### **Bölüm 2**

### **Flutter'da Geliştirme**

Önceki bölümde gördüğümüz gibi, Flutter bize Web, masaüstü bilgisayarlar ve mobil cihazlarda (ana çekicilik gibi görünen) çalışan uygulamalar oluşturma olanağı sağlar. Ama bir dakika, bu uygulamaları tam olarak nasıl oluşturuyoruz? Hangi editörü kullanmalıyız? Bir Flutter projesinde neler bulunmalıdır? Dart kaynak kodunu nasıl derleriz? Projeyi desteklemek için başka araçlara ihtiyacımız var mı? Test etmek için bir tarayıcıya veya cihaza nasıl aktarırız? Güzel sorular, değil mi?

Bu bölümde bu sorulara ve daha fazlasına yanıt vereceğiz. İki önemli konuyu ele alacağız:

1. Gerekli Araçlar – Bunları nasıl yükleyip yöneteceğimiz
2. Geliştirme Süreci – Uygulamayı nasıl oluşturup çalıştıracağımız ve nasıl hata ayıklayacağımız

|  |
| --- |
| **Dikkat** Çapraz platform uygulama geliştirme araçları, çeşitli organizasyonlardan gelen ve birbirleriyle koordinasyon sağlamadan değişiklik yapan birçok bileşeni içerir. Ayrıca, gelişmekte olan bir teknolojiyle çalıştığımız için değişiklikler sık sık olur. Bu bölümde zaman aşımına uğramayacak bilgiler vermeye çalıştık, ancak yine de zamanla güncelliğini yitirebilir. Lütfen en yeni bilgiler için ilgili araçların resmi kaynaklarını kontrol edin. |

© Rap Payne 2019 9

R. Payne, Beginning App Development with Flutter, 9 <https://doi.org/10.1007/978-1-4842-5181-2_2>

## **Flutter Araç Zinciri**

Geliştirici topluluğunun ürettiği faydalı araçların sonu yok. Bu gerçekten bunaltıcı olabilir. Hepsini kapsamaya çalışmıyoruz. Size yeterli olacak kadar bilgi vermeyi amaçlıyoruz, ancak sizi aşırı yüklemek istemiyoruz. Eğer favori bir aracınızı atladıysam, affedin.

### **Flutter SDK**

Flutter SDK, vazgeçilmez tek araçtır. Flutter derleyicisi, proje oluşturucu, cihaz yöneticisi, test çalıştırıcı ve Flutter yapılandırmasıyla ilgili sorunları teşhis eden (ve hatta düzelten) araçları içerir.

### **Flutter SDK’nın Kurulumu**

Kurulum talimatlarını şu adreste bulabilirsiniz: <https://flutter.dev/docs/get-started/install> Kısaca özetlemek gerekirse, en son araç setinin sıkıştırılmış dosyasını indirmeniz ve açtığınız klasörü PATH değişkeninize eklemeniz gerekecek. Adımlar işletim sistemine göre değişir, ancak ilgili web sitesinde çok açık bir şekilde belirtilmiştir.

**İpucu:**  
Bu adım düşük seviyeli ve göz korkutucu görünebilir, ancak bundan sonra işler daha kolay ve daha az hata eğilimli hale gelir. Sakın gözünüzü korkutmasına izin vermeyin. 

## **IDE’ler**

Teoride bir IDE’ye gerçekten ihtiyacımız yoktur. Flutter, herhangi bir editörle yazılabilir ve ardından yüklediğiniz Flutter SDK ile derlenip çalıştırılabilir. Ancak gerçekte bunu yapan neredeyse hiç kimse yoktur. Neden yapsınlar ki? Aşağıdaki IDE’ler, Flutter desteğiyle birlikte gelir!

10

### **Microsoft’tan VS Code**

VS Code, Microsoft tarafından geliştirilmiştir. Resmi adı "Microsoft Visual Studio Code" olup, çoğumuz ona "VS Code" deriz. Ancak lütfen bu aracı, Microsoft'un başka bir ürünü olan "Microsoft Visual Studio" ile karıştırmayın. İsimleri benzer olsa da aynı şey değiller.

