# **Anforderungsanalyse (Für die Feature-Test-Anwendung oder für „das Testen“?) -> Für das Testen allgemein UND für die Feature-Test-App (eigenes Unterkapitel)**

# **FA: Nutzung von eingebauten Hardware- und Software-Komponenten**

Lagesensor/Gyroskop

Es soll geprüft werden, ob der Lagesensor von einer Anwendung, welche mit dem zu testenden Framework entwickelt wurde, ansprechbar ist um die Lage des Smartphones ermitteln zu können.

Näherungssensor

Es soll geprüft werden, ob der Näherungssensor eines Smartphones muss weiterhin nutzbar ist.

Kommunikation

Es soll getestet werden, ob Kommunikationen via Bluetooth und WiFi für Gerät-zu-Gerät- und Gerät-zu-Netzwerk-Kommunikation möglich sind.

Notifications

Es soll überprüft werden, ob Notifications über eine in dem zu testenden Framework entwickelten Anwendung ausgegeben werden können.

Speicher

Es soll geprüft werden, ob Dateien in den lokalen Speicher des Smartphones geschrieben, gelesen und auch wieder gelöscht werden können. Hierbei soll zusätzlich überprüft werden, welche Datenhaltungsformate unterstützt werden.

Umgebungslichtsensor

Der Umgebungslichtsensor des Smartphones soll auch bei Anwendungen, welche mit dem zu testenden Framework entwickelt wurden, weiterhin aktiv sein und die Helligkeit des Displays dem Umgebungslicht anpassen.

# **FA: Die grafische Benutzeroberfläche**

Ein natives Look-And-Feel einer mobilen Anwendung bedeutet, dass die grafische Benutzeroberfläche dem Benutzer das Gefühl einer nativen Anwendung vermittelt. Die ausgewählten Frameworks müssen hierzu die Möglichkeit bieten, native Bedienelemente bei der Entwicklung von Anwendungen zu verwenden. Es muss möglich sein die Oberfläche einer Anwendung nach dem Design-Standard des entsprechenden Betriebssystems (Android, iOS) aufzubauen. Hier wird im Falle des Betriebssystems Android der aktuelle Design-Standard ‚Material Design‘ (X) gefordert. Um dies zu testen ist es notwendig in die Test-Anwendung gängige Bedienelemente wie zum Beispiel Action-Bars, Floating Buttons und Navigationselemente zu integrieren.

# **FA: Betriebssysteme**

Für jedes zu untersuchende Framework soll ermittelt werden, für welche Plattformen sich Anwendungen damit entwickeln lassen. Der Fokus liegt hierbei auf den mobilen Betriebssystemen Android und iOS, da diese den Großteil des Marktes ausmachen (vgl. Kapitel X). Hierbei soll auch berücksichtigt werden, wie viele und welche ältere Versionen der Betriebssysteme vom entsprechenden Framework unterstützt werden und inwieweit eine Anwendung abwärtskompatibel ist. Auch soll untersucht werden wie viel Zeit die Hersteller der Frameworks durchschnittlich benötigen um auf ein Betriebssystemupdate seitens iOS und Android zu reagieren.

Zudem wird ermittelt auf welchen Betriebssystemen die Frameworks für die Entwicklung installiert werden können.

# **FA: Sicherheit**

Bezüglich der Sicherheitsaspekte von mobilen Anwendungen soll überprüft werden, ob und inwieweit das Rechtemanagement der jeweiligen Plattform (Android, iOS) nativ unterstützt wird. Dies beinhaltet unter anderem die Regelung der Zugriffe auf Gerätefunktionen wie Speicher oder die Kameras. Auch wird untersucht, ob es Möglichkeiten zur Hinterlegung von Sicherheitszertifikaten gibt und ob ein eigener privater Zertifikatspeicher eingerichtet werden kann.

# **FA: Interoperabilität/Erweiterbarkeit**

Unter dem Sammelpunkt Interoperabilität und Erweiterbarkeit finden sich Prüfungen, die sich auf eine mögliche Anpassbarkeit der Frameworks und die Nutzung von Bibliotheken von Fremdanbietern beziehen. In diesem Zuge wird auch ermittelt, ob sich die Frameworks in gängige IDEs, wie zum Beispiel Visual Studio oder Eclipse integrieren lassen und ob es eine Auswahl an Programmiersprachen gibt, mit denen entwickelt werden kann.

