smf
YOUR IT PARTNER

Die 8 Erfolgsfaktoren
bei der App-Entwicklung

INHALT

INTRO

DIE 8 ERFOLGSFAKTOREN BEI DER APP-ENTWICKLUNG

Mobile Endgeräte sind aus unserem Alltag nicht mehr wegzudenken: Wir buchen Kino- und Konzerttickets über unsere Smartphones, bestellen bequem auf dem heimischen Sofa kulinarische Leckereien und müssen uns nicht mehr in überfüllte Warenhäuser begeben, um unsere Einkaufsliste abzuarbeiten.

Mitte 2024 standen sowohl im Apple Appstore als auch im Google Playstore mehr als 4,3 Mio. mobiler Anwendungen zum Download bereit. Die Einsatzmöglichkeiten bzw. Anwendungsszenarien sind vielfältig und die Bandbreite bewegt sich von komplexen Geschäftsanwendungen oder nützlichen Fitness-Trackern bis hin zu „Augmented Reality“-Computerspielen.

Für Unternehmen bedeutet dies, eigene mobile Strategien zu entwickeln, um die steigende Nachfrage nach mobilen Lösungen zu bedienen. Wo lassen sich Optimierungspotenziale in den eigenen Geschäftsprozessen erschließen? An welchen Stellen ist die Einführung spezieller, individueller Applikationen (Apps) sinnvoll? Und welche Voraussetzungen sind notwendig für das Entwickeln erfolgreicher Apps?

Unser Ziel ist es, Ihnen zu helfen, einen Mehrwert für Ihr Unternehmen zu generieren und die Potenziale der Datenrevolution voll auszuschöpfen.

GRENZENLOSE TECHNIK?

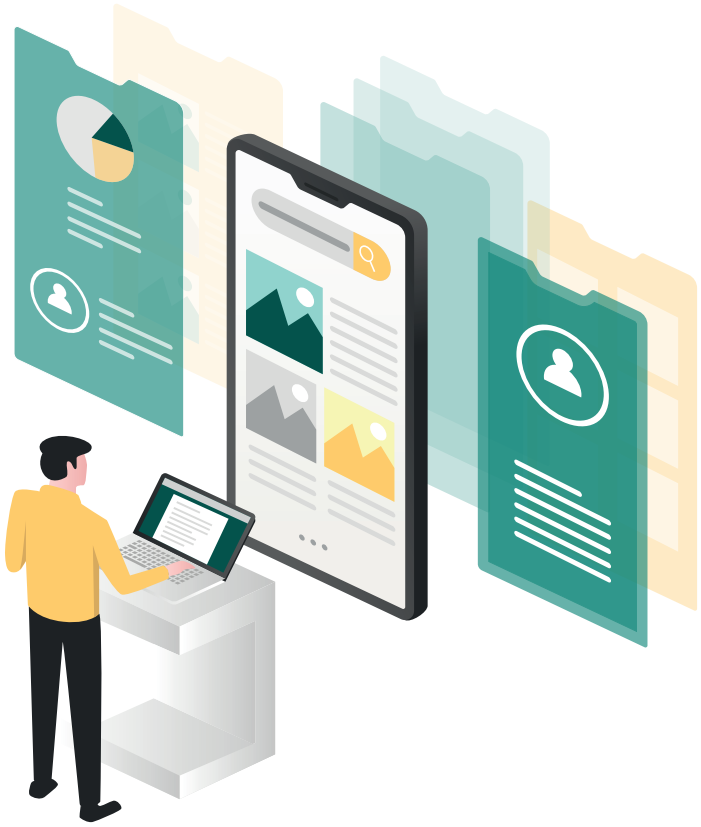
Technisch sind die Voraussetzungen so gut wie nie zuvor:

Zunehmende Rechenleistung
Technische Fortschritte in der Miniaturisierung
Schnelle Anbindung mobiler Endgeräte an das Internet
Aktuelle Mobilfunkstandards wie 5G

Existieren womöglich keine technischen Grenzen für eine Vielzahl von Anwendungen? Doch, sie existieren. Denn obwohl der Einstieg in die Programmierung mobiler Anwendungen aufgrund der softwaretechnischen Fortschritte der letzten Jahre sehr leicht ist, müssen diese technischen Möglichkeiten in einem reproduzierbaren, industriellen Softwareerstellungsprozess je nach Anwendungsfall sinnvoll eingesetzt werden.

Besondere Aufmerksamkeit gilt beispielsweise schon während der Definitionsphase der Usability (Gebrauchstauglichkeit) und der User Experience, um die Anwenderinnen und Anwender bei der Verarbeitung der angezeigten Informationen nicht zu überfordern.

Lassen Sie sich von der App-Idee über smarte Software-Lösungen bis hin zur AppFactory von uns unterstützen. Wir sorgen für das Zusammenspiel aller System-Schnittstellen oder kompletter Backend-Systeme unter iOS und Android, auf Smartphone, Tablet oder Watch. Gemeinsam machen wir Ihre Welt mit Apps mobil!



HERAUSFORDERUNGEN UND BESONDERHEITEN ODER: WAS NÜTZT DIE NEUESTE TECHNIK, WENN SIE SICH NICHT BEDIENEN LÄSST?

Da die Bildschirmgröße bei Smartphones geringer ist als bei Desktop-Anwendungen, kommen User Experience und Usability eine hohe Bedeutung zu. Komplexe Informationen können nicht so eingeblendet werden, wie am PC üblich. Doch auch durch die Art der Interaktion über die Gestensteuerung entsteht eine weitere Einschränkung: Es stehen weder eine Maus noch eine Tastatur zur Verfügung, so dass mobile Anwendungen nach Möglichkeit mit einem Finger bedient werden müssen.

