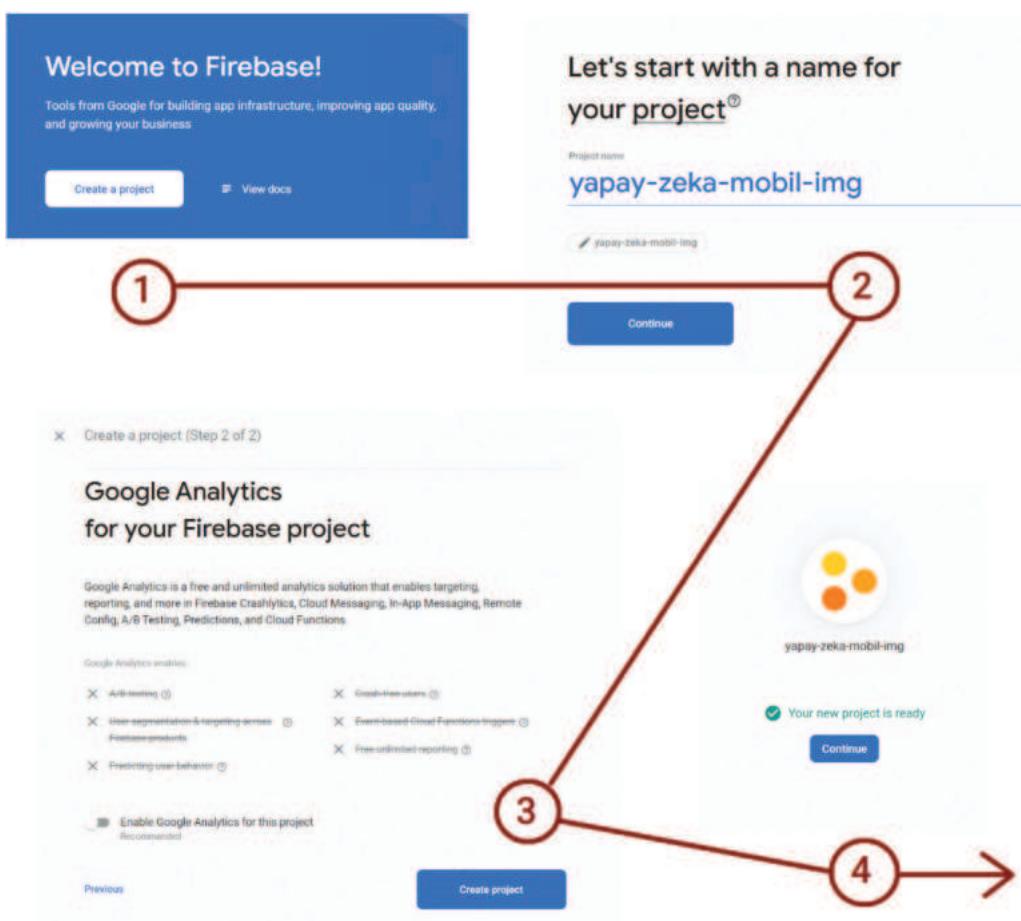
dikkat

Yapay Zeka, kapsamı oldukça geniş bir konudur. Yapay Zekayı daha iyi anlamak için kullanılan yöntemlerden (*sınıflandırma, regresyon, kümeleme, zeki optimizasyon... vb.*) algoritmalarla (örneğin *karar ağaçları, yapay sinir ağları, genetik algoritma gibi zeki optimizasyon teknikleri*) kadar birçok farklı konuyu öğrenmek gerekmektedir.



Google Firebase Web sitesine firebase.google.com adresinden ulaşabilirsiniz.

API: Bir yazılım içerisinde tanımlanan fonksiyonların başka yazılımlar tarafından kullanılması için hazırlanmış iletişim alt-yapısıdır.



Şekil 7.12 Google Firebase Ortamında Proje Açılışının Gerçekleştirilmesi



Şekil 7.13 JSON Dosyası ve Build Script Düzenlemeleri

Firebase ML Kit ile etkileşim sağlayarak metin tanıma, nesne tanıma, yüz tanıma ve doğal dil işleme yönelik farklı uygulamalar geliştirilebilmektedir. Örneğin, mobil cihaz ortamındaki görsellerde nesne tespiti yapan uygulama için ön tanımlamalar yapılrken **FirebaseVisionLabelDetector** kullanılır:

```
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';
import 'package:mlkit/mlkit.dart';
import 'dart:async';

void main() => runApp(new MyApp());

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => new _MyAppState();
}

class _MyAppState extends State<MyApp> {
  File _file;
  List<VisionLabel> _currentLabels = <VisionLabel>[];
  FirebaseVisionLabelDetector detector = FirebaseVisionLabelDetector.instance;

  @override
  initState() {
    super.initState();
  }
}
```

İlgili tanımlamalar sonrasında uygulama arayüzünde **FloatingActionButton** kullanmak suretiyle görselin çağrılması ve çağrılan görseldeki tespitin Firebase ML Kit üzerinden yapılmasını takiben, tespit edilen nesnelerin isimlerini (**label**) ve doğruluk (*güven*) değerlerini (**confidence**) belirten döntülerin ilgili **widget** bileşenleri yardımıyla elde edilmesi sağlanır:

```
@override
Widget build(BuildContext context) {
  return new MaterialApp(
    home: new Scaffold(
      appBar: new AppBar(
        title: new Text('Nesne Tespit'),
      ),
      body: _buildBody(_file),
      floatingActionButton: new FloatingActionButton(
        onPressed: () async {
          try {
            var file =
              await ImagePicker.pickImage(source: ImageSource.gallery);
            setState(() {
              _file = file;
            });
            var currentLabels =
              await detector.detectFromBinary(_file?.readAsBytesSync());
            setState(() {
              currentLabels = currentLabels;
            });
          } catch (e) {
            print(e.toString());
          }
        },
        child: new Icon(Icons.select_all),
      ),
    ),
  );
}
```

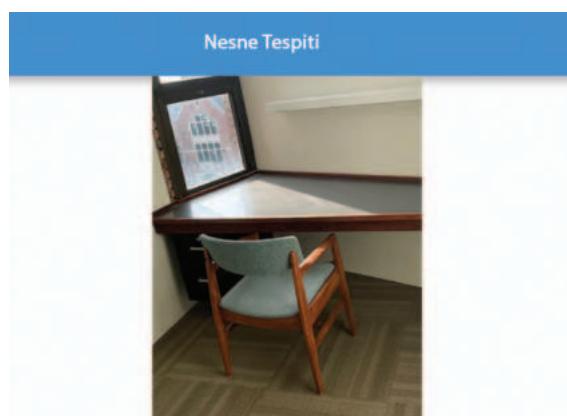
```

Widget _buildBody(File _file) {
    return new Container(
        child: new Column(
            children: <Widget>[displaySelectedFile(_file), _buildList(_currentLabels)],
        ), );
}

Widget _buildList(List<VisionLabel> labels) {
    if (labels == null || labels.length == 0) {
        return new Text('Boş', textAlign: TextAlign.center);
    }
    return new Expanded(
        child: new Container(
            child: new ListView.builder(
                padding: const EdgeInsets.all(1.0),
                itemCount: labels.length,
                itemBuilder: (context, i) {
                    return _buildRow(labels[i].label, labels[i].confidence.toString());
                },
            ), );
}

Widget displaySelectedFile(File file) {
    return new SizedBox(
        width: 150.0,
        child: file == null
            ? new Text('Seçili öğe yok.')
            : new Image.file(file),
    );
}

Widget _buildRow(String label, String confidence, String entityID) {
    return new ListTile(
        title: new Text(
            "\nTespit: $label\nDoğruluk: $confidence",
        ),
        dense: true,
    ); } }
```



Etiket: Chair
Doğruluk: 0.7723041033801630

Etiket: Room
Doğruluk: 0.7030221058977400

Etiket: Window
Doğruluk: 0.6230014110211078

Etiket: Table
Doğruluk: 0.6110388702016390



Şekil 7.14 Yapay Zeka Tabanlı Nesne Tespiti Uygulaması



Öğrenme Çıktısı

4 Yapay Zeka içeren Flutter tabanlı uygulamalar geliştirebilme

Araştır 4

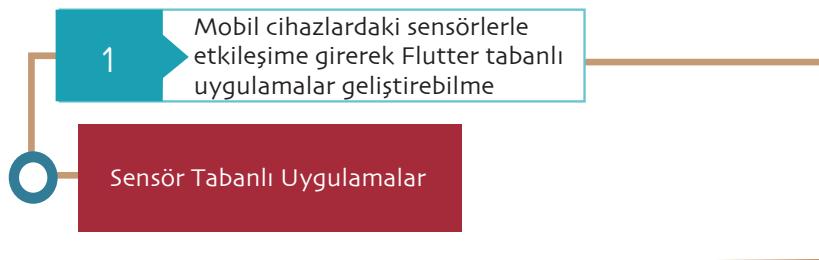
Mobil cihazlarda ve uygulamalarda yaygın görülen Yapay Zeka çözümleri nelerdir?

İlişkilendir

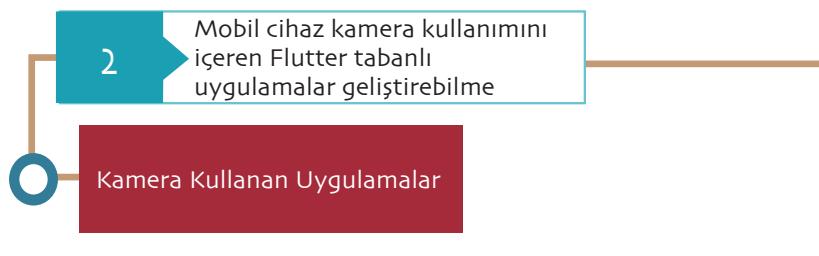
Benzer çözümler için Yapay Zeka kullanan mobil uygulamaları avantaj ve dezavantajları bakımından karşılaştırın.

Anlat/Paylaş

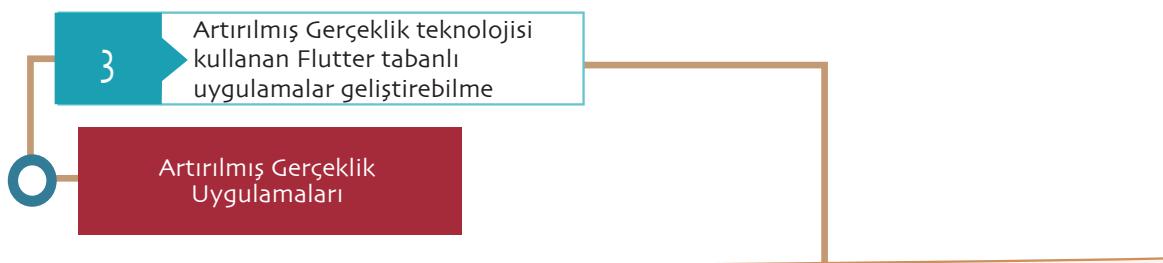
Gelecekteki mobil uygulamalarda ihtiyaç duyulabilecek Yapay Zeka kullanım senaryolarını, nedenleriyle birlikte açıklayın.



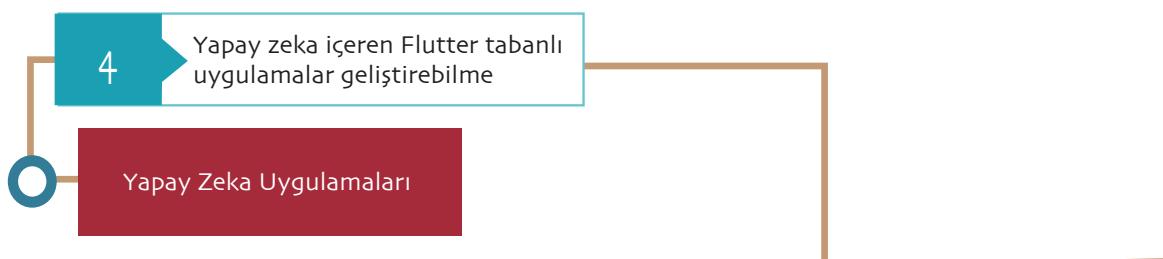
Sensörler, güncel mobil teknolojilerin vazgeçilmez bileşenleri arasında yer almaktadır. Mobil taşınırlığı sağlamak ya da cihazı giyilebilir bir biçimde kullanmak adına önceki geliştirilen mobil cihazlarda yazılım kabiliyetlerinin artırılması sensörler sayesinde mümkün olmaktadır. Sensörler, çevre ortamındaki ışık, ses, basınç gibi unsurları algılayan birçok farklı makinede / cihazda kullanılıyor olsa da, mobil teknolojiler sayesinde seviye atlamlı ve yenilikçi mobil uygulamaların doğmasını kolaylaştırmıştır.. Diğerlerine göre daha çok dikkat çeken, uygulamalara daha kritik mekanizmalar kazandıran en önemli sensörler şunlardır: *İvmeölçer*, bağlı olduğu cihaz kapsamındaki ivmeyi algılayarak, yerçekimi düzeyini ya da hızlanma, durma gibi olayları ölçmektedir. *Jiroskop*, üzerinde bulunduğu cihazın X-Y-Z eksenlerindeki açısal konumunu ölçerek, açısal hızı tespit etmektedir. *Manyetometre*, çevredeki manyetik alan yoğunluğunu ve yönünü ölçebilen bir sensördür. *Yönlendirme Sensörü*, mobil cihazın yatay-dikey kullanım değişikliklerinin tespitini sağlayan sensördür. Flutter ortamında farklı sensörler için uygulama kodlamayı sağlayan birçok farklı eklenti bulunmaktadır. Bunlar arasında *İvmeölçer*, *Jiroskop*, *Manyetometre* ve *Yönlendirme Sensörü*'nun dördünün de destekleyen eklenti *motion_sensors* eklentisidir. İlgili sensörleri içeren uygulamalarda öncelikli olarak Flutter projesindeki pubspec.yaml dosyasına sensör kullanımını işaret eden dependencies bilgisini eklemek gerekmektedir. İlgili eklenti, dart kod dosyası içerisinde `import 'package:motion_sensors/motion_sensors.dart'`; komutu ile çağrılmaktadır. Eklenti sayesinde *listen* fonksiyonları üzerinden sensör okumaları yapılabilmektedir.