# **FA: Tests und Performance**

Bei der Evaluierung von Frameworks für Crossplatform- beziehungsweise hybrider Anwendungs-entwicklung wird auch untersucht inwiefern die einzelnen Frameworks Werkzeuge zum Schreiben und Durchführen von (automatisierten) Tests selbst anbieten oder unterstützen. Die Anwendung, die als Evaluationshilfe im Rahmen dieser Arbeit entwickelt wird, soll zudem Möglichkeiten bieten die Performance der Cross-Platform-Anwendungen mit einer nativen Anwendung zu vergleichen. Hierzu muss die zu entwickelte Test-Anwendung Messungen der Reaktionszeiten der Anwendung ermöglichen.

# **NFA: Hersteller**

Als eine der nichtfunktionalen Anforderungen wird die Bedeutsamkeit der Hersteller/Anbieter der einzelnen Frameworks recherchiert. Hierzu zählen Attribute wie Unternehmensgröße, Bekanntheitsgrad (Score), Produktpalette und Einsätze. Zudem werden in diesem Rahmen auch die Entwicklungsstadien der Frameworks näher durchleuchtet und Fragen hinsichtlich der aktuellen Version und der Anzahl vergangener Updates beantwortet. Auch wird untersucht, in welchem Umfang versprochene Funktionalitäten bereits umgesetzt wurden und welche Erweiterungen und Verbesserungen in Planung sind.

# **NFA: Support und Community**

Die Untersuchung des Umfangs des Hersteller-Supports findet unter Berücksichtigung folgender Gesichtspunkte statt: Werden Tutorials und Beispiellösungen für verschiedene Problemstellungen geboten? Werden Online-Schulungen und/oder Schulungen mit Anwesenheit angeboten? Gibt es eine Dokumentation, die regelmäßig gepflegt wird? Gibt es ein Forum, in dem Fragen gestellt werden können, die von einem Support-Team des Herstellers beantwortet werden?

Zusätzlich zum Hersteller-Support soll auch die Größe der jeweiligen Community betrachtet werden. Die Größe wird dabei anhand der Internetpräsenz bemessen.

# **NFA: Entwicklung**

Ein wichtiger Faktor bei der Evaluation von Frameworks für hybride und Cross-Platform-Anwendungsentwicklung ist der Entwicklungsprozess mit den entsprechenden Frameworks. Unter diesem Aspekt werden unter anderem die Leichtigkeit der Installation und der Benutzung der Frameworks verglichen. Gemessen wird dies an der notwendigen Einarbeitungszeit und der Zeit die insgesamt für Installation und Einrichtung aller für die Entwicklung notwendigen Werkzeuge benötigt wird. Hier fließt auch der Umfang der Bibliotheken, welche die Frameworks mit sich bringen, in Form folgender Fragestellungen ein: Welche Bausteine für die Entwicklung werden angeboten und was muss alles selbst implementiert werden? Zentrales Kriterium ist auch der Anteil des plattformübergreifend nutzbaren Codes am Gesamtcode, hier am Beispiel der Funktionstest-Anwendung, welche als Werkzeug zur Evaluation dient.

# **NFA: Lizenz und Kosten**

Bei der Evaluation werden auch etwaige Kosten und Lizenzmodelle der zu untersuchenden Frameworks betrachtet. Hier werden die einzelnen Frameworks hinsichtlich etwaiger Kaufpreise, Monats- oder Jahreslizenzpreise und Kosten für Support sowie Minor und Major Upgrades untersucht.

# **Abgrenzungskriterien**

Im Rahmen dieser Projektarbeit wird eine Marktanalyse durchgeführt, um die 5 aktuell relevantesten Frameworks aus dem Bereich hybrider und Cross-Platform-Anwendungsentwicklung auszuwählen. Die als Evaluationswerkzeug dienende Funktionstest-Anwendung wird einmal nativ als Referenz und anschließend nur mit diesen 5 ausgewählten Frameworks implementiert. Weitere Frameworks werden nur aufgrund ihrer Spezifikationen und Literatur untersucht. Aufgrund der zur Verfügung stehenden Hardware und Software wird die native Variante der mobilen Anwendung, welche als Evaluationswerkzeug entwickelt wird, ausschließlich in Android für ein Android Smartphone entwickelt. Somit werden sich sämtliche Teile der Evaluation, welche mit Hilfe der Funktionstest-Anwendung erarbeitet werden, nur auf die Android-spezifischen Bereiche der Frameworks beziehen können. Bei der Evaluation des nativen Look-And-Feel anhand des GUI-Designs wird sich an den aktuellen Richtlinien für das Material Design von Android orientiert.

