IN DIESEM KAPITEL

- Lernen Sie die Geschichte der App-Entwicklung kennen
- Bekommen Sie einen Überblick über alternative Technologien zu Flutter
- Lesen Sie ein paar objektive und ein paar subjektive Gründe, warum und wann Flutter die beste Wahl ist
- Erhalten Sie einen kurzen Einblick, wie Flutter funktioniert und was es mit Dart auf sich hat

Kapitel 1
Flutter und das große Feld der App-Entwicklung

Willkommen im allerersten Kapitel Ihrer Reise in die Flutter-Welt! Wir starten mit einem kleinen Warm-up, indem wir Sie zunächst auf einen kleinen Exkurs entführen und Ihnen ein bisschen Grundwissen zu Flutter, seinen Vorteilen und Grenzen an die Hand geben. Anhand von alternativen Formen der App-Entwicklung lässt sich das wunderbar darstellen. Am Ende dieses Kapitels wissen Sie auf jeden Fall, was uns an Flutter so begeistert und warum wir Sie möglicherweise auch bald im Kreise der Begeisterten begrüßen dürfen.

Flutter in a Nutshell

Flutter ist ein Cross-Platform-Framework von Google, das User-Interface-Elemente zur Entwicklung einer App zur Verfügung stellt. Seit 2021 ist es nicht einfach nur ein x-beliebiges, sondern *das* weltweit am meisten genutzte Cross-Platform-Framework auf dem Markt (`https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/`, Stand: 20.03.2023).

Flutter-Apps laufen sowohl auf Android- und iOS-Smartphones als auch im Web und auf Windows-, Linux- und macOS-Desktops. Programmiert wird mit der Sprache Dart. In diesem Buch wollen wir Ihnen vor allem die Entwicklung von mobilen Apps für Android und iOS mit Flutter vorstellen – das meiste Gelernte lässt sich aber auch auf die anderen Plattformen anwenden.

Ein Framework ist eine Ansammlung von fertigen Komponenten und Tools, die Entwickelnde benutzen und anpassen können, mit dem Ziel, die Entwicklung zu beschleunigen. Sie können sich damit auf wichtige Aufgaben konzentrieren, anstatt mühselig jedes Mal von Neuem die Basisfunktionalitäten und -komponenten zu programmieren.

Seit Flutter 2017 veröffentlicht wurde, wird es unter Entwickelnden lebhaft diskutiert. Es ist eine neue Herangehensweise an die App-Entwicklung. Von manchen wird es als kurzlebiger Trend eingestuft, andere denken, dass Flutter die anderen Programmiersprachen und Frameworks, die sich zur App-Entwicklung eignen, in Zukunft in den Schatten stellen wird.

Anfänge der App-Entwicklung

App-Entwicklung war nicht immer ein eigenes Feld in der Software-Entwicklung – und es war auch nicht immer so prominent gespalten in Android- und iOS-Entwicklung, wie es das heute ist.

1994 kommt das erste Smartphone auf den Markt, aber es soll noch ein paar Jahre dauern, bis das Gerät massentauglich wird. Apps werden noch aus dem Internet heruntergeladen und wie Webseiten programmiert. Es entsteht eine kleine Hobby-Programmierszene für App-Entwicklung.

2007 kommt das erste iPhone auf den Markt und ein Jahr später öffnet der Apple App Store mit 500 Apps im Angebot seine virtuellen Pforten. Diese Apps laufen nur auf Apple-Geräten und können nicht aus dem Internet heruntergeladen, sondern nur über den Apple Store bezogen werden. Das ist bis heute so. Diese iPhone-Apps werden nicht mehr mit Web-Programmiersprachen geschrieben, sondern mit Objective-C, das schließlich 2014 von Swift abgelöst wird.

2008, ein Jahr nach dem iPhone, kommt das erste Android-Smartphone auf den Markt und der Google Play Store liefert die passenden Apps dazu. Diese Apps können auf allen Geräten mit Android-Betriebssystem laufen. Das Konzept ist etwas freier als in der Apple-Welt: Apps müssen nicht über den Google Play Store installiert werden, sie können auch aus dem Internet heruntergeladen und installiert oder per E-Mail verschickt werden. Doch die Programmiersprache ist auch keine Web-Sprache – Android-Apps werden erst mit Java, heute meist mit Kotlin programmiert.