VS Code’u şu adresten edinebilirsiniz: [https://code.visualstudio.com](https://code.visualstudio.com/)

### **JetBrains’ten Android Studio / IntelliJ**

Android Studio ve IntelliJ, temelde aynı şeydir. Aynı kod tabanından türetilmiş olup, aynı özelliklere sahiptirler.

Android Studio’yu şu adresten edinebilirsiniz: <https://developer.android.com/studio> IntelliJ IDEA’yı şu adresten edinebilirsiniz: [www.jetbrains.com/idea/download](https://www.jetbrains.com/idea/download)

## **Hangi IDE’yi Kullanmalıyım?**

Hem VS Code hem de Android Studio/IntelliJ ücretsiz ve açık kaynaklıdır. Windows, Mac ve Linux’ta çalışırlar. Flutter geliştiricileri arasında popülerlikleri benzerdir ve biri diğerine üstün değildir. Hangisini seçerseniz seçin, yanlış yapmış olmazsınız.

Ancak seçim yapmak zorundaysanız, geçmişinizin hangi aracı daha çok seveceğinizi etkileyebileceğini unutmayın: Web geliştirme dünyasından gelenler (HTML, CSS, JavaScript, NodeJS, React, Angular, Vue gibi teknolojileri kullananlar) genellikle VS Code tercih eder. Java tabanlı bir geçmişe sahip olanlar, özellikle Android geliştiricileri, Android Studio/IntelliJ kullanma eğilimindedir.

Bu seçim çok önemli değildir. Bir proje üzerinde çalışırken bile editör değiştirmek oldukça kolaydır. Birini deneyin, sevmezseniz diğerine geçin. Geri dönüp tekrar deneyin, en rahat hissettiğiniz IDE’yi seçin. Değiştirmek büyük bir mesele değildir.

  
Flutter ekibinden Andrew Brogdon (@redbrogdon) tarafından Flutter geliştiricileri arasında yakın zamanda yapılan bir anket, %53'ünün VS Code, %30'unun Android Studio ve %15'inin IntelliJ kullandığını gösterdi. Bkz. <http://bit.ly/flutter_devtools_poll>

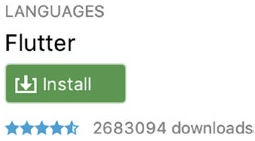
11

nasıl sevdiğinizi. Eğer beğenmezseniz, diğerini bir süre test sürüşüne çıkarabilirsiniz. Güçlü bir tercih yapana kadar birkaç kez ileri geri gidin. Değiştirmek gerçekten büyük bir mesele değil.

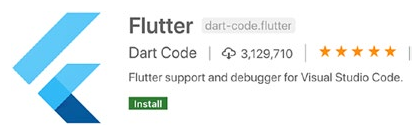
## IDE DevTools

Bu IDE’ler harika olsa da, Flutter için özel olarak tasarlanmamışlardır; başka diller ve framework'ler için de kullanılırlar. **Flutter geliştirme sürecini daha iyi hale getirmek için Flutter DevTools’u yüklemeliyiz.** Flutter DevTools şu avantajları sağlar: Debugger desteği , Logları görüntüleme , Emülatörlerle kolay entegrasyon ,Ekstra hata ayıklama araçları

DevTools'un yüklenmesi her IDE'nin içinden yapılır. Android içinde Studio/IntelliJ, ana menüden “Tercihler ➤ Eklentiler” seçeneğine gidin (Şekil 2-1). VS Code'da “Görünüm ➤ Uzantılar ”a gidin (Şekil 2-2). Flutter devtools basitçe “Flutter” olarak adlandırılır ve bir arama onları ortaya çıkaracaktır. Her iki platformda da yeşil “Yükle” düğmesine basın.



Şekil 2-1. Android Studio'da DevTools kurulumu



Şekil 2-2. VS Code'da DevTools kurulumu

Kurulumdan sonra IDE'yi yeniden başlatmanız gerekebilir.

