Mobile App Development Frameworks

Wenn sie beginnen, Ihre Lösung für die Erstellung einer neuen mobilen Anwendung zu wählen, stehen Sie oft vor einer Vielzahl von Optionen, sind sich aber nicht sicher, wo Sie anfangen sollen. Ist die Erstellung Ihrer Anwendung ausschließlich mit einer nativen Lösungen die richtige Wahl? Sollte Ihr Team nur die Entwicklung mit Web-Technologien in Betracht ziehen? Was ist mit einigen der Hybridlösungen da draußen? Dies sind nur einige der vielen Fragen, die sich stellen, wenn Sie darüber nachdenken, welcher Weg der Beste für Ihr Vorhaben ist.

Bevor sie mit der Entscheidung beginnen, ist es wichtig, sich daran zu erinnern, dass die Auswahl der richtigen Lösung für Ihre nächste Anwendung von mehreren Faktoren abhängt: Ihr Budget, der Zeitplan, die Erfahrung von Ihnen bzw. von ihrem Team, und schließlich auch von ihrer Zielgruppe.

Deswegen möchte ich zu aller Anfang die drei primären Genres der Entwicklung mobiler Anwendung erklären und einige der Vor- und Nachteile jeder dieser Lösungen erläutern. Am Ender dieser Präsentation sollten Sie ein besseres Verständnis für die Auswahl haben und die Richtige Lösung für Ihre mobile Anwendung zu finden.

Web App

Reine Webanwendungen zu entwickeln, gibt die Freiheit, außerhalb der App-Stores zu existieren und ihre Anwendung anderen mobilen und Desktop-Benutzern anzubieten. Dies sind nur traditionelle Webanwendungen, die in HTML, CSS und JavaScript geschrieben sind, für die Sie eine Vielzahl von Frameworks und Bibliotheken nutzen können, wie Angular, React, Vue oder sogar Plain-Vanilla JavaScript.

Bei diesem Entwicklungsansatz beschränkt sich Ihre mobile Anwendung jedoch auf die Funktionen des mobilen Browsers des Benutzers. Dies bedeutet, dass sie keinen vollständigen Zugriff auf das Benutzergerät für Dinge wie z.B. sein Adressbuch haben. Während sich dies im Laufe der Jahre mit dem Zugriff auf Funktionen wie GPS und Kamera verbessert hat, könnte dies je nach den für Ihre Anwendung benötigten Funktionen auch zu einem Problem werden.

Eine der größeren Herausforderungen bei der Entwicklung von Anwendungen mit Web-Technologien besteht darin, dass viele der gängigen Anwendungs-UI-Controls, z. B. der Tab-Navigator, nicht nativ existieren und neu erstellt werden müssen, was dazu führen kann, dass Ihre Anwendung nicht richtig funktioniert. Des Weiteren ist die Gestensteuerung schwieriger zu implementieren. Allerdings haben viele der UI-Bibliotheken große Sorgfalt bei der Replikation der meisten gängigen UI-Komponenten aufgewendet, die für Ihre App notwendig sind, so dass dieses Risiko reduziert wird.

Im Jahr 2017 führte Google das Konzept der Progressiven Webanwendungen (PWAs) ein, die es diesen Anwendungen ermöglichen, mehr applikationsähnliche Funktionen wie Push-Benachrichtigungen, Offline-Funktionen und mehr zu übernehmen.

Für einige Entwicklungsteams ist diese Lösung sehr attraktiv, da sie einfach eine Anwendung aus einer einzigen Codebasis erstellen können, die dann auf einer Vielzahl von Plattformen verwendet läuft und schnell mit einer neuen Funktion oder Fehlerbehebung aktualisiert werden kann.

Was ist eine Native Mobile App

Wenn Menschen von einer nativen mobilen Anwendung sprechen, beziehen sie sich in der Regel auf eine Anwendung, die mit der nativen Entwicklungssprache und plattformspezifischen Tools

geschrieben wurde. Zum Beispiel: Eine native iOS-Anwendung wird entweder in Swift oder Objective-C geschrieben und mit Xcode kompiliert, während eine native Android-Anwendung mit Kotlin oder Java entwickelt und mit Android Studio kompiliert wurde.

