Fachhochschule Lübeck

Fachbereich Elektrotechnik und Informatik

BACHELORARBEIT

Evaluierung von Cross Mobile Frameworks

von

Markus Domin

Vorgelegt von: Markus Domin

Erstbetreuer: Prof. Dr. rer. nat. Nane Kratzke

Zweitbetreuer: Prof. Dr. rer. nat. Monique Janneck

Erklärung zur Bachlorarbeit

Ich versichere, dass ich die Arbeit selbständig, ohne Hilfe verfasst habe.

Bei der Abfassung der Arbeit sind nur die angegebenen Quellen benutzt worden.
Wörtliche oder dem Sinne nach entnommene Stellen sind als solche gekennzeichnet.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, insbesondere dass die Arbeit Dritten zur Einsichtnahme vorgelegt oder Kopien der Arbeit zur Weitergabe an Dritte angefertigt werden.

_____ _____

(Datum) Unterschrift

Danksagung

Hiermit möchte ich mich bei Herrn Prof. Dr. rer. nat. Nane Kratzke für seine sehr gute Betreuung und Unterstützung wären dieser Arbeit bedanken.

Inhaltsverzeichnis

1 Einleitung

1.1 Aufgabenstellung

Durch die stetig steigende Bedeutung von Smartphones und Tablets für Unternehmen und private Anwendungen stehen App-Entwickler häufig vor der Aufgabe Anwendungen für mehrere Zielplattformen (iOS, Android, Symbian, etc.) ggf. unter Nutzung plattformspezifischer Programmiersprachen entwickeln zu müssen. Dieser mehrfache Entwicklungsaufwand kostet Zeit und Geld. Ein Ansatz, um die Entwicklung für diese verschiedenen Plattformen kostengünstiger und einfacher zu gestalten, sind sogenannte „Cross Mobile“ Plattformen. Das spezifische entwickeln für mehrere Geräte entfällt, da der Code zum Großteil plattformunabhängig ist. Die meisten Vertreter dieser Entwicklungsumgebungen erstellen sogenannte „Hybrid Apps“. Diese nutzen die Funktionalität von HTML5 zur Darstellung und binden plattformspezifische Schnittstellen, über eine die Ziel-Plattform abstrahierende „Cross Mobile“ Plattform API an.
In dieser Bachelorarbeit soll es darum gehen, die gängigsten Entwicklungsumgebungen anhand einer App-Entwicklung zu vergleichen. Die App Entwicklung konzentriert sich dabei vor allem auf die Integration von Mobilephone typischer Sensorik (Ortsbestimmung, Lagebestimmung, Beschleunigung, etc.). Dazu wird ein und dieselbe App auf mehreren „Cross Mobile“ Plattformen und zusätzlich noch für die jeweilige Mobileplattform nativ entwickelt.
Die in dieser Bachelorbereit zu betrachtenden Mobile Plattformen sind
iOS (in Version 6.1)
Android (in Version 4.1)
und Windows Phone (in Version 8).
Die in dieser Bachelorarbeit zu evaluierenden „Cross Mobile“ Plattformen sind

- MoSync
- Titanium
- PhoneGap
- Corona Labs
- und Sencha.

Die zu entwickelnde App ist eine Tracking App, die Wanderer bei ihren Ausflügen unterstützen und sensorlastige Funktionalitäten anbieten soll.
Eines der Hauptfunktionen ist die Positionsbestimmung mittels Ortungsdiensten (z.B. GPS).

Des Weiteren soll die App zur Orientierungshilfe einen Kompass anzeigen.
Um nun auch die Aktivität des Benutzers festzustellen wird der Beschleunigungssensor ausgewertet.
Zusätzlich soll die App es ermöglichen mit der Kamera Fotos aufzunehmen und diese per UMTS, WLAN oder Bluetooth zu versenden.
Der Entwicklungsprozess soll dazu genutzt werden, die verwendeten „Cross Mobile“ Plattformen in mehreren Punkten zu evaluieren.

- Installationsaufwand
- Kosten der Plattformen
- Ansprechbarkeit der Sensoren
- Unterstützte Mobile Plattformen (iOS, Android, WindowsPhone)
- Test- und Simulationsmöglichkeiten (insbesondere der Sensorik)
- Feldtest der App auf den Testgeräten (alle Apps sollten beispielsweise dieselben Sensordaten wie die nativ entwickelte App generieren)

1.2 Motivation

Seitdem das erste iPhone vorgestellt wurde, steigen die Verkaufszahlen von Smartphones und später auch Tablets stetig, während die von klassischen Desktop PCs weiter abnehmen. Angetrieben durch diesen demographischen Wandel der User wollen auch viele Unternehmen eigene Apps anbieten. Was kein Problem wäre, wenn es nur ein mobiles Betriebssystem gäbe, welches unterstützt werden muss. Leider ist die Situation eine andere. Momentan ist es so, dass es viele verschiedene mobile Betriebssysteme gibt. Als Unternehmen gilt es nun zu entscheiden, ob nur ein oder mehrere mobile Betriebssysteme unterstützt werden. Bei der Unterstützung für nur ein mobiles Betriebssystem ist es nicht möglich alle Kunden zu erreichen, dafür sind die Kosten bei der Entwicklung geringer. Sollen nun mehrere mobile Betriebssysteme unterstützt werden, multiplizieren sich die Entwicklungskosten mit der Anzahl der zu unterstützenden Systeme. Eine Lösung um die Entwicklungskosten gering zu halten und doch mehrere Systeme zu unterstützen ist die Mobile Cross Entwicklung. Hierbei wird die App in einem Framework entwickelt und auf die entsprechenden Systeme übertragen.

1.3 Zielsetzung

Das Ziel dieser Bachelorarbeit ist es, zu evaluieren in wie weit die Frameworks für Cross Entwicklung den Zugriff auf die Sensorik unterstützen. Hierzu wird eine App mit folgenden Features als Evaluationsgegenstand entwickelt.

- Darstellung eines Kompass zum Navigieren
- Darstellung der aktuellen GPS-Position
- Auswertung des Beschleunigungssensors
- Aufnehmen von Fotos
- Versenden von Fotos mit Bluetooth
- Versenden von Fotos an einen Webserver

2 Mobile Devices

2.1 Geräte

Bei den Mobile Devices[1] gibt es viele Typen von Geräten, beispielsweise Pager, Handys, Smartphones, Tablets und PDAs. Neuere Geräte, die in diese Kategorie fallen, sind die Smartwatches[2] und Google Glasses[3]. Im Weiteren Verlauf werden aber nur die Smartphones und Tablets näher betrachtet, da diese momentan die größte Rolle spiele.(1)

2.2 Betriebssysteme

In diesem Kapitel werden die aktuell vier wichtigsten mobilen Betriebssysteme beschrieben. Der Kern der Beschreibung ist hier die Entwicklungsgeschichte der Betriebssysteme.

2.2.1 iOS

Das iOS ist ein von Apple entwickeltes Betriebssystem, welches das Derivat zu dem Appel Desktop-Betriebssystem OSX ist. Es kommt beim iPod, iPad und beim iPhone zum Einsatz. Vorgestellt wurde das Betriebssystem im Jahre 2007. Zu diesem Zeitpunkt trug es noch den Namen „iPhone OS“. Erst im Jahre 2010 wurde dann der Name auf iOS geändert. Der App Store wurde von Apple aber erst 2008 mit der „iPhone OS“ Version 2.0 in das Betriebssystem integriert. (2)

2.2.2 Android

Android ist ein auf Linux basierendes Betriebssystem welches von der Firma „Android Inc“ entwickelt wurde. Diese Firma wurde von Google im Jahre 2005 für 50 Millionen US Dollar gekauft. Im Jahre 2007 wurde dann von Google die „Open Handset Alliance“ gegründet. In dieser Vereinigung haben sich namenhafte Firmen zusammengeschlossen, um ein mobiles Betriebssystem namens Android auf den Markt zu bringen. Zu diesen Firmen zählen Samsung,

[1] Mobile Devices dienen dazu, den Benutzer zu unterstützen.
[2] Smartwatches sind Uhren mit Farbdisplay. Diese können sich mit Smartphones kabellos verbinden und Benachrichtigungen als auch eingegangene Emails anzeigen.
[3] Google Glasses ist eine Art Brille, die Informationen auf einem kleinen Display vor einem Auge anzeigen kann.