# **Bewertungskriterien**

Für die Evaluierung der Frameworks wird zunächst ein Katalog mit Bewertungskriterien angefertigt, anhand dessen die einzelnen Frameworks bewertet werden. Der Katalog besteht dabei aus Kriterien, nach denen alle in Kapitel X vorgestellten Frameworks allein durch Recherche untersucht werden können. Zusätzlich werden noch Kriterien aufgenommen, die sich auf die praktische Anwendung der in Kapitel X ausgewählten 5 Frameworks beziehen, mit welchen die in Kapitel X vorgestellte Funktionstest-Anwendung testweise implementiert wird.

Um oben beschriebenen Kriterienkatalog aufzustellen, wurde zunächst eine Umfrage durchgeführt. Die Umfrage richtete sich an (ehemalige) Studierende der Fachhochschule Dortmund, welche angaben, bereits Erfahrungen in der mobilen Anwendungsentwicklung gesammelt zu haben. Die Fragestellung der Umfrage lautete: „Welche Kriterien halten Sie für wichtig in der Entwicklung mit Frameworks für hybride, bzw. Cross-Platform-Anwendungen?“ (E-Mail siehe Anhang) Die Befragten antworteten hierauf mit einem Freitext, indem sie die für sich wichtigen Kriterien zur Bewertung von Frameworks aufzeigten. Diese Kriterien wurden in einem Dokument (Anhang) gesammelt und Mehrfachnennungen entsprechend gekennzeichnet. Ein Kriterium, welches von mehreren Befragten als wichtig beschrieben wurde, bekommt bei der Evaluation eine höhere Gewichtung zugeschrieben, als ein Kriterium, welches nur von einem Befragten genannt wurde. Aus dieser Zusammenfassung der ermittelten Bewertungskriterien wurde anschließend eine Matrix (Abbildung X) erstellt. Der Grad der Erfüllung eines Kriteriums wird an der Anzahl an Punkten zu erkennen sein, die vergeben werden. Hier ist eine Skala von 1 bis 5 (gar nicht bis voll erfüllt) vorgesehen. Wobei hier zu jedem Kriterium zunächst definiert werden muss, was „voll erfüllt“ bedeutet. Neben diesen Kriterien wird es auch Kriterien geben, welche nur mit „vorhanden“ und „nicht vorhanden“ bewertet werden können.

--- Bild Matrix ---

## **Kategorie Kosten, Lizenz**

In der Kategorie Kosten und Lizenz gibt es einen Sonderfall bei dem Punkt „Lizenzmodell“. Es ist hier nicht möglich eine konkrete Abstufung von „gut“ nach „schlecht“ in 5 Kategorien zu schaffen, da verschiedene Lizenzmodelle verschiedene Vor- und Nachteile aufweisen können. So wird sich darauf beschränkt, lediglich eine kurze Beschreibung des jeweiligen Lizenzmodells, sofern es eins gibt, zu formulieren.

Die Punkte „Fixkosten“, „Lizenzkosten“ und „Supportkosten“ werden in 5 Abstufungen von „gratis“ bis „teuer“ unterteilt. Die exakten Geldwerte für die Grenzen der Abstufungen werden sich aus dem günstigsten und teuersten Preis, welche sich zu den Frameworks ermitteln lassen, ergeben.