2010 bringt Microsoft das Windows Phone auf den Markt. Apps für das Windows Phone können in C# geschrieben werden. Das Windows Phone setzt sich aber nicht gegen das iPhone und die Android-Smartphones durch und verliert rasch seinen Marktanteil. 2017 wird die Produktion schließlich eingestellt.

Der Markt der App-Entwicklung wächst. 2022 werden rund 1,6 Millionen iOS-Apps im Apple Store angeboten und über 3,5 Millionen Android-Apps im Google Play Store (`https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores`, 20.03.2023). Pro Tag werden mittlerweile über 700 iOS-Apps veröffentlicht und fast 2000 Android-Apps.

Alternativen zur App-Entwicklung mit Flutter

Die Tatsache, dass Sie dieses Buch in den Händen halten, lässt bereits vermuten, dass Sie zumindest daran interessiert sind, in die Flutter-Welt einzutauchen. Trotzdem möchten wir Ihnen davor die Alternativen zur Flutter-Entwicklung aufzeigen, um zu erfahren wie (teilweise beschwerlich und zeitaufwendig) der Weg zur eigenen App bisher aussah oder teilweise immer noch aussieht.

Die App-Entwickelnden-Szene teilt sich heute in verschiedene Plattformen: Entweder Sie programmieren nativ für Apple mit Swift und Xcode als Entwicklungsumgebung oder für Android mit Java oder Kotlin und Android Studio als Entwicklungsumgebung. Alternativ dazu wählen Sie eine Cross-Platform-Technologie wie Flutter, mit der Sie mehrere Betriebssysteme bedienen können. Auch eine Progressive Web-App (PWA) könnte als Alternative zu einer nativen oder einer Cross-Platform-App infrage kommen. In den folgenden Abschnitten gehen wir kurz auf alle drei Möglichkeiten ein.

Native Entwicklung

Native Entwicklung mit Swift für Apple oder Kotlin und Java für Android ist immer noch ein wichtiges Feld. Speziell wenn eine App hauptsächlich auf Hardware-nahe Funktionen wie den Einsatz von Kamera, Bluetooth, verschiedenen Sensoren oder Audio angewiesen ist, sollten Sie über eine native Entwicklung nachdenken. Diese Implementierungen sind meist mit viel nativem plattformspezifischen Code verbunden und können daher nicht für mehrere Plattformen verwendet werden. Sie können diese Funktionalitäten in Flutter zwar mit Plug-ins meistens abdecken, bei der Performance müssen Sie aber teilweise mit Einbußen rechnen.

Als mit Flutter entwickelnde Person werden Sie auch um kleine Einblicke in die Android- und iOS-Welt manchmal nicht herumkommen. Wenn Sie aus der nativen Entwicklung kommen, ist dies auf jeden Fall ein Vorteil. Ihr Wissen wird Ihnen in der Flutter-Entwicklung sicher zugutekommen.

Mit Flutter ist es mittlerweile auch möglich, plattformspezifisch jeweils andere Designelemente anzuzeigen. Die Libraries für Android- und iOS-spezifisches Design sind schon automatisch integriert.

Cross-Platform-Entwicklung

In den letzten Jahren kristallisiert sich immer mehr heraus, dass der App-Markt zwischen Android und iOS gespalten bleiben wird. Gleichzeitig gibt es kaum noch Firmen, die es sich leisten wollen, eine der Plattformen zu ignorieren. Firmen, die eine App auf den Markt bringen, wollen in der Regel beide Smartphone-Betriebssysteme bedienen – und oft auch eine Webvariante ihres Produktes.

Um nicht für ein Produkt zwei oder drei verschiedene Apps parallel entwickeln und pflegen zu müssen, entscheiden sich heute viele Firmen für eine Cross-Platform-Lösung. Eine Cross-Platform-App ist eine Software, die nicht nur auf einem, sondern auf mehreren Betriebssystemen laufen kann.

Diese Cross-Platform-Apps haben häufig Nachteile gegenüber den nativ programmierten Apps. Sie sind oft weniger performant und anfangs konnten die Cross-Platform-Frameworks nicht auf alle Hardware-Funktionen zugreifen, wie zum Beispiel Bluetooth oder GPS.