Mobil uygulamalarda gerçek dünya ile görsel etkileşime girmenin en kolay yolu kameraları kullanmaktır. Flutter ortamında Dart programlama diliyle geliştirilen uygulamalarda kamera kullanım izinlerinin denetlenmesi için *permission_handler* eklentisi kullanılabilir. Eklentiyi kullanabilmek adına projedeki pubspec.yaml dosyasına dependencies: satırına `permission_handler`: çağrı yapılmaktadır Dart dosyası kapsamında *permission_handler* eklentisinin `import 'package:permission_handler/permission_handler.dart'`; komutuyla çağrıması ile kamera izinleri kullanıma açılmaktadır. Uygulama içerisinde herhangi bir anda kamera kullanım izni durumunu kontrol etmek için `permission.camera.status` metodunu kullanılabilmektedir. Kullanıcıdan izin talebinde bulunmak için `permission.camera.request()` komut çağrılmalıdır. Kullanıcı dönütünü beklemek için `await Permission.camera.request().isGranted` komut bütünü kullanılabilmekte, kullanıcı kullanım iznini onaylarsa uygulamada kullanım izni alınmış kamera üzerinden işlemlere devam edilebilmektedir. Uygulamalarda kamera üzerinden fotoğraf çekimi sağlamak için Flutter ve Dart programlama dili ortamına *camera* eklentisi kullanılmalıdır. Çekilen fotoğrafların mobil cihaz galerisine kaydedilebilmesi için *path_provider* ve *path* eklentileri de *yaml* dosyasına eklenmelidir. İlgili camera eklentisi dart dosyası ortamına `import 'package:camera/camera.dart'`; komutu ile çağrılmaktadır.



Kullanıcıların mobil uygulamalar üzerinden sanal ve gerçek dünyayı aynı anda görüntüleyebilmeleri ve böylelikle etkileşim düzeylerini artırmaları, son yıllarda uygulama geliştirme süreçlerinde sıkılıkla tercih edilen bir yaklaşım olmuştur. *Artırılmış Gerçeklik (AR: Augmented Reality)*, gerçek dünyadaki görüntüye sanal unsurların dahil edildiği Karma Gerçeklik türüdür. Artırılmış Gerçeklik içeren uygulamalar sayesinde eğitimden turizme, ticari uygulamalardan video oyun dünyasına, hatta sağlığa kadar hayatın farklı alanlarında etkili uygulamalar geliştirilebilmektedir. Flutter ve Dart tabanlı mobil uygulamalara Artırılmış Gerçeklik entegre etmek için *arcore_flutter_plugin* eklentisi kullanılabilir. Söz konusu eklenti, Google firmasının sunduğu ARCore teknolojisinin kullanımını sağlamaktadır. Uygulamanın dart dosyası içerisinde AR Core eklentisinin *import 'package:arcore_flutter_plugin/arcore_flutter_plugin.dart'*; komutuyla eklenmesi suretiyle Artırılmış Gerçeklik özellikleri çağrılmaktadır. AR Core sayesinde uygulama ortamına çeşitli üç boyutlu (3D / 3B) şekillerin dahil edilmesi oldukça kolaydır. Bunun için Artırılmış Gerçeklik süreçlerini uygulamada hayatı geçirmemizi sağlayan *ArCoreController* sınıfını çağırılmamız gerekmektedir. *ArCoreController* sınıfı *_onArCoreViewCreated* metodunu altında çağrılmakta ve ardından farklı şekillerin eklenmesi bağımsız şekil ekleme metodları altından gerçekleştirilebilmektedir. Kod ortamında her bir şekil çağrı metodunda *ArCoreMaterial* şeklin renk ve materyap yapısını tanımlamayı sağlamakta, *ArCoreNode* altında ise nihai şekil ve şeklin konum değerleri tanımlanabilmektedir. *ArCoreNode* bir bakıma kamera ile görüntülemede sanal nesnelerin yerleştirileceği konumlarla tekabül etmektedir. Şekil çağrılarının da eklenmesi akabinde *widget* çağrıları ve *body* altında *ArCoreView* çağrıları yapılmak suretiyle, Artırılmış Gerçeklik görüntüleme ortamının dahil edildiği uygulama arayüzü elde edilebilmektedir. *ArCoreNode* altında imaj dosyalarının şekillere eklenmesi sayesinde spesifik nesneler de oluşturulabilmektedir.



Yapay Zeka, hedef problemleri irdeleyip esnek çözümler üretebilen, gelişmiş yazılımlar geliştirilmesini mümkün kıyan, önemli bir bilim ve teknoloji alanıdır. Flutter ortamında, Dart programlama dili desteğiyle Yapay Zeka uygulamaları geliştirmek çeşitli eklentilerle mümkün olmaktadır. Bunlar arasında en dikkat çekeni Google tarafından sunulan *Firebase ML Kit* (Machine Learning Kit) eklentisidir. Eklentinin kullanılabilmesi için Google Firebase Web platformu üzerinden proje oluşturmak gerekmektedir. Bunun için *create project* seçeneğiyle Flutter uygulamasının iletişim kuracağı API için proje açılışı yapılmalıdır. Proje platformu açıldıktan sonra Android uygulama seçeneği seçilip, uygulama isminin kayıt altına alınması (*register*) gerekmektedir. Kayıt sonrası Flutter ortamındaki projeye sunulan *JSON* dosyasının ve build kodlarının (*build.gradle* dosyasına) eklenmesi gerekmektedir. Firebase ML Kit taraflı düzenlemeler sonrasında Flutter ortamında projenin *pubspec.yaml* dosyası altındaki *dependencies* kapsamında *mlkit:* ibaresi eklenir ve uygulama dart dosyasında *import 'package:mlkit/mlkit.dart'*; kodu ile eklentinin çağrıları yapılır. Firebase ML Kit ile etkileşim sağlayarak metin tanıma, nesne tanıma, yüz tanıma ve doğal dil işleme yönelik farklı uygulamalar geliştirilebilmektedir. Örneğin, mobil cihaz ortamındaki görsellerde nesne tespiti yapan uygulama için ön tanımlamalar yapılrken *FirebaseVisionLabelDetector* kullanılır. Bu bağlamda, görselin çağrılmaması ve çağrılan görseldeki tespitin Firebase ML Kit üzerinden yapılmasını takiben, tespit edilen nesnelerin isimlerini (*label*) ve doğruluk (*güven*) değerlerini (*confidence*) belirten döntütlerin ilgili widget bileşenler yardımıyla gösterilmesi sağlanabilmektedir.

1 Aşağıdakilerden hangisi, mobil uygulamalardaki kullanım tecrübeini etkileşim boyutunda artırmak için dikkate alınması gereken hususlardan biri **değildir**?

- A. Donanım-yazılım etkileşiminden faydalananmak
- B. Gerçek dünyayı mobil ortama taşımak
- C. İnsanların aynı cihazları kullanmasını sağlamak
- D. Kolay kullanım sağlamak
- E. İleri düzey veri işleme süreçleri işletmek

2 Aşağıdakilerden hangisi mobil cihazlardaki Jiroskop (Gyroscope) sensörünün ölçüm şeklini doğru bir biçimde tanımlamaktadır?

- A. Çevredeki manyetik alan yoğunluğunu ve yönünü ölçmektedir.
- B. Cihazdaki sıcaklık, nem gibi iklimsel değişiklikleri ölçmektedir.
- C. İvmeyi algılayarak, yerçekimi düzeyini ya da hızlanma, durma gibi olayları ölçmektedir.
- D. Cihazın X-Y-Z eksenlerindeki açısal konumunu ölçerek, açısal hızı tespit etmektedir.
- E. Cihazın yatay-dikey kullanım değişikliklerini tespit etmektedir.

3 Flutter ortamı ve Dart programlama dili kullanmak suretiyle geliştirilen sensör tabanlı mobil uygulamalar ve motion_sensors eklentisi ile ilgili olarak aşağıdakilerden hangisi **yanlıştır**?

- A. Motion_sensors kapsamında sensör okumaları yapmak için listen fonksiyonu kullanılabilir.
- B. Motion_sensors eklentisi ile mobil cihaz koymuşmasını tespit etmek mümkün değildir.
- C. Motion_sensors, sensör kontrolü için kullanılabilen tek eklenti değildir.
- D. Motion_sensors eklentisi kapsamında Manyetometre sensöründen gelen değerler okunabilemektedir.
- E. Motion_sensors eklentisi kullanılırken dependencies altında metadata tanımlaması yapılmalıdır.

4 Mobil uygulamalarda kamera kullanımı ile ilgili olarak aşağıdakilerden hangisi doğrudur?

- A. Kamera kullanımının mobil uygulamalar açısından herhangi bir avantajı yoktur.
- B. Flutter ortamı için herhangi bir kamera eklenmesi bulunmamaktadır.
- C. Android işletim sistemi, kamera kullanan uygulama geliştirmeye izin vermemektedir.
- D. Dart programlama dili ile kamera uygulamaları kodlamak mümkün değildir.
- E. Kamera kullanımı, uygulamalar nezdinde izne tabidir.

5 Flutter ortamında, Dart programlama dili kullanılarak, camera eklentisiyle geliştirilecek bir fotoğraf çekme uygulaması için aşağıdakilerden hangisi yanlış olacaktır?

- A. Kaydedilecek görüntünün dosya uzantısını belirtmeye izin verilmemektedir.
- B. Camera eklentisinin dependencies altına tanımlanması gerekmektedir.
- C. O andaki görüntünün fotoğrafını çekmek için takePicture metodunu kullanılabilmektedir.
- D. Çekilen fotoğrafların kaydedilebilmesi için path_provider ve path eklentileri de kullanılmalıdır.
- E. Kamera kullanımını sağlamak için CameraController sınıfı kullanılmaktadır.

6 Flutter ortamında Artırılmış Gerçeklik uygulamaları geliştirilmesini sağlayan ARCore eklentisi ile ilgili olarak aşağıdakilerden hangisi **yanlıştır**?

- A. ARCore eklentisi Google firması tarafından sunulmaktadır.
- B. Eklentinin kullanımı için ArCoreController sınıfının çağrılması gerekmektedir.
- C. Ekranda sanal unsurların oluşturulması için listen metodunu kullanılmaktadır.
- D. Flutter projelerinde arcore_flutter_plugin ismiyle tanımlanmaktadır.
- E. Görüntülemeler için ArCoreView çağrıları kullanılmaktadır.

7 "Kamera ile görüntülemede sanal nesnelerin yerleştirileceği konumları işaret etmektedir"

Yukarıdaki ifade ile tam olarak tanımlanan ARCore eklentisi bileşeni aşağıdakilerden hangisidir?

- A. ArCoreView
- B. ArCoreNode
- C. ArCamera
- D. ArShape
- E. ArCoreController

8 Flutter ortamında Yapay Zeka uygulamaları geliştirmek için kullanılan Firebase ML Kit eklentisiyle ilgili olarak aşağıdaki ifadelerden hangisi **yanlıştır**?

- A. Firebase ML Kit eklentisi sadece nesne tespiti uygulamaları yapılmasına izin vermektedir.
- B. Firebase ML Kit eklentisi Dart dilinde import 'package:mlkit/mlkit.dart'; koduyla çağrılır.
- C. Firebase ML Kit eklentisi dependencies altında mlkit: ibaresi ile tanımlanır.
- D. Firebase ML Kit eklentisi Google tarafından geliştirilmiştir.
- E. Firebase ML Kit eklentisinin Dart ortamında kullanımı için Firebase Web platformunda proje tanımı yapılmalıdır.

9 Firebase ML Kit eklentisiyle nesne tespiti yapmak için kullanılan bileşen aşağıdakilerden hangisidir?

- A. FirebaseObjectAnalyse
- B. FirebaseClassifier
- C. FirebaseVisionLabelDetector
- D. FirebaseObjectDetector
- E. FirebaseClassification

10 "Firebase ML Kit eklentisi Yapay Zeka çözümlerini bir tür ----- üzerinden sunmaktadır."

Yukarıda cümlede boş bırakılan yeri aşağıdakilerden hangisi doğru şekilde tamamlar?

- A. XML
- B. TEXT
- C. MDB
- D. API
- E. HTML

1. C	Yanıtınız yanlış ise “Giriş” konusunu yeniden gözden geçiriniz.	6. C	Yanıtınız yanlış ise “Artırılmış Gerçeklik Uygulamaları” konusunu yeniden gözden geçiriniz.
2. D	Yanıtınız yanlış ise “Sensör Tabanlı Uygulamalar” konusunu yeniden gözden geçiriniz.	7. B	Yanıtınız yanlış ise “Artırılmış Gerçeklik Uygulamaları” konusunu yeniden gözden geçiriniz.
3. B	Yanıtınız yanlış ise “Sensör Tabanlı Uygulamalar” konusunu yeniden gözden geçiriniz.	8. A	Yanıtınız yanlış ise “Yapay Zeka Uygulamaları” konusunu yeniden gözden geçiriniz.
4. E	Yanıtınız yanlış ise “Kamera Kullanan Uygulamalar” konusunu yeniden gözden geçiriniz.	9. C	Yanıtınız yanlış ise “Yapay Zeka Uygulamaları” konusunu yeniden gözden geçiriniz.
5. A	Yanıtınız yanlış ise “Kamera Kullanan Uygulamalar” konusunu yeniden gözden geçiriniz.	10. D	Yanıtınız yanlış ise “Yapay Zeka Uygulamaları” konusunu yeniden gözden geçiriniz.

7

Araştır Yanı
Anahtarı

Araştır 1

Günümüz mobil cihazlarında, -bölümde anılan sensörler haricinde- farklı görevlerde, alternatif sensörler de vardır. Örneğin dokunmatik ekranlar esasında özel bir sensör yapısı (*touch sensor*) sayesinde çalışmaktadır. Yine harita ve konum tespit uygulamaları *GPS sensörü* sayesinde çalışmaktadır. Yine cihazın yakınlarındaki nesnelerle etkileşimiini sağlayan yakınlık (*proximity*) sensörü, ışık (*light*) sensörü, sıcaklık (*temperature*), nem (*humidity*) ölçümleri yapan sensörler ve parmak izi (*fingerprint*) sensörleri de mobil cihazlarda görülebilen diğer sensörler arasındadır. Sensör çeşitliliği mobil cihazın boyutları kadar teknik yeterlikleriyle de ilişkilidir. Özellikle Nesnelerin İnterneti gibi teknolojilerin ivmeli bir şekilde yükselişine devam etmesi, gelecekte yeni sensör türlerinin ortaya çıkma ihtimali canlı tutmaktadır.

Araştır 2

Kameralar, mobil cihazlar ortamında dış ortamın görüntüsünü alma ihtiyaçları doğrultusunda ilk olarak 1999 yılında ortaya çıkmış ve günümüzde kadar hızlı gelişmeler göstermişlerdir. Önceleri mobil cihazların sadece arka yüzünde yer alan kameralar artık *görsel gürültüye* ihtiyaçlarına cevap vermek adına cihaz ön yüzlerine de yerleştirilmektedir. Dolayısıyla tipik bir mobil cihazda en az iki kamera kesinlikle bulunmaktadır. Bununla birlikte çekilen fotoğraflarda ve videolarda kaliteye olan ihtiyacın artması, kameralar içerisinde kullanılan merceklerin dikkatli seçilmesini ve yine otomatik odaklılama yapan mekanik mekanizmaların yer aldığı daha gelişmiş kameraların da tasarılanmasını gereklî kılmıştır. Ayrıca panaromik ya da üç boyutlu, özel efekt destekli görüntü çekimleri sağlamak için güncel cihazlarda birbirine yakın konumlanmış, çoklu kameralar da kullanılmaktadır. Çekim kalitesi anlamında incelediğinde ise ilk ortaya çıktığında 10-megapixel (MP) düzeyine bile erişemeyen mobil cihaz kameraları, 2008 yılından sonra bu sınırı aşmış, 2010 yılından sonra 50-MP sınırlarına doğru yol almaya başlamış ve 2018 yılından sonra 50-MP sınırını aşan kameralar da cihazlarda sıradan hale gelmeye başlamıştır. 2022 yılı itibarıyle kameraları 100-MP üzerinde geçen cihazlar vardır.