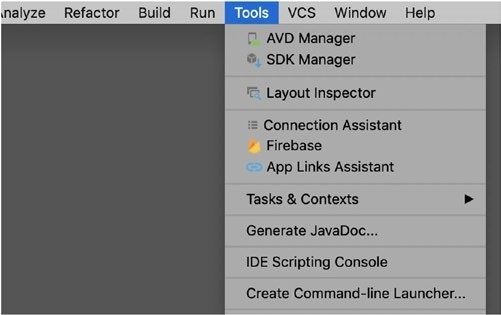
12

## **Android Emülatörü**

Tıpkı tonlarca Android modeli olduğu gibi, tonlarca Android emülatörü de var, ancak bunlarla etkileşim kurmanın yalnızca iki popüler yolu var: Genymotion ve AVD Manager. Genymotion kâr amacı güden bir şirkettir, bu nedenle web sitelerini ziyaret ettiğinizde sizi ücretli sürümlerine yönlendirmek için ellerinden yapacaklardır. Bu anlaşılabilir bir durum. Biz AVD Manager'a odaklanacağız çünkü tamamen ücretsiz ve Flutter geliştiricileri arasında daha popüler.

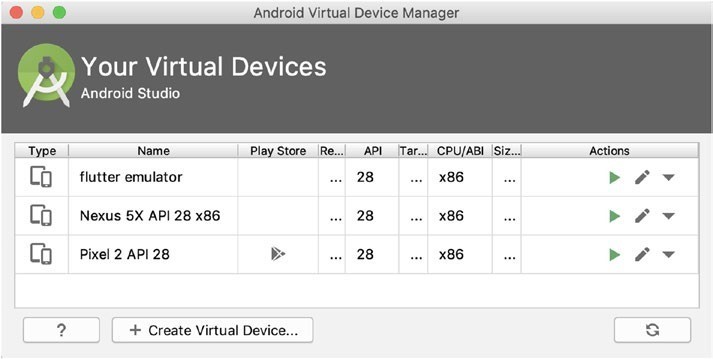
AVD, "Android Sanal Aygıt" anlamına gelir. AVD Yöneticisi Android Studio'da

Araçlar altında bulunur (Şekil [2-4](bookmark://_bookmark1)).



**Şekil 2-4.** Android Studio'da AVD Yöneticisini Bulma

Açıldığında, şu anda yüklü olan emülatörlerinizin bir listesini göreceksiniz. İlk başta boş olarak başlamalıdır. Alttaki "+ Sanal Cihaz Oluştur..." düğmesine basarak mevcut yüzlerce Android cihaz emülatöründen birini veya daha fazlasını yükleme olanağına sahip olacaksınız (Şekil [2-5](bookmark://_bookmark0)).



**Şekil 2-5.** AVD Manager mevcut cihazların bir listesine sahiptir. Daha fazla eklemek için "+" düğmesine tıklayın

Tıklayın ve her türlü cihaz arasından seçim yapabilir veya kendiniz bir tane oluşturabilirsiniz. Bir cihazı yalnızca bir kez yüklemeniz gerekir. Yüklendikten sonra, bu taklit cihaz IntelliJ/Android Studio veya VS Code gibi herhangi bir IDE'den kullanılabilir. VS Code üzerinde ayrı bir kuruluma gerek yoktur.

# **Araçların güncel tutulması**

İlk , Xamarin ve React Native gibi araçlarla çapraz platform geliştirme, ilgili araçların çokluğu ve aralarındaki bağımlılıklar nedeniyle son derece zordu. Hâlâ bunun acısıyla terapi görüyorum.

Ancak Flutter sahneye daha sonra çıktığı için diğerlerinin hatalarından ders çıkarabiliyor. Flutter ekibi, bu sorunlu noktaların farkına vararak bize araç zincirinin geri kalanını yönetmek için yenilikçi bir araç verdi. Geliştirme makinenizi inceleyerek Flutter uygulamalarını geliştirmek için ihtiyaç duyacağınız tüm araçları, sahip olduğunuz sürümleri, mevcut sürümleri ve bunlar arasındaki bağımlılıkları arayacak ve ardından sorunlara bir teşhis koyacaktır. Hatta bu sorunlara bir çözüm bile önerecek. Kulağa bir doktor gibi geliyor, değil mi? O zaman sizi Flutter Doctor ile tanıştırayım!