HTC, LG, Motorola, Intel, Nvidia, Qualcomm, T-Mobile, China Mobile, NTT DoCoMo, Sprint, Telefónica und Texas Instruments. Das erste Smartphone mit Android wurde dann im Jahr 2008 veröffentlicht. Die heute aktuelle Versionsstufe auf den Android-Geräten ist die Version 4.1 oder 4.2, die auch als „Jelly Bean“ bezeichnet werden. In dieser Version hat Google die Bildschirmwiederholfrequenz verbessert, so dass die Übergänge der Anwendungen jetzt sanfter in einander übergehen.(3)

2.2.3 Windows Phone

Windows Phone ist ein von Microsoft entwickeltes Betriebssystem für Smartphones. Im inneren basiert es auf einem Windows NT Kernel. Es ist als der Nachfolger von Windows Mobile 6.5 anzusehen, wobei es mit Anwendungen von Windows Mobile 6.5 nicht mehr kompatibel ist. Die erste Version war Windows Phone 7. Durch die schnelle Entwicklung dieser Version mangelte es an Funktionen, so dass Microsoft im Jahre 2011 ein Update mit der Versionsnummer 7.5 herausgebracht hat. In dieser Version war eine mobile Version des Internet Explorers 9 enthalten. Des Weiteren wurde die Multitaskingunterstützung für Apps von Drittherstellern implementiert. Auch die Unterstützung für Windows Live SkyDrive wurde in dieser Version implementiert. Im Januar 2013 wurde dann die Version 7.8 veröffentlicht. In dieser Version ist es dem Benutzer gestattet, die Oberfläche persönlicher zu gestalten. Nach der Version 7.8 wurde dann die Version 8 vorgestellt. In dieser Version gab es einige Verbesserungen. So unterstützt das Betriebssystem jetzt NFC, MicroSD Karten, Mehrkernprozessoren, Bildschirmauflösungen bis 720p sowie Geräteverschlüsselung für Firmen.(4)

2.2.4 Blackberry OS

BlackBerry OS ist ein Betriebssystem für Smartphones. Entwickelt wird es seit 1999 von der Firma Research in Motion (RIM) in Waterloo. Der Vorteil vom BlackBerry OS ist, dass es sich in die IT-Struktur von Firmen gut integrieren und administrieren lässt. In der Anfangszeit wurde das Betriebssystem ausschließlich über die Tastatur und einen Navigationsknopf gesteuert.(5)

Die Aktuelle Version 10 vom Blackbarry OS lässt sich nun auch über einen Touchscreen bedienen. Außerdem bietet die aktuelle Version die Möglichkeit Android-Anwendung laufen zu lassen, sowie eine Abgrenzung zwischen Arbeits- und Privatumfeld.(6)

2.2.5 Symbian

Symbian ist ein Betriebsystem für Smartphones und PDAs. Seine Ursprünge hat Symbian bei der EPOC- Plattform von Psion. Im Jahre 1998 gründeten dann Ericsson, Motorola, Nokia und Psion ein Konsortium mit dem Namen Symbian, welches die Entwicklung fortführte. Die Symbian Ltd wurde später komplett von Nokia übernommen und in die gemeinnützige Organisation Symbian Foundation überführt. Diese stellte im Februar 2010 das Symbian Betriebssystem unter die Open Source Lizenz.(7)

2.3 Betriebssystemverbreitung

Aus einer aktuellen Statistik des EITO und des IDC ist zusehen, dass der Absatz von Smartphones in den letzten Jahren kontinuierlich gestiegen ist. Es ist in der Zukunft auch nicht damit zu rechnen, dass sich dieser Trend stark verändert. (8)

Ein Grund für diesen kontinuierlichen Anstieg ist der immer weiter sinkende Preis für ein Smartphone. So kostet aktuell ein Android Smartphone unter 100 Euro. Ein Windows Phone ist noch ca. 80 Euro teurer. Eine Ausnahme beim Preis machen die iPhones, denn ihr Ausgangspreis ist deutlich teurer.

Anhand der weltweiten Verkaufszahlen aus dem vierten Quartal 2012 ist ersichtlich, dass Android das meist genutzte mobile Betriebssystem ist.

Tabelle 2-1 Weltweite Smartphone-Plattformen im vierten Quartal 2012 (9)

Smartphone-Plattform	Verkaufte Smartphones	Marktanteil
1. Android	149,8 Millionen	69,2 Prozent
2. iOS	47,8 Millionen	22,1 Prozent
3. Blackberry	7,6 Millionen	3,5 Prozent
4. Windows Phone	5,1 Millionen	2,4 Prozent
5. Symbian	3,2 Millionen	1,5 Prozent

In der folgenden Grafik ist aber zu erkennen, dass das Android Betriebssystem nicht in allen Ländern die Nummer 1 ist. Besonders in Japan und den USA ist iOS das meist genutzte mobile Betriebssystem. Die anderen mobilen Systeme spielen momentan eine eher untergeordnete Rolle.

Ihr Marktanteil ist meist nur im einstelligen Prozentbereich.

Abbildung 2-1 Weltweite Verteilung der Smartphone Verkaufszahlen (8)

	12 w/e 25 Dec 2011 %	12 w/e 23 Dec 2012 %	Change %
GB	100,0%	100,0%	0,0
iOS	34,1	32,4	-1,7
Android	43,9	54,4	10,5
RIM	16,0	6,4	-9,6
Symbian	3,0	0,7	-2,3
Windows	2,2	5,9	3,7
Bada	0,4	0,1	-0,3
Other	0,4	0,2	-0,2
Germany	100,0%	100,0%	0,0
iOS	23,8	24,7	0,9
Android	60,7	66,6	5,9
RIM	0,9	1,6	0,7
Symbian	8,2	3,2	-5,0
Windows	3,0	2,6	-0,4
Bada	2,7	1,0	-1,7
Other	0,7	0,3	-0,4
France	100,0%	100,0%	0,0
iOS	23,1	25,6	2,5
Android	46,5	58,7	12,2
RIM	12,9	4,7	-8,2
Symbian	3,5	1,0	-2,5
Windows	3,7	4,1	0,4
Bada	9,7	5,3	-4,4
Other	0,7	0,5	-0,2
Italy	100,0%	100,0%	0,0
iOS	20,0	24,9	4,9
Android	50,7	51,8	1,1
RIM	4,8	2,4	-2,4
Symbian	20,1	5,3	-14,8
Windows	2,8	13,9	11,1
Bada	1,2	1,4	0,2
Other	0,5	0,2	-0,3
Spain	100,0%	100,0%	0,0
iOS	8,2	6,4	-1,8
Android	62,2	86,4	24,2
RIM	16,6	2,5	-14,1
Symbian	12,6	2,3	-10,3
Windows	0,4	1,8	1,4
Bada	0,0	0,0	0,0
Other	0,0	0,6	0,6