Araştır Yanıt Anahtarı

7

Araştır 3

Metaverse (Metaevren) kavramı, ilk olarak Neal Stephenson tarafından 1992 yılında yayınlanan Snow Crash romanındaki *sanal evren* yapısını tanımlamak için kullanılmıştır. Anlaşılacığı üzere normalde bilimkurgusal bir ölçekte ortaya konulan bu kavram, özellikle 21. Yüzyılda bilişim teknolojilerinde meydana gelen gelişmeler neticesinde bir teknoloji formuna kavuşmaya başlamıştır. Metaverse, Sanal Gerçeklik ve Artırılmış Gerçeklik gibi teknolojilerin kullanımı sayesinde insanların sanal bir dünyada varlıklarını ve faaliyetlerini sürdürmesi felsefesi üzerine kurulu durumdadır. Teknolojik gereklilikler, donanımsal bileşenlerin yanında üç boyutlu tasarlanan ortamları da gerekli kılmaktadır. Metaverse teknolojisinin bilinen sanal uygulamaların en önemli farkı, insanların gerçek dünyada yerine getirdikleri; başkallarıyla iletişim kurma, ticaret yapma, iş sahibi olma, ekonomik ve kültürel etkileşim içerisinde girme, mekanları ziyaret etme gibi süreçlerin Metaverse ortamında da mümkün olmasıdır. Bu ikinci yaşam faktörü, Metaverse teknolojisinin hızla popülerleşmesinde anahtar rol oynamıştır. Halihazırda birçok farklı Metaverse uygulaması yoğun bir şekilde kullanılmakta ve gelişimini sürdürmektedir. Uzun vadede Metaverse ortamlarıyla gerçek yaşam arasında daha güçlü bağların kurulacağı (örneğin, her iki tarafta da hukuki sonuçların birbirini etkilemesi, faaliyetlerin her iki dünyayı da etkilemesi...vs.) düşünülmektedir.

Araştır 4

Yapay zeka çözümleri, biz farkına varmasak bile mobil cihazların ve uygulamaların birçok farklı noktasında kullanılmaktadır. Örneğin, gelişmiş mobil cihazlarda mekanik ve elektronik bileşenlerin bulunan ortama göre konumlanması, cihaz kaynaklarının verimli ve etkin kullanılması gibi süreçlerde Yapay zeka algoritmalarının kabiliyetlerinden yararlanılmaktadır. Mobil uygulama boyutunda ise fotoğraf çekimlerinde kişi yüzlerinin tespiti, filtrelerin gerçekçi bir şekilde uygulanması, fotoğrafları işleyerek otomatik video klipler ve galeri organizasyonlarının sağlanması, özel yazılım tabanlı asistanlarla cihazların-uygulamaların sesli kontrol edilmesi Yapay zeka ile sağlanmaktadır. Bu tür çözümler sosyal medya uygulamalarını da desteklemekte hatta bu tür ortamlarda kişi önerileri, ilgi alanlarına yönelik profillerin, grupların, sayfaların tespiti gibi süreçler de Yapay Zeka ile şekillendirilmektedir. Diğer yandan, giyilebilir cihazlar üzerinden kişisel sağlık, spor, etkinlik süreçlerinin takibi aşamalarında Yapay zeka çözümleri kullanılmaktadır. Yine farklı mobil uygulamalar kullanıcıya özgü kullanım süreçleri tasarlama, analitik akıllı döntütlerde bulunma, paylaşım ve iletişim önerileri sunma gibi mekanizmaları Yapay Zeka sayesinde işe koşabilmektedir. Son olarak, kullanıcıların sanal karakterlere ve senaryolarla karşı mücadele verdiği mobil oyunlar da söz konusu mekanizmaları Yapay Zeka tabanlı yöntemler ve algoritmalar-tekniklere borçludur.

Kaynakça

- Agarwal, P. ve Bhattacharyya, O. (2018). Mobile technologies in healthcare: systematising the move from point solutions to broad strategies. *BMJ Quality & Safety*, 27(11), 865-867.
- Avila-Garzon, C., Bacca-Acosta, J., Duarte, J. ve Betancourt, J. (2021). Augmented Reality in Education: An Overview of Twenty-Five Years of Research. *Contemporary Educational Technology*, 13(3).
- Aytekin, P., Yakin, V. ve Çelik, B. H. (2019). Augmented Reality Technology in Marketing. *AJIT-e*, 10(39), 87.
- Benford, S. ve Giannachi, G. (2011). *Performing Mixed Reality*. MIT Press.
- Craig, A. B. (2013). *Understanding Augmented Reality: Concepts and Applications*. Newnes.
- Eckert, M., Volmerg, J. S. ve Friedrich, C. M. (2019). Augmented reality in medicine: systematic and bibliographic review. *JMIR mHealth and uHealth*, 7(4), e10967.
- Hunt, E. B. (2014). *Artificial Intelligence*. Academic Press.
- Keengwe, J. ve Bhargava, M. (2014). Mobile learning and integration of mobile technologies in education. *Education and Information Technologies*, 19(4), 737-746.
- Kelleher, J. D., Mac Namee, B. ve D'arcy, A. (2020). *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. MIT press.
- Köse, U. (2017). *Yapay Zeka Tabanlı Optimizasyon Algoritmaları Geliştirilmesi*. Doktora Tezi. Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği ABD.
- Köse, U. (2021). Yapay Zeka Kavramı. *Yapay Zeka Etiği*. (Ed. U. Köse). Nobel Akademik Yayıncılık.
- Meder, A. M. ve Wegner, J. R. (2015). iPads, mobile technologies, and communication applications: A survey of family wants, needs, and preferences. *Augmentative and Alternative Communication*, 31(1), 27-36.
- Nabiiev, V. V. (2005). *Yapay Zeka: Problemler-Yöntemler-Algoritmalar*. Seçkin Yayıncılık.
- Schroeder, R. (1993). Virtual reality in the real world: history, applications and projections. *Futures*, 25(9), 963-973.
- Shaikh, A. A., & Karjaluoto, H. (2015). Mobile banking adoption: A literature review. *Telematics and informatics*, 32(1), 129-142.
- Sparkes, M. (2021). What is a metaverse. *New Scientist*. 251(3348), 18.
- Ternier, S., Klemke, R., Kalz, M., Van Ulzen, P. ve Specht, M. (2012). ARLearn: augmented reality meets augmented virtuality. *Journal of Universal Computer Science-Technology for learning across physical and virtual spaces*, 18(15), 2143-2164.
- Wetzel, R., Blum, L., Broll, W. ve Oppermann, L. (2011). Designing mobile augmented reality games. In *Handbook of augmented reality* (pp. 513-539). Springer, New York, NY.
- Yung, R. ve Khoo-Lattimore, C. (2019). New realities: a systematic literature review on virtual reality and augmented reality in tourism research. *Current Issues in Tourism*, 22(17), 2056-2081.

Bölüm 8

Örnek Mobil Uygulama Geliştirme

Öğrenme Çıktıları

1 Firebase Veri Tabanının Kurulması

- 1 Firebase veri tabanı işlemlerini yapabilme

2 Telefon Defteri Uygulamasının Sayfalarını Tasarlama

- 2 Telefon defteri uygulaması için gerekli widgetleri oluşturabilme
- 3 Firebase'de kimlik doğrulama ve veri tabanı işlemlerini yapabilme
- 4 Gyro sensörünü kullanmak için gerekli paketleri kurarak, ilgili konutlar ile telefonun sallanma durumunun tespit edilerek arama işlemini başlatabilme

Anahtar Sözcükler: • Telefon Defteri Uygulaması • Firebase • Kullanıcı Giriş • Veri Tabanı
• Sensör kullanımı



GİRİŞ

Bu üitede kitapta daha önceki ünitelerde öğrenilen konuları pekiştirmek amacıyla baştan sona bir flutter uygulaması gerçekleştirilecektir. Bu uygulamada kullanıcı girişinin olduğu ve verilerin bir Firebase veritabanında saklandığı telefon defteri uygulaması yapılacaktır. Ayrıca son olarak sensor kullanımına örnek olması açısından telefon defterindeki kişinin telefonu sallama yöntemiyle aranabilmesi sağlanacaktır.

Telefon defteri uygulamasında kullanıcı girişini olacak online Firebase Cloud Firestore veritabanı kullanacaktır. Dolayısıyla kullanıcı istediği telefon dan giriş yaptığında daha önce başka bir telefondan oluşturduğu kişilerine erişim sağlayabilecektir.

Öncelikle veritabanı olarak Firebase seçilmiştir. Bunun birkaç sebebi bulunmaktadır. Flutter, Firebase gibi Google'nın bir ürünüdür ve Firebase'yi doğal olarak desteklemektedir. Dolayısıyla kullanıcı giriş gibi nispeten karmaşık olabilecek ve çeşitli güvenlik tedbirlerinin alınmasını gerektirecek durumlar için hazır araçları bulunmakta ve çok kolaylıkla uygulanabilmektedir. Firebase proje geliştirdirken ve küçük çaplı kullanımlarda ücretsiz olarak kullanılabilmektedir. Örneğin veritabanı için 1 GB'a kadar olan kullanımlar ücretsizdir. 1 GB'in geçilmesi durumunda ücret ödemesi gerekmektedir. 1 GB'in oldukça makul bir değer olduğu ve çeşitli optimizasyon teknikleri kullanıldığından birçok projede 1 GB'in üzerinde çıkmak gerekmeyeceği söylenebilir. Ayrıca kullanıcı girişinde ise aylık 10.000 girişe kadar ücretsizdir. Sonrasında ise ücret talep edilmektedir. Firebase dışında farklı çözümlerin tercih edilmesi durumunda da mutlaka bir ücret ortaya çıkacaktır. Örneğin veritabanını kendi sunucumuzda barındırmamız durumunda, sunucu kiralama ücreti, alan adı ücreti gibi çeşitli ücretlerle karşılaşılması normaldir. Bu durumda yapılması gereken şudur, aylık kullanım miktarları tahmin edilerek, en uygun (ucuz, kullanımı kolay ve güvenli) çözümün uygulanmasıdır. Firebase'de bu çözümlerden biri olabilir.

Bu üitede projede yazılan tüm kodlar burada verilmemiştir. 100'lere satır kod yazılılığı için tüm kodların burada verilmesi uygun olmayacağından. Onun yerine kritik öneme sahip olan kodlara yer verilmiş ve açıklamaları yapılmıştır. Projenin kaynak kodları aşağıdaki bağlantıdan indirilebilir.



Kaynak kodlara https://serkancankaya.com/FlutterTelDefteri_Kaynak_Kodlar.zip adresinden ulaşabilirsiniz.



Firebase: Firebase, "uygulamanızı oluşturmak, geliştirmek ve büyütmek" için Google tarafından sunulan bir araç setidir. Bu araçlar, uygulama geliştiricilerinin normalde kendilerinin oluşturmak zorunda oldukları, ancak gerçekle oluşturulmak istemekleri hizmetlerin büyük bir bölümünü kapsar. Uygulama geliştiricileri normalde uygulama deneyiminin kendisine odaklanmayı tercih etmektedir. Firebase ise analistik, kimlik doğrulama, veritabanları, yapılandırma, dosya depolama, push mesajlaşma gibi şeyleri içerir ve uygulama geliştiricilerinin bu gibi konularda çok daha hızlı entegrasyon yapabilmelerini sağlar. Firebase hizmetleri bulutta barındırılır ve uygulama geliştirici tarafından çok az veya hiç çaba harcamadan ölçeklenebilir. Bu örnekte kimlik doğrulama ve veritabanı uygulamaları kullanılmıştır.

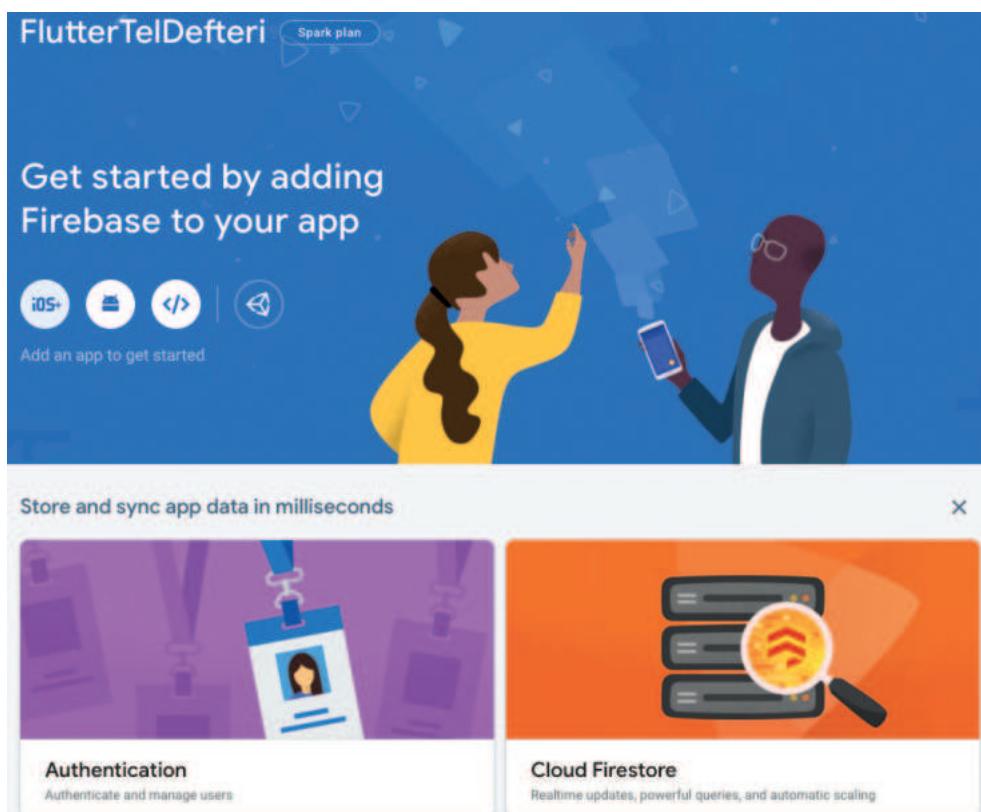
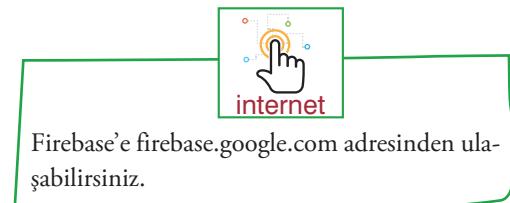


dikkat

İlişkisel veri tabanı yönetim sistemi (RDBMS), uygulama geliştiricilerinin ilişkisel bir veri tabanı oluşturmasını, güncellemesini ve yönetmesini sağlayan bir sistemdir. Genellikle RDBMS'lerde, veri tabanına erişmek için Yapılandırılmış Sorgu Dili (SQL) kullanılır ve veriler tablolar biçiminde depolanır. Ancak SQL olmadan da RDBMS kullanımı mümkün olabilmektedir. RDBMS, dünyada en popüler veritabanı sistemidir. Sistem performansı ve uygulama kolaylığının yanında büyük miktarda veriyi depolamak ve almak için güvenilir bir yöntem sağlar.