## **Kategorie Support, Community**

Bezüglich des Punktes 2)a) „Support/Wartung vom Hersteller“ wird der Umfang des Supports pro Framework ermittelt und anhand dessen die Abstufung vorgenommen. Dies umfasst zum Beispiel telefonischen Support, ein Ticket-System oder ein moderiertes Forum. Die Qualität von Punkt 2)b) „Dokumentation“ wird unter anderem am Umfang der Dokumentation selber und am Update-Zyklus bemessen, d.h. wie schnell wird die Dokumentation entsprechend Änderungen am Framework aktualisiert? Bei der Untersuchung der Größe der Community werden, falls vorhanden, das Framework eigene Forum und weitere Entwicklerforen nach Anzahl Threads mit Themen, die sich auf das jeweilige Framework beziehen, durchleuchtet. Das Schulungsangebot wird sowohl nach Qualität als auch nach Quantität beurteilt: Das bedeutet, dass unterschieden wird, ob nur Online- oder auch Anwesenheitsschulungen angeboten werden. Hierbei spielt es auch eine Rolle an wie vielen unterschiedlichen Standorten und wie häufig Schulungen angeboten werden. Zudem wird der Angebotsumfang an Schulungen hinsichtlich einer Aufteilung von Anfänger- bis Spezialschulungen bewertet.

## **Entwicklung**

Die Kategorie „Entwicklung“ betrifft nur die 5 ausgewählten Frameworks, mit denen die Funktionstest-Anwendung entwickelt wird. Für die Leichtigkeit der Installation und Benutzung des Frameworks werden Punkte von 1 bis 5 vergeben: 1 bedeutet kompliziert und aufwändig und 5 bedeutet automatisiert und intuitiv. Als „Einarbeitungszeit“ wird in diesem Falle die Zeit von der Fertigstellung der Installation bis nach dem Schreiben der ersten funktionierenden Seite der Anwendung angesehen. Hier gilt je kürzer, desto besser. Bei Punkt 3)c) „plattformübergreifend nutzbarer Code“ inwieweit der mit dem jeweiligen Framework erzeugte Code plattformübergreifend nutzbar ist, beziehungsweise wie hoch der Anteil des plattformspezifischen Codes ist. Hier gilt entsprechend, je mehr Code plattformübergreifend nutzbar ist, desto besser. Unter „Umfang Bibliotheken/Templates“ wird verstanden, wie viele Bausteine mit dem Framework sozusagen mitgeliefert werden, die einzelne Funktionalitäten kapseln und einfach in den Code zu integrieren sind.

## **Hersteller**

Die „Bekanntheit/Größe des Herstellerunternehmens“ des jeweiligen Frameworks wird anhand der Unternehmensgröße, des Bekanntheitsgrads, der Produktpalette und Einsätze der Produkte gewertet. Bei dem Punkt „Entwicklungsstadium des Frameworks“ wird in die Wertung mit einbezogen, ob sich das Framework noch in einer Beta-Version befindet, oder es schon ein Release gibt. Zudem wird betrachtet, wie viele Updates und Bugfixes bereits vorgenommen wurden und wie viel von dem was das Framework laut Hersteller leisten soll bereits umgesetzt oder noch in Planung ist. Eine „Ja“-„Nein“-Unterscheidung gibt es bei dem Punkt „Weiterentwicklung des Frameworks“. Hierbei ist positiv zu bewerten, wenn das Framework nachweislich noch weiterentwickelt werden soll, um mit den Änderungen und der Weiterentwicklung der einzelnen Plattformen mithalten zu können.

## **OS-Versionen**

„Reaktionszeit auf OS-Update“ beschreibt die Zeit, die benötigt wurde um nach einem Update eines Betriebssystems, wie zum Beispiel Android, das Framework entsprechend anzupassen oder zu erweitern. Mit dem Punkt „Abwärtskompatibilität“ wird bewertet, bis zu welcher OS-Version das Framework anwendbar ist. Hier ist zum Beispiel für Android nicht ohne Bedeutung, dass weiterhin Version 4.x unterstützt wird, die unter den Android Versionen den größten Marktanteil annimmt (QUELLE!).

# **Marktanalyse**

## **ReactNative**

React Native nutzt die Bibliothek React von Facebook zur Entwicklung für die Plattformen Android, iOS und die Universal Windows Platform. Programmiert werden React Native Anwendungen in JavaScript und HTML. Der Code wird anschließend in native Quelldateien übersetzt, weswegen React Native verspricht native Komponenten ohne großen Aufwand einbinden zu können und flüssige Animationen zu ermöglichen.