User erwarten heutzutage eine einfach zu bedienende, mobile Anwendung, weil sie es eben vom Wettbewerb kennen und gewohnt sind. Mobile Unternehmensanwendungen sollen sich genauso einfach und schnell bedienen lassen wie Social-Media- oder Onlineshop-Apps, die man privat intuitiv einsetzt, ohne dass ein dickes Handbuch gewälzt, ein Tutorial angeschaut oder sogar eine Schulung gebucht werden muss.

WAS BRAUCHT EINE APP, UM ERFOLGREICH ZU SEIN?

Die Zauberworte heißen hier „Nutzerorientierte Gestaltung“. Um den Ansprüchen an eine moderne, intuitive App gerecht zu werden, bedienen wir uns bei SMF dieser Methodik.

Das Ziel ist, ein Softwaresystem entlang der Wünsche, Aufgaben und Eigenschaften der zukünftigen Benutzerinnen und Benutzer zu entwerfen.

Schon in der Definitionsphase stehen die künftigen Anwenderinnen und Anwender im Mittelpunkt, Softwareergonomie und Usability umgeben das Ganze.

Als Ergebnis entsteht ein interaktives System, welches leicht zu erlernen und intuitiv zu benutzen ist, eine geringe Fehlerrate bei der Eingabe aufweist und damit die Zufriedenheit der Anwendenden sicherstellt (nach DIN 9241).

MIT DIESEN 8 FAKTOREN WERDEN APPS ERFOLGREICH

Besondere Aufmerksamkeit gilt den Bedürfnissen der Anwenderinnen und Anwender. Sie müssen erkannt werden, um die Benutzungsoberfläche daraufhin optimal auszurichten. Den folgenden Ansprüchen sollte eine App daher entsprechen, um erfolgreich zu sein:

01

Aufgabenangemessenheit:
Die Anwenderinnen und Anwender können mit einem optimalen Einsatz von Zeit, Geduld sowie einer Gedächtnis- und Transferleistung ihre Aufgaben bewältigen.

02

Selbstbeschreibungsfähigkeit:
Die Anwendung bietet Orientierung. Den Benutzenden ist anhand der Navigationselemente jederzeit klar: „Wo komme ich her?“, „Wo bin ich?“ und „Wo kann ich von hier aus hin?“. Es wird dauerhaft ein Feedback durch die Benutzungsoberfläche gegeben (z. B. durch Swiping), das ein Gefühl von Sicherheit und Kontrolle erzeugt.

03

Erwartungskonformität:
Die mobile Anwendung unterstützt konsistent die Richtlinien (GUI-Guidelines und Styleguides) der jeweiligen Plattformhersteller. Die Art der Navigation und generell die Interaktion mit der Anwendung wird somit strikt eingehalten.

04

Fehlertoleranz:
Der Korrekturaufwand für Fehler ist gering. Die Interaktion ist „fehlertolerant“ und das beabsichtigte Ergebnis kann trotz möglicher Fehleingaben mit keinem oder minimalem Korrekturaufwand weiterhin erzielt werden.

05

Steuerbarkeit:
Die User können den Ablauf der Interaktion mit der App beeinflussen. Funktionen der Benutzerinteraktion sind unmittelbar ersichtlich.

06

Individualisierbarkeit:
Die Benutzungsoberfläche lässt sich individuell anpassen, z. B. die Kontrastanpassung für farbenblinde Anwenderinnen und Anwender oder die Schriftgröße für eine verbesserte Lesbarkeit von Texten.

07

Lernförderlichkeit:
Die mobile Anwendung kann intuitiv, d. h. ohne den Einsatz von Schulungen oder die Bearbeitung eines Handbuches bedient werden. Die Benutzungsoberfläche ist logisch aufgebaut und besitzt nachvollziehbare Abläufe.

08

Software-Qualitätssicherung:
Für den Massenmarkt: Zahlreiche Anpassungen und Testläufe wurden auf einer nennenswerten Anzahl von Geräten (z. B. über cloud-basierte Testumgebungen) durchgeführt, um die Marktdurchdringung sicherzustellen. Für Unternehmen: Die mobile Anwendung wurde ausreichend auf den im Unternehmen im Einsatz befindlichen Geräten getestet.

Wir bei SMF können hier auf langjährige Erfahrung unserer UI/UX-Expertinnen und -Experten zurückgreifen. Sie unterbreiten bei jedem Entwicklungsprojekt Empfehlungen und stellen den Kundennutzen sowie die Bedienbarkeit in den Fokus.

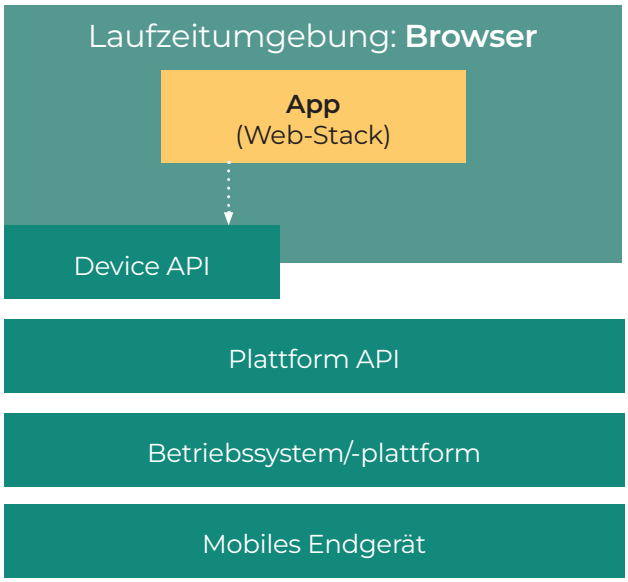
WAS DARF ES SEIN? PARADIGMEN FÜR DIE ENTWICKLUNG MOBILER ENDGERÄTE

Je nach Anwendungsszenarium existieren unterschiedliche Entwicklungsparadigmen. Sie alle haben Vor-, aber auch Nachteile, die es in der Definitionsphase zu beachten gilt.