FIREBASE VERİ TABANININ KURULMASI

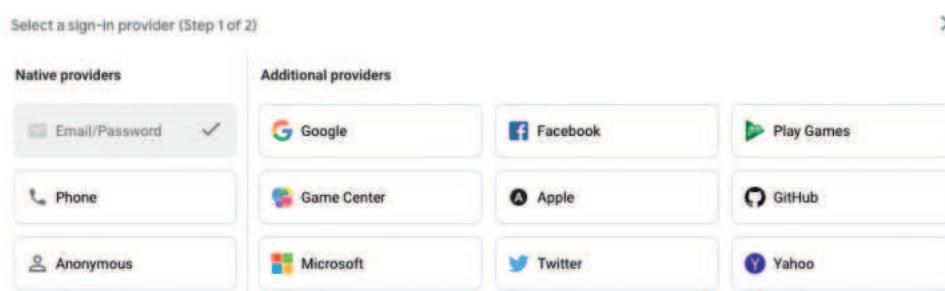
Firebase veri tabanını kullanabilmek için öncelikle Google hesabının olması gerekmektedir. Daha sonra firebase.google.com adresine Google hesabı ile giriş yaptıktan sonra Goto Console seçeneğine tıklanmalıdır. Burada yeni proje oluşturulabilir veya daha önce oluşturulmuş projeler düzenlenebilir. Yeni proje oluşturmak için “Add Project” seçeneği seçilmelidir. İki aşamalı proje oluşturma sayfasında ilk olarak proje adı belirlenmelidir. Eğer proje adı daha önce başka bir kullanıcı tarafından alınmış ise Firebase proje adı ile başlayan eşsiz bir isim üremektedir. Örneğimizde FlutterTelDefteri ismi kullanılmıştır. Bu isim daha önce başka bir kullanıcı tarafından kullanılmadığı için eşsiz isim olarak kabul edilmiş ve yeni eşsiz bir isim üretme durumu olmamıştır. İkinci aşamada “Google Analytics for your Firebase Project” araçlarını kullanılıp kullanılmayacağı sorulmaktadır. Bu örnekte bu seçenek iptal edilerek devam edilmiştir. Böylece Firebase projesi oluşturulmuş olur (Görsel 8.1).



Görsel 8.1 Firebase Proje Sayfası

Proje oluşturulduktan sonra bu bölümde yapılması gereken iki işlem bulunmaktadır: Authentication ve Firestore Database (Cloud Firestore). Authentication geliştirilecek uygulamada kullanıcı girişlerini yönetmek için kullanılan modüldür. Authentication bölümünden Get Started seçeneği seçilmelidir. Firebase kullanarak farklı sistemler ile kullanıcı giriş yapabilmektedir (Görsel 8.2). Bu projede Email/Password seçeneği kullanılmıştır. Kullanıcılar e-posta adreslerini kullanarak şifrelerini oluşturmuşlardır. Farklı pro-

jelerde diğer kullanıcı giriş seçenekleri de değerlendirilebilir. Kullanıcılar açısından internetteki en büyük problemlerden biri de çok sayıda platforma üye olmak ve bu platformların şifrelerini hatırlamaktır. Alternatif kullanıcı giriş seçenekleri sayesinde kullanıcılar yeni şifre belirlemek zorunda kalmadan örneğin Google hesabı ile sisteme giriş yapabilirler. Ayrıca birden fazla giriş seçeneğinin aynı anda kullanılması da mümkünür. Örneğin uygulamalar hem Google hem de Apple ile kullanıcı girişini yapabilecek şekilde yapılandırılabilir. Böylece kullanıcılar için farklı kullanıcı girişleri arasından tercih yapabilecekleri bir esneklik sunulmuş olacaktır.

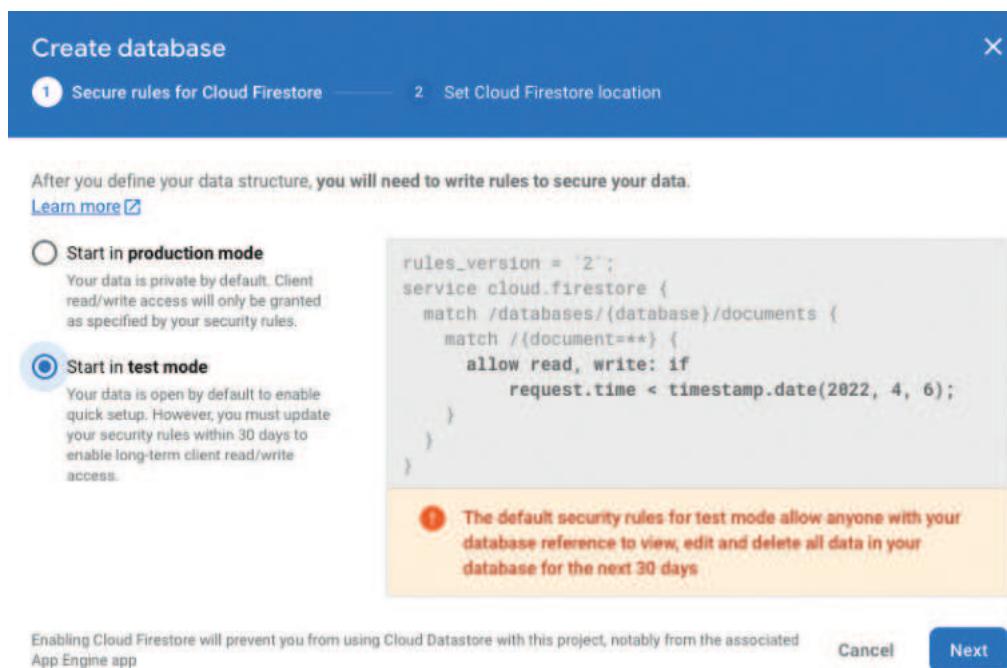


Görsel 8.2 Firebase tarafından desteklenen kullanıcı giriş sistemleri.

Bu çalışmada sadece Email/Password seçeneği tercih edilmiştir. Dolayısıyla kullanıcıların uygulama öncelikle üyelik oluşturması gerekecektir. Uygulamaya kayıtlı kullanıcıların yönetim işlemleri de yine Authentication bölümündeki Users sekmesinden yapılmaktadır. Bu bölüm kullanılarak yeni kullanıcı oluşturulabilir. Böylelikle uygulama geliştirilirken kullanıcı girişi sayfaları test edilebilecektir. Ayrıca kullanıcı silme, pasif duruma alma ve kullanıcının şifresini sıfırlama işlemleri de yapılmamıştır. Authentication bölümündeki Templates sekmesinde e-posta adres doğrulama, şifre sıfırlama, e-posta adresi değiştirme gibi e-posta iletleri ayarlanabilmektedir. Ayrıca bu bölümde **SMTP** ayarının da yapılması gerekmektedir. Böylece gönderilecek e-postalar burada ayarlanan SMTP e-posta sunucusu tarafından gönderilebilecektir. Buraya kurumsal mail sunucusunun adresi yazılabileceği gibi Gmail gibi bir sunucunun da adresi ve kullanıcı bilgileri yazılabilir. Ancak Gmail gibi ücretsiz e-posta sunucuları ile günlük gönderilebilecek e-posta sayısı sınırlıdır. Bu durumda günlük gönderilecek e-posta sayısı örneğin Gmail için 500'ü geçiyorsa ücretli üyelik satın almak veya farklı e-posta sunucuları kullanmak gerekecektir.

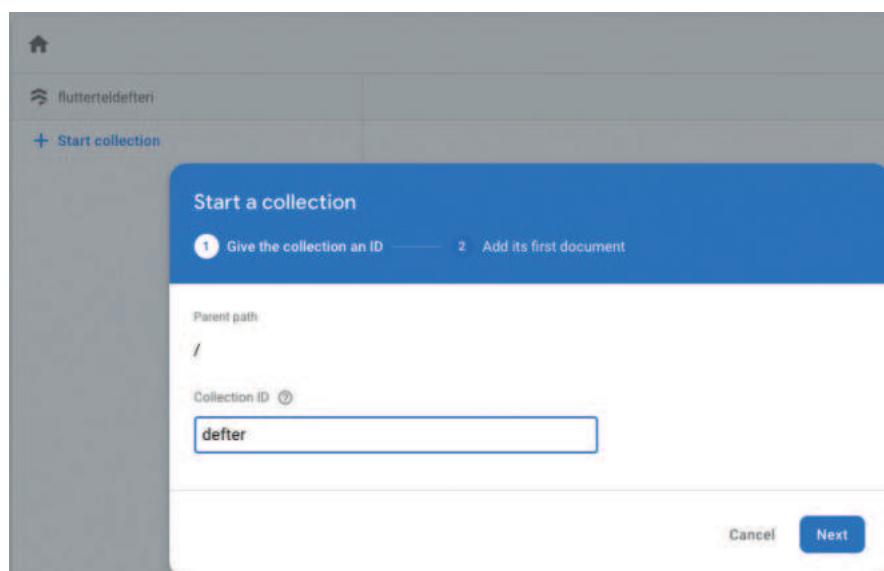
Firestore Database bölümünde Create Database ile veritabanı oluşturulmalıdır. Projede öncelikle test mode kullanılmıştır (Görsel 8.3). Ancak bu mod güvenli değildir. Proje bittikten sonra daha uzun kullanımlara yönelik olarak veritabanına erişim kuralları güncellenmelidir.

SMTP: SMTP, Basit Posta Aktarım Protokolü anlamına gelir ve e-posta sunucuları tarafından e-posta göndermek, almak ve/veya iletmek için kullanılan bir uygulamadır. Bir SMTP e-posta sunucusunun, kullandığınız posta istemcisi veya uygulama tarafından ayarlanabilen ve genellikle smtp.serveraddress.com şeklinde bir adresi olacaktır. Örneğin, Gmail'in kullandığı SMTP sunucusu smtp.gmail.com'dur.



Görsel 8.3 Create database sayfası.

Bir sonraki aşamada veritabanının hangi coğrafi bölgede barındırılacağı seçildikten sonra Enable ile veritabanı kullanıma hazır hale gelecektir. Uygulamanın hedef kitlesinin bulunduğu coğrafi bölgeye yakın bir yer seçilmesi durumunda performansın daha iyi olacağı söylenebilir. Bu çalışmada Türkiye'ye yakın olması bakımından Avrupa merkezli bir sunucu tercih edilmiştir. Firebase Cloud Firestore klasik veri tabanlarından farklı yapıda bir veritabanı sistemidir. Detaylı bilgi için kendi web sayfasındaki yardım belgeleri incelenebilir. Veritabanı oluşturulduktan sonra kişilerin telefon ve e-posta bilgilerini barındırmak için bir koleksiyon oluşturulmalıdır. Bu örnekte defter isminde bir koleksiyon oluşturulmuştur (Görsel 8.4). Aslında koleksiyon oluşturulmadan da uygulamanın çalışacağı söylebilir. Cloud Firestore koleksiyonlar ilk kullanıldıkları anda otomatik olarak oluşturulmasını sağlamaktadır.



Görsel 8.4 Start a collection sayfası.

Böylece veritabanı ile ilgili süreçler bitmiştir. Bundan sonra yeni kullanıcı kaydı, kullanıcı girişi, veritabanına yeni kayıt ekleme, düzenleme ve silme işlemleri flutter projesi ile gerçekleştirilecektir. Öncelikle bu bölümde Cloud Firestore hakkında biraz bilgiye yer verilmiştir.



Cloud Firestore

Cloud Firestore, NoSQL türünde belge odaklı bir veritabanıdır. NoSQL veritabanları ilişkisel veritabanı yönetim sistemlerinden farklı olarak verilerin tablo şeklinde depolanmadığı veri tabanlarıdır. NoSQL veritabanları, veri modellerine göre çeşitli türlerde olabilmektedir. Bu türler ise belge, anahtar/değer çifti, geniş sütun ve grafik şeklinde sıralanabilir. NoSQL veritabanları esnek şemalar sağlarlar ve büyük miktarda veri ve yüksek kullanıcı yükleriyle kolayca ölçeklenirler. Cloud Firestore veritabanında da tablo veya satır yoktur. Bunun yerine, verileri koleksiyonlar halinde düzenlenmiş belgelerde depolanır. Her belge bir dizi anahtar/değer çiftini içerir. Tüm belgeler koleksiyonlarda saklanmalıdır. Belgeler, her ikisi de dizeler gibi ilkel alanlar veya listeler gibi karmaşık nesneler içerebilen alt koleksiyonlar ve iç içe nesneler içerebilir. Koleksiyonlar ve belgeler, Cloud Firestore'da dolaylı olarak oluşturulur.