## **Flutter Doctor**

Komut satırından flutter doctor'ı çalıştıracaksınız. Araç zincirinizdeki tüm araçları kontrol eder ve karşılaştığı sorunları rapor eder. Burada Xcode'un yardıma ihtiyaç duyduğu bir durum var:

$ flutter doctor

Doctor summary (to see all details, run flutter doctor -v):

[✓] Flutter (Channel beta, vX.Y.Z, on Mac OS X X.Y.Z, locale

en-US)

[✓] Android toolchain - develop for Android devices (SDK

version X.Y.Z)

[ ! ] Xcode - develop for iOS and macOS (Xcode X.Y)

✘ **Xcode requires additional components to be installed in**

**order to run.**

**Launch Xcode and install additional required components**

**when prompted.**

[✓] Android Studio (version X.Y)

[✓] VS Code (version X.Y.Z)

[ ! ] Connected device

**! No devices available**

! Doctor found issues in 2 categories.

$

"Kullanılabilir cihaz yok" hatası yaygındır ve genellikle bunu görmezden gelebilirsiniz. Bu sadece o anda hiçbir emülatörün çalışmadığı anlamına gelir.

İşte görmeyi tercih ettiğimiz şeyin bir örneği - her şey kontrol

$ flutter doctor

Doctor summary (to see all details, run flutter doctor -v):

[✓] Flutter (Channel beta, vX.Y.Z, on Mac OS X X.Y.Z, locale

en-US)

[✓] Android toolchain - develop for Android devices (SDK

version X.Y.Z)

[✓] Xcode - develop for iOS and macOS (Xcode X.Y)

[✓] Android Studio (version X.Y)

[✓] VS Code (version X.Y.Z)

[✓] Connected device (1 available)

• No issues found!

flutter doctor yalnızca sorunları tespit edip bildirmekle kalmaz, aynı zamanda genellikle her biri için bir çözüm önerir. Hatta "flutter upgrade" aracılığıyla kendisini yükseltme zamanının geldiğini bile size söyleyecektir.

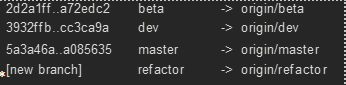
## **Flutter Geliştirmesi**

Evet, Flutter SDK'nın ilk kurulumu biraz göz korkutucuydu ancak geliştirme çok kolay. Kelimenin tam anlamıyla iki kelime yazacaksınız, "flutter upgrade":

$ flutter upgrade

Upgrading Flutter from /usr/local/bin/flutter...

From <https://github.com/flutter/flutter>



\*

<snip>

\* [new tag] v1.10.5 **->** v1.10.5

Updating c382b8e..a72edc2

11 files changed, 413 insertions( + ), 302 deletions ( - )

Building flutter tool...

Upgrading engine...

Downloading ios- deploy...

0.3s

Flutter X.Y.Z • channel beta • <https://github.com/flutter/>

flutter.git

Framework • revision a72e06 (23 hours ago) • 20XX-YY-ZZ

15:41:01 -0700

Engine • revision b863200c37

Tools • Dart X.Y.Z

Running flutter doctor...

Doctor summary (to see all details, run flutter doctor -v):

[✓] Flutter (Channel beta, vX.Y.Z, on Mac OS X X.Y.Z, locale

en-US)

[✓] Android toolchain - develop for Android devices (SDK

version X.Y.Z)

[✓] Xcode - develop for iOS and macOS (Xcode X.Y)

[✓] Android Studio (version X.Y)

[✓] VS Code (version X.Y.Z)

[✓] Connected device (1 available)

• No issues found!

Flutter doktorunun son adım olarak otomatik olarak çalıştırıldığını ve her şeyin yolunda olduğunu onayladığını unutmayın. Yükseltme işlemi çocuk oyuncağı.

**Flutter Geliştirme Süreci**

Artık tüm araçları yüklediğimize ve güncel olduğuna göre bir uygulama oluşturalım ve hata ayıklayıcıdan çalıştırın.

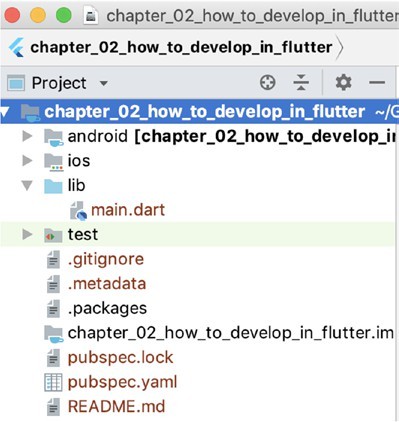
**Uygulamayı ve Dosyaları Oluşturma**

Çalıştırarak tamamen yeni bir Flutter uygulaması oluşturun...