Cordova und Ionic

Cross-Platform-Entwicklung begann ursprünglich mit hybriden Apps. Ein Beispiel dafür ist das Apache-Cordova-Framework – auch bekannt unter dem Namen »PhoneGap«. Entwickelnde programmieren eine Webseite und das Framework packt diese Webseite in einen nativen Android- oder iOS-Container. Diese Apps haben leider oft eine mäßige Performance, lange Ladezeiten, vorgegebene und nur bedingt anpassbare Designelemente und gelten darum als »billige« Variante der nativen Entwicklung für iOS und Android. Das Ionic-Framework funktioniert nach demselben Prinzip und wird heute noch genutzt – ist aber nicht weit verbreitet.

Xamarin

Xamarin ist ein kostenpflichtiges Framework für .NET-Entwickelnde. Mit diesem Framework können in C# Apps für Android, iOS und Windows geschrieben werden. Es wird in Deutschland noch teilweise in mittelständischen Firmen eingesetzt, hat sich aber nie richtig durchsetzen können.

Unity und die Unreal Engine

Cross-Platform-Spiele können mit Unity in C# oder der Unreal Engine in C++ programmiert werden. Diese Cross-Platform-Spiele-Apps sind übrigens für den Großteil des Umsatzes des Apple App und Google Play Stores verantwortlich (`https://www.statista.com/statistics/266489/earnings-forecast-for-mobile-apps-providers/`, 20.03.2023). Für Flutter gibt es mittlerweile die Flame-Engine; mit der sich 2D-Spiele gut umsetzen lassen. Für komplexe Spieleentwicklung und vor allem 3D-Spiele sollten Sie allerdings weiterhin Unity oder die Unreal Engine in Betracht ziehen.

React Native

Mit React Native hat Meta vor ein paar Jahren eine ziemlich gute Cross-Platform-Lösung entwickelt. Der programmierte JavaScript-Code wird direkt in einen iOS- und Android-kompatiblen Code übersetzt, was der Performance zugutekommt und im Vergleich zum hybriden Ansatz deutlich näher am Look-and-feel von nativen Apps liegt. React Native ist weit verbreitet und war vor Flutter die beliebteste Cross-Platform-Lösung unter App Entwickelnden.

2021 hat sich Flutter offiziell gegenüber React Native als das meistgenutzte Cross-Platform-Framework weltweit durchgesetzt. Hauptgrund dafür ist, dass es gegenüber nativen Apps kaum Qualitätseinbußen gibt.

PWA-Entwicklung

Progressive Web-Apps sind keine mobilen Apps im engeren Sinne, sondern eigentlich Webseiten, die mithilfe einer extra Service-Worker-Datei zu einer offline-fähigen, herunterladbaren Web-App erweitert werden können. Sie können prinzipiell mit jedem Web-Framework geschrieben werden (auch mit Flutter übrigens). Sogar WordPress Webseiten können mithilfe eines Plugins zu einer PWA erweitert werden.

Die Eigenarten einer PWA

Die Apps können sowohl im Google Play Store, als auch auf normalen Webseiten im Internet gefunden und heruntergeladen werden. Eine Bereitstellung über den Apple App Store ist allerdings aktuell nicht möglich. Sie sehen aus wie native Apps und nutzen den Cache des Smartphone Browsers, um Daten zu speichern und offline zur Verfügung zu stellen.

Genau wie Flutter gibt diese Technologie das Versprechen mit nur einer Codebase drei große Plattformen zu bedienen. Während man bei Flutter nach einer Anpassung des Codes drei verschiedene Builds anfertigen und dann auf drei verschiedenen Plattformen veröffentlichen muss, wird eine PWA meist nur im Web veröffentlicht und ist dann sofort von allen Plattformen aus zugänglich. Auf den ersten Blick ist eine PWA also noch schneller zu updaten als in Flutter programmierte Apps.

Die Probleme einer PWA

Das Problem mit PWAs ist leider, dass es viele verschiedene Browser gibt und die Firmen dahinter Probleme haben, sich auf gemeinsame Standards zu einigen. Es geht nur langsam in die richtige Richtung. Safari zum Beispiel stellt sich gern quer, wenn es um die Frage geht, auf welche Smartphone-Funktionen der Browser und damit die PWA zugreifen darf. Das ständige Anpassen der App nach Browser Updates und das kompatibel Halten mit veralteten Browser Versionen ist umständlich und zeitaufwendig.

Ein weiterer wichtiger Punkt gegen eine Entscheidung für eine PWA-Entwicklung ist, dass diese Apps für viele Smartphone-Nutzende noch unbekannt sind. Sie suchen eher im App Store nach Apps als im Internet, und wenn sie eine PWA im Internet finden, wissen sie meist nicht um die Möglichkeit, diese herunterzuladen und wie eine native App zu benutzen.