Da diese Anwendungen mit den Standardlösungen der Plattform entwickelt werden, haben Entwickler vollen und einfacheren Zugriff auf die Funktionen des Geräts, wie z.B. alle Sensoren des Geräts, das Adressbuch des Benutzers, etc. Native Anwendungen sind in der Regel auch performanter. Sie sind nicht nur schneller, sondern haben auch Zugriff auf alle nativen Bedienelemente und Layouts der Benutzeroberfläche.

Allerdings kann jede Anwendung, die für iOS mit Swift geschrieben wurde, nicht auf Android laufen und umgekehrt. Das bedeutet, dass Sie speziell für jede Plattform entwickeln müssen, was zu einem größeren Budget und einer größeren Teamgröße führen kann. Vorausgesetzt, dass Sie Ihre Anwendung sowohl für iOS als auch für Android freigeben möchten. Darüber hinaus ist Ihre Anwendung nur über den App-Store der jeweiligen Plattform verfügbar und unterliegt deren jeweiligen Regeln und Einschränkungen. Das bedeutet, dass für jedes Release, egal ob es sich um ein neues Feature oder eine Fehlerbehebung handelt, der gleiche Genehmigungsprozess stattfinden muss. Dies kann von einem Tag bis zu zwei Wochen für den Apple App Store dauern.

Hybrid Apps

Diese Lösung ist eine Mischung aus einem nativen Ansatz und einer Web-Lösung, daher der Name Hybrid. Dabei wird der Kern der Anwendung mit Web-Technologien (HTML, CSS und JavaScript) geschrieben, die dann in einer nativen Anwendung gekapselt werden. Durch die Verwendung von Plugins können diese Anwendungen vollen Zugriff auf die Funktionen des mobilen Geräts haben.

Das Herzstück einer hybrid-mobilen Anwendung ist nach wie vor nur eine Anwendung, die mit HTML, CSS und JavaScript geschrieben wird. Statt dass die App jedoch im Browser des Benutzers angezeigt wird, wird sie von einer nativen Anwendung und einem eigenen eingebetteten Browser aus ausgeführt, der für den Benutzer im Wesentlichen unsichtbar ist. Eine iOS-Anwendung würde beispielsweise den WKWebView verwenden, um unsere Anwendung anzuzeigen, während sie auf Android das WebView-Element verwenden würde, um die gleiche Funktion auszuführen.

Dieser Code wird dann mit einer Lösung wie Apache Cordova oder Ionic's Capacitor in einen nativen Application Wrapper eingebettet. Diese Lösungen erstellen eine native Shell-Anwendung, die nur die Webview-Komponente der Plattform ist, in die sie Ihre Webanwendung lädt. Dies gibt Ihnen die Möglichkeit, echte native Anwendungen zu erstellen und zu veröffentlichen, die in jedem der App-Stores der Plattformen zum Download angeboten werden können.

Darüber hinaus verfügen sowohl Cordova als auch Capacitor über ein Plugin-System, mit dem Sie über die Grenzen des „Browsers“; hinausgehen und auf die gesamte Palette der Funktionen des mobilen Endgeräts eines Benutzers zugreifen können. Wenn Sie also TouchID auf einem iOS-Gerät als Anmeldeoption verwenden oder sich mit einem Bluetooth-Gerät verbinden möchten, kann dies ganz einfach durch die Installation eines Plugins geschehen. Diese Plugins werden von einer Vielzahl von Entwicklern erstellt und viele werden aktiv unterstützt. Ionic bietet sogar ein komplettes Ökosystem von unterstützten Plugins als Teil seiner Enterprise Engine Lösung. So lassen sich die Einschränkungen einer reinen Webanwendung leicht überwinden, so dass Ihre Anwendung in ihren Funktionen die gleiche Qualität wie native Anwendungen aufweist.

Es gibt jedoch einige Nachteile bei dieser Option. Ähnlich wie bei der reinen Web-Anwendung muss die UI-Komponenten neu erstellt werden. Hier kommen Lösungen wie Ionic, Flutter, Xamarin, React Native und andere zum Einsatz. Diese Optionen bieten alle robuste

Benutzeroberflächenkomponenten, die wie ihre ursprünglichen Gegenstücke aussehen und sich auch so anfühlen und somit Ihnen eine vollständige Palette von Bausteinen für Ihre Anwendung bieten.

Die einzige andere zu berücksichtigende Überlegung ist, ob Ihre Anwendung noch im nativen Browser des Geräts ausgeführt wird. Wenn ja, können Performance-Probleme oder andere Besonderheiten auftreten, die für jede Plattform oder Betriebsversion spezifisch sind.