01

Mobile Entwicklung

Die mobilen Webanwendungen, kurz „Web-Apps“ genannt, stellen die portabelste Variante der mobilen Anwendungsentwicklung dar. Sie werden mit Hilfe gängiger Webtechniken wie HTML5, CSS3 und Typescript bzw. Javascript entwickelt und innerhalb eines Webbrowsers ausgeführt. Sowohl Android als auch iOS können die Navigationsleiste ihrer Browser ausblenden. Dadurch füllt der Browser den ganzen Bildschirm aus. Durch eine geeignete Definition von Cascading Style Sheets (CSS) und durch Verwendung bestimmter Frameworks lässt sich ein durchdachtes Look-And-Feel entwickeln. Die Frameworks ermitteln über Javascript-API-Funktionen, in welchem Browser die Anwendung ausgeführt wird. Anschließend lassen sich über sogenannte „Browser-Weichen“ unterschiedliche CSS-Einstellungen für Android und iOS laden. Dadurch wird eine plattformspezifische Gestaltung der App ermöglicht.



Architekturübersicht mobiler Webanwendungen

Vorteile mobiler Entwicklung

Da bereits bekannte Programmiertechniken zum Einsatz kommen, ist der Einstieg für Web-Entwickelnde besonders einfach. Außerdem kommen die großen plattform-spezifischen Unterschiede zwischen Android und iOS nicht zum Tragen, da die durch die Browser (Safari auf iOS und Chrome für Android) zur Verfügung gestellte Laufzeitumgebung nahezu identisch ist. Dadurch werden die Entwicklungskosten erheblich reduziert.

Diese Klasse der mobilen Anwendungen wird nicht klassisch über den Appstore bzw. Playstore installiert. Stattdessen müssen die User zu einer Webseite navigieren, die die mobile Webanwendung bereitstellt. Anschließend kann über eine integrierte Funktion eine Referenz auf diese Anwendung auf dem Homescreen des mobilen Endgerätes platziert werden. Durch die Vermeidung des klassischen Vertriebsweges mit seinen zeitaufwändigen Test-, Freigabe- und Veröffentlichungsprozessen, werden weitere Kosten vermieden.

Da die mobile Webanwendung immer von einem Webserver geladen werden muss, benötigten Web-Apps in der Regel einen Netzwerkzugang, um korrekt arbeiten zu können. Die unter dem Begriff der „Progressive Web-Apps“ (PWA) bekannten Techniken erlauben zwar rudimentäre Offline-Fähigkeiten, dennoch können diese nicht mit einer klassischen nativen Entwicklung mithalten. Webbrowser, die diesen Standard implementieren, stellen beispielsweise eine Javascript-API (auch Device-API genannt) bereit, um auf bestimmte Hardwareressourcen zuzugreifen, wie zum Beispiel die Sensoren.

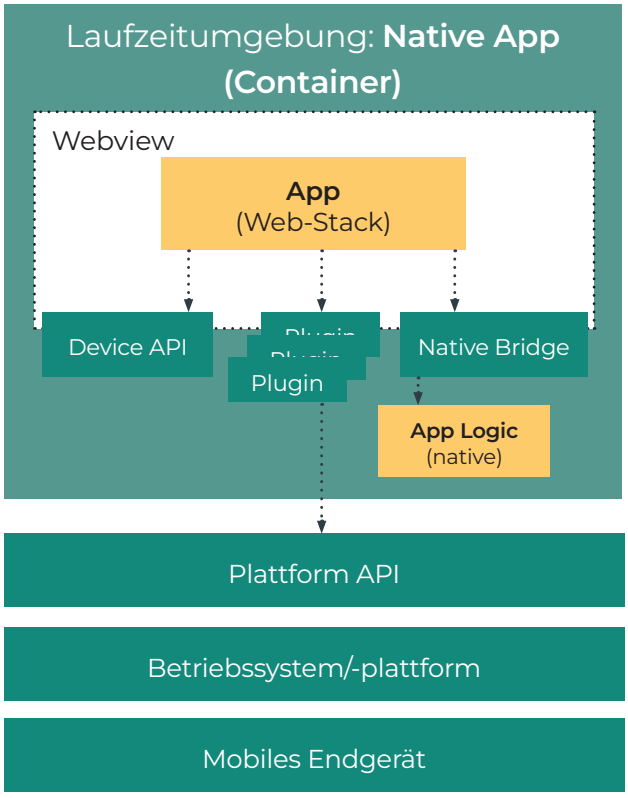
Nachteile mobiler Entwicklung

Da HTML und CSS deklarative Beschreibungssprachen für Oberflächen sind, müssen sie zeitintensiv interpretiert werden. Dies geht in der Regel zu Lasten der Performanz. Selbstverständlich werden auch hier die technischen Grenzen von Jahr zu Jahr verschoben, da moderne Webbrowser den Zugriff auf den Grafikprozessor erlauben und damit auch komplexe 3D-Grafik möglich machen. Dennoch sind die Anforderungen in manchen Anwendungsszenarien mit diesen Mitteln nicht zu erfüllen.