Vorteile von Flutter

Sie sehen, es gibt also einige Möglichkeiten, eine App zu entwickeln. Wir halten Flutter jedoch in den meisten Fällen für die sinnvollste. Hier noch einmal die wichtigsten Gründe, sich für Flutter zu entscheiden, zusammengefasst und ein paar weiterführende im Überblick:

- ✔ **So gut wie nativ:** Mit Flutter können Apps entwickelt werden, die im Google Play Store und im Apple Store genau wie native Apps gefunden werden können und ein ebenbürtiges »Look-and-feel« wie native Apps bieten.

- ✔ **Viele Plattformen mit einer Codebase:** Flutter Skills haben eine große Hebelwirkung – mit einem Framework können Sie sehr viele verschiedene Plattformen bedienen. Da kommt sonst kaum ein Cross-Platform-Framework ran.
- ✔ **Das impliziert auch Plattform-Skalierbarkeit:** Sie können mit einer beliebigen Plattform starten, schnell ein MVP an Benutzende ausrollen und jederzeit neue Plattformen nachziehen, wenn der Markt reif dafür ist – ohne großen Mehraufwand.
- ✔ **Flutter-Entwicklung ist schnell:** Mit Flutter können Sie ganz fix ein professionell aussehendes UI schaffen. Es eignet sich hervorragend, um einen App-Prototypen oder MVP (ein Minimum Viable Product ist die erste funktionsfähige Iteration eines Produktes) zu erstellen, weil der Weg von der Idee zum ersten MVP-Release so schnell gehen kann.
- ✔ **Dart macht Spaß:** Dart, die Sprache, mit der in Flutter entwickelt wird, ist einfach zu lernen, vor allem wenn man schon eine Programmiersprache beherrscht. Oft wird es als eine Mischung zwischen Kotlin, Swift und TypeScript beschrieben.
- ✔ **Flutter ist Open Source:** Ein Team von Google arbeitet ununterbrochen an der Weiterentwicklung von Flutter, aber Sie können auch selbst aktiv werden und an der Weiterentwicklung mitwirken. Außerdem haben Sie Zugriff auf den gesamten Source-Code und können die Funktionsweise von Flutter jederzeit selbst nachschlagen.
- ✔ **Spaß beim Entwickeln:** Flutter soll nicht nur für die App-Nutzenden eine Bereicherung sein, sondern auch für die Entwickelnden. Von Anfang an ist an das Entwicklungserlebnis für Entwickelnde gedacht worden und das sorgt dafür, dass Flutter einfach Spaß macht. Mit Hot-Reload, einer Funktion, die Sie in diesem Teil noch praktisch kennenlernen werden, kann die App zum Beispiel in Sekunden mit neuen Änderungen geupdatet werden, ohne einen langen Build-Prozess und einhergehende Wartezeit auszulösen.
- ✔ **Eine hilfsbereite Community:** Es hat sich mittlerweile auch eine große, aktive und sehr hilfsbereite Flutter Community entwickelt. Es gibt unzählige Tutorials, Hilfeforen und Blogs, die bei Problemen Hilfestellung leisten können. Auf Meet-ups und Konferenzen können Sie viele leidenschaftliche Flutter-Entwickelnde kennenlernen und sich austauschen.
- ✔ **Premium-Dokumentation:** Flutter ist sehr gut dokumentiert. Wenn Sie bereits mit anderen Frameworks, Programmiersprachen und Software-Entwicklungstools gearbeitet haben, wissen Sie wahrscheinlich, dass das nicht selbstverständlich ist.
- ✔ **Package- und Plug-in-Vielfalt:** Für nahezu jeden Use-Case gibt es mittlerweile ein passendes Package oder Plug-in, mit dem Sie Ihre Flutter-App um Funktionalität und Designelemente erweitern können.
- ✔ **Zukunftsfähige Technologie:** Flutter ist »in« und das Interesse steigt, nicht nur bei Entwickelnden, auch bei potenzieller Kundschaft. Ein perfekter Zeitpunkt einzusteigen!

Wie funktionieren Flutter und Dart?

Flutter ist ein Framework (von Google auch als UI-Toolkit bezeichnet) und Dart die Programmiersprache, mit der innerhalb dieses Frameworks programmiert wird – so viel wissen Sie bereits. Im Folgenden möchten wir aber noch einen Schritt weiter gehen und Ihnen einen kurzen Einblick geben, wie Flutter und Dart unter der Haube funktionieren und warum gerade Dart als Programmiersprache ausgewählt wurde.