Cloud Firestore'da depolama birimi belgedir. Belge, değerlerle eşleşen alanları içeren bir kayıt olarak düşünülebilir. Ancak belgeler koleksiyonları da içerebileceği için kayıttan daha fazlasıdır. Her belge bir adla tanımlanır. Örneğin ali kullanıcısını temsil eden bir belge şöyle olabilir:



ali
adi : "Ali"
soyadi : "Veli"
dogum_tarihi : 1815

Bu örnekte görüldüğü gibi belgeler anahtar/değer çiftlerini barındırmaktadır. Burada adı anahtar "Ali" ise onun değeridir. Bir belgedeki karmaşık, iç içe nesnelere map denir. Örneğin, yukarıdaki örnekteki kullanıcının adı aşağıdaki gibi bir map ile yapılandırılabilir.



ali
isim:
adi : "Ali"
soyadi : "Veli"
dogum_tarihi : 1815

Göründüğü üzere bu yapı JSON'a çok benzemektedir. Aslında temelde de Cloud Firestore'un bir tür JSON veritabanı olduğu da söylenebilir. Belge adları bu örnekte olduğu gibi elle verilebileceği gibi, Cloud Firestore tarafından otomatik, eşsiz bir değer olarak üretilebilmektedir. 4ly08ultXrbDNuyeev8 değeri Cloud Firestore tarafından üretilmiş örnek bir değerdir. Belgeler, yalnızca belgeler için kapsayıcı olan koleksiyonlarda yaşar. Örneğin, her biri bir belgeyle temsil edilen çeşitli kullanıcılarınızı içeren bir kullanıcı koleksiyonu olabilir:

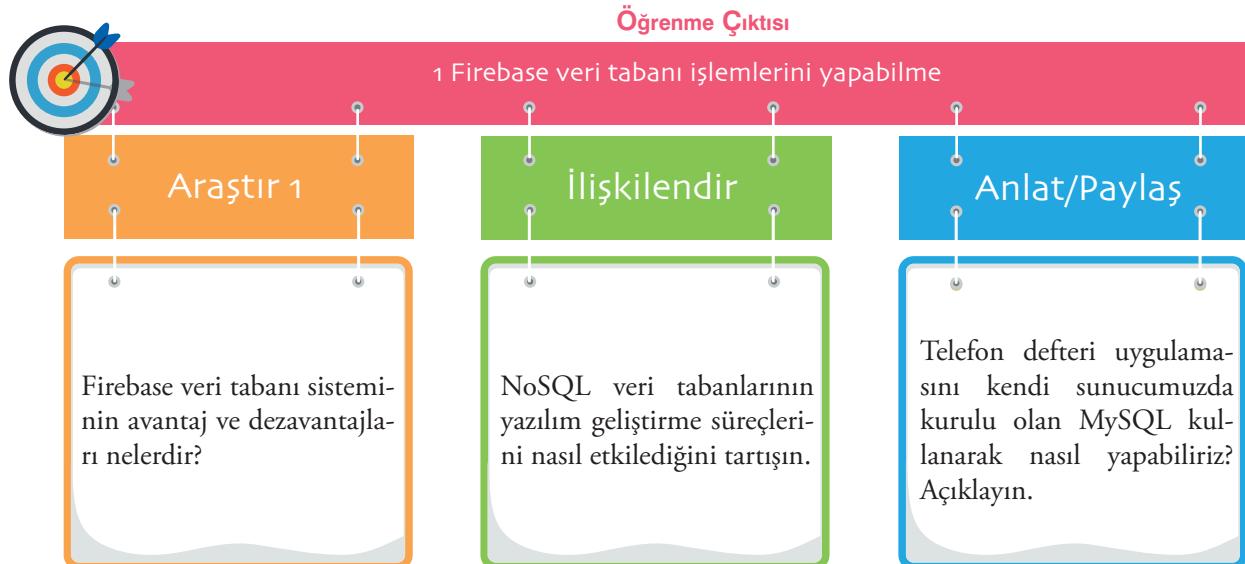
 kullanıcılar
 HNjhUdlUO88olr0MHwl7
 adı : "Ali"
 soyadı : "Veli"
 dogum_tarihi : 1995
 4ly08ultXbrbDNuyeev8
 adı : "Hasan"
 soyadı : "Hüseyin"
 dogum_tarihi : 2000

Örnekte kullanıcılar koleksiyonunda iki belge bulunmaktadır. Bu belgelerde ise kullanıcılar ait bilgiler yer almaktadır. Cloud Firestore şemasız olduğu için belgelere hangi alanların konulacağı ve bu alanlarda hangi veri türlerinin saklanacağı konusunda herhangi sınırlama bulunmamaktadır. Aynı koleksiyondaki belgelerin tümü farklı alanlar ve veri türleri içerebilir. Ancak, klasik tablolarda olduğu gibi belgeleri daha kolay sorgulayabilmek için aynı alan ve veri türlerini ilgili koleksiyonda bulunan tüm belgede kullanmak mantıklı olacaktır. Bir koleksiyonda sadece belgeler bulunabilir. Koleksiyonlarda doğrudan değer saklamak mümkün değildir. Ayrıca koleksiyonlar diğer koleksiyonları da içeremez. Firebase'de alt koleksiyon oluşturmak mümkündür. Alt koleksiyon, belirli bir belgeyle ilişkili bir koleksiyondur. Bir başka deyişle belgelerin altında alt koleksiyon oluşturulabilir. Örneğin kullanıcı ile ilgili belgenin içinde notlar alt koleksiyonu oluşturulabilir. Notlar koleksiyonunun içinde de o kullanıcının yazmış olduğu notlardan oluşan belgeler bulunabilir.

 kullanıcılar
 HNjhUdlUO88olr0MHwl7
 adı : "Ali"
 soyadı : "Veli"
 dogum_tarihi : 1995
 notlar
 KAj7jGqPxTZCghRvw5VS
 baslik : "Ders Çalış"
 mesaj : "C++ ilgili ders çalışmam lazım"
 KbYWG1wCAF9ttiYBIVZK
 baslik : "Proje"
 mesaj : "Proje başvurusu hazırlamam lazım."

Böylece bu notlar sadece Ali Veli isimli kullanıcıya ait olmuş olur. Bir koleksiyon içindeki belgelerin adları benzersizdir. Koleksiyonların oluşturulmasına veya silinmesine gerek yoktur. Bir koleksiyondaki ilk belge oluşturulduktan sonra koleksiyon var olur. Bir koleksiyondaki tüm belgeler silinirse, koleksiyonda silinmiş olur. Sonuç olarak Firebase Cloud Firestore sistemi yaygın kullanılan RDBMS'lerden çok farklıdır. Ancak kolay kullanımı ve yüksek performanslı olması bakımından avantajları bulunmaktadır. Spark isminde ücretsiz bir planımasına rağmen, çok yoğun kullanımlarda ciddi kullanım ücretleri ortaya çıkabilir. Geliştiriciler, gereksinimlerini tespit ederek, Firebase veya MySQL, PostgreSQL, Oracle, SQL Server

gibi RDBMS'lerden birini tercih edebilirler. Farklı sistemlerin avantaj ve dezavantajları bulunmaktadır. Örneğin MySQL açık kaynak kodludur ve ücretsiz olarak kullanılabilir. Ancak MySQL'i kurabilmek için bir alan adı ve sunucuya ihtiyaç vardır. Alternatif olarak bulut sistemler üzerinde barındırılan veritabanı sistemlerinden de yararlanılabilir. Doğal olarak bulut sistemlerin, alan adı ve sunucu hizmetinin de bir maliyeti olacaktır.



TELEFON DEFTERİ UYGULAMASININ SAYFALARINI TASARLAMA

Flutter projesi oluşturmak için çeşitli araçlar kullanılabilir. Ancak bu çalışmada Android Studio tercih edilmiştir. Android Studio'da yeni bir flutter projesi oluşturulmalıdır. Proje oluşturulduktan sonra Terminalden aşağıdaki kod yazılarak firebase_core kütüphanesi projeye eklenmelidir.

```
flutter pub add firebase_core
```

Bu kod çalıştırıldıktan sonra pubspec.yaml dosyası içine "firebase_core: ^1.13.1" satırının eklendiği görülecektir. Versiyon numarası güncel sürümü göre değişiklik gösterebilir. Sonrasında ise flutter pub get yapılacak projedeki bağımlılıklar güncellenmelidir. Bir sonraki aşamada ise FlutterFire yazılımı kullanılmıştır. Bu yazılım, oluşturulan Firebase projesi ile oluşturulan flutter projesini ilişkilendirmek için kullanılmaktadır. Bu ilişkilendirme işlemi manuel olarak yapılmaktadır. Ancak FlutterFire bu süreci oldukça basitleştirmektedir. FlutterFire yazılımı daha önce kurulmamışsa sisteme kurulması gerekmektedir. Bunun için Terminal sayfasında sırayla aşağıdaki komutlar çalıştırılmalıdır.

```
dart pub global activate flutterfire_cli
flutterfire configure
```

İkinci komut çalıştırıldığında Google hesabı ile giriş yapılması istenmektedir. Giriş yapıldıktan sonra otomatik olarak daha önce oluşturulan Firebase projeleri listelenir. Bu projelerden ilgili olan proje seçilmiştir. Bu örnekte FlutterTelDefteri projesi seçilmiştir. Proje seçildikten sonra projenin hangi platformları destekleyeceği sorulmaktadır. Bu örnekte Android, IOS ve Web seçenekleri seçilmiştir. Bir sonraki aşama-

da FlutterFire yazılımı Firebase projesi ile flutter projelerinin ilişkilendirme işini gerçekleştirmiş ve proje içerisinde `firebase_options.dart` isminde bir dosya oluşturmuştur. Projede Firebase'i etkinleştirmek için aşağıdaki gibi import satırları yazılmalı ve main fonksiyonu için ilgili komutlar çalıştırılmalıdır.

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/cupertino.dart';
import 'firebase_options.dart';
void main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp(
        options: DefaultFirebaseOptions.currentPlatform,
    );
    runApp(MyApp());
}
```



Giriş Sayfasını Oluşturma

Firebase veritabanı ve kullanıcı giriş sistemlerini kullanabilmek için aşağıdaki komutlar çalıştırılmalıdır. Bu komutlar çalıştırıldıktan sonra pubspec.yaml dosyasına ilgili satırlar otomatik olarak eklenecektir.

```
flutter pub add cloud_firestore
flutter pub add firebase_auth
flutter pub get
```

Projedeki ilk sayfa Giriş sayfasıdır. Giriş sayfasının görüntüsü Görsel 8.5'te görüldüğü gibidir.

Giriş sayfası ayrı bir dosyada StatefulWidget sınıfı miras alınan `Giris` isminde bir sınıf ile oluşturulmuştur. Projede her bir sayfa ayrı dosyalar içinde oluşturulmuş olan StatelessWidget veya StatefulWidget sınıfları miras alınarak oluşturulmuştur. Ayrı dosyalarda barındırılmasının sebebi ise yazılan kodların karmaşıklığının azaltılarak bir dosyadaki yazılan satır sayısını düşük seviyelerde tutma ihtiyacıdır. Ayrı dosyalarda oluşturulan bu sınıflar kullanılmak istendiğinde import edilerek rahatlıklar kullanılabilmektedir.

Giriş sayfasında kullanılan bazı widget'lar diğer sayfalarda da kullanılacağı için StatelessWidget sınıfı miras alınarak AuthButton, CustomFormField, CustomHeader, CustomRichText gibi yeni sınıflar oluşturulmuş ve bu sayfalarda kullanılmıştır. Örneğin CustomFormField sınıfı aşağıdaki gibi tanımlanmıştır.



Görsel 8.5 Giriş sayfası

```
class CustomFormField extends StatelessWidget {
    final String headingText;
    final String hintText;
    final bool obsecureText;
    final Widget suffixIcon;
    final TextInputType keyboardType;
    final TextInputAction textInputAction;
    final TextEditingController controller;
    final int maxLines;

    const CustomFormField(
        {Key? key,
        required this.headingText,
        required this.hintText,
        required this.obsecureText,
        required this.suffixIcon,
        required this.keyboardType,
        required this.textInputAction,
        required this.controller,
        required this.maxLines})
        : super(key: key);

    @override
    Widget build(BuildContext context) {
        return Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Container(
                    margin: const EdgeInsets.only(
                        left: 20,
                        right: 20,
                        bottom: 10,
                    ),
                    child: Text(
                        headingText,
                        style: KTextStyle.textFieldHeading,
                    ),
                ),
                Container(
                    margin: const EdgeInsets.only(left: 20, right: 20),
                    decoration: BoxDecoration(
                        color: AppColors.grayshade,
                        borderRadius: BorderRadius.circular(15),
                    ),
                    child: Padding(
                        padding: const EdgeInsets.symmetric(horizontal: 12.0),
                        child: TextField(
                            maxLines: maxLines,
                            controller: controller,
                            textInputAction: textInputAction,
                            keyboardType: keyboardType,
                            obscureText: obsecureText,
                            decoration: InputDecoration(
                                hintText: hintText,
                                hintStyle: KTextStyle.textFieldHintStyle,
                                border: InputBorder.none,
                                suffixIcon: suffixIcon),
                        ),
                    ),
                );
            ],
        );
    }
}
```

Bu tür widgetlar sayesinde ekranda benzer görüntütü elde etmek için çok daha az kod yazmak yeterlidir. Projedeki tüm metin kutuları CustomFormField olarak tasarlanmıştır. Eğer bu şekilde bir widget tanımı yapılmamış olsaydı her metin kutusu kullanılması gereken yerde yaklaşık 60 satır kod yazmak gerekecektir. Bunun yerine bu projede aşağıdaki CustomFormField sınıfına ait bir widget aşağıdaki gibi rahatlıkla oluşturulabilmektedir.

```
CustomFormField(
    maxLines: 1,
   textInputAction: TextInputAction.done,
    keyboardType: TextInputType.emailAddress,
    controller: _cEposta,
    headingText: "Eposta Adresi",
    hintText: "Eposta adresini giriniz",
    obscureText: false,
    suffixIcon: const SizedBox(),
),
```

CustomFormField sınıfı incelendiğinde çeşitli parametrelerin kullanıldığı görülmektedir. Bu parameteler sayesinde bu sınıf farklı ihtiyaçlara uyum sağlayabilecek bir esneklik kazanmış olmaktadır. Örneğin obscureText parametresi true olarak ayarlanarak metin kutusunun şifre girişine uygun hale gelmesi sağlanabilmektedir. Şifre girişi metin kutularında karakterler * veya farklı bir karakter ile gizlenmektedir. Ayrıca sık kullanılan kodları widget olarak ayrı dosyalarda saklamak daha önce bahsedildiği gibi yazılan programın karmaşaklığını azaltmak açısından da önem arz etmektedir. Giriş sayfası ilk açıldığında aşağıdaki kod ile kullanıcının daha önce giriş yapıp yapmadığı kontrol edilmektedir.

```
@override
void initState() {
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((_) {
        if (FirebaseAuth.instance.currentUser != null) {
            Navigator.push(
                context,
                MaterialPageRoute<void>(
                    builder: (context) => Defter(),
                ),
            );
        }
    });
}
```

Burada kodlar, StatefulWidget sınıfından türetilen Giriş sınıfı için initState fonksiyonunu içine yazılmıştır. initState fonksiyonu StatefulWidget sınıfından miras alınmakta override edilmektedir. Bu fonksiyon, StatefulWidget nesnesi ağaç eklendiğinde bir kez çağrılır. Böylece bu kodlar Giriş sınıfı çağrıldığında anda çalışmış olacaktır. super.initState() ile miras alınan StatefulWidget sınıfının ilgili fonksiyonunun çalıştırılması sağlanmıştır. WidgetsBinding.instance.addPostFrameCallback olayı ile Giriş sayfası ekranda oluşturulduktan sonra tetiklenecek yeni bir olay tanımlanmaktadır. Bu olay sayesinde Giriş sınıfının widgetleri ekranda oluşturulduktan sonra bu olay ile belirtilen fonksiyon çalıştırılmaktadır. Bu örnekte ise addPostFrameCallback fonksiyonuna parametre olarak yazılmış fonksiyon tanımlaması yapılmıştır.

Dolayısıyla sayfa ekranda oluştuğu anda bu isimsiz fonksiyon çalıştırılır. Bu isimsiz fonksiyonda ise FirebaseAuth.instance.currentUser değerinin null olup olmadığı kontrol edilmektedir. Eğer null değil ise daha önce kullanıcı girişini yaptığı anlamına gelecektir. Bu durumda tekrar kullanıcı girişini yaptırmamak için Navigator.push ile sistem Defter sayfasını açmaktadır. Giriş sayfasında kullanıcı adı ve şifre girdikten sonra Giriş düğmesine tıklandığında ise aşağıdaki tıkla fonksiyonu çalışmaktadır.