	12 w/e 25 Dec 2011 %	12 w/e 23 Dec 2012 %	Change %
US	100,0%	100,0%	0,0
iOS	44,9	51,2	6,3
Android	44,8	44,2	-0,6
RIM	6,1	1,1	-5,0
Symbian	0,2	0,1	-0,1
Windows	2,2	2,6	0,4
Bada	0,0	0,0	0,0
Other	1,7	0,9	-0,8
Australia	100,0%	100,0%	0,0
iOS	41,8	38,4	-3,4
Android	48,1	55,8	7,7
RIM	1,1	0,5	-0,6
Symbian	5,9	1,5	-4,4
Windows	1,9	2,8	0,9
Bada	0,0	0,3	0,3
Other	1,2	0,8	-0,4
Urban China	100,0%	100,0%	0,0
iOS	n/a	21,9	
Android	n/a	72,5	
RIM	n/a	0,0	
Symbian	n/a	3,9	
Windows	n/a	0,9	
Bada	n/a	0,0	
Other	n/a	0,9	
Japan	100,0%	100,0%	0,0
iOS	n/a	66,2	
Android	n/a	31,9	
RIM	n/a	0,3	
Symbian	n/a	0,1	
Windows	n/a	0,8	
Bada	n/a	0,0	
Other	n/a	0,8	
EU5	100,0%	100,0%	0,0
iOS	25,4	25,6	0,2
Android	50,8	61,1	10,2
RIM	10,3	4,0	-6,4
Symbian	7,4	2,2	-5,2
Windows	2,6	5,4	2,8
Bada	3,0	1,5	-1,4
Other	0,5	0,3	-0,2

Laut einer Prognose des IDC wird sich die Aufteilung der Marktanteile bis 2016 noch zugunsten des Windows Phone Betriebssystems ändern. Beim Windows Phone Betriebssystem wird mit einem kontinuierlichen Wachstum gerechnet, so dass es im Jahre 2016 auf dem Niveau vom iOS ist. In der nachfolgenden Grafik ist diese Entwicklung dargestellt.

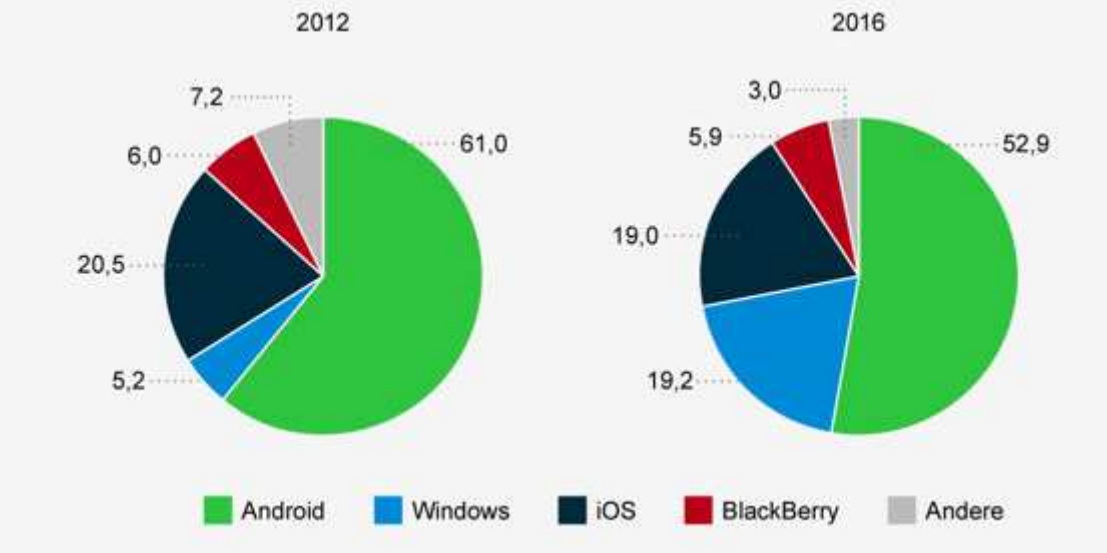


Abbildung 2-2 Prognose zum Marktanteil der Smartphone- Betriebssysteme (in %) (9)

2.4 Vertriebsmöglichkeiten Mobiler Apps

App Store ist eine Bezeichnung für digitale Plattformen die zum Vertrieb von Software für mobile Plattformen gedacht ist. Auf diese Plattformen können dann die App Entwickler ihre App hochladen und festlegen, wie viel sie kosten sollen. Für den Kunden ist es dann ein leichtes sich eine App aus dem Anwendungskatalog auszusuchen und zu installieren. Einige Anbieter haben das Angebot ihrer App- Stores um Musik, Bücher und Filme erweitert.

Die bekanntesten App Stores sind aktuell:

- App Store (iOS)
- BlackBerry World
- Google Play
- Windows Phone Store

In der nachfolgenden Tabelle ist zusehen, welche alternativen App Stores es für die jeweiligen Betriebssysteme gibt. Auffällig ist, das es fast nur bei Android Alternativen gibt.

Tabelle 2-2 App Stores von Drittherstellern

Store	iOS	Android	Windows Phone
Nokia Store		JA	
Samsung Apps		JA	
Amazon Appstore		JA	
Cydia	Nur bei Jailbreak [4]IOS Geräten		
Opera Store		JA	
GetJar		JA	
Handster		JA	

2.5 Gewinnerzielung mit Apps

Um mit Apps Gewinn zu erzielen gibt es verschiedene Möglichkeiten. Zum einen kann eine App zu einem bestimmten Preis im jeweiligen App Store angeboten werden. Eine andere Möglichkeit ist es, die Apps kostenlos anzubieten, um dann in der App selber Werbung zu schalten. Diese wird dann als In-App Werbung bezeichnet. Hier bekommt der Anbieter, der ausgehend von der Popularität der App, für jeden Klick auf die Werbung einen bestimmten Betrag gutgeschrieben. Eine weitere Möglichkeit sind die sogenannten In-App Käufe. Hierbei wird in der App selber eine eigene Währung angeboten. Mit dieser Währung lassen sich dann Zusatzfunktion freischalten. Um diese Währung zu bekommen wird meist ein Wechselportal in der App eingebaut, bei dem der Nutzer echtes Geld in In-App Währung tauschen kann. Von dieser Möglichkeit machen hauptsächlich Spiele App Gebrauch.(10)

2.6 Rolle in der Industrie

Das Android Betriebssystem ist so beliebt, dass es sogar in Fabriken und Industrieanlagen eingesetzt wird. Dass es durch den Einsatz einiger Apps aus dem Googles Play Store möglich ist mit der integrierten Kamera Barcodes einzuscannen ist keine Neuheit. Neu ist nur, dass es speziell

[4] Jailbreak bezeichnet das entfernen der Nutzerbeschränkungen die vom Hersteller vergeben wurden.

für den Einsatz in Fabriken jetzt robuste Android Handhelds gibt, die wasserfest sind und Stürze aushalten. Bei diesen Handhelds wird auch nicht nur die Kamera zum einscannen von Barcodes verwendet, sondern auch spezielle 2D Laserscanner. Eine andere Lösung sind Bluetoothadapter mit 2D Laserscannern. Diese senden dann die aufgenommenen Daten an das gekoppelte Smartphone. Für Android ist dieses Arbeitsumfeld neu, ganz anders sieht das bei Microsoft aus. Denn Microsoft ist mit seinem mobilem Betriebssystem Windows Mobile schon lange auf den Handheld Barcodescannern vertreten.

3 Grundlagen der Entwicklung mobiler Anwendungen

3.1 Native Apps

Native Apps werden speziell für das jeweilige Betriebssystem entwickelt. Zur Entwicklung wird die die vom entsprechenden Hersteller verfügbare API verwendet. Durch dieses Vorgehen bei der Entwicklung hat der App Entwickler die Möglichkeit, über die vom Hersteller freigegebenen Schnittstellen die Hardware anzusprechen. Über diese Schnittstellen kann der Entwickler nun die Daten der Sensoren abrufen und daraus moderne Apps entwickeln. Ein weiterer Vorteil der nativen App Entwicklung ist, dass die Apps mit der maximalen Performance laufen. Der Grund hierfür ist die Compelierung der App speziell auf die Hardware der Zielplattform. Die Performance spielt bei einfachen Anwendungen zwar nicht die größte Rolle, aber bei Spielen sieht das schon anders aus. Hier erwartet der Benutzer ein flüssiges Verhalten der App und will keine Wartezeiten in Kauf nehmen. Zur besseren Übersicht sind die Vor- und Nachteile noch einmal in der nachfolgenden Tabelle aufgeführt.(11)

Tabelle 3-1 Vor- und Nachteile von Native Apps (11)

Vorteile	**Nachteile**
Hohe Performance	plattformgebunden
Vertrieb über den App- Store	Hohe Entwicklungskosten
In- App Verkäufe	
Zugriff auf Hardware- und Softwareschnittstellen des Smartphones	

3.2 Apple iOS

3.2.1 Voraussetzung für die Entwicklung

Als aller erstes wird bei der iOS Entwicklung ein Mac mit einer Intel CPU benötig. Auf dem Mac sollte OSX in der Version Snow Leopard oder eine neuere Version installiert sein. Zusätzlich wird

noch eine Appel ID benötigt. Durch das Anmelden bei Apple ist die Appel ID kostenlos zu bekommen. Um mit der iOS Entwicklung zu beginne ist es auch erforderlich XCODE zu installieren, was wiederum bei Appel kostenlos heruntergeladen werden kann.(12)

Für die Entwicklung von iOS Apps mit Zugriff auf die Sensoren wird auch noch ein iPhone benötigt, da der Emulator nicht alle Sensoren simulieren kann.