Zusammenfassung Hybrid – Native - Web

Welche ist also die richtige Wahl für Ihre Anwendung? Wenn Sie kein hoch performantes Spiel oder eine ähnliche Anwendung erstellen, könnte die hybride Anwendungsentwicklung die richtige Wahl für Sie sein. Da sie einen einfacheren Entwicklungsansatz, Kosteneinsparungen und die Kompatibilität auf einer Vielzahl von Plattformen bietet. Mit bekannten Frameworks wie z. B. Ionic, Flutter etc. ist solch eine Anwendung schnell gebaut. Und zu solchen Frameworks werde ich jetzt auch noch einige Worte verlieren.

Vergleich

Einer der bekanntesten Frameworks dafür ist definitiv Ionic. Doch er konkurriert zur Zeit sehr stark mit dem aufgehenden Stern von Google namens Flutter. Diese beiden Frameworks werde ich jetzt in einigen Gesichtspunkten vergleichen und nebenher sowohl Vorteile als auch Nachteile erzählen.

Ionic vs Flutter

Ionic und Flutter teilen die gemeinsame Vision, schöne, leistungsstarke Anwendungen zu entwickeln, die überall funktionieren. Die Kernphilosophien könnten jedoch nicht unterschiedlicher sein.

Philosophie

Bei Ionic geht es darum die vorhandenen Dinge bestmöglich zu nutzen, indem sie wo immer es möglich ist offene Webstandards bzw. Fähigkeiten nutzen. Tatsächlich, wenn man sich für Ionic entscheidet, setzt man nicht wirklich auf Ionic. Sie setzen auf das Internet. Das liegt daran, dass das Ionic Framework und die Tools alle auf offenen Webtechnologien basieren, von den Websprachen, mit denen Sie ionische Apps (HTML, CSS, JavaScript) erstellen, bis hin zu den standardbasierten UI-Komponenten, die in Ihrer App ausgeführt werden.

Flutter hat sich entschieden, seine eigene Vision zu verwirklichen, indem es ein völlig neues, in sich geschlossenes Ökosystem von Grund auf neu geschaffen haben. Von Dart, der Nicht-Standard-Sprache, die Sie verwenden, um Flutter-Anwendungen zu erstellen, bis hin zu seiner benutzerdefinierten Rendering-Engine, basiert fast alles an Flutter auf seinen eigenen Standards.

Wie funktioniert Flutter?

Die Kernsprache von Flutter ist Dart, eine wenig bekannte Sprache aus dem Jahr 2011. Obwohl es schon eine Weile da ist, sind nur wenige Entwickler heute damit vertraut (weniger als 2%, laut StackOverflow Survey 2019), und es wird selten außerhalb der Flutter-Community verwendet.

Beim Erstellen für Mobilgeräte verwendet Flutter den Dart Compiler, um Ihren Dart Code in nativen Maschinencode zu konvertieren, der auf der Geräteplattform ausgeführt wird. Innerhalb einer benutzerdefinierten Rendering-Engine wird Ihre mobile Anwendung schlussendlich angezeigt. Flutter bietet eine eigene Bibliothek von UI Widgets und nutzt somit nicht Web Components wie Ionic.

Flutter Apps greifen auf native Gerätefunktionen über eine Plugin-Bibliothek zu, die ähnlich wie Ionic und React Native ist. Außerdem ist es möglich Ihren eigenen benutzerdefinierten plattformspezifischen Code schreiben, wenn das Package oder Plugin, nach dem Sie suchen, nicht verfügbar ist.

Überblick über Ionic

Ionic Apps werden mit den Sprachen der klassischen Webentwicklung erstellt: HTML, CSS und JavaScript. Meist mit einem Framework wie Angular, React oder Vue.

Wenn Ionic auf dem Handy ausgeführt wird, läuft es innerhalb eines nativen Containers unter Verwendung von Cordova oder, in jüngster Zeit, von Capacitor, der den vollen Zugriff auf alle nativen Gerätefunktionen oder APIs ermöglicht. Die Benutzeroberfläche Ihrer Ionic App läuft in einer WebView, das praktisch ein kopfloser Browser ist, der für den Benutzer unsichtbar ist. In einer Desktop-Implementierung läuft Ionic in einem nativen Desktop-Container wie Electron.