$ flutter create my\_app

### **Bir Flutter projesinin anatomisi**

Yeni oluşturduğunuz projedeki tüm dosya ve klasörler hakkında bilgi sahibi olmanız kritik değildir. Ancak merak ediyorsanız, Şekil 2-6'da gösterilen yeni oluşturulmuş bir Flutter projesini hızlıca .



#### **Şekil 2-6.** Flutter create tarafından oluşturulmuş yeni bir Flutter projesi

Bu klasörlere sahip olacaksınız:

* android ve ios - Bunlar projenin platforma özgü kısımlarıdır. Çoğunlukla bunlara dokunmanız gerekmeyecektir.
* lib - Burası tüm Dart kaynak kodunuzun yuvasıdır. Uygulamanızın hiyerarşisini burada oluşturacaksınız. Neredeyse tüm zamanınızı ve dikkatinizi burada harcayacaksınız.
* test - Birim testleriniz varsa (ve muhtemelen sonunda

yapmalısınız), onları buraya koyun.

Ve bu dosyalara sahip olacaksın:

* pubspec.yaml - Bu aslında Dart projeleri için proje dosyasıdır. Proje adımızı, açıklamamızı, bağımlılıklarımızı ve daha fazlasını burada belirleriz.

Nelerin önerildiğini ve mümkün olduğunu daha iyi anlamak için buradaki yorumları okuduğunuzdan emin olun.

* .gitignore ve README.md - Bunlar kaynak kod depoları için git ve github kullanan geliştiricilere çok tanıdık gelecektir. Diğerleri umursamayacaktır.
* .metadata ve .packages - Bunlar asla açmayacağınız önemli yapılandırma dosyalarıdır. Ancak Flutter'ın bunlara ihtiyacı var.

**İpucu** Bilmeniz gereken bir dosya daha var: analysis-options. yaml. Bu dosyaya sahip olmak gerekli değildir, ancak sahip olursanız daha iyi kod yazarsınız. Bu dosya, siz kodu yazarken IDE'ye kod üzerinde linting (diğer adıyla statik analiz) yapması için sinyal gönderir. analysis-options.yaml ile, en iyi uygulamaları kullanmadığınızda IDE sizi uyaracaktır.

Sıfırdan bir tane yazmak yerine, başka birininkiyle başlamanızı önerebilir.

İşte Flutter'da çok popüler olan bir tane

Topluluk: <https://github.com/flutter/plugins/blob/>

master/analysis\_options.yaml.

Agresiftir. Bazı kuralları kapatmak istiyorsanız, satırları silmeniz veya yorum yapmanız yeterlidir. Tüm linting kurallarının bir açıklaması burada bulunabilir: http://dart-lang.github.io/linter/lints/.

## **Uygulamanızı çalıştırma**

Artık oluşturulmuş bir Flutter uygulamanız var. Hadi onu çalıştıralım. Uygulamanızı çalıştırmanın birden fazla yolu vardır. En popüler yol, Android Studio/IntelliJ veya VS Code'da yeşil "Play" düğmesine basmaktır. Bunu komut satırından "flutter run" kullanarak da yapabilirsiniz:

$ flutter run

Running "flutter pub get" in chapter\_02\_how\_to\_develop\_in\_

flutter... 0.5s

Launching lib/main.dart on iPhone X in debug mode...

Running Xcode build...

├─Assembling Flutter resources... 6.1s

└─Compiling, linking and signing... 5.9s

Xcode build done. 13.8s

Syncing files to device iPhone X... 1,852ms

**Çalışırken değişiklikleri yeniden yüklemek için "r" tuşuna basın. Sıcak yeniden başlatmak (ve durumu yeniden oluşturmak) için "R" tuşuna basın.**

iPhone X üzerinde bir Observatory hata ayıklayıcı ve profil oluşturucu

şu adreste mevcuttur: http://127.0.0.1:52550/8m0h8zacV58=/

Daha ayrıntılı bir yardım mesajı için "h" tuşuna basın. Ayrılmak için "d" tuşuna basın; çıkmak için "q" tuşuna basın.