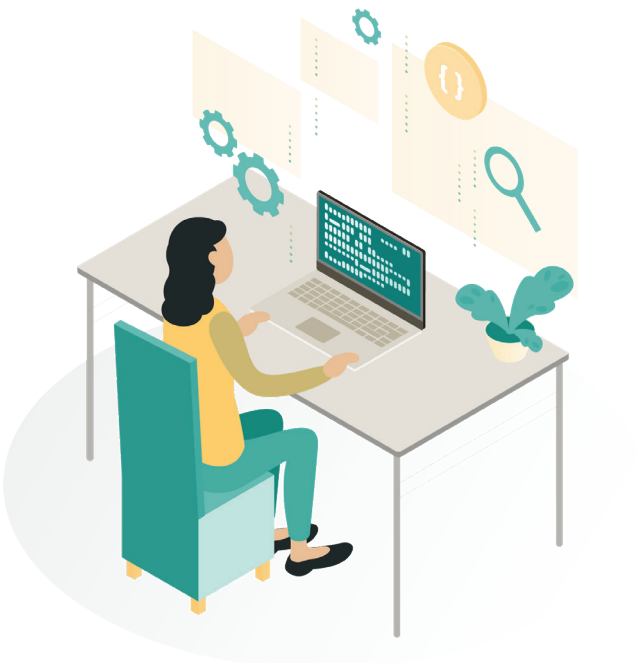
02

Hybride Entwicklung

Bei den hybriden mobilen Anwendungen handelt es sich ebenfalls um mobile Web-Apps, die als Laufzeitumgebung jedoch nicht den auf der Plattform gängigen Standard-webbrowser verwenden, sondern einen exklusiven, eigenen Container mitbringen. Dabei handelt es sich um eine native Anwendung, die die sogenannte „Webview“ verwendet, um die Weboberfläche darzustellen. Diese native Anwendung wird anschließend mit allen Ressourcen der Webanwendung verpackt und über die klassischen Vertriebskanäle (Appstore und Playstore) vertrieben. Im Gegensatz zu den mobilen Webanwendungen müssen diese Ressourcen demnach nicht von einem Webserver geladen werden, sondern sind beim Ausführen der mobilen Anwendung auf dem Endgerät vorhanden.



Architekturübersicht hybrider, mobiler Webanwendungen



Vorteile hybrider Entwicklung

Das Besondere an diesen Anwendungen ist, dass weiterhin bekannte Webtechniken für die Darstellung der Benutzungsoberfläche zum Einsatz kommen. Über die Device-API des PWA-Standards kann die mobile Anwendung zudem auf bestimmte Hardwarefunktionen zugreifen.

Da jedoch der Container exklusiv für diese Anwendung bereitgestellt wird, können aufwändige Berechnungen der Anwendungslogik auch mit nativen Mitteln umgesetzt werden. Über die sogenannte „Native Bridge“ kann die Webanwendung diese Funktionen anschließend aufrufen.

Um die Herstellkosten einer nativen App zu vermeiden, können bei hybriden Webanwendungen auch existierende Frameworks verwendet werden. Das Apache Cordova ist eines der bekanntesten Frameworks für hybride Anwendungen und kann auf Basis der Apache License 2.0 in eigene Anwendungen integriert werden. Darüber hinaus stehen aber auch noch weitere Frameworks wie zum Beispiel Capacitor zur Verfügung.

Sind bestimmte Gerätefunktionen über die Device-API nicht erreichbar, können bei Apache Cordova zusätzliche Plugins geladen werden, die die Aufgabe der „Native Bridge“ übernehmen. Plugins dienen demnach dazu, den hybriden Anwendungen die durch die Plattform API angebotene Funktionstiefe über eine zusätzliche Javascript-API zugänglich zu machen.

Nachteile hybrider Entwicklung

Hybride Apps kombinieren zwar die Vorteile nativer Entwicklungen mit mobilen Webanwendungen, aber auch die Nachteile. Zwar sind die Entwicklungskosten der Benutzungsoberfläche durch die Verwendung von Webtechniken im Vergleich zu einer nativen Entwicklung geringer, aber die hybride App muss weiterhin den komplexen Freigabeprozess der Plattformanbieter (Apple und Google) durchlaufen. Wird zudem die Anwendungslogik aus Laufzeitgründen auf die native Seite übertragen, muss die Logik in der Regel doppelt implementiert werden. Dadurch sind die Entwicklungskosten einer hybriden App teurer als bei mobilen Web-Apps.

03

Native Entwicklung

Native Apps werden auf Basis eines mobilen Betriebssystems und unter Zugriff auf plattform-spezifische APIs entwickelt. Dabei kommen auch spezielle Programmiersprachen zum Einsatz. Google hat mit der Einführung der Android-Plattform einen Schwerpunkt auf die Programmiersprache Java gesetzt. Native Apps werden dort mittlerweile überwiegend in Kotlin entwickelt, der von Google offiziell bevorzugten Programmiersprache für die Android-Entwicklung.

Zudem empfiehlt Google für die native UI-Entwicklung das Toolkit Jetpack Compose, mit dem Benutzeroberflächen deklarativ erstellt werden können, ähnlich wie SwiftUI für iOS. Mit Jetpack Compose können UI-Oberflächen und Elemente schneller erstellt werden, als mit dem klassichen XML-basierten View-System und belegen deutlich weniger Zeilen Code.

Um den Einstieg für Entwicklerinnen und Entwickler so gering wie möglich zu gestalten, hat das Unternehmen eine kostenfreie Entwicklungsumgebung, das sogenannte „Android Studio“ bereits im Jahre 2013 veröffentlicht. Da der Android Kernel in C/C++ entwickelt wurde, steht es den Entwickelnden frei, besonders in Bezug auf die Laufzeit anspruchsvolle Funktionen ebenfalls in diesen Programmiersprachen zu implementieren.