```
Future<void> tıkla() async {
    try {
        UserCredential userCredential = await
    FirebaseAuth.instance.signInWithEmailAndPassword(
        email: _cEposta.text,
        password: _cSifre.text
    );
    if(FirebaseAuth.instance.currentUser!=null){
        Navigator.push(
            context,
            MaterialPageRoute<void>(
                builder: (context) => Defter(),
            ),
        );
    }
} on FirebaseAuthException catch (e) {
    if (e.code == 'user-not-found') {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            content: Text('Kullanıcı Bulunamadı'),
        ));
    } else if (e.code == 'wrong-password') {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            content: Text('Şifre Yanlış'),
        ));
    }
}
}
```

Tıkla fonksiyonu Future<void> türünde ve async olarak tanımlanmıştır. Bunun sebebi await ifadesinin sadece async olarak tanımlanan fonksiyonlarda çalışabilmesidir. Bir fonksiyonun async olabilmesi için geri dönüş türünün Future<T> olması gerekmektedir. T fonksiyonun asıl geri döndüreceği türdür. Bu örnekte geriye döndürülmesi gereken bir değer olmadığı için T yerine void yazılmıştır. Async olarak tanımlanan fonksiyonlar çağrıldığında ayrı bir thread (paralel bir süreç) olarak çağrılmaktadır. Özellikle zaman alacak süreçlerde bu yöntem tercih edilmektedir. Örneğin internette indirme yapılmasını gerektiren bir işlem gerçekleştirilecek ise async fonksiyonlardan yararlanılmalıdır. Böylece kullanıcılar indirme işlemi devam ederken dahi uygulama ile etkileşim kurmaya devam edebileceklerdir. Internetten indirme yapılması gereken bir süreçte async kullanılmamış olsaydı, uygulama indirme süresi boyunca donacak ve tepki vermeyecekti. Özellikle internetin yavaş olduğu senaryolarda bu donma süresi çok uzayabilir. Kullanıcı uygulamanın kilitlendiğini düşünüp, uygulamayı sonlandırma yoluna gidebilir. Bu gibi durumlarla karşılaşmamak için async fonksiyonlardan yararlanılmalıdır. signInWithEmailAndPassword fonksiyonu da bir async fonksiyondur. Ancak burada await ifadesi ile bu fonksiyonun sonucunun beklenmesi sağlanmıştır. Eğer bu yapılmamış olsaydı signInWithEmailAndPassword fonksiyonu ayrı bir thread olarak çalışacak ve fonksiyonun sonucu beklenmeden if satırı ile giriş yapılp yapılmadığı kontrol edilecektir. signInWithEmailAndPassword işleminin internete bağlı olarak

uzun süreceği düşünülürse altında bulunan if satırının her zaman false sonucunu döndüreceği varsayılabılır. Buradaki await ifadesi ile signInWithEmailAndPassword işlemi tamamlanana kadar burada beklenilmesi sağlanmıştır. Böylece if ifadesi doğru bir karşılaştırma yapabilmektedir. tikla fonksiyonu ise Giriş düğmesine basıldığından çağrılmaktadır. Burada da bir async süreç olduğu için programın donma durumu söz konusu olmamaktadır. Özette tikla fonksiyonda FirebaseAuth bileşeni kullanılarak kullanıcının girdiği eposta adresi ve şifre ile kullanıcı girişini yapılmaya çalışılmaktadır. Eğer giriş başarılı olursa sistem Defter sayfasına yönlendirmektedir. Eğer giriş başarılı olmaz ise üretilen hata koduna göre "Kullanıcı bulunamadı" veya "Şifre Yanlış" uyarıları ScaffoldMessenger ile ekranın en altına gösterilmektedir.

Bu kodda try on catch mekanizmasından yararlanıldığı görülmektedir. Programlama dillerinde çalışma zamanında hata üretmesi beklenen süreçler bulunmaktadır. Örneğin bir işletim sisteminde bir dosyaya erişim veya internette bulunan bir veri tabanı sunucusu ile iletişim kurma süreçleri bağlamında hata ortaya çıkabilir. Erişim istenen dosya silinmiş olabilir. O an internet bağlantısı olmaması sebebiyle veritabanına erişim sağlanamayabilir. Bu gibi durumlarda programlar hata yapıp çalışmasını sonlandırır. Ancak programın hata verip sonlanması istenilen bir durum değildir. Bu gibi durumlarla baş etmek için programlama dillerinde try catch veya bu örnekte olduğu gibi try on catch mekanizmaları geliştirilmiştir. Özellikle hata üretebilecek satırlar try bloğu mekanizması içine yazılmalıdır. Hata üretme potansiyeli bulunan fonksiyonlar throw ifadesi ile Exception sınıfını veya Exception sınıfından türetilmiş özel bir sınıfı kullanarak hata üretebilirler. Örneğin signInWithEmailAndPassword fonksiyonu Exception sınıfından türetilmiş FirebaseAuthException türünde bir hata üretebilmektedir. Bu durumda on catch mekanizması ile hata olması durumunda bu hata yakalanabilir. Bu örnekte oluşabilecek hatalar arasında kullanıcı şifresinin yanlış olması olasılığı e.code == 'wrong-password' karşılaştırma ifadesi ile kontrol edilerek giriş yapmaya çalışan kullanıcı bu konuda uyarılmıştır. Exception'ın yakalanması durumunda uygulama çalışmaya devam edecek ve sonlandırılmayacaktır. Exception'ın yakalanmaması durumunda ise uygulama hata verip kapanacaktır.

Kayıt Ol Sayfasını Oluşturma

Kullanıcı Giriş sayfasında Kayıt Ol linkine tıklandığında Görsel 8.6'te görülen ekran açılmaktadır. Bu sayfada yeni bir kullanıcı e-posta adresini ve şifresini yazarak sisteme kaydolabilmektedir. Kullanıcıdan şifresini iki defa yazması istenmektedir. Böylece hatalı şifre yazımının önüne geçilmeye çalışılmıştır.



Görsel 8.6 Kayıt Ol sayfası

Kayıt ol sayfasında kullanıcı e-posta adresini ve şifresini belirleyerek sisteme kayıt olabilmektedir. Kayıt ol düğmesine tıklandığı anda aşağıda kodlarda verilen tiklaKayıt fonksiyonu çalışmaktadır. Fonksiyon içinde await ifadesi kullanıldığı için tiklaKayıt fonksiyonu async olarak tanımlanmıştır.

```
Future<void> tiklaKayit() async {
    if(_cSifre.text!= _cSifre2.text){
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            content: Text('Şifreler Uyumsuz...'),
        )));
        return;
    }
    try {
        await FirebaseAuth.instance.createUserWithEmailAndPassword(
            email: _cEposta.text,
            password: _cSifre.text,
        );
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            content: Text("Kayıt Başarılı. Giriş yapabilirsiniz."),
        ));
        Navigator.push(
            context,
            MaterialPageRoute<void>(
                builder: (context) => Giris(),
            ),
        );
    } on FirebaseAuthException catch (e) {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            content: Text(e.message!),
        )));
    }
}
```

tiklaKayıt fonksiyonunda öncelikle kullanıcının iki defa girmiş olduğu şifrelerin tutarlı olup olmadığı kontrol edilmekte ve tutarsız ise “Şifreler Uyumsuz..” mesajı verilmektedir. Daha sonra FirebaseAuth.instance.createUserWithEmailAndPassword fonksiyonu ile kullanıcı kaydı gerçekleştirilmektedir. Hatalı bir durum olması durumunda Firebase sisteminin ürettiği hata mesajı doğrudan ScaffoldMessenger ile ekranın altında gösterilmektedir. Kullanıcı kaydının başarılı olması sonucunda ise “Kayıt Başarılı. Giriş yapabilirsiniz.” mesajı verilmekte ve kullanıcı tekrar giriş sayfasına yönlendirilmektedir. Giriş sayfasında başarılı bir şekilde giriş yapmış bir kullanıcı için Defter sayfası açılmaktadır.

Defter Sayfasını Oluşturma

Defter sayfasında bir ListView ile kullanıcının daha önce kaydetmiş olduğu kişilerin listesi yer almaktadır (Görsel 8.7).



Görsel 8.7 Defter sayfası

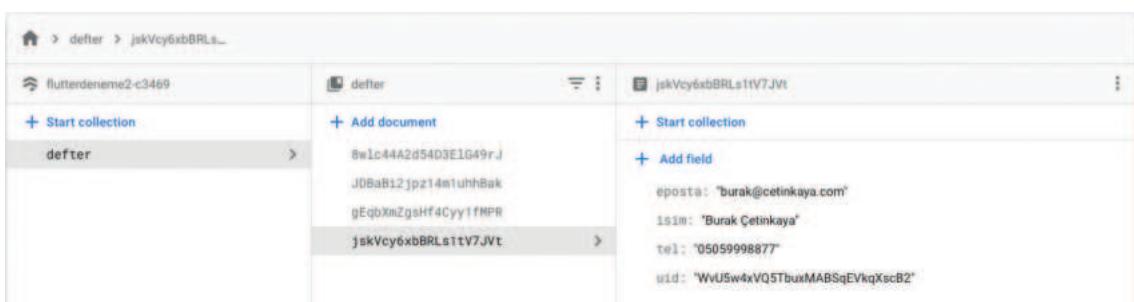
Defter sayfasında kişilerin veritabanından okunarak listelenmesi getir fonksiyonu ile sağlanmaktadır. Getir fonksiyonunun kodları aşağıda verilmiştir. Bu fonksiyon 3 yerde çağrılmaktadır. İlk olarak _DefterState sınıfının kurucu fonksiyonunda çağrılmaktadır. Böylece kullanıcı bu sayfayı açtıktı anda kurucu fonksiyon çalışacak ve veritabanındaki bilgileri getirerek ListView'da listelenmesi sağlanacaktır. Ayrıca getir fonksiyonu Ekle ve Detay sayfalarından tekrar defter sayfasında dönündüğünde de çağrılmaktadır. Böyle yeni kişi eklendiğinde, silindiğinde ve düzenleme yapıldığında yapılan değişiklikler ListView'da güncellenmiş olacaktır.

```
Future<void> getir() async {
  kisiler=[];
  var user = FirebaseAuth.instance.currentUser;
  await FirebaseFirestore.instance
    .collection('defter')
    .where('uid', isEqualTo: user?.uid)
    .get()
    .then((QuerySnapshot querySnapshot) {
      querySnapshot.docs.forEach((doc) {
        kisiler.add(Kisi(doc["isim"].toString(), doc["tel"].toString(),
          doc["eposta"].toString(), doc.id.toString()));
        print(doc["isim"].toString()+"->"+doc.id.toString());
      });
    });
  setState(() {});
}
```

Getir fonksiyonundaki kodlar incelendiğinde kullanıcı temelli bir sorgulama yapıldığı görülmektedir. Dolayısıyla her kullanıcı sadece kendi kaydettiği kişileri listelemektedir. Bunun için where('uid', isEqualTo: user?.uid) özelliğinden yararlanılmıştır. Veriler çekildikten sonra kişiler ismindeki bir List<Kisi> değişkenine aktarılmaktadır. Bu değişken _DefterState sınıfına üye değişken olarak tanımlanmıştır. Getir fonksiyonunun tekrar tekrar çalışması gerekeceği için öncelikle kisiler= [] ile değişkenin içeriği sıfırlanmaktadır. kişiler listesinin içinde Kisi türünde değerler yer almaktadır. Kisi sınıfının kodları aşağıda verilmiştir. Bu sınıf kişi.dart olarak ayrı bir dosyada tanımlanmış ve kullanılmak istenilen sayfalarda import edilmesi sağlanmıştır.

```
class Kisi {
    final String isim;
    final String tel;
    final String eposta;
    final String kod;
    const Kisi(this.isim, this.tel, this.eposta, this.kod);
}
```

Kişi sınıfında veritabanı ile uyumlu olacak şekilde isim, tel, e-posta ve kod özellikleri yer almaktadır. Firebase'de her oluşturulan belge için otomatik veya manuel bir document id üretilmesi gerekmektedir. Kişi ile ilgili üretilmiş olan bu document id değeri Kişi sınıfının kod özelliğinde saklanmaktadır. Bu değer daha sonra güncelleme veya silme işlemi gerçekleştirmek için kullanılmaktadır. Kişi bilgileri veri tabanındaki Görsel 8.8'deki gibi görülmektedir.

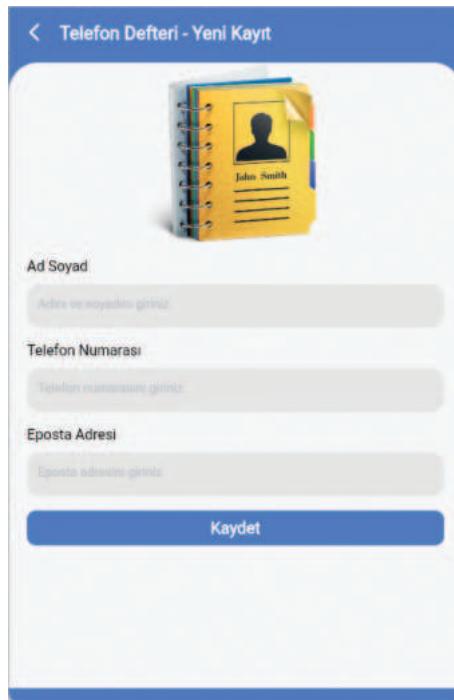


Görsel 8.8 Veritabanında kişi kaydı

Görsel de görüldüğü üzere "Burak Çetinkaya" kişisinin document id'si jskVcy6xbBRLs1tV7JVt şeklindedir. Bu kişi ise WvU5w4xVQ5TbuxMABSqEVkqXscB2 uid'li serkan.cankaya@idu.edu.tr e-posta adresine sahip kullanıcının kişi listesinde yer almaktadır. Bu id'ler sistem tarafından otomatik üretilen eşsiz değerlerdir. Veritabanında her bir kişi kaydı aslında defter koleksiyonunun içinde birer doküman şeklinde yer almaktadır. Oluşturulan bu dokümanlar ise alan (field) veya başka koleksiyonlar barındırılabilir. Bu örnekte ise sadece e-posta, isim, tel ve uid alanları bulunmaktadır. Bu bakımdan firebase'in klasik ilişkisel veritabanı yönetim sistemlerinde veriler yapısı ve ilişkileri daha önceden belirlenmiş tablolarda yer almaktadır. Firebase'de ise her doküman bağımsızdır ve farklı içeriklere sahip olabilmektedir.

Yeni Kayıt Sayfasını Oluşturma

Defter sayfasında yeni kayıt eklenmek istendiğinde sayfanın sağ al bölümünde yer alan + simgesine tıklanmalıdır. Bu simgeye tıklandığında Yeni Kayıt sayfası açılır (Görsel 8.9).



Görsel 8.9 Yeni Kayıt sayfası

Yeni kayıt sayfasında ad soyad, telefon numarası ve e-posta adresi bilgileri girilerek yeni kişi kaydı yapılmaktedir. Kaydet düğmesine tıklandığında aşağıda kodları verilen ekle fonksiyonu çalışmaktadır. Ekle fonksiyonu da içinde await olması bakımından async olarak tanımlanmıştır.