Als nächsten Punkt verlangt Appel noch, dass sich jeder Entwickler oder jedes Unternehmen was Apps entwickeln will, im iOS Developer Programm anmeldet. Hier gibt es die folgenden Möglichkeiten:

Tabelle 3-2 Lizenz-Überblick iOS Developer Programm(13)

Typ	**Preis**	**Beschreibung**
iOS Developer Program Individual	$99 /Jahr	Für Einzelentwickler die Apps im App Store veröffentlichen wollen.
iOS Developer Program Company	$99 /Jahr	Für Unternehmen die in Teams Apps entwickeln und diese im App Store veröffentlichen wollen.
iOS Developer Enterprise Program	$299 /Jahr	Für Unternehmen die unternehmensinterne Apps entwickeln und diese im App Store veröffentlichen wollen.
iOS Developer University Program	Kostenlos	Für Hochschulen die iOS Entwicklung vermitteln wollen

Zusammengefasst braucht ein Entwickler für iOS Apps einen aktuellen Mac, ein aktuelles iPhone sowie eine Entwicklerlizenz.

3.2.2 Designvorgaben

Appel überprüft jede App, die über den App Store vertrieben werden soll. Hierbei hat Appel sehr strenge Regel was das Design und die Steuerung der App betrifft. Um sich besser mit den

Vorgaben vertraut zu machen, hat Apple mehrere Regelwerke für Entwickler bereit gestellt. Das sind zum einen die "iOS Human Interface Guidlines" und zum anderen die "App Store Review Guidelines". Verstoßen Apps gegen diese Vorgaben, werden sie von Apple abgelehnt und müssen nachgebessert werden, um in den App Store zu kommen.(14)

Typisch für iOS Apps ist die Kopfzeile, die zum einen den Titel und die Button zur Navigation enthält. Beides sollte in den Standardfarben des iOS Betriebssystem sein, damit sich der Nutzer schnell zurecht findet. Auch bei der Eingabe von Datum und Zeit hat Apple klare Vorgaben gemacht. Hier gibt es eigens entwickelte Picker[5] ,an denen der Nutzer die Einstellungen vornimmt. Auch bei den Farben der Button hat Apple klare Vorgaben gemacht. Hier ist standartmäßig eine graue Farbe zu verwenden. Bei besonderen Eingaben ist aber auch Grün und Rot erlaubt.(15)

3.2.3 In-App Werbung

Apple bietet momentan eine eigene Werbeplattform namens iAd an. Mit dieser ist es möglich Werbung zu eigenen Produkten zu schalten. Eine weitere Möglichkeit ist das Anzeigen von Werbebannern in einer iOS App. Hierzu wird in die jeweiligen App ein Werbebanner eingeblendet auf dem dann zufällig Werbeeinblendungen stattfinden. Durch die Einblendung der Werbebanner ist es auch möglich Gewinn zu erzielen (siehe Kapitel 2.5). (16)

3.2.4 Vertriebsmöglichkeiten

Wie schon im Kapitel 2.4 erwähnt, gibt es bei iOS Apps nur die offizielle Möglichkeit sie in Appels App Store zu veröffentlichen. Um eine App in dem App Store zu veröffentlichen muss sie erst bei Apple eingereicht werden. Apple überprüft dann, ob die in Kapitel 3.2.2 erwähnten Designrichtlinien eingehalten wurden. Erst wenn das in Ordnung ist, wird die App im App Store aufgenommen. Von jeder kostenpflichtigen App bekommt Appel Gebühren von 30 Prozent. Dafür übernimmt Apple dann auch die Zahlungsabwicklung mit den Kunden. (17)

[5] Picker sind drehende Walzen zum Einstellen.

3.3 Google Android

3.3.1 Voraussetzung für die Entwicklung

Bei der Entwicklung von Android Apps kann sich der Entwickler zwischen einer der folgenden Betriebssystemplattformen entscheiden. Um alle Sensor Funktionen zu Testen ist es erforderlich ein Android Smartphone zu besitzen, da der Emulator die Sensoren nicht simuliert.

Anforderungen(18):

Windows

- XP (32-bit)
- Vista (32- / 64-bit)
- Windows 7 (32- / 64-bit)

Apple

- OS X 10.5.8 oder neuer (x86)

Linux

- GNU C Library (glibc) 2.7 oder neuer ist erforderlich
- Ubuntu version 8.04 oder neuer
- 64-bit Distributionen müssen 32-bit Programme ausführen können

Um bei Google die Android App im Play Store zu veröffentlichen ist eine Entwicklerlizenz bei Google nötig, welche einmalig 25$ kostet.(19)

3.3.2 Designvorgaben

Bei Android gab es in der Anfangszeit kaum vorgaben wie ein Entwickler eine App gestallten sollte. Im Jahre 2012 veröffentlichte Google dann zum ersten Mal eine detaillierte Designvorgabe. Anders als bei Apple und Windows ist das aber nur eine Empfehlung und keine klare Forderung. Ein Punkt dieser Vorgabe ist das die App eine ansprechende Oberfläche haben sollte und auch mit den Effekten sollte sparsam umgegangen werden, um den Nutzer nicht zu verwirren. Ein weiterer Punkt ist, das nur was wichtig ist auch angezeigt wird. Dies ist für die Allgemeineübersicht sehr von Vorteil. Ein sehr wichtiger Punkt ist auch, das Elemente die gleich Aussehen auch immer gleich

funktionieren sollten.(15)

3.3.3 In App Werbung

Bei Android bietet Google die Möglichkeit der In App Werbung. Hierzu gibt es für Android ein eigenes Software Development Kit (SDK)[6] namens Ad. Mit diesem SDK ist es möglich Banner Werbung in Android Apps zu platzieren. Durch diese Einblendung der Werbebanner ist es auch möglich einen Gewinn zu erzielen (siehe Kapitel 2.5). (20)

3.3.4 Vertriebsmöglichkeiten

Wie im Kapitel 2.4 zusehen gibt es bei Android nicht nur die Möglichkeit die Apps bei Googles eigenem Play Store zu veröffentlichen sondern auch bei Drittanbietern. Bei Googles Play Store werden die Apps ohne Überprüfung der Designvorgaben veröffentlicht. Es findet hier nur eine Überprüfung auf jugendgefährdende Inhalte, Urheberrechtsverletzungen und Malware [7]statt. Genau wie Apple verlangt auch Google eine Gebühr in Höhe von 30 Prozent vom Verkauf jeder App aus dem Play Store.(21)

3.4 Windows Phone

3.4.1 Voraussetzung für die Entwicklung

Bei der Entwicklung für Windows Phone 8 ist die erste Voraussetzung ein Rechner mit Windows 8 sowie Visual Studio 2012. Als nächste Softwarekomponente wird das Windows Phone SDK benötigt. Um als Entwickler auch alle Funktionen von Visual Studio 2012 und des Windows Phone SDK zu nutzen ist es wichtig eine CPU mit Hardware Virtualisierung zu besitzen da sonst der Windows Phone Emulator nicht lauffähig ist.(22)

Zusätzlich ist noch eine Windows Phone Entwicklerlizenz notwendig. Diese kostet 99$, außer für Studenten, da gibt es sie kostenlos.(23)

Weitere Kosten entstehen durch ein Windows Phone welches zur Entwicklung von Apps von Vorteil ist ,da der Emulator auch hier nicht alle Sensoren unterstützt.