## **Ionic**

…

# **Framework Analyse**

## **Xamarin**

### Kosten, Lizenz

Die Kostenstaffelung bezieht sich hauptsächlich auf die Visual Studio – Version, die erworben werden kann. Es gibt 2 kostenlose Community Varianten: einmal eine Visual Studio Community Version und eine Xamarin Studio Community Version für die Entwicklung auf Apple Geräten. In beiden Varianten sind die namensgebenden Entwicklungsumgebungen enthalten. Die Preise der kostenpflichtigen Versionen entsprechen dem Preis für die jeweilige Visual Studio Version, die mit erworben wird.

Quelle: https://www.xamarin.com/compare-visual-studio

Community Version

* Kostenlos
* Enthält das Xamarin SDK
* Open Source Software-Development
* Alle Komponenten für die Cross-Platform Entwicklung sind enthalten
* Support über Community und Foren

Professional

* Visual Studio Professional
* Xamarin Studio Professional (für Mac)
* Kosten siehe Visual Studio Professional
* Keine zusätzlichen Cross-Platform Entwicklungskomponenten
* Technischer Support inklusive (bei subscription)

Enterprise

* Visual Studio Enterprise
* Xamarin Studio Enterprise (für Mac)
* Zusätzliche Enterprise Features:
  + Bytecode Hiding
  + Live App Inspection
  + Profiler
  + Test Recorder DIE add-ins
* 25% Ermäßigung für die Xamarin Test Cloud
* Technischer Support inklusive (bei subscription)

Preise:

Professional:

* 499$ ohne Subscription
* 1199$ mit Subscription; Renew: 799$

Enterprise:

* Nur mit Subscription möglich!
* 5999$; Renew: 2569$

Eine Subscription hält für 2 Jahre.

### Support, Community

Technischer Support siehe >Kosten, Lizenz<.

Entwickler-Support:

* Foren:
  + Xamarin Forum mit Unterforen zu einzelnen Themen wie Android, iOS, Tools, etc. Insg. Ca. 80K Threads mit ca. 400K Posts
  + Stack Overflow Community
* FAQ
* Email Support (Business und Enterprise Kunden)
* Developer Center
  + Dokumentation (mit API Spezifikationen); nicht erkennbar wann und wie häufig geupdatet wird
  + Guides
  + Code Rezepte
  + Beispiele
  + Etc.
* Consulting Partners (Schulungen)
  + 2-10 Tage; 1500$/Tag plus Reisekosten
  + Via Konferenzschaltung: 2-6h; 200$/Stunde
* 158 public Repositories auf GitHub
* Video Tutorials
* Webinare

### Hersteller, Publisher

* Xamarin Inc. (Microsoft)
* Entwicklungsstadium: “Cycle 9” (Major Release), wöchentliche Releases; aktuellste unterstützte Android Version: 7.1, iOS Version: 10.4; <https://developer.xamarin.com/releases/>
* Framework wird weiterentwickelt

### OS-Versionen

* Reaktionszeit auf OS Update:
  + Android 7.1: 4. Oktober 2016; Xamarin.Android 7.1: 19. Oktober 2016
* Abwärtskompatibilität: Android: ab Version 4.0; iOS: ab Version iOS 6; Windows Phone 7 (ohne Xamarin.Forms)
* Systeme, auf denen das Framework läuft: Windows: mind. Windows 7, für Xamarin.Form: Windows 8.1, für Forms UWP Windows 10; Apple: mind. OS Version OS X El Capitan (10.11); <https://developer.xamarin.com/guides/cross-platform/getting_started/requirements/>
* OS, für die entwickelt werden kann: iOS, Android, Windows Phone, Mac, Windows 10

### Interoperabilität/Erweiterbarkeit

* Webservice Aufrufe: Ja
* Fremdanbieter-Bibliotheken können eingebunden werden
* Integration in vorhandene IDE: Xamarin kann nur mit Visual Studio auf Windows und mit dem Xamarin Studio auf Mac genutzt werden
* Support-Tools?