In iOS können Entwicklerinnen und Entwickler ihre Apps ebenfalls in C/C++ programmieren. Ursprünglich wurde iOS aber in Objective-C implementiert. Seit ihrer Einführung im Jahr 2014 hat sich die Programmiersprache Swift zum Standard für die Entwicklung von iOS-Anwendungen etabliert. Besonders vorteilhaft ist, dass Swift-Code nahtlos neben bestehendem Objective-C-Code im selben Projekt verwendet werden kann.

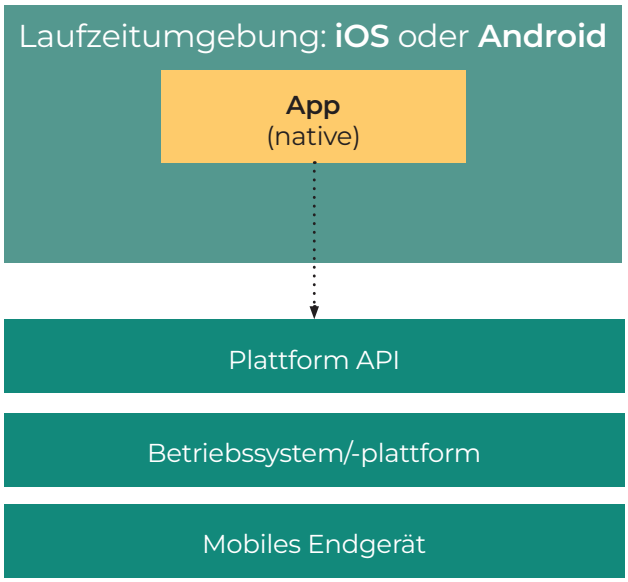
Dies erleichtert die Migration von Objective-C zu Swift erheblich, da neue Funktionen in Swift entwickelt werden können, während bestehender Objective-C-Code weiterhin genutzt wird. Ergänzend dazu bietet SwiftUI ein modernes Framework zur benutzerfreundlichen UI-Entwicklung. Sofern man einen Mac besitzt, kann man die kostenfreie Entwicklungsumgebung Xcode verwenden, um entsprechende native Apps für iOS zu entwickeln. Neben einer Entwicklungsumgebung benötigen Softwareentwickelnde für den Zugriff auf die plattformspezifischen Funktionen ein sogenanntes „Software Development Kit“ (SDK). Beide Plattformen stellen daher SDKs kostenlos bereit.

Vorteile nativer Entwicklung

Native Apps haben vollen Zugriff auf alle öffentlich zugänglichen Betriebssystemfunktionen und können in der Regel auch alle Hardwareressourcen ausschöpfen. Das betrifft sowohl Grafikfunktionen als auch gerätespezifische Sensoren. Native Apps werden daher bei laufzeitkritischen Anwendungen oder bei speziellen Anwendungsszenarien eingesetzt, bei denen die Steuerung sowie der Zugriff auf bestimmte Hardwarekomponenten von Bedeutung sind.

Nachteile nativer Entwicklung

Da eine native App für iOS nicht automatisiert in eine vergleichbare Android-App konvertiert werden kann, erfordert die Bereitstellung einer mobilen Anwendung auf beiden Plattformen häufig den doppelten Implementierungsaufwand. Das ist nicht gleichzusetzen mit der Verdoppelung der gesamten Herstellkosten, da die Aufwände in den vorgelagerten Entwicklungsphasen (Spezifikation und Entwurf) nicht erneut anfallen. Für ein kleines Softwareunternehmen oder eine kleine Entwicklungsabteilung kann das Vorhalten von spezialisierten Softwareentwicklerinnen und Softwareentwicklern für beide Plattformen aus wirtschaftlichen Gründen nicht tragfähig sein.



Architekturübersicht nativer Webanwendungen

04

Cross-Plattform-Entwicklung

Eine spezielle Form der nativen Entwicklung mobiler Anwendungen stellen die Cross-Plattform-Apps dar. Cross-Plattform-Apps werden mit Hilfe eines Frameworks entwickelt und dann auf dem mobilen Endgerät nativ ausgeführt. Man unterscheidet zwei verschiedene Vorgehensweisen.

Die erste Kategorie von Ansätzen stellt das Framework sowie das SDK für die Cross-Plattform-Entwicklung zur Kompilierungszeit bereit. Eine solche Cross-Plattform-App wird während des Kompilierungsvorganges in die native App der Zielplattform kompiliert. Dabei werden die eingesetzten API-Funktionen des Cross-Plattform-Frameworks auf entsprechende API-Funktionen der Zielplattform konvertiert. **Der prominenteste Vertreter dieses Ansatzes ist aktuell Flutter.**

Die zweite Kategorie kompiliert eine Cross-Plattform-App in eine plattformunabhängige Zwischensprache und setzt auf eine spezielle Laufzeitumgebung, die eine Konvertierung der API-Aufrufe des Cross-Plattform-Frameworks auf die Funktionen der Zielplattform zur Laufzeit durchführt.

Ein bekannter Vertreter dieses Ansatzes ist .NET MAUI (Multi-platform App UI), die als Nachfolger von Xamarin von Microsoft entwickelt wurde. .NET MAUI basiert auf der Mono-Laufzeitumgebung und ermöglicht die Entwicklung von Cross-Plattform-Apps auf Basis von C# mit einer einheitlichen Codebasis, die auf Android, iOS, macOS und Windows ausgeführt werden kann.

Ein relativ neues Framework auf diesem Feld ist Kotlin Multiplatform (KMP). Kotlin Multiplatform ist ein Framework, das Entwicklern ermöglicht, plattformübergreifenden Code zu schreiben und zu teilen, um die Entwicklung für verschiedene Plattformen wie Android, iOS, Web und auch Desktop zu erleichtern. Es basiert auf der Programmiersprache Kotlin und erlaubt es, gemeinsame Logik in einem einzigen Codebase zu implementieren, während plattformspezifische Implementierungen für UI und andere spezifische Funktionen weiterhin möglich sind.