[6] Software Development Kit (SDK) ist eine Software Werkzeugsammlung zum Erstellen von Software.
[7] Malware ist ein Sammelbegriff für schadhafte Software

3.4.2 Designvorgaben

Microsoft setzt mit seinem Windows Phone auf ein schlichtes einheitliches Design. Typische Apps haben hier weiße Schrift auf schwarzem Grund als Ausgangsdesign. Diese Farbgebung ist bei der Entwicklung voreingestellt. Einen wesentlichen Vorteil hat das Windows Phone bei der Auflösung, denn hier haben alle Geräte die vorgegebene Auflösung von 800x 480 Pixeln, das ist beim iOS- und Android Geräten nicht so. Für die Entwickler ist diese feste Größe eine gute Basis um Bilder entsprechend zu skalieren. Eine weitere Vorgabe für Apps sind die im beiden Titel am oberen Rand jeder App. Der obere der beiden ist der Titel der aktuellen Seite und der untere beinhaltet den Name der Kategorie. Ein Bsp.: für den Titel wäre "Einstellungen" und für die Kategorie wäre es "Design". Wie bei Apple gibt es auch beim Windows Phone fertige Picker um das Datum und die Uhrzeit auszuwählen, so dass ein Entwickler diese benutzen sollte.(15)

Eine Besonderheit bei der Entwicklung mit Windows Phone stellt das Panoramalayout dar. Hier wird die App als eine Art Laufleiste implementiert. Bei der Bedienung verschiebt der Nutzer die aktuelle Ansicht horizontal um zur nächsten Ansicht zu gelangen. Im nachfolgenden Bild ist das Beispiel einer Windows Phone Panorama App zu sehen. Klar zu erkennen ist das durchgehende Hintergrundbild. Die Panorama Apps sind ein klares Alleinstellungsmerkmal vom Windows Phone, da es bei Android und iOS zur Zeit keine vergleichbare Technik gibt.(15)

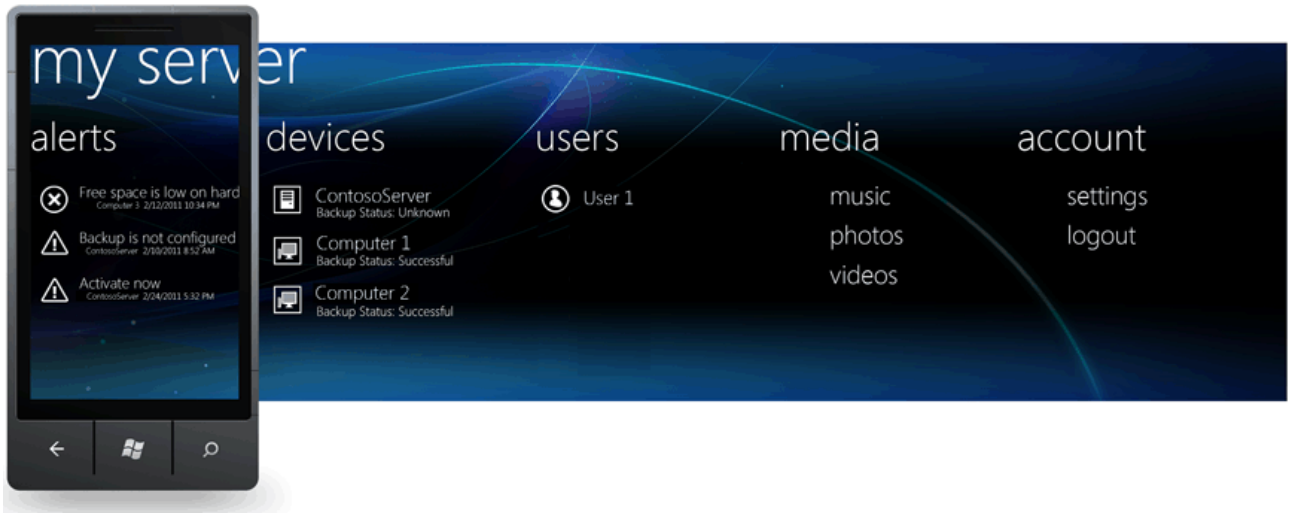


Abbildung 3-1 Windows Phone Panorama App Beispiel(24)

3.4.3 In App Werbung

Bei Mircosoft gibt es für die In App Werbung beim Windows Phone auch ein eigenes SDK. Dieses

SDK ist schon Bestandteil der Windows Phone Developer Tools. Mit diesem ist es nun Möglich Werbebanner in einer App anzuzeigen. Die Werbung für die Werbebanner wird dann von Microsoft zufällig ausgewählt. Durch die Einblendung der Werbebanner ist es auch möglich Gewinn zu erzielen (siehe Kapitel 2.5).(25)

3.4.4 Vertriebsmöglichkeiten

Wie im Kapitel 2.4 zusehen gibt es beim Windows Phone nur die Möglichkeit die Apps über den Windows Phone Store zu verkaufen. Auch Microsoft verlangt wie Google und Apple 30 Prozent Gebühren der verkauften Apps.(26)

4 Cross- Plattformen

4.1 Motivation von Cross Frameworks

Bei der Entwicklung einer App für verschiedenen Plattformen entstehen für Unternehmen vermeidbare Kosten. Der Grund hierfür ist, dass die App für jede Plattform einzeln entwickelt und implementiert werden muss. Ein weiterer Punkt ist die Wartung, denn hier muss jede Plattform eigenständig gewartet werden. Tritt ein Fehler auf wird der nur in der jeweiligen App behoben, da sich die Apps im Aufbau und in der Programmiersprache unterscheiden. Hier setzen jetzt Cross Frameworks an. Diese bieten eine einheitliche Codebasis, so dass ein Fehler nur einmal behoben werden muss. Das erhöht sehr die Wartbarkeit der App.(27)

4.2 Technologieüberblick

Nachfolgend werden die einzelnen Ansätze für die Cross- Plattform Entwicklung vorgestellt.

4.2.1 Web Apps

Als Web Apps werden aktuell nicht nur Anwendungen bezeichnet die auf mobilen Devices laufen. Es ist vielmehr so, dass alle Webanwendungen oder Webapplikationen als Web App bezeichnet werden können. Im nachfolgenden wird das Augenmerk auf den Web Apps liegen, die auf Mobile Devices zum Einsatz kommen. Eine Web App sieht und fühlt sich für den Benutzer im Idealfall genauso an wie eine native App. Sie bietet nach dem Start eine Benutzeroberfläche an und fügt sich damit optisch in das System ein. Das bedeutet, dass der Benutzer gar nicht mitbekommt das er eine Web App ausführt. Um diesen Effekt zu erzielen, werden die Web Apps meistens mit JavaScript und HTML5 entwickelt.

Da Web App Applikationen für Browser sind, ist es im Prinzip auch möglich, die Mobile Web Apps auf Desktop Browsern auszuführen. Es ist aber auch möglich dieses zu blockieren.