### Tests

* Automatisierte (UI) Tests in der Test Cloud (kostet extra); günstigstes Abo: 99$ pro Monat für ein Gerät gleichzeitig (man kann auch für mehr Geld auf mehreren unterschiedlichen Geräten gleichzeitig testen) für eine ‚Device Hour‘ (Device Hour: Zeit, die Tests auf den Geräten laufen, beispiel: 5 min test auf 5 geräten = 25 Device Minutes) ; Xamarin bietet ein Test Framework namens Calabash für das Erstellen automatisierter Tests; <https://www.xamarin.com/test-cloud>
* Emulator

### Programmiersprache

* Nur C#

### Sicherheit

* Permissions: nativ (Android)
* Nativer Zugriff über das Android SDK auf Gerätefunktionen
* VPN Verbindungen möglich, ebenfalls nativ

## **Cordova mit Ionic**

### Kosten, Lizenz

Die Nutzung von Cordova ist kostenlos, es gibt keine Bezahl-Modelle. Ionic hat folgende Abstufungen:

Free (Community)

* Zugriff auf die Ionic Framework OSS (Open Source Software)
* Entwickeln auf allen Plattformen enthalten
* Build & live-test fertige Apps

Indie (25$ pro Monat)

* Alles wie Community PLUS:
* App Prototyping mit Ionic Creator (GUI per drag and drop zusammenbasteln)
* Einfaches share & test mit anderen Ionic (Indie) Entwicklern
* Möglichkeit native Binaries in der Cloud zu bauen
* Custom PWA (progressive web app) hosting

Business (99$ pro Monat)

* Alles wie Indie PLUS:
* Collaborative app development
* Bypass the app stores with live app updates
* Basic app analytics & insights
* Einfache 3rd-Party Integrationen
* Email Support

Enterprise (Preis auf Anfrage)

* Alles wie Business PLUS:
* Dedicated Account Manager
* Enterprise-class 3rd-Party Integrationen
* SSO (Single Sign On) Support & enterprise mobile application management
* Anpassbare App Analytics
* Enhanced App security features
* Hot fixes & enterprise releases
* Code review & consulting services
* On-prem or VPC deployment
* One business day Support SLA

Lizenz: MIT-Lizenz (Ionic), Apache Lizenz (Cordova) <https://de.wikipedia.org/wiki/MIT-Lizenz>

* Ionic Cloud: “Basis”: 20$ pro Monat pro App
  + Notifications an ausgewählte Benutzer der Anwendung
  + Updates vorbei am App Store
  + Umwandeln des Ionic Codes in native App Binaries für verschiedene Plattformen
  + Traditionelle email und passwort authentifizierung

Ionic ist komplett Open Source

### Support, Community

* Ionic Forum, stack overflow community (über 30.000 threads https://ionicframework.com/) , World Wide Slack, GitHub (über 28.300 Github Stars)
* Dokumentation
  + Ionic: Guides zu verschiedenen Themen wie Third Party Libs, Windows Setup, …
  + Ionic: Videos: Crash Kurse etc.
  + Ionic: Bücher (müssen gekauft werden), gibt’s auch im Paketpreis mit Video-Kurs etc.
  + Ionic: Kurse (eLearning) kosten auch extra
  + Cordova: Installationsanleitungen, generell „nur“ Anleitungen für den Start (Plugin erstellen, App bauen)
  + Ionic: große Übersicht der UI Elemente mit Beispielcode für die 3 Plattformen iOS, Android und Windows <https://ionicframework.com/docs/v2/components/#overview>
  + Ionic: Viel Beispiel-Code für diverse Funktionen: Standortbestimmung, Barcode-Scanner, etc.
* Trusted Partners: Unternehmen, die helfen/ unterstützen die Ionic App umzusetzen, bzw. die Implementierung ganz übernehmen <https://ionic.io/trusted-partners?_ga=1.105667203.1450521604.1486547246>
* FAQ

### Hersteller / Publisher

* Cordova: Apache
* Entwickler: Drifty /Ionic ist einziges großes Produkt von Drifty)
* Version 2 des Ionic Frameworks wurde am 25.01.2017 veröffentlicht
* Regelmäßige Bugfixes und Weiterentwicklungen, mehrere Releases im Monat <https://github.com/driftyco/ionic/releases>
* Cordova ebenfalls regelmäßige Fixes etc. <https://github.com/apache/cordova-android/blob/master/RELEASENOTES.md>