Wenn Sie gleich Ihre erste Flutter-App entwickeln, werden Sie sich zwei APIs (Schnittstellen) bedienen, der Flutter-API und der Dart-API.

Flutter-API

Die Flutter-API gibt Ihnen eine Kollektion von UI-Elementen an die Hand, aus der Sie sich bedienen können. Bei Flutter heißen diese Elemente »Widgets«. Es gibt zum Beispiel Container, Buttons, Textfelder, Menüs, Bilder, Icons, Pop-ups und viele mehr.

Sie können aus der Material Library Elemente im Android-Stil auswählen oder aus der Cupertino Library Elemente im iOS-Stil. Die meisten Widgets haben anpassbare Parameter, wie Farben, Größen, Abstände und Inhalte.

Dart-API

Die Dart-API gibt Ihnen Zugriff auf die üblichen Funktionen, die in der Regel jede Programmiersprache anbietet: Sie können zum Beispiel eine Funktion aufrufen, die eine Kommazahl rundet, eine Collection sortiert oder Ihnen das derzeitige Datum zurückgibt.

Mit Dart-Code können Sie die Logik der App schreiben und bestimmen, wann welche Daten geladen und wann welche UI-Elemente angezeigt werden sollen.

Warum Dart?

Aber warum muss es eigentlich die unbekannte Sprache Dart sein? Hätten JavaScript, Java, Kotlin, Python oder irgendeine andere Sprache, die auch in anderen Bereichen der Software-Entwicklung eingesetzt wird, es nicht auch getan? Viele Entwickelnde, die wir treffen, haben ähnliche Bedenken.

Was ist so besonders an Dart?

Dart ist eine leicht zu erlernende, moderne und elegante, für die UI-Entwicklung optimierte Programmiersprache. Sie ist objektorientiert und die Syntax ähnelt der von Kotlin, Swift und TypeScript.

Seit Version 2.12 ist Dart Null-Safe und sorgt dafür, dass Flutter-Entwickelnde immer im Blick haben müssen, wann eine Variable null werden kann.

Das Flutter-Team hat sich vor allem für Dart entschieden, weil Dart schnell ist und gleichzeitig eine gute User Experience für App-Nutzende sowie App-Entwickelnde bietet. Aber warum ist das so?

JIT und AOT – bitte, was?

Das Geheimnis dahinter? Dart unterstützt sowohl JIT- als auch AOT-Kompilierung. Diese Kombination lässt sich in anderen Sprachen nur selten finden, denn meist wird nur eine Form der Kompilierung unterstützt.

JIT steht für Just-in-Time-Kompilierung. Wenn eine Sprache JIT kompiliert, heißt das, dass sie kompiliert, während die App läuft. Das führt für die App-Nutzenden zwar zu einer geringeren Performance, bietet aber vor allem Geschwindigkeitsvorteile beim Entwickeln – Stichwort Hot-Reload.

Dart benutzt den JIT-Compiler, wenn Sie Ihre App während des Entwickelns auf einem Emulator oder auf einem angeschlossenen Smartphone debuggen. Führen Sie eine Änderung in Ihrem Code durch und speichern Sie diese, sehen Sie sie dank Hot-Reload innerhalb von Millisekunden auf Ihrem Gerät.

Erst wenn Sie Ihre App veröffentlichen wollen und dafür einen sogenannten »Release Build« generieren, benutzt Dart den AOT-Compiler (Ahead-of-Time-Compiler). Dart kompiliert also, bevor die App von der benutzenden Person aus dem Store heruntergeladen oder im Web aufgerufen wird, zu nativem ARM- oder Intel-x64-Maschinencode oder im Web zu JavaScript-Bytecode und sorgt so dafür, dass App-Nutzende eine qualitativ ebenbürtige User Experience bekommen – analog einer nativ programmierten App. Es gibt nahezu keine Einbußen an Auflösung oder Geschwindigkeit.

Es gibt also ein paar gute Gründe, warum sich das Flutter-Team für Dart entschieden hat. Auch wir sind uns sicher: Sie werden die Sprache bestimmt genauso schnell zu schätzen lernen wie wir.

Sie sind nun bestens mit Hintergrundwissen ausgestattet – wagen wir uns an den praktischen Teil.