```
Future<void> ekle() async {
    CollectionReference defter = FirebaseFirestore.instance.collection('defter');
    await defter.add({
        'isim': _cIsim.text,
        'eposta': _cEposta.text,
        'tel': _cTel.text,
        'uid': FirebaseAuth.instance.currentUser?.uid
    }).then((value) => print("Ekleme Başarılı"));
    Navigator.pop(context,"");
}
```

Ekle fonksiyonu çalıştığında kullanıcının girmiş olduğu bilgiler kullanılarak Firebase Cloud Firestore'da defter koleksiyonunun içine yeni doküman eklenmektedir. Bu dokümanda ise isim, eposta, tel ve uid alanları yer almaktadır. Bu noktada uid parametresi önemlidir. Uid parametresine FirebaseAuth.instance.currentUser?.uid değeri verilmektedir. Bu değer uygulamaya giriş yapmış olan kullanıcının id'sidir. Böylece girilen kişi bilgilerinin hangi kullanıcıya ait olduğu belirlenmiş olur. Aslında Cloud Firestore veritabanında klasik ilişkisel veritabanında kullanılabilen bir yapıya benzer bir yapı oluşturulmuştur. İlişkisel veritabanları dü-

şünündüğünde uid değeri yabancı anahtar (foreign key) olarak değerlendirilebilir. Document Id'de aslında o belgenin veya kaydın birincil anahtarı (primary key) olarak değerlendirilebilir. Bu örnekte document id değeri özellikle yazılmadığı için otomatik eşsiz bir değer Cloud Firestore tarafından üretilmektedir.

Defter sayfasındaki listeden bir isim üzerine tıklandığında Kayıt Düzenle sayfası açılmaktadır. Kayıt düzleme sayfası arka planda Detay sınıfı tarafından yönetilmektedir. ListView'da listelenen kişinin isminin üzerine tıklandığında ilgili kişiye ait Kişi nesnesi kisiler[index] şeklinde DetaySayfasi fonksiyonuna gönderilmesi gerekmektedir. Bu fonksiyon ise Detay sayfasının açılmasından sorumludur.

```
child: ListView.builder(
  itemCount: kisiler.length,
  itemBuilder: (context, index) {
    return ListTile(
      title: Text(kisiler[index].isim),
      onTap: () {
        DetaySayfasi(kisiler[index]);
      },
    );
  },
),
```

Detay sınıfının kurucu fonksiyon tanımında Kişi parametresi bulunmaktadır. Dolayısıyla kayıt düzleme sayfası açılırken Detay sınıfının kurucu fonksiyonuna ilgili kişi bilgilerini barındıran Kişi nesnesinin gönderilmesi gerekmektedir. Detay(kisi: k) ile kişi nesnesi Detay sınıfının kurucu fonksiyonuna iletilemektedir.

```
Future<void> DetaySayfasi(Kisi k) async{
  await Navigator.push(
    context,
    MaterialPageRoute<void>(
      builder: (context) => Detay(kisi: k),
    ),
  );
  getir();
}
```

Böylece Detay sayfası açıldığı anda düzenlenecek kişi bilgilerine sahip olmuş olmaktadır. Bu durumda Detay sınıfının kişi bilgisini kabul edebilmesi için Detay sayfasının kurucu fonksiyonu da aşağıdaki gibi tanımlanması gerekmektedir.

```
class Detay extends StatefulWidget {
  const Detay({Key? key, required this.kisi}) : super(key: key);
  final Kisi kisi;
```

Kayıt Düzenle Sayfasını Oluşturma

Kayıt düzenle sayfasının ekran görüntüsü Görsel 8.10'de verilmiştir. Kayıt düzenle sayfasında, kayıt düzenleme, kayıt silme ve kişinin telefon numarasını arama işlemleri yapılmaktadır. Bu işlemlerin yapılması için formun altın 3 adet düğme yerleştirilmiştir.



GörSEL 8.10 Kayıt Düzenle sayfası

İlk olarak kişiye ait telefon numarasını arama işlemi açıklanmıştır. Aslında burada uygulanan işlem uygulama içinden doğrudan telefon etme durumu değildir. Onun yerine cep telefonunun Telefon uygulamasını açmak ve aranacak telefon numarası bölümünde ilgili kişinin telefon numarasının gelmesini sağlamaktır. Bunun sonucunda kullanıcının Telefon uygulamasından sadece Ara düğmesine basması yeterli olacaktır. Telefon ile istenilen telefon numarasını arayabilmek için url_launcher eklentisine ihtiyaç bulunmaktadır. Bu eklenti “flutter pub get url_launcher” komutu ile projeye eklenebilir. Komut çalıştırıldığın sonra pubspec.yaml dosyasında dependencies bölümünde ilgili satır eklenmektedir. Sonrasında ise gerekli import satırları yazıldıktan sonra Ara düğmesi ile ilişkilendirilen aşağıdaki fonksiyonu çalıştırılmak yeterli olmaktadır.

```
_ara() async {
  var url = "tel:"+ _cTel.text;
  await launch(url);
}
```

Uygulama üzerinden telefon etmek için launch(“tel: 0555 555 55 55”) şeklinde launch fonksiyonunu çağırma gerekmektedir. Örnekte de benzer şekilde “tel:” ifadesi ile kullanıcının telefon numarası birleştirilerek bu fonksiyona parametre olarak veildmiştir. Bu durumda ilgili işletim sistemi Telefon uygulamasına geçiş yapmakta ve bu numara ekranda belirmektedir.

Sensör kullanımına örnek olması açısından kullanıcıların telefonu sallayarak ta arama yapabilmeleri sağlanmıştır. Telefonun salladığının tespit edilebilmesi için shake eklentisinden yararlanılmıştır. Bu eklenti sayesinde telefonun sallanıp sallanmadığı çok kolaylıkla tespit edilebilmektedir. Shake eklentisi “flutter pub get shake” komutu ile projeye eklenebilir. Komut çalışıktan sonra pubspec.yaml dosyasında dependencies bölümünde ilgili satır eklenmektedir.

```
void initState() {
super.initState();
_cEposta.text = widget.kisi.eposta;
_cTel.text = widget.kisi.tel;
_cIsim.text = widget.kisi.isim;
detector = ShakeDetector.autoStart(onPhoneShake: () {
    _ara();
    detector.stopListening();
});
}
```

shake paketi import edildikten sonra ShakeDetector ile telefonun sallanıp sallanmadığı tespit edilebilmektedir. Örnekte telefon sallandığında _ara fonksiyonu çağrılmaktadır. Sallantı olayını tesip etme ile ilgili satırlar override yapılan initState fonksiyonu içine yazılmıştır. Böylelikle Detay sınıfına ait nesne oluşturulduktan hemen sonra bu satırlar çalışacak ve telefonun sallantı durumu takip edilmeye başlanacaktır. Telefonun sallanması durumunda _ara() fonksiyonu çalıştırılmakta ve detector.stopListening() ile de dinleme olayı sonlandırılmaktadır. Shake eklentisi aslında arka planda jiroskop sensörünü kullanmaktadır. Doğrudan jiroskop sensörünü kullanarak ta benzer bir sonuç elde etmek mümkündür. Telefonlardaki jiroskop sensörünün kontrolü ise sensor_plus eklentisi ile yönetilebilmektedir. Bu eklenti “flutter pub get sensor_plus” komutu ile projeye eklenebilir.

```
import 'package:sensors_plus/sensors_plus.dart';
accelerometerEvents.listen((AccelerometerEvent event) {
    print(event);
});
userAccelerometerEvents.listen((UserAccelerometerEvent event) {
    print(event);
});
gyroscopeEvents.listen((GyroscopeEvent event) {
    print(event);
});
magnetometerEvents.listen((MagnetometerEvent event) {
    print(event);
});
```

jiroskop sensöründen gelen bilgiler yukarıda belirtildiği gibi belirlenecek bir fonksiyon tarafından işlenebilir. accelerometerEvents, yerçekimi etkileri de dahil olmak üzere telefonun hızını tanımlar. Basitçe ifade etmek gerekirse, telefonun belirli bir yönde hareket edip etmediğini anlamak için accelerometerEvents ile event değişkenine gelen değerler kullanılabilir. userAccelerometerEvents, accelerometerEvents ile

aynırı, ancak yerçekimini içermez. Bir başka deyişle, kullanıcının cihaz üzerindeki etkisi olarak düşünülebilirler. gyroscopeEvents, cihazın dönüşü ile ilgili bilgiler verir. MagnetometerEvents, cihazı çevreleyen ortamındaki manyetik alan ile ilgili bilgiler verir. Pusula, bu verilerin örnek bir kullanımıdır. Sallama olayı ise iki tür gerçekleştirebilir: kullanıcı telefonu döndürerek sallayabilir veya ileri geri hareket ettirerek sallayabilir. Bu durumda sallama oyununu en iyi şekilde tespit edebilmek için userAccelerometerEvents ve gyroscopeEvents olaylarının birlikte kullanılması uygun olacaktır. Bu örnekte bu detaylara girilmemiştir. Ancak internette yapılacak basit bir arama ile bu sensor_plus paketinin nasıl kullanılabileceği ile ilgili çok sayıda kaynağı ulaşmak mümkün olabilmektedir.

Kayıt düzenleme sayfasında Düzenle düğmesine basıldığında ise aşağıdaki `_duzenle()` fonksiyonu çalışmaktadır. `_duzenle()` fonksiyonu await ifadesi içerdığı için async olarak tanımlanmıştır. Aslında bu fonksiyon içinde FirebaseFirestore ile işlem yapıldığı için hata oluşma riski bulunmaktadır. Bu bağlamda yazılan kodların try on catch mekanizması içinde olması çok daha mantıklı olurdu. Internet bağlantı problemi, Firebase login problemi gibi sorunlar rahatlıkla tespit edilip, uygulamanın olası hataları bertaraf edilebilirdir. Ancak bu örnekte bu detaylara girilmemiştir.

```
Future<void> _duzenle() async {
    CollectionReference defter = FirebaseFirestore.instance.
    collection('defter');
    await defter.doc(widget.kisi.kod).update({
        'isim': _cIsim.text,
        'eposta': _cEposta.text,
        'tel': _cTel.text
    }).then((value) => print("Düzenleme Başarılı"));
    Navigator.pop(context, "uuu");
}
```

Veritabanındaki kaydı güncellemek için `doc().update` fonksiyonundan yararlanılmaktadır. Burada önemli olan Kisi sınıfındaki kod özelliğidir. Defter sayfasında kişi bilgileri veritabanından çekilirken firebase tarafından oluşturulan otomatik document id değeri de alınmış ve Kisi sınıfının kod özelliğine aktarılmıştır. Bu sayede document id özelliği kullanılarak kişi üzerinde düzenleme ve silme işlemleri kolaylıkla yapılabilmektedir. Bu sayfada dikkat edilecek bir başka nokta ise `widget.kisi.kod` değeridir. Bu kod `_DetayState` sınıfında çalışmaktadır. Ancak kisi özelliği Detay sınıfına aittir. Bu durumda `_DetayState` sınıfının `widget` özelliği üzerinden Detay sınıfının özelliklerine erişilebilmektedir. Bu örnekte de widget üzerinden Detay sınıfının kisi özelliğine erişim sağlanmıştır. Aşağıda da sil düğmesine tıklandığında çalışan kodlara yer verilmiştir. Burada da benzer olarak kodların try on catch mekanizması içinde olması hataların bertaraf edilebilmesi açısından önemlidir.