### OS-Versionen

* Reaktionszeit auf OS-Update: Android 7 wird seit dem 20. 10. 2016 von Cordova unterstützt (Ionic ist hier irrelevant, da API-Zugriffe nur über die Cordova Plugins erfolgen)
* Abwärtskompatibilität:
  + Cordova: Android: ab 4.0 (API level 14) (Versionen verlieren Unterstützung wenn sie unter die 5% Grenze auf Googles Verteilungs Dashboard sinken <https://developer.android.com/about/dashboards/index.html> ), iOS: ab Version iOS 8, Blackberry: ab 4.6, OS X: OS X 10.9, Windows Phone 8: ab WP 8.1, Windows: ab Windows 8.1, ubuntu: ab Ubuntu 14.04
* Cordova: OS, für die entwickelt werden kann: Android, iOS, Bada, Blackberry, Firefox OS, Symbian, Tizen, webOS, Ubuntu Touch, Windows Phone, Windows, OS X
* Cordova: Systeme, auf denen das Framework läuft: Windows, Linux, iOS
* Ionic: : OS, für die entwickelt werden kann: Android, iOS, Windows Phone, Windows 10
* Ionic: Systeme, auf denen das Framework läuft: Windows, Linux, iOS

### Interoperabilität/Erweiterbarkeit

* Webservice Aufrufe: Ja (z.B. SOAP Web Services)
* Fremdanbieter-Bibliotheken können eingebunden werden (mit NPM <https://ionicframework.com/docs/v2/resources/third-party-libs/> )
* Entwicklung im Text Editor nach Wahl; bauen, starten, Konfigurationen über Kommandozeile

### Tests

* Emulator
* Live Browser Test (automatisches updaten der im Browser angezeigten App sobald etwas am Code geändert wurde <http://ionicframework.com/docs/guide/testing.html> )
* Siehe Pricing
* Unit Tests können selber geschrieben werden, keine Tools, die Testgenerierung vereinfachen

### Programmiersprache

* JavaScript (Logik), nutzt CSS, HTML5 und Sass für das Design

<https://ionicframework.com/docs/v2/theming/platform-specific-styles/> Material Design Ionic

<https://cordova.apache.org/docs/en/6.x/guide/support/index.html> Sensor/API Unterstützung Cordova

## **React Native**

### Kosten, Lizenz

* Keine kostenpflichtigen Angebote
* React: Open Source Library zum Bauen von Benutzeroberflächen
* Lizenz: [3-clause BSD license](https://en.wikipedia.org/wiki/BSD_licenses#3-clause)

### Support, Community

* Stack Overflow
* React Native Facebook Gruppe (12635 Mitglieder 09.03.2017)
* Basics Tutorial (Hello world)
* Beispiel-Anwendungen auf GitHub
* Dokumentation
  + Mit Beispielcode und Guides für verschiedene Features und GUI Elemente
  + API Spezifikation
* Bücher

### Hersteller / Publisher

* Facebook, Instagram
* Entwicklungsstadium: Version 0.42 (Februar 2017)
* Es gibt Feature Requests, man kann selber ein Feature Request erstellen; von vielen Entwicklern gewünschte Requests werden dann umgesetzt; auf der Seite der Feature Requests kann man sehen, an welchen neuen Features momentan gearbeitet wird
* Zyklus: seit 2017 Monatliche Releases <https://facebook.github.io/react-native/versions.html>

### OS-Versionen

* Reaktionszeit auf OS-Update: aus den release-notes nicht erkennbar
* Abwärtskompatibilität: Android: 4.1 (API 16), iOS: 8.0 <https://github.com/facebook/react-native>
* Mobile OS: iOS, Android
* Development OS: macOS, Linux, Windows

### Interoperabilität/Erweiterbarkeit

* Webservice-Aufrufe: JA (react-native-web-service-handler: wird über npm installiert <https://www.npmjs.com/package/react-native-web-service-handler> ), oder mit der [XMLHttpRequest API](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest) (in React Native enthalten)
* 2 Typen von 3rd Party Libraries: native modules, JavaScript modules <https://blog.nativebase.io/adding-third-party-native-modules-using-react-native-cbcb9e3786ba#.50r1o7hor>
* IDE: Editor nach Wahl

### Tests

* Android Emulator
* Jest Tests (JS-only Tests) über Kommandozeile (npm); React Native Repository stellt verschiedene Tests zur Verfügung
* Unit Tests für Android
* Integration Tests (android und iOS)
* Snapshot Tests für iOS

### Programmiersprache

* JavaScript (Logik)