Die Oberflächenkomponenten, die Sie in einer Ionic Anwendung verwenden können, verwenden alle den Standard Web Components, so dass sie in jedem Webbrowser ausgeführt werden können und mit jedem JS-Framework kompatibel sind. Ionic bietet eine Bibliothek mit über 100 UI-Komponenten, die Sie mit CSS anpassen können. Sie können auch Stencil, einen Open-Source-Compiler für Webkomponenten, verwenden, um Ihre eigene Bibliothek mit benutzerdefinierten Webkomponenten aufzubauen.

Somit nehmen Sich die beiden Frameworks nicht viel in dieser Hinsicht. Doch in welchen Punkt macht es sich bezahlt das Flutter auf eine eigene Engine setzt?

Performance

Ganz klar. Die Performance.

Eines der Dinge, von denen Entwickler mit Flutter oft schwärmen, ist die Leistung. Dies scheint der Ort zu sein, an dem sich all das nicht-standardisierte Material, in das sie investiert haben, ausgezahlt hat.

Hier hat Flutter wirklich die Nase vorne. Jedoch sollte man bedenken, dass die Leistung fast immer davon abhängt, wie der Code geschrieben wird und nicht primär davon, welche Plattform oder welches Framework Sie wählen.

Code portability

Wenn es darum geht, Ihre App auf Mobilgeräten und auf Desktops bereitzustellen, erscheinen sowohl Ionic als auch Flutter gleichmäßig aufeinander abgestimmt. Die frühen Demos von Flutter zeigen, dass man einige gut aussehende iOS- und Android-Anwendungen aus einer einzigen Codebasis erstellen kann. Und während sich die Desktop-Unterstützung von Flutter noch in der Entwicklung befindet. Kann Ionic bereits eine Windows App erzeugen.

Die Frage ist, ob Sie Ihre App über das Web bereitstellen möchten. Die Einschränkungen der Web-Implementierung von Flutter werden wahrscheinlich nie für Anwendungen funktionieren, die schnelle Ladezeiten und schnelle Leistung erfordern. Da Ionic auf dem Web basiert und vollständig auf Webcomponents setzt, gewinnt Ionic den Punkt im Bereich Code Portability (Übertragbarkeit).

Knowledge and skillset

Ein weiterer Kritikpunkt einiger Gegner von Flutter ist die Programmiersprache Dart. Wenn Sie nicht einer der 2% der Entwickler sind, die Dart bereits kennen, müssen Sie darüber nachdenken, ob Sie eine neue Sprache erlernen wollen. Eine weitere zu berücksichtigende Sache ist, wie marktfähig Ihr Skillset sein wird, sobald Sie Dart gelernt haben. JavaScript ist natürlich eine ziemlich sichere Lösung für jeden Entwickler, ob Web oder nicht. Der Bedarf an Dart Entwicklern wird wahrscheinlich allein vom Erfolg oder Misserfolg von Flutter als praktikable Lösung auf lange Sicht abhängen.

Des weiteren erlernen Sie nur die Flutter Arbeitsweise, da es ja in einem benutzerdefinierten Ökosystem betrieben wird. Diese Arbeitsweise lässt sich nur schwierig auf Nicht-Flutter-Projekte

übertragen. Wenn Sie beispielsweise Probleme mit der Benutzeroberfläche beheben, lernen und beherrschen Sie die benutzerdefinierte Flutter-Rendering-Engine, und nicht generell Web-Apps. Und wenn Sie mit nativen Geräteplattformen interagieren, lernen Sie Flutters Interpretation von Android und iOS, nicht von Android oder iOS im native Sinn. Dies ist einer der größten Kompromisse, die zu berücksichtigen sind, wenn Sie mit Flutter entwickeln.

Im Vergleich dazu ist Ionic eine Spur besser denke ich. Da Ionic mit Angular, Vue oder React bennutzt werden kann, lernen Sie ein Framework, dass auch abseits von Ionic verwendet werden kann. Dazu können die meisten Entwickler bereits mit CSS, JavaScript und HTML umgehen und somit fällt der Einstieg wesentlich leichter.

Beispiele herzeigen

Damit Sie sich besser etwas darunter vorstellen können, werde ich jetzt live Beispiele herzeigen. Wie wird in Ionic bzw Flutter eine App erstellt. Welche Tools benötigen Sie für die Entwicklung? Und vieles mehr werde ich sofort beantworten.