Dadurch können Entwickler die Wiederverwendbarkeit von Code maximieren und gleichzeitig plattformgerechte Benutzererlebnisse bieten. Kotlin Multiplatform unterstützt auch eine Vielzahl von Bibliotheken und Tools, was die Integration in bestehende Projekte erleichtert. **Mit seiner Flexibilität ist es besonders attraktiv für Teams, die schnellere Entwicklungszyklen anstreben und Ressourcen effizienter nutzen möchten.**

Vorteile der Cross-Plattform-Entwicklung

Da die Cross-Plattform-Apps zur Laufzeit in die Zielplattform überführt werden, besitzen sie nahezu dieselben, sehr guten Performanzeigenschaften wie native Apps und sind dabei aber auch plattformunabhängig. Dieses Vorgehen senkt die Entwicklungskosten im direkten Vergleich zu einer nativen Entwicklung.

Nachteile der Cross-Plattform-Entwicklung

Alle Cross-Plattform-Frameworks abstrahieren die native API der spezifischen Plattform und stellen der Anwendungsentwicklung daher eine eigenständige SDK zur Verfügung. Da die Hersteller mobiler Betriebssysteme mit jeder aktuellen Version neue Betriebssystemfunktionen integrieren, sind bei einer Cross-Plattform-Entwicklung diese neuen Funktionen nicht erreichbar, sofern die SDKs nicht überarbeitet wurden. Durch die zusätzliche Abstraktion geht bisweilen auch zusätzliche Laufzeit verloren, so dass die Cross-Plattform-Apps zwar deutliche bessere Antwortzeiten besitzen, aber dennoch nicht an das Leistungsprofil nativer Anwendungen herankommen.

ENTSCHEIDUNGSHILFE: ENTWICKLUNGSPARADIGMEN IM DIREKTEN VERGLEICH

Die folgende Grafik bietet einen Überblick über die Vor- und Nachteile der einzelnen Entwicklungsparadigmen. Je nachdem, welches Kriterium für ein konkretes Entwicklungsszenario von Bedeutung ist, kann anhand dieser Gegenüberstellung eine bestimmte Entwicklungstechnik ausgewählt werden.

Kriterien	Web-Apps	Hybrid-Apps	Cross-Plattform-Apps	Native Apps
Performanz	-	+	+ +	+ +
Offline Nutzbarkeit	o	+	+ +	+ +
Native Funktionen	-	o +	+	+ +
Natives Look&Feel	-	-	+	+ +
Installation	+ +	+	+	+
Erreichbarkeit	+	o	o	o
Kosten	+ +	+	+	-
Wartung und Updates	+	o	o	o
Plattformunabhängigkeit	+ +	+	+	-
Wearables	- -	-	+	+ +

Legende: - schlecht o neutral + gut ++ sehr gut
Gegenüberstellung der Entwicklungsparadigmen

Performanz:
Native Apps sind im Hinblick auf die Performanz die beste Wahl. Da die Cross-Plattform-Apps aber ebenfalls nativ ausgeführt werden, sind sie in dieser Übersicht gleichgestellt. Da die hybriden Apps aufgrund der »Native Bridge« zeitkritische Funktionen nativ implementieren können, sind sie wesentlich schneller als mobile Webanwendungen.

Offline Nutzbarkeit:
Die nativen und hybriden Apps sind problemlos offline nutzbar. Bei den Web-Apps unterstützt zwar das Browser-Caching den Offline-Modus. Die Kapazitäten und Möglichkeiten sind dennoch stark eingeschränkt.

Zugriff auf native Funktionen:
Anforderungen, die einen engen Zugriff auf hardwarenahe Funktionen des mobilen Endgerätes verlangen, bedingen eine native Entwicklung. Obwohl die Web-Apps über die Device-API auf Funktionen des Betriebssystems und des Endgerätes zugreifen können, ist der Umfang kaum mit nativen Apps vergleichbar.

Native Apps können hingegen im vollen Umfang auf die Funktionen des Betriebssystems zugreifen. Dazu zählen das GPS, die Kamera, die Kontaktdaten, Gesten, Benachrichtigungen, u. v. m.

Hybride Apps können über Plugins sowie die »Native Bridge« auf alle Funktionen zugreifen. In der Praxis gestaltet sich die Fehlersuche bei Problemen aber schwierig.

Natives Look-And-Feel:
Mobile Webanwendungen und hybride Apps emulieren mit Frameworks und CSS-Einstellungen das native Look-And-Feel der Zielplattform. Für viele Anwendungen ist die Emulation ausreichend.

Dennoch kann diese Art der Darstellung nicht mit einer direkten, plattformspezifischen Ausführung der Benutzungsoberfläche mithalten. Sogar die Cross-Plattform-Apps haben zwar geringe, aber erkennbare Unterschiede zu der nativen Darstellung.

Installation:
Da mobile Webanwendungen nicht über den App Store und Playstore installiert werden müssen, sind sie am besten und einfachsten zu installieren, aber auch zu aktualisieren. Andererseits haben sich alle Anwenderinnen und Anwender an diesen Vertriebs- und Installationsweg für Apps gewöhnt und bei der Herstellung von mobilen Betriebssystemen wird dieser Installationsweg für die Anwendung so einfach wie möglich gestaltet.

Für Gelegenheitsanwendungen erscheint daher der Installationsweg einer Web-App vermeintlich als zu komplex. Aus technischer Sicht ist dieser Weg aber für die Softwareentwicklung am einfachsten.