Die ersten Web Apps waren noch relativ einfache Apps denen grundlegende Funktionen fehlten. Eine davon ist das Offlinespeichern von Daten. Erst mit dem Einsatz von HTML5 bekamen Web Apps diese Funktionalitäten. So ist es heute durch eine Schnittelle in HTML5 möglich die GPS Position des Smartphone auszulesen.(28)

4.2.2 Webbasierte Hybrid Apps

Hybrid Apps reihen sich zwischen den Web Apps, die weit von der Hardware entfernt sind, und zwischen den Native Apps ein. Sie sind besonders für Anwendungen geeignet, die auf vielen verschiedenen Mobile Devices laufen sollen. Hier bieten sie das gewohnte Userinterface sowie ein Optimum an Geschwindigkeit, als auch den Zugriff auf die Hardware- und Softwareschnittstellen. Das IT-Research- und Beratungsunternehmen Gartner sagt sogar voraus, dass bis zum Jahre 2016 mehr als die Hälfte aller Apps Hybride Apps sind. Ein Grund für diese Entwicklung laut Gartner ist, dass die Offlinefunktionen von Web Apps nicht immer funktionieren. (29)

Die Entwicklung einer Hybriden App wird nicht, wie bei den native Apps, in einer plattformspezifischen Sprache entwickelt. Es werden hier vielmehr Web- Apps mit einem nativen Container verbunden. Durch diese hybride Verbindung von Web App und nativen Container wird sichergestellt, dass die Web App Zugriff auf die Sensorik erhält. Es können aber auch Dienste wie Telefonie oder die Kontakte über den Container angesprochen werden. Es ist jetzt aber falsch zu denken, dass diese nativen Container alles was eine Native App an Funktionen benutzen kann auch ansprechen können. Diese Container können meist nur die wichtigsten Schnittstellen zum Host System bedienen.(29)

4.2.3 Cross compiled Apps

Diese Art von Apps werden meistens in einer Script Sprache entwickelt. Nachdem die App fertig entwickelt ist wird sie für die einzelnen Plattformen compiliert. Beim Compilieren wird die App technisch an die Zielplattform angepasst. So wird sichergestellt, dass sich die App am Ende optisch und ergonomisch in das Zielsystem einfügt. (30)

4.3 Vergleich der Technologien

Wie in der nachfolgenden Tabelle zusehen ist, bieten einige App Typen mehr Funktionen als andere. Der Grund hierfür liegt in der unterschiedlichen Entwicklung. Die größten Unterschiede treten hier bei der Geschwindigkeit, Hardwarezugriff, Kosten und dem Implementierungsaufwand auf.

Tabelle 4-1 Vergleich von App Typen

App Type	Vorteile	Nachteile	Beispiel Framework
Native Apps	• Zugriff auf die Sensorik • Vertrieb über den App-Store • Höchste mögliche Performance	• Nicht plattform-unabhängig	• Android SDK • XCODE
Web Apps	• plattformunabhängig • Unabhängig vom App Store (erleichtert eigenen Update-Zyklus)	• dauerhafte Online Verbindung erforderlich • nicht so performant wie Native Apps • keine In- App Verkäufe • wenig Zugriff auf Hardware- und Softwareschnitt-stellen des Host Systems	• Sencha • jQuery Mobile
Webbasierte Hybride Apps	• plattformunabhängig • Zugriff auf die Sensorik • Vertrieb über den App-Store	• nicht so performancestark wie Native Apps	• PhoneGap • Mosync
Cross Compiled	• plattformunabhängig • hohe Performance	• höhere Kosten als bei Web Apps	• Corona • Titanium

Apps	• Zugriff auf die Sensorik • Vertrieb über den App-Store	• Xamarin

Bei den Apps sind diese Unterschiede aber miteinander verbunden. Die nachfolgende Grafik zeigt den Zusammenhang zwischen Kosten, Implementierungsaufwand, Hardwarezugriff und Geschwindigkeit dar. Bei dem Hardwarezugriff der Apps bedeutet niedrig ,dass es nur möglich ist einen Sensor oder gar keinen Sensor anzusprechen. Hoch hingegen bedeutet vollen Zugriff auf die Hardwareschnittstellen. Bei der Geschwindigkeit steht hoch für die maximal mögliche Geschwindigkeit auf dem Gerät, wobei es bei niedrig schon zu Verzögerungen kommen kann. Beim Implementierungsaufwand bedeutet niedrig, dass die App nur einmal entwickelt werden muss. Im Gegensatz dazu muss die App dann bei einem hohen Aufwand so häufig entwickelt werden, wie es zu unterstützende Betriebssystem gibt.

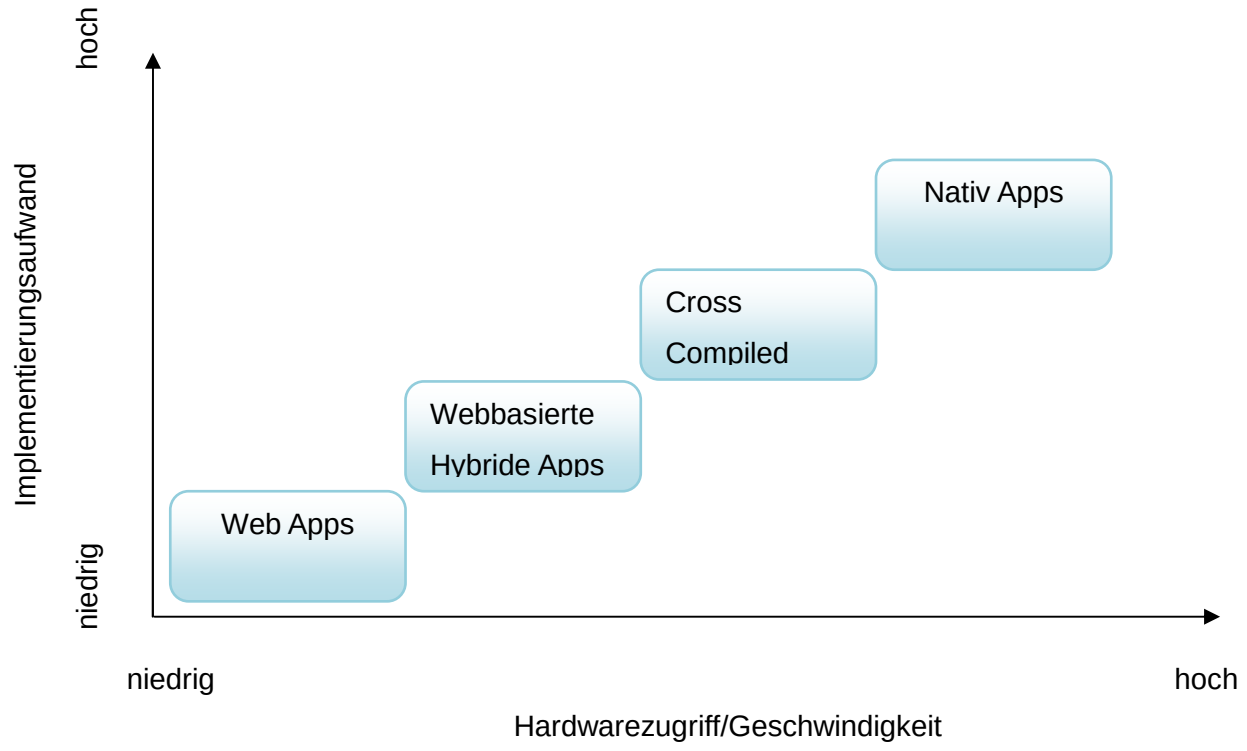


Abbildung 4-1 Vergleich der App Typen zwischen Kosten und Hardwarezugriff (64)

4.4 Verbreitung

Anhand der folgenden Grafik ist zu erkennen, dass die beiden meist genutzten Cross Plattformen die sind, welche die Apps mit Hilfe von Webtechnologien wie HTML5, CSS und JavaScript erstellen. Das ist damit zu begründen, dass viele Entwickler schon mit der Webentwicklung vertraut sind. Wobei auch das Xamarin Mono mit 26 Prozent sehr gerne genutzt wird. Hier wird aber nicht mit Webtechnologien entwickelt sondern in der Hochsprache C#.