Ancak IDE'nizdeki yeşil Oynat/Ayıkla düğmesine basarsanız (Şekil 2-7), kesme noktaları belirleyerek ve geliştirici araçlarını kullanarak kodda adım adım ilerleyerek uygulamanızda hata ayıklama seçeneğine sahip olursunuz (Şekil 2-8).

metin, ekran görüntüsü, yazı tipi, logo içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

#### **Şekil 2-7.** Android Studio'da Oynat ve Hata Ayıkla düğmeleri en üsttedir

*metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.*

***Şekil 2-8.*** *Oynat düğmesi VS Code'da sol üsttedir*

Açıkçası uygulamanızı bir tür cihazda çalıştırmanız gerekecek. Bir web uygulaması için Chrome tarayıcısı, emülatörler veya geliştirme makinenize bir kabloyla bağlı fiziksel bir cihaz. Play/Debug düğmesine tıkladığınızda, o anda hangi cihazı çalıştırmak istediğinizi seçebilirsiniz. Android Studio'nun önceki ekran görüntüsünde, mevcut cihazların bir listesini içeren bir açılır menü olduğuna dikkat edin. VS Code'da ise Play düğmesine bastığınızda hemen seçeneklerinizi içeren bir menü açılır. Her iki IDE'de de kontrol sizdedir.

**İpucu** Komut satırından "flutter devices" komutunu çalıştırarak şu

anda hangi cihazların mevcut olduğunu kontrol edebilirsiniz.

$ flutter devices

3 connected devices:

Vivo XL3 • 55S...KF • android-arm64 • Android

8.0.0 (API 26)

Android SDK • emul...4 • android-x86 • Android 9

(API 28) (emulator)

İPhone X • E6...39A • ios • com.

apple...OS-12-1 (simulator)

Yukarıdaki örnek çıktı bize üç cihazımız olduğunu söylüyor. Birinci ve ikinci cihaz Android, üçüncü cihaz ise iOS çalıştırıyor. İlk cihaz bağlı bir fiziksel cihazdır. İkinci ve üçüncü ise emülatördür.

Bu komutun, aralarından seçim yapabileceğiniz tüm olası emülatörleri size söyleyen "flutter emulators" komutundan farklı olduğunu unutmayın.

Flutter devices komutu, uygulamanızı çalıştırmak için şu anda hangi

cihazların mevcut olduğunu söyler.

### **Web uygulaması olarak çalıştırma**

Flutter, bir web uygulaması olarak çalıştırdığınızda tarayıcınızı bir cihaz olarak kabul eder. Bu nedenle, bir web uygulaması olarak çalıştırmak için gereken tek şey Google Chrome web tarayıcısını bir cihaz olarak etkinleştirmektir. Bu tek seferlik komut ile etkinleştirebilirsiniz:

$ flutter config --enable-web

"enable-web" değerinin "true" olarak ayarlanması.

O andan itibaren, uygulamanızı çalıştırabileceğiniz cihazların bir listesini aldığınızda, "Chrome" bunlardan biri olarak görünecektir. Uygulamanızı Chrome'da çalıştırmayı seçmeniz yeterlidir; IDE web uygulamanızı Chrome'a yükleyecektir.

### **Bağlı bir cihaz üzerinde çalıştırma**

Uygulamanızı fiziksel bir cihaz üzerinde çalıştırmanız gereken zamanlar olabilir. Örneğin, Bluetooth ile bağlı fiziksel bir yazıcıya etiket yazdırmayı içeren bir proje geliştiriyordum. Emülatörler, Bluetooth üzerinden

Bluetooth. Yazdırma işlemini test etmek için Bluetooth yazıcımla eşleştirilmiş gerçek bir fiziksel cihaza ihtiyacım .

Fiziksel bir cihazı geliştirme makinenize bağlamak için çoğu Android cihaz için bir USB kablosu ve çoğu iPhone için bir Lightning kablosu kullanacaksınız.

**İpuçları** #1 Bir Android cihazı bağlarken, başlangıçta onu şarj etmeye veya fotoğraf aktarmaya çalıştığınızı düşünecektir. Hata ayıklamaya çalıştığınızı bildirmek için cihazdaki Geliştirici Seçenekleri ekranını açın ve "USB hata ayıklamayı etkinleştir "i seçin.