```
Future<void> _sil() async {
    CollectionReference defter =
        FirebaseFirestore.instance.collection('defter');
    await defter.doc(widget.kisi.kod).delete().then((value) =>
        print("Silme Başarılı"));
    Navigator.pop(context, "");
}
```

Silme işlemi de doc(widget.kisi.kod).delete() fonksiyonu ile gerçekleştirilmektedir. Benzer şekilde widget.kisi.kod değerinden yararlanılmıştır. Bu kod çalıştırıktan sonra silme işlemi gerçekleştirilerek kullanıcı tekrar Defter sayfasına yönlendirilmektedir.



Öğrenme Çıktısı



- 2 Telefon defteri uygulaması için gerekli widgetları oluşturabilme
- 3 Firebase'de kimlik doğrulama ve veri tabanı işlemlerini yapabilme
- 4 Gyro sensörünü kullanmak için gerekli paketleri kurarak, ilgili konutlar ile telefonun sallanma durumunun tespit edilerek arama işlemini başlatabilme

Araştır 2

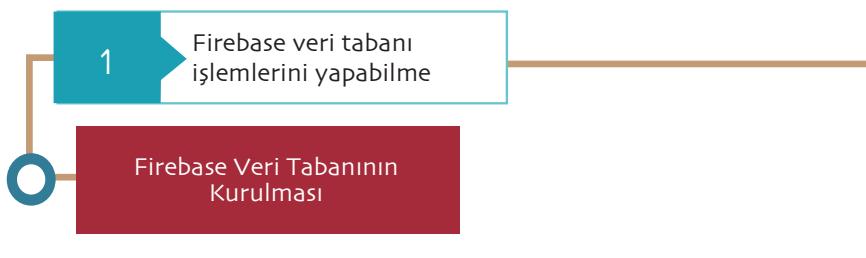
Telefon defteri uygulamasında Firebase veritabanı yerine Sqlite veritabanı sisteminin kullanımının avantaj ve dezavantajları nelerdir?

İlişkilendir

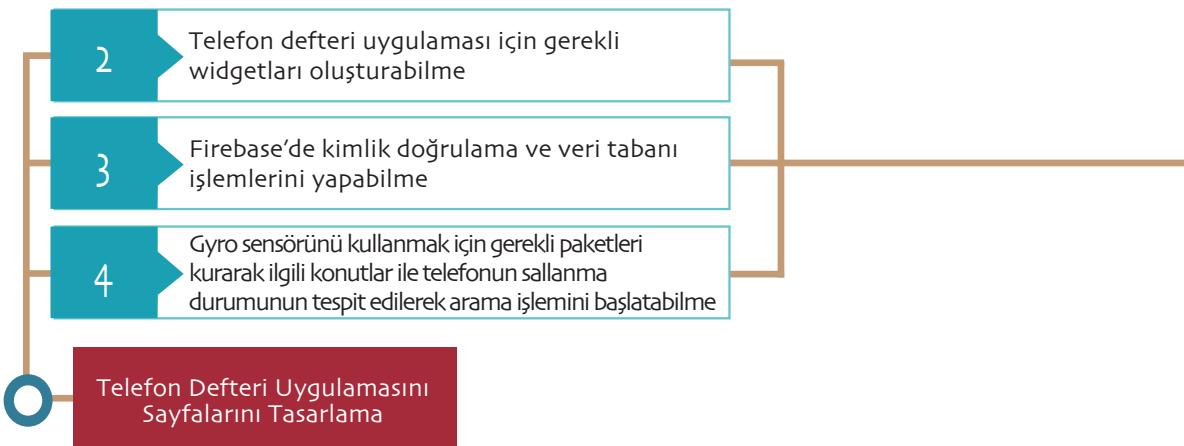
Firebase veri tabanı ve Sqlite veri tabanları birlikte kullanılabilir miydi? Tartışın.

Anlat/Paylaş

Özel widgetlar oluşturmanın yararları nelerdir? Açıklayın.



Firebase, “uygulamanızı oluşturmak, geliştirmek ve büyütmek” için Google tarafından sunulan bir araç setidir. Firebase analitik, kimlik doğrulama, veritabanları, yapılandırma, dosya depolama, push mesajlaşma gibi şeyleri içerir ve uygulama geliştiricilerinin bu gibi konularda çok daha hızlı entegrasyon yapabilmelerini sağlar. Bu projede Firebase’ın kimlik doğrulama ve Cloud Firestore veritabanı özelliklerinden yararlanılmıştır. Firebase veritabanını kullanabilmek için öncelikle Google hesabının olması gerekmektedir. firebase.google.com adresinden uygulamada kullanılacak projeler oluşturulup yönetilebilmektedir. Authentication geliştirilecek uygulamada kullanıcı girişlerini yönetmek için kullanılan modüldür. Authentication modülü çok sayıda farklı giriş yöntemini desteklemekle birlikte bu örnekte Email/Password seçeneği tercih edilmiştir. Cloud Firestore NoSQL türünde belge odaklı bir veritabanıdır. NoSQL veritabanları ilişkisel veritabanı yönetim sistemlerinden farklı olarak verilerin tablo şeklinde depolanmadığı veri tabanlarıdır. NoSQL veritabanları, veri modellerine göre çeşitli türlerde olabilmektedir. NoSQL veritabanları esnek şemalar sağlarlar ve büyük miktarda veri ve yüksek kullanıcı yükleriyle kolayca ölçeklenirler. Cloud Firestore veri tabanında da tablo veya satır yoktur. Bunun yerine, verileri koleksiyonlar halinde düzenlenen belgelerde depolanır. Her belge bir dizi anahtar/değer çiftini içerir. Tüm belgeler koleksiyonlarda saklanmalıdır. Belgeler, her ikisi de dizeler gibi ilkel alanlar veya listeler gibi karmaşık nesneler içerebilen alt koleksiyonlar ve iç içe nesneler içerebilir. Koleksiyonlar ve belgeler, Cloud Firestore’da dolaylı olarak oluşturulur.



İlk olarak Android Studio'da yeni bir flutter projesi oluşturulduktan sonra firebase_core kütüphanesi projeye eklenmelidir. Firebase kütüphanesi projeye eklendikten sonra uygulama ile firebase.google.com'da oluşturulan projenin ilişkilendirilmesi gereklidir. Bu noktada FlutterFire uygulamasından yararlanılabilir. Bu uygulama ile Terminal'de birkaç satır koy yazarak bu ilişkilendirme işlemi tamamlanabilir. İlişkilendirme işlemi tamamlandıktan sonra projede Firebase'in Cloud Firestore ve Authentication özelliklerinin çalışabilmesi için cloud_firestore ve firebase_auth eklentilerinin projeye eklenmesi gerekmektedir. Sonrasında ise Giriş ve Kayıt Ol sayfaları yapılarak kullanıcıların sisteme giriş yapabilmeleri ve eğer daha önce üye olmadılarsa da üye olmaları sağlanmaktadır. Projede sık kullanılan yapılar, ayrı bir dosyada StatelessWidget sınıfı olarak oluşturulmuştur. Bu tür widgerler sayesinde ekranda benzer görüntüyü elde etmek için çok daha az kod yazmak yeterli olmaktadır. Örneğin projedeki tüm metin kutuları CustomFormField olarak tasarlanmıştır. Defter sayfasında kişiler Cloud Firestore veritabanından okunarak bir ListView'da listelenmektedir. Bunun için Firestore eklentisinin ilgili komutları ile veritabanı sorgulamaları gerçekleştirilebilmektedir. Yeni kayıt oluşturma sayfasında kullanıcı telefon defterine yeni bir kişi eklemektedir. Kayıt düzende sayfasında ise kullanıcı kişiye telefon etme, kişi kaydını düzenleme ve silme işlemlerini gerçekleştirebilmektedir. Flutter'da Telefon uygulamasını açabilmek için url_launcher eklentisine ihtiyaç vardır. Sensör kullanımına örnek olması açısından bu bölümde telefon sallandığında telefon edebilme özelliği eklenmiştir. Flutter'da telefonun sallanmasını tespit edebilmek için shake eklentisini ihtiyaç vardır.

1 Aşağıdakilerden hangisi Firebase sisteminde kullanılan araçlardan biri **değildir**?

- A. Web sayfası barındırma
- B. Kimlik doğrulama
- C. Veritabanları
- D. Dosya depolama
- E. Analitik

2 Aşağıdakilerin hangisinde verilere erişmek için Yapılandırılmış Sorgu Dili (SQL) kullanılır ve veriler tablolar biçiminde depolanır?

- A. NoSQL veritabanları
- B. İlişkisel Veritabanı Yönetim Sistemleri
- C. Sosyal ağlar
- D. Firebase
- E. Cloud Firestore

3 Firebase sisteminin sahibi aşağıdakilerden hangisidir?

- A. Apple
- B. Oracle
- C. MySQL
- D. Google
- E. Microsoft

4 Aşağıdakilerden hangisi, Firebase Authentication'da alternatif kullanıcı giriş seçeneklerinden biri **değildir**?

- A. Facebook
- B. Apple
- C. Microsoft
- D. Google
- E. Kurumsal sistemler

5 Cloud Firestore'da veriler aşağıdaki birimlerin hangisinde depolanır?

- A. Tablo
- B. Kayıt
- C. Belge
- D. Grafik
- E. Alan

6 Cloud Firestore ile ilgili aşağıdaki ifadelerden hangisi **yanlıştır**?

- A. SQL ile sorgulama yapılabilir.
- B. Cloud Firestore şemasızdır.
- C. Aynı koleksiyondaki belerler farklı alan ve veri türleri içerebilir.
- D. Belgelerin içinde alt koleksiyonlar bulunabilir.
- E. Koleksiyonların oluşturulmasına veya silinmeye gerek yoktur.

7 Flutter projesi ile Firebase projesini ilişkilendirmek için kullanılan yardımcı yazılımın adı aşağıda-kilerden hangisidir?

- A. Custom Firebase
- B. Firebase Core
- C. Authentication
- D. Cloud Firestore
- E. FlutterFire

8 Flutter'da Firebase kimlik doğrulamasını kullanabilmek için aşağıdaki eklentiden hangisine ihtiyaç vardır?

- A. Cloud_firestore
- B. Firebase_auth
- C. Authentication
- D. Flutter pub get
- E. Firebase_options

9 Bir StatefulWidget nesnesi ağaca eklendiğinde bir kez çağrılan fonksiyon aşağıdakilerden hangisidir?

- A. CallBack
- B. Push
- C. InitState
- D. PostFrameCallBack
- E. Future

10 Bir fonksiyon içinde await ifadesinin kullanılması ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

- A. Try on catch mekanizması içinde kullanılması şarttır.
- B. Fonksiyon çağrılrken async ifadesi ile çağrılmalıdır.
- C. Fonksiyon mutlaka geriye bir değer döndürmelidir.
- D. Bu fonksiyon çalıştırıldığında uygulama kilitlenir ve fonksiyonun tamamlanması beklenir.
- E. Fonksiyonun Future<T> türünde ve async olarak tanımlanması gereklidir.

- | | | | |
|------|--|-------|---|
| 1. A | Yanıtınız yanlış ise “Giriş” konusunu yeniden gözden geçiriniz. | 6. A | Yanıtınız yanlış ise “Firebase Veritabanının Kurulması” konusunu yeniden gözden geçiriniz. |
| 2. B | Yanıtınız yanlış ise “Giriş” konusunu yeniden gözden geçiriniz. | 7. E | Yanıtınız yanlış ise “Telefon Defteri Flutter Projesini Oluşturma” konusunu yeniden gözden geçiriniz. |
| 3. D | Yanıtınız yanlış ise “Firebase Veritabanının Kurulması” konusunu yeniden gözden geçiriniz. | 8. B | Yanıtınız yanlış ise “Telefon Defteri Flutter Projesini Oluşturma” konusunu yeniden gözden geçiriniz. |
| 4. E | Yanıtınız yanlış ise “Firebase Veritabanının Kurulması” konusunu yeniden gözden geçiriniz. | 9. C | Yanıtınız yanlış ise “Telefon Defteri Flutter Projesini Oluşturma” konusunu yeniden gözden geçiriniz. |
| 5. C | Yanıtınız yanlış ise “Firebase Veritabanının Kurulması” konusunu yeniden gözden geçiriniz. | 10. E | Yanıtınız yanlış ise “Telefon Defteri Flutter Projesini Oluşturma” konusunu yeniden gözden geçiriniz. |

8

Araştır Yanıt
Anahtarı

Araştır 1

Firebase, Google tarafından sunulan bir yazılım geliştirme platformudur. Alt yapıyı yönetmeden hızlı bir şekilde web uygulamaları oluşturmaya yardımcı olur. Platformun NoSQL veritabanı, tüm verilerin koleksiyonlarda ve belgelerde güvenli bir şekilde saklanması sağlar. Firestore veritabanı ilişkisel değildir; daha esnek ve ölçeklenebilir yapısıyla büyük miktarda veri depolamak için daha iyidir. Firebase, yazılım geliştirme döngüsünü daha kısa ve daha basit hale getirir. Firebase Google'a ait olduğu için Google Analytics gibi ürünleriyle kolayca entegre olabilmektedir. Firebase, uygulamaları daha kişisel hale getirmek için kullanıcıların davranışını izlemeye olanak tanır. Müşteri davranışları analiz edilerek, yeni özellikler ekleme konusunda kararlar verebilmeyi ve böylece müşteri deneyimini ve müşteriyi elde tutmayı iyileştirebilmeye yardımcı olabilir. Firebase, BT topluluğunda geniş bir takipçi kitlesine sahiptir ve yeni öğrenmek isteyenler için yeterli teknik belgeleri bulunmaktadır. Yeni öğrenenler takıldıkları sorularına çevrimiçi topluluklara danışarak rahatlıkla cevap bulabilirler. Esnek ve kabul edilebilir bir ücretlendirme politikası vardır. Kapsamlı belgeler, önceden hazırlanmış API'ler, basit UI'ler ve diğer özellikleri ile entegrasyonu ve kurulumu basittir. Karmaşık konfigürasyonlar yapmaya gerek kalmadan uygulamalar oluşturmak mümkün hale gelmektedir. Firebase push notifications özelliği ile uygulamanın kullanıcılarına bildirim göndermek için araçlar sunmaktadır. Firebase'in zayıf yönleri arasında ise sınırlı veri tabanı sorgulama kapasitesi, klasik veri tabanları ile uyumsuzluk ve data migration yapmanın zor olması, iOS desteğinin Android kadar gelişmiş olmaması sayılabilir.

Araştır 2

Sqlite yerel bir veritabanıdır. Sqlite kullanıldığından kullanıcı girişine gerek kalmayacaktır. Geliştirilen telefon defteri sadece kurulduğu telefonda çalışacaktır. Firebase'de olduğu gibi her seferinde internete bağlanıp veri çekmek zorunda olmadığı için çok daha hızlı çalışacaktır. Ancak sadece Sqlite kullanılması durumunda verilerin başka bir telefona aktarılması çok zorlaşacaktır. Hatta telefonun bozulması durumunda verileri tamamen kaybolma ihtimali bulunmaktadır. Aslında Firebase ile birlikte bir de yerel bir Sqlite veri tabanı kullanmak çok mantıklı bir seçenek olacaktır. Halihazırda geliştirilen uygulama her seferinde Firebase veri tabanına bağlanmakta tüm kişi listesini indirmektedir. Daha gelişmiş bir uygulamada daha fazla alan, daha fazla veri ve resim gibi ek özelliklerde bulunabilir. Buna ek olarak kayıt sayısının 100'lerle ifade edilecek bir sayıya ulaşması durumda indirilecek veri miktarı çok fazla olacak ve her seferinde bu verilerin indirilmesi bant genişliğinin gereksiz kullanımına gelecektir. Ayrıca yüksek miktarda veri indirildiği için kullanıcını bekleme süresi artacak ve Firebase tarafında da fazla veri kullanımından dolayı ek ücretlerin oluşmasına sebep olacaktır. Bunun yerine veriler Sqlite veritabanında yerelde saklanabilir ve Firebase sadece senkronizasyon için kullanılabilir. Yeni bir kişi eklendiğinde, kişi silindiğinde veya kişi düzenlenliğinde senkronizasyon işlemi gerçekleştirilebilir. Bu işlemler dışındaki tüm süreçler lokal veritabanı üzerinden gerçekleştirilir. Kullanıcı telefon değiştirdiğinde yapması gereken tek şey kullanıcı adı ve şifresi ile yeni sistem giriş yapmak olacaktır. Tüm verilerin bir kopyası Firebase'de saklandığı için ilk giriş sırasında bir senkronizasyon işlemi ile Firebase'deki veriler ile local Sqlite veritabanı oluşturulabilecektir.

Kaynakça

- Alessandria, S. (2013). *Flutter projects: A practical, project-based guide to building real-world cross-platform mobile applications and games*. Pact.
- Alessandria, S. ve Kayfitz, B. (2021). *Flutter cookbook: Over 100 proven techniques and solutions on mobile development with flutter 2.2 and dart*. Pact.
- Biswas, N. (2022). *Beginning React and Firebase*. Apress. <https://doi.org/10.1007/978-1-4842-7812-3>
- Clow, M. (2018). *Learn Google flutter fast*. Independently Published.
- Domes, S. (2017). *Progressive web apps with react: Create lightning fast web apps with native power using React and Firebase*. Pact.
- Katz, M., Moore, K. D., Ngo, V. ve Guzzi, V. (2021). *Flutter apprentice: Learn to build cross-platform apps*. Raywenderlich.
- Kumar, A. (2018). *Mastering Firebase for Android Development*. Pact. www.packtpub.com%0APacktPub.com
- Moroney, L. (2017). *The definitive guide to Firebase*. Apress. <https://doi.org/10.1007/978-1-4842-2943-9>
- Napoli, M. L. (2019). *Beginning flutter: A hands on guide to app development*. Wrox. <https://doi.org/10.1002/9781119550860>
- Prajyot Mainkar ve Giordano, S. (2019). *Google flutter mobile development: Quick start guide*. Pact.
- Windmill, E. (2019). *Flutter in action*. Manning.
- Zammetti, F. (2019). *Practical flutter: Improve your mobile development with Google's latest open-source SDK*. Apress.