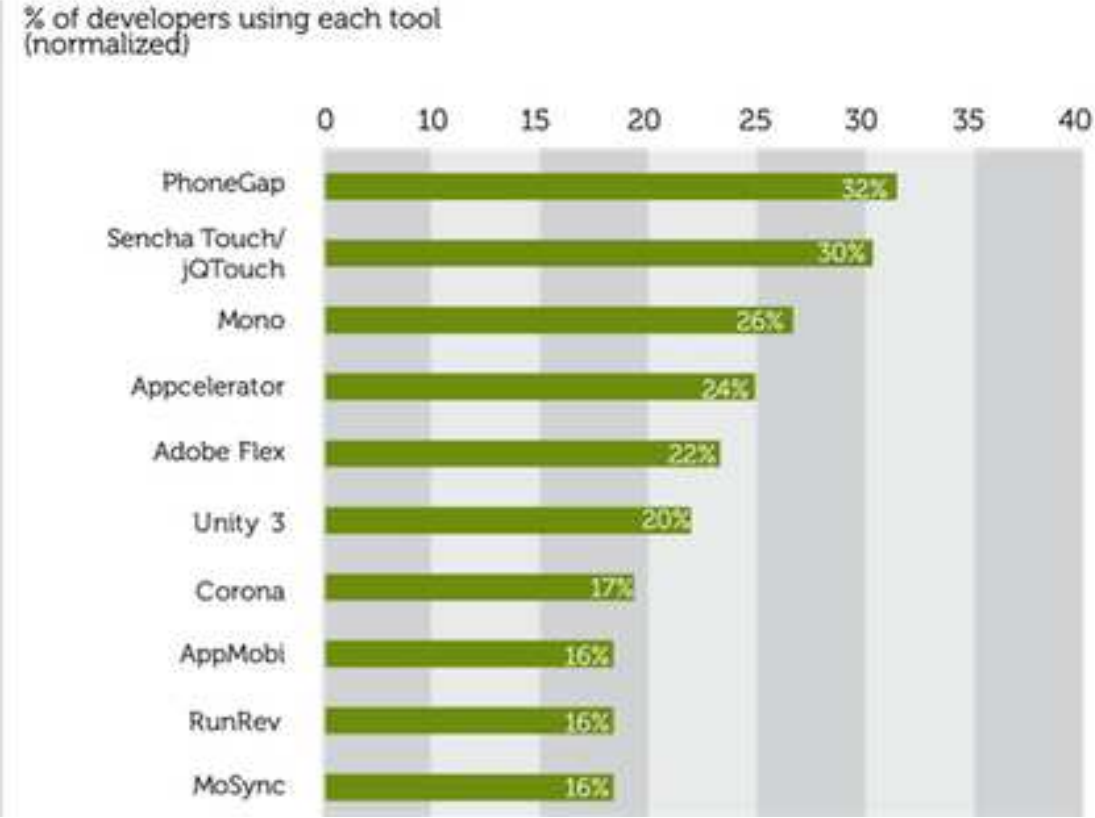


Abbildung 4-2 Cross Framework Nutzerübersicht 2012 (31)

4.5 Lizenzkosten

Die nachfolgende Tabelle zeigt einen Überblick der Kosten der meist verwendeten Frameworks. Bei den Kosten ist zu erkennen, dass die meisten Anbieter, der hier aufgeführten Frameworks, sie komplett kostenlos anbieten oder zumindest eine kostenlose Start Edition.

Tabelle 4-2 Cross Frameworks Kostenübersicht (Stand 20.06.2013)

Framework	**Preis**
PhoneGap	Kostenlos(32)
Sencha Touch	Kostenlos Support ab 299,00 Dollar(33)
Xamarin Mono	Start Edition Kostenlos IDIE 299,00 Dollar BUSINESS 999,00 Dollar ENTERPRISE 1899,00 Dollar(34)
Appcelerator	Titanium Kostenlos **Appcelerator Platform** (PUBLIC CLOUD) ab 999$ /Monat **Appcelerator Platform** (VIRTUAL PRIVATE CLOUD) ab 2667$ /Monat (35)
Adobe Flex	Open Source(36)
Unity 3D	Unity3D Kostenlos Unity 3D Pro 1500,00 Dollar Add-On ANDROID PRO 1500,00 Dollar Add-On iOS PRO 1500,00 Dollar(37)
Corona	Start Edition Kostenlos Pro Edition 599,00 Dollar / Jahr Enterprise Edition ab 999,00 Dollar / Jahr(38)
MoSync	Standard Kostenlos Basic Pro 199 Euro Gold Pro 2999 Euro(39)
Rhodes	Kostenlos(40)

4.6 Features laut Hersteller

Die nachfolgende Tabelle zeigt die einzelnen Frameworks und deren Unterstützung für bestimmte Betriebssysteme. Des Weiteren sind die von den jeweiligen Herstellern angegeben Hauptfunktion zusehen.

Tabelle 4-3 Cross Framework Featureübersicht

Framework	**Betriebssysteme**	**Features**
PhoneGap	iPhone, Android,Blackberry, WebOS, Windows Phone 7+8, Symbian, Bada(41)	Zugriff auf Beschleunigungssensor, Kamera, Kompass, Kontakte, Dateien, GPS Sensor, Medien, Benachrichtigung Alarm/Sound/ Vibration und Speicher(41)
Sencha Touch	Geräteabhängige Unterstützung bei iOS, Blackberry, Windows und Android(42)	Verschiedene Elemente zur grafischen Darstellung, Multitouch Unterstützung (42)
Xamarin Mono	Android, iOS, Windows, Mac(43)	Eigene IDE, hohe Geschwindigkeit der App(43)
Appcelerator Titanium	iOS, Android, Windows, Blackberry, Webbrowser(44)	Eigene IDE sowie ein eigener Marktplatz um Erweiterungen zur IDE zu kaufen(44)
Adobe Flex	Android, BlackBerry, iOS und Standart PC mit Webbrowser(36)	Zugriff auf Kamera, GPS Sensor, Beschleunigungssensor und auf die lokale Datenbank(36)
Unity 3D	Android, iOS, Windows, Wii U, OSX, Linux, PS3, XBOX, Unity Webmediaplayer(45)	Eigene IDE, eigener Marktplatz für Erweiterungen, Vorbereitete Profile für einige Spiel Szenarien (45)
Corona	iOS, Android, Kindle Fire, NOOK(46)	Eigener Simulator, Facebookzugriff, in-App Verkäufe möglich (46)
MoSync	Moblin, Blackberry, Windows Mobile, Android, iOS, Java, MeeGo, Symbian OS(47)	Eigene IDE, Programmiersprachen C/C++ oder HTML5(47)
Rhodes	iOS, Android, RIM, Windows Mobile and Windows Phone 7(40)	Zugriff auf GPS Sensor, Kontakte, Kamera, NFC[8](40)

[8] NFC (Near Field Communication) ist ein Standard zum drahtlosen Austausch von Daten.

4.7 Auswahl der zu evaluierenden Frameworks

In der nachfolgenden Tabelle werden die zu evaluierenden Frameworks auf Grundlage der Verbreitung sowie der Unterstützung für verschiedene mobile Betriebssysteme und der vorhandenen Features ausgewählt. Hierzu werden die Daten aus Kapitel 4.4, Kapitel 4.5 und dem Kapitel 4.6 zu Rate gezogen.