#2 Birçok bağlantı sorunu kalitesiz bir USB kablosundan kaynaklanabilir. Mantıksız bir şekilde, tüm USB kabloları eşit yaratılmamıştır. Ayarları değiştirdikten sonra hala bağlanamıyorsanız daha kaliteli bir kabloya geçin.

### **Sıcak doldurma**

Uygulama emülatörünüzde/tarayıcınızda/fiziksel cihazınızda/her neyse çalıştıktan sonra, kaynak kodunda değişiklikler yapmak ve yeniden çalıştırmak isteyeceksiniz. İşte gerçekten harika olan şey: kaynak koddaki bir değişikliği her kaydettiğinizde, yeniden derlenir ve yeni sürüm anında yüklenir. Uygulamanız kaldığınız yerden devam eder - aynı noktada, aynı durumda ve aynı verilerle Biz buna "sıcak yeniden yükleme" diyoruz ve geliştirme döngüsünü gülünç derecede hızlı ve sürtünmesiz hale getiriyor.

### **Hata Ayıklama**

Her iki IDE de temel olarak tüm IDE'lerde alışık hata ayıklama araçlarına sahiptir. Projenizi çalıştırmaya başladığınızda, hata ayıklama araçları görünecektir.

Android Studio'da hata ayıklama penceresi genellikle IDE'nin alt kısmında

açılır. Şekil 2-9'a benzeyen küçük bir araç çubuğu vardır.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

***Şekil 2-9.*** *Android Studio'da hata ayıklama araç çubuğu*

Seçenekler soldan sağa doğru "üzerine bas", "içine bas", "içine basmaya zorla" ve "dışına bas" şeklindedir.

VS Code'da araç çubuğu kaynak kodunuzun üzerinde yüzer şekilde görünür (Şekil 2-10).

yazı tipi, metin, grafik, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

***Şekil 2-10.*** *VS Code'da hata ayıklama araç çubuğu*

Seçenekleri "oynat/duraklat, "üzerine gel, "içine gir", "dışına çık", "sıcak yeniden yükle, "yeniden başlat" ve hata ayıklamayı durdur" şeklindedir.

**Not** Flutter, hata ayıklama sırasında bir cihazda gerçekten çalıştığında olduğundan daha seçicidir. Bu iyi bir şeydir çünkü hata ayıklama sırasında muhtemelen düzeltmeniz gereken ancak ölümcül olması gerekmeyen bazı hataları belirgin hale getirir. Yayın sürümünde, aynı hataları yutar ve (umarım) kullanıcılarımızın uygulamamızı çalıştırmaya devam etmesine izin verir.

Bu hataların bir ailesi "çalışma zamanı "dır. Hata ayıklayıcı size gibi bir hata verdiğinde bunlardan biriyle karşı karşıya olduğunuzu anlarsınız:

========tarafından yakalanan istisna ========

Bir hareket işlenirken aşağıdaki iddia atıldı:

setState() geri arama argümanı bir Future döndürdü.

\_FooState#236 üzerindeki setState() yöntemi bir closure veya Future döndüren bir yöntemle çağrıldı. Belki de "async" olarak işaretlenmiştir.

vs. vs. vs.

Sizin çıkarımınız şu: Bunlardan birini gördüğünüzde sorunu çözün. Yapılması gereken doğru şey budur. Ancak dağıtımı yaptıktan sonra aynı sorunu kafanız karışmasın.

# **Sonuç**

Bakın, bunun özümsenmesi gereken çok fazla şey olduğunu biliyorum. Çapraz platform geliştirmenin doğası, araçları zorlaştırıyor. Ancak en kötüsü geride kaldı. Flutter SDK'sını ve bir IDE'yi (VS Code/Android Studio/IntelliJ IDEA) kurduktan sonra ihtiyacınız olan tek şey bu. DevTools ve bir ya da iki emülatör gerçekten yardımcı olabilir. Geriye kalan tek şey pratik yapmak için biraz tekrar yapmak. Harika olacaksın!

Artık Flutter araç zincirini gördüğümüze göre, widget oluşturmaya başlayalım!