

Mobile App Development Medientechnik 5





Arten von Apps:

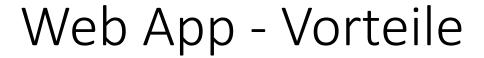
- Web Apps
- Native Apps
- Hybride Apps
- Cross Platform Apps





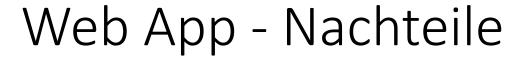


- Eine Web App ist eine vollständig browserbasierte Anwendung.
- Die Bereitstellung erfordert einen Webserver.
- Die Funktionalität wird mit Technologien wie
 - HTML
 - CSS
 - Javascript React, Angular
 - Java/Kotlin Spring
 - C# ASP.NET
 - PHP Laravel
- Die Web App ist für die Verwendung von mobile Geräten optimiert, kann aber plattformunabhängig überall aufgerufen werden.





- Einsatz bewährter Webtechnologien
- Zielgruppen sind mit dem Umgang modernen Webapplikationen vertraut
- Halbe Entwicklungszeit im Vergleich zu nativen Apps
- Verfügbarkeit für alle Plattformen mit kompatiblen Browsern
- Keine Kompatibilitätsprobleme, da alle User die gleiche Version verwenden (das reduziert gleichzeitig Kosten für Support und Wartung)





 Internetzugang notwendig / nicht alle App-Funktionen sind offline verfügbar

Keine Veröffentlichung in Stores (Google Play und App Store)

 Nicht alle Funktionen mobile Betriebssysteme sind über den Browser zugänglich









- PWAs kombinieren das Beste aus Web- und App-Technologien
- PWAs sind progressiv, was bedeutet, dass sie auf jedem Gerät oder in jedem Browser funktionieren, unabhängig von der Fähigkeit des Geräts.
- Reaktionsfähig (**responsive**): PWAs passen sich unterschiedlichen Bildschirmgrößen und –auflösungen an.
- PWAs können teilweise offline verwendet werden oder in Bereichen mit schlechter oder keiner Internetverbindung funktionieren (Caching von bereits geladenen Inhalten)
- App-ähnliches Verhalten (installierbar): Ablage am Startbildschirm, Push-Notifications und Gerätefunktionen wie Kamera, GPS, Mikrofon, ...





Veröffentlichte PWA: https://www.findpwa.com/

- Checklist:
 - Performante Website
 - Funktionalität in allen Browsern gewährleistet
 - Responsive Design für alle Auflösungen
 - Eine Seite die angezeigt wird, falls der Anwender offline ist (offline page)

Beispiel Temperatur-Konverter





Native App

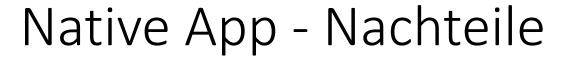
- Eine native Anwendung wird speziell für **EIN** (mobiles) Betriebssystem des jeweiligen Endgerätes konzipiert und entwickelt
- Vertrieb über App Stores des jeweiligen Betriebssystems
- Native App-Komponenten werden für User Interfaces verwendet und gewährleisten ein einheitliches Aussehen/Verhalten für das jeweilige Betriebssystem

 - Native iOS App (Objective C oder Swift) → erfordert ein iOS-Endgerät (App Store)





- Bestmögliche Leistung, Geschwindigkeit und Benutzererfahrung (UX) auf dem jeweiligen Betriebssystem
- Zugriff auf spezifische Gerätefunktionen (Kamera, Push-Benachrichtigungen, GPS, Gyroskop, usw.)
- Bestimmte Funktionen und Daten sind offline verfügbar
- Betriebssystem-Updates können in der nativen App Entwicklung sofort genutzt werden (andere Ansätze erfordern ein Update des eingesetzten Frameworks, was oft zeitverzögert passiert)





- Entwicklung individuell f
 ür das jeweilige Betriebssystem
 - höhere Kosten und Aufwände wenn die App plattformübergreifend angeboten werden soll
 - parallele Entwicklung von iOS und Android Version spart Zeit, die Kosten sind aber oft fast doppelt so hoch
 - doppelter bzw. dreifacher Quellcode (iOS, Android, Webbrowser)







- Eine hybride App ist eine Browseranwendung in native Gewand.
 - basiert auf gängigen Web-Technologien
 - läuft in einem Webview Container in dem sie als native App verpackt wird
 - greift über Wrapper auf native APIs und Funktionen des Betriebssystems zu

Export der entwickelten Webanwendung auf bestimmtes (mobiles)
 Betriebssystem

• Tools: Capacitor, Cordova, Electron







- Vorteile
 - Zeit und Kostenersparnis durch einmaligen Aufwand für die Erstellung einer Website
 - Wartung von nur einer gemeinsamen Codebase
 - Bereitstellung von hybriden Apps in Stores möglich
- Nachteile
 - Teilweise eingeschränkter Zugriff auf native Funktionen







- Cross Platform Apps werden in einer Programmiersprache entwickelt, die nicht zum Betriebssystem des Geräts gehört.
- Code wird mit Hilfe eines plattformübergreifenden Framework in die native Systemsprache des Ziel-Betriebssystems kompiliert.
- Ein Großteil des Codes wird für alle Plattformen geteilt (Shared), nur kleine Teile werden plattformspezifisch mehrfach entwickelt (meist hardwarenahe Funktionalitäten)
- Frameworks: Flutter, React Native, .NET MAUI (Xamarin)







• Vorteile:

- Annähernd native Leistung und gute Benutzerfreundlichkeit
- Zeit und Kostenersparnis, da der Code für verschiedene Plattformen geteilt wird
- Wartungsfreundlichkeit Änderungen müssen nur an einer Stelle vorgenommen

Nachteile:

- Längere Wartezeit für neue native Features (Framework-Anbieter müssen neue Features erst bereitstellen)
- Extra-Aufwand für Entwickler durch plattformübergreifende Einschränkungen bezüglich UI/UX-Anpassung
- Abhängigkeit von Drittanbieter-Frameworks