Tabelle 4-4 Auswahl der zu evaluierenden Frameworks

Framework (Sortiert nach Verbreitung(31))	**iPhone**	**Windows Phone 8**	**Andorid**	**Verfügbar**	**Ausgewählt**
PhoneGap	Ja	Ja	Ja	Ja	Ja
Sencha Touch	Ja		Ja	Ja	Ja
Xamarin Mono	Ja	Ja	Ja	Nein (Start Edition ist zu eingeschränkt)	nein
Appcelerator Titanium	Ja		Ja	Ja	Ja
Adobe Flex	Ja		Ja	Nein	nein
Unity 3D	Ja		Ja	Ja (hauptsächlich für die Spieleentwicklung gedacht)	nein
Corona	Ja		Ja	Ja	Ja
MoSync	Ja		Ja	Ja	Ja
Rhodes	Ja		Ja	nein	nein

5 Zu evaluierende Cross Plattform Frameworks

5.1 MoSync

MoSync ist ein SDK zur Erstellung von mobilen Anwendungen. Entwickelt wird MoSync von der schwedischen Softwarefirma MoSync AB. Diese veröffentlichte die erste Version des MoSync SDK im Frühjahr 2005. Es ist in die Entwicklungsumgebung Eclipse integriert worden. In dieser Entwicklungsumgebung werden nun die Apps mit C, C++ oder HTML5 und JavaScript implementiert. MoSync bietet nativen Zugriff auf das darunter liegende Betriebssystem. Dieser Zugriff ist seit Android und iOS seit Version 2.5 möglich, bei Windows Phone Geräten seit Version 3.0. Für den Nativen Zugriff auf das System bietet MoSync für JavaScript eine Technologie namens Wormhole. Durch diese Technologie werden Funktionsaufrufe aus JavaScript für Systemressourcen an die darunterliegende C API weitergeleitet. So ist es für Entwickler sehr komfortabel eine App in HTML5 und JavaScript zu erstellen und trotzdem auf die Systemressourcen zuzugreifen. (48)

Ist die App fertig programmiert wird sie in Nativen Code für die jeweilige Plattform umgesetzt.

(48)

5.2 Sencha Touch

Sencha Touch ist ein spezielles Framework zur Erstellung von Mobile Webpages. Entwickelt wird es von der Firma Sencha. Die erste stabile Version wurde im November 2010 veröffentlicht. In dieser Version unterstützt Sencha Touch Android und iOS Geräte. Mit Version 1.1.0 kam dann auch die Unterstützung von BleckBerry OS 6 dazu. Mit der Veröffentlichung von Version 2.2.1 wurden dann auch die folgenden Plattformen unterstützt: Google Chrome for Android, BlackBerry 10, Bada Mobile Browser, Kindle Fire Browser, Windows Phone 8 und Windows 8 IE10 und Mobile Safari. (49)

Apps können bei Sencha Touch einfach in HTML5,CSS3 und JavaScript geschrieben werden. Zur einfacheren Oberflächengestaltung bietet Sencha Touch eine Fülle an graphischen Komponenten. Einige davon sind z.B. Textfelder, Datumsauswahlboxen sowie Scrollbars. Außerdem wird die Steuerung mit Touch- Gesten unterstützt, so das es dem Nutzer später auch Möglich ist, durch

wischen die App zu bedienen.

Eine fertig erstellte App kann in Nativen Code für die jeweilige Plattform übersetzt werden. (49)

5.3 Corona

Das Corona SDK wurde zunächst nur für die Spieleentwicklung konzipiert. Es hat sich dann aber auch in der für die Entwicklung von Lern-, E-Book-, sowie Business-Apps bewährt. Entwickelt wird Corona von der Firma Corona Labs. Für die Entwicklung von Apps setzt das Framework auf bewährte Technologien. Zur Entwicklung einer App liefert Corona selbst aber keine IDE mit, hier sollte daher auf einen externen Quellcodeeditor zurückgegriffen werden. Dabei muss beachtet werden, dass diese die Programmiersprache LUA unterstützen muss, da die gesamte App in LUA geschrieben werden muss, um von Corona verarbeitet werden zu können. Innerhalb der Entwicklung ist es dann Möglich mit dem Corona SDK auf über 500 APIs zuzugreifen. Zum Testen der App bietet Corona einen eignen Simulator an. (50)

Ist die Entwicklung der App abgeschlossen kann sie sehr leicht über den Simulator in Native Apps exportiert werden. Anschließend kann die App dann auf ein Smartphone oder Tablet übertragen werden, auch die Veröffentlichung in einem App Store ist möglich.

5.4 Phonegap

Phonegap ist ein Framework für die Entwicklung von Apps. Ursprünglich wurde Phonegap von der Firma Nitobi entwickelt. Nitobi wurde dann im Oktober 2011 von Adobe Systems übernommen und der Phonegap Code wurde der Apache Software Foundation für ein neues OpenSource Projekt zur Verfügung gestellt. Diese Projekt bekam am Anfang den Name Apache Callback und wurde später in Cordova umbenannt. Aktuell ist Phongap damit eine Distribution von Cordova, und kann somit auch Funktionen enthalten die Cordova selbst nicht zur Verfügung stellt.(51)

Zur Entwicklung von Apps kommen bei Phonegap clientseitige Webtechnologien zum Einsatz. Diese sind Javascript , HTML und CSS. Durch diesen Entwicklungsansatz soll die Erstellung von Cross- Plattform Apps erleichtert werden. Dadurch ist es auch möglich, Phongap zusammen mit anderen Cross Frameworks zu nutzen, deren Hauptfunktion die Gestaltung der Benutzeroberfläche ist. Frameworks die hier in Frage kommen sind z.B. "Sencha Touch"[9] und "jQuery Mobile"[10]. (51)

[9] Sencha Touch ist ein Cross Framework und wurde in Kapitel 5.2 erklärt.
[10] jOery Mobile ist ein Cross Framework und dient hauptsächlich zur Gestaltung von Benutzeroberflächen.

Ein wichtiger Punkt bei der Entwicklung mit Phonegap ist der Zugriff auf einen Teil der Hardware des Smartphones. Um diese Funktionen anzubieten setzt Phonegap ein Foreign Function Interface[11] ein. Hat der Programmierer nun den Webcode fertig, wird dieser mit den Native Codeteilen, die Phongap für jedes Betriebssystem bereitstellt, zusammen gepackt. In der fertigen App wird dann der Webcode in der betriebssystemeigenen Webengine ausgeführt. Phonegap umschließt dieses dann so, das dem Nutzer nicht mehr ersichtlich ist das die Webengine genutzt wird. Dieser Vorgang wird auch als Wrapper bezeichnet. Zusammen gefasst besteht die fertige App dann zum Teil aus Webcode und zum anderen aus Native Code. Dieses wird dann zusammen als webbasierte Hybride App (siehe Kapitel 4.2.2) bezeichnet.

5.5 Titanium

Titanium ist ein Framework zur Entwicklung von Cross Compiled Apps für Smartphones, Tablets und Desktops. Momentan unterstützt Titanium iOS, Android, Windows, BlackBerry, OSX und Linux. Bei der Entwicklung setzt Titanium ganz auf JavaScript und kommt auch ohne HTML5 und CSS aus. Die Oberfläche wird im Code durch JavaScript beschrieben. Titanium bietet von sich aus schon eine Menge Softwaremodule an. Zusätzlich gibt es aber auch noch die Möglichkeit weitere Module über den Titanium Store zu erwerben.

Bei der Entwicklung einer App stellt Titanium dem Programmierer eine vollständige IDE zur Verfügung. Diese enthält einen Assistenten zur Quellcodevervollständigung und ermöglicht es alle Projekte zu verwalten, sowie das Debuggen in Emulatoren oder auch auf den Endgeräten. Noch dazu gibt es die Möglichkeit die erstellten Apps direkt in den App Store hochzuladen.

Ist eine App fertig entwickelt wird sie für jede Zielplattform in nativen Code umgesetzt. Das hat einen erheblichen Performancevorteil gegenüber Web Apps (siehe Kapitel 4.2.1) und webbasierten Hybrid Apps (siehe Kapitel 4.2.2).

[11] Foreign Function Interface (FFI) ist ein Schnittstelle, mit dem es einem Programm ermöglicht wird Routinen oder Dienstleistungen eines anderen Programms, was in einer anderen Sprache geschrieben wurde, aufzurufen.

6 Zu evaluierende Native Frameworks

6.1 Google Android SDK

Das von Google entwickelte Android SDK enthält alle Programme die zum Erstellen von Android Apps nötig sind. Diese sind dann z.B. Debuggger und Emulatoren. Das SDK selbst hat keine eigene IDE, es wurde aber von Google in die Eclipse IDE mittels Plug-In integriert. Es ist auch möglich andere IDEs wie z.B. NetBeans zu benutzen, offiziell wird aber Eclipse unterstützt. Zur Oberflächengestaltung einer App kann der Android eigene GUI Editor verwendet werden