Erreichbarkeit:
Sowohl Google als auch Apple haben ein großes Interesse daran, dass die auf diesen Plattformen angebotenen, nativen Apps gefunden und installiert werden. Daher investieren sie Aufwand darin, ihre App Stores und Playstores suchmaschinentauglich zu machen. Web-Apps können direkt im Browser ausgeführt werden und sind für Suchmaschinen daher einfach zu finden und zu nutzen, so dass sie eine größere Erreichbarkeit aufweisen. Da Web-Apps zudem nicht installiert werden müssen, lassen sie sich einfacher ausprobieren.

Kosten:
Soll eine App für ein breites Publikum, eventuell sogar für den Massenmarkt, entwickelt werden, verdoppeln sich bei einer nativen Entwicklung die Implementierungskosten. Daher ist es in den meisten Fällen günstiger, entweder eine Cross-Plattform-App, eine hybride App oder eine Web-App zu realisieren. Steht aber die Zielplattform aufgrund organisatorischer und technischer Vorgaben fest, sind native Entwicklungen durchaus auch aus Kostensicht wettbewerbsfähig.

Wartung und Updates:
Es gelten die gleichen Argumente wie bei der Installation. Da aber der Freigabeprozess über die konventionellen Vertriebswege der Plattformhersteller komplexer ist, sind native Anwendungen nachteiliger als Web-Apps.

Plattformunabhängigkeit:
Die Plattformunabhängigkeit ist bei einer klassischen nativen Entwicklung nicht gegeben. Daher muss eine App für Android und iOS komplett manuell entwickelt werden. Selbstverständlich teilen sich beide Apps dieselben Anforderungen und Konzepte. Dennoch ist die App unter der

Berücksichtigung der plattformspezifischen Funktionen jeweils separat zu entwickeln. Aufgrund der einheitlichen Laufzeitumgebung bieten Web-Apps auch hier Vorteile.

Wearables:
Unter Wearables werden intelligente Geräte verstanden, die man am Körper trägt. Das können beispielsweise Brillen oder Uhren sein. Da die Hardwareausstattung dieser Geräte bei weitem nicht mit der Ausstattung eines Smartphones mithalten kann, ist aufgrund der technischen Restriktionen lediglich die Entwicklung von Cross-Plattform-Apps oder nativen Apps möglich.

Obwohl die nativen Apps zahlreiche Vorteile gegenüber anderen Entwicklungsansätzen bieten, sollte man nicht den Fehler machen, in jedem Projekt immer „auf dasselbe Pferd" zu setzen. Denn unter den Kostengesichtspunkten sind die Web-Apps nicht zu schlagen. Werden demnach keine hohen Anforderungen in Bezug auf die Antwortzeiten oder den Zugriff auf die Sensorik gestellt, kann man mit Hilfe dieses Entwicklungsparadigmas wirtschaftlich solide Lösungen kreieren.

Ein guter Mittelweg kann die Nutzung von Cross-Plattform-Technologien wie Flutter sein. Flutter bietet die Möglichkeit, mit einer einzigen Codebasis performante und plattformübergreifende Apps zu entwickeln, die nahezu native Performance bieten, aber kosteneffizienter als reine native Apps sind. So lassen sich sowohl wirtschaftlich solide als auch leistungsfähige Lösungen realisieren.

WARUM FRAMEWORKS EINE SO GROSSE BEDEUTUNG HABEN

Aufgrund des hohen Bedarfs für mobile Anwendungen und den verschiedenen Entwicklungsparadigmen ist ein sich kontinuierlich ändernder Markt für Frameworks und Entwicklungsumgebungen entstanden. Im Folgenden sind die gängigsten Frameworks kategorisiert.

Die Frameworks, die sich in der Kategorie der mobilen Web-Apps befinden, können selbstverständlich auch für die Benutzungsoberfläche hybrider Apps verwendet werden. Hybride Entwicklungsframeworks dienen der Entwicklung des nativen Containers aus der Ausführungsumgebung und erlauben es, besonders laufzeitkritische Funktionen unter Zugriff nativer Operationen umzusetzen.

Web-Apps	Hybrid-Apps	Cross-Plattform-Apps	Native Apps	IDEs
Sencha Kendo UI Onsen UI Quasar Vue.js SVELTE	CORDOVA IONIC Framework NativeScript	Flutter React Native .NET MAUI kivy CODENAME ONE	android PowerApps	Visual Studio apache Flex eclipse

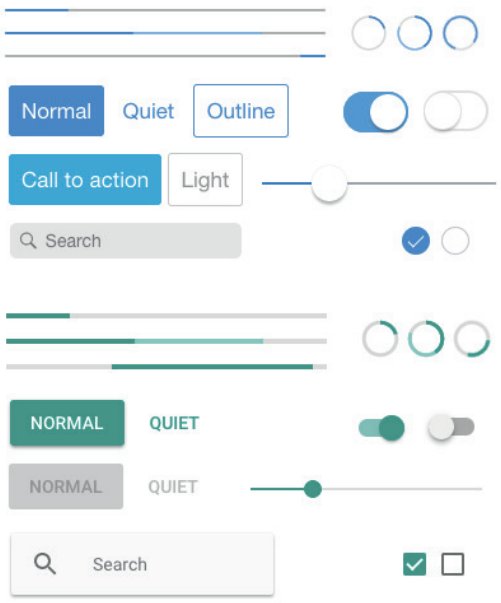
Übersicht gängiger Frameworks und Entwicklungsumgebungen

Die Entwicklerinnen und Entwickler dieser Frameworks legen aus Gründen der hohen Innovationsgeschwindigkeit den Quellcode häufig offen, so dass eine Vielzahl der abgebildeten Frameworks „Open Source“ sind. Bei den von Apple angebotenen SDKs für die App-Entwicklung auf iOS Endgeräten konnte keine exakte Einordnung stattfinden, da das SDK nicht separat, also unabhängig von einer bestimmten Entwicklungsumgebung, bezogen und installiert werden kann. Daher befinden sich sowohl die Entwicklungsumgebung Xcode als auch die Programmiersprache und Laufzeitumgebung Swift zwischen den beiden letzten Spalten.