Beginnen wir mit Ionic. Ich habe mich dazu entschieden die Ionic App mithilfe von dem Framework Angular zu erstellen, da es die meisten bereits kennen.

Bevor wir ein Projekt erstellen können benötigen wir folgende Dinge: Node.js mit npm, des Weitern noch einen Code Editor wie Visual Studio Code. Als letztes benötigen wir noch ionic und Cordova was mittels npm installiert wird.

Das heißt somit, wir sind bereit unsere erste Ionic App zu entwickeln. Mittels ionic start project-name tabs legen wir ein Ionic Angular Projekt mit dem Standard Template tabs an. Dann wechseln wir in das Verzeichnis und starten die App mittels ionic serve. Und voila unsere erste Ionic App läuft in einem Webbrowser. Welches natürlich die Compile Zeit etwas verlängert. Aber so schnell ist ein Projekt mit Ionic erstellt.

Nun möchte ich Ihnen ein Tool zeigen das nur Ionic hat. Und zwar bietet Ionic eine App an, über welche es möglich ist die zu entwickelnde App per Wireless zu übertragen und auszuprobieren. Wenn das Projekt nämlich mittels ionic serve --devapp gestartet wird, und Sie sich im selben Netzwerk befinden müsste das Projekt sogleich in der App erscheinen und per click können Sie den aktuellen Stand testen.

Das wars zu Ionic

Nun möchte ich noch kurz die Vorgehensweise von Flutter erklären. Die folgendermaßen funktioniert.

Bevor man beginnt muss man sich natürlich die Flutter SDK downloaden, welche auf der offiziellen Website ganz einfach zu finden ist. Diese wird an einem gewünschten Ort abgelegt und entpackt. Dies kann je nach Computer schonmal etwas länger dauern.

Parallel dazu kann man sich schon einen Editor wie Visual Studio Code oder Android Studio installieren. Ich habe mich dazu entschieden Flutter in Android Studio zu entwickeln, da ich hier schon den Emulator dabei habe und selbst Flutter empfiehlt diese Entwicklungsumgebung. Dafür benötigt wird ein Flutter Plugin, welches installiert werden muss (File > Settings > Plugin > Flutter). Anschließend wird die IDE neu gestartet. Nun sollte es möglich sein ein neues Flutter Projekt zu erstellen.

Dies funktioniert folgendermaßen:

1. File > New > Start new Flutter Project

2. Flutter Application
3. Project Name, Flutter SDK (wenn noch keine installiert wurde -> Install SDK und Path auswählen)
4. Projekt Pfad angeben und next
5. Domain noch angeben (at.htl)
6. Jetzt noch auf fertigstellen und somit ist unsere erste Flutter App erstellt.

Diese startet man ganz einfach oben rechts mit dem Emulator. Leider gibt es für Flutter nicht eine vergleichbare Lösung wie für Ionic die DevApp. Dennoch ist es natürlich möglich die App am Handy zu testen nur halt über ein USB Kabel.

Ein Feature welches Flutter wirklich extrem gut gemacht hat ist der sogenannte Hot Reload. Mit dem ist es möglich ist in Sekundenschnelle Änderungen zu Übernehmen.

https://github.com/lxieyang/ionic-audio-player

Resümee

Alles in einem kann man sagen, das jede dieser zwei Frameworks Vor und Nachteile hat. Die Dev App von Ionic ist unschlagbar cool finde ich. Aber der Hot reload von Flutter ist ebenfalls ziemlich beeindruckend. Wenn man Dart sich einmal angeeignet hat, ist Dart wahrscheinlich die bessere Option. Dennoch wird es für die meisten mehr Sinn machen Ionic zu nutzen, da bereits jeder von Ihnen Angular Kenntnisse besitzt und somit auch Ionic zu einem Großteil bereits programmieren kann.

Ich hoffe somit, Sie haben einen guten Überblick über die beiden Frameworks bekommen. Und bedanke mich somit bei Ihrer Aufmerksamkeit.

Quellen

https://ionicframework.com/resources/articles/ionic-vs-flutter-comparison-guide

https://flutter.dev/

https://medium.com/swlh/what-is-phonegap-why-you-should-use-it-for-app-development-5ac5e19eb19a

https://stackshare.io/stackups/ionic-vs-phonegap

https://ionicframework.com/resources/articles/ionic-vs-react-native-a-comparison-guide#where-we-are-the-same

https://ionicframework.com/resources/articles/what-is-hybrid-app-development