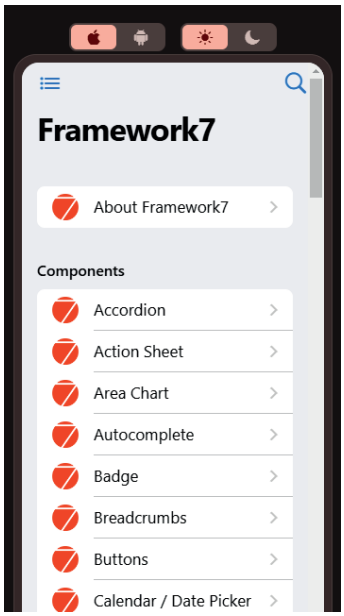
Der Einsatz von verfügbaren Frameworks zur mobilen Entwicklung sollte bei Web-Apps immer bevorzugt werden, bevor eine eigene Entwicklung auf Basis von gängigen Web-Frameworks aufgenommen wird, da der Reifegrad und das Look-And-Feel dieser Ansätze sehr weit fortgeschritten sind.

BEISPIELE EINER WEBBASIERTEN OBERFLÄCHE

Die nachfolgenden beiden Beispiele zeigen webbasierte Oberflächen, welche mit Hilfe der Frameworks Onsen UI und Framework7 realisiert wurden:



Darstellungsbeispiel Onsen UI Framework



Darstellungsbeispiel Framework7

ECHTE KUNDEN, ECHTE LÖSUNGEN: UNSERE CASE STUDIES

Wir entwickeln maßgeschneiderte, intuitive mobile Applikationen und sorgen für das Zusammenspiel aller System-Schnittstellen oder kompletter Backend-Systeme unter iOS und Android, auf Smartphone, Tablet oder Watch. Zwei Case Studies zeigen, wie erfolgreich das funktioniert:

Eurowings: Mobile Smartphone-Applikation FOCUS

Die deutsche Fluggesellschaft bietet täglich rund 700 Flüge an und bedient weltweit über 160 Stationen. Um die jeweiligen Dienstleister am Boden in Echtzeit und mobil mit allen benötigten Informationen zu versorgen, haben wir die mobile Smartphone-Applikation „FOCUS“ konzipiert und umgesetzt – und begeistern seitdem damit Ramp Agents, Cockpit Crews und Fluggast-Betreuende.

Stadt Monheim: Smart-City Fahrrad-App

Für die smarte und stationsgebundene Realisierung eines Bike-Sharing-Systems für Bürgerinnen, Bürger und touristische Gäste einer mittelgroßen Stadt in NRW haben wir eine spezielle App entwickelt. Diese zeigt für 31 Stationen über 450 Fahrräder auf einer Map an, die sich dann auswählen, buchen und online bezahlen lassen. Die App wurde erfolgreich in den Tourist-Verleih, das Stadt- und Pass-System sowie ein Rental-System integriert und mit Backend, IAM, Keycloak und REST-API realisiert.

REMONDIS: RETRACK App

REMONDIS ist das größte deutsche Unternehmen für Umwelt- und Recyclingdienstleistungen. Es bietet Lösungen im Abfallmanagement, Recycling und Umweltschutz. Im Rahmen einer Vielzahl von Geschäftsprozessen der Abfallwirtschaft werden dabei unterschiedliche Behälterarten verwendet. Um papierlose und damit effizientere UVV-Prüfungen zu erreichen, wurde ein digitalisiertes, zentrales Behältermanagement geschaffen: RETRACK, eine Kombination aus Asset Management und Prozessdigitalisierung.

FAZIT UND HANDLUNGSEMPFEHLUNG

Unternehmen müssen eigene mobile Strategien entwickeln, um die steigende Nachfrage nach mobilen Lösungen zu bedienen. Wir bei SMF bieten dafür die notwendige Technologieberatung und sind der richtige Partner für erfolgreiche App-Entwicklung. Wir digitalisieren bereits seit 1985 und setzen seit den frühen Anfängen des Mobile Computing zahlreiche Projekte erfolgreich um.
Unsere Expertinnen und Experten wissen, welche Voraussetzungen für eine erfolgreiche App-Entwicklung notwendig sind, sie kennen alle gängigen Entwicklungsparadigmen und helfen Ihnen bei der Auswahl der richtigen Strategie, damit auch Ihre App erfolgreich wird.

Im Gespräch mit Ihnen erarbeiten wir, wo sich Optimierungspotenziale in den Geschäftsprozessen erschließen lassen und an welcher Stelle die Einführung spezieller, individueller Applikationen (Apps) sinnvoll ist. Gemeinsam wählen wir dann die wirtschaftlichste Lösung aus.

Vereinbaren Sie noch heute einen unverbindlichen Termin mit uns.
Wir freuen uns auf das Gespräch mit Ihnen!

Ihr Ansprechpartner

Jan Metzmacher
Segment Manager
Industry

SMF GmbH

j.metzmacher@smf.de

WO WIR SIND

SMF GmbH
Paul-Henri-Spaak-Straße 5, 44263 Dortmund

T. +49 231 9644-0
F. +49 231 9644-100

info@smf.de
www.smf.de

smf
YOUR IT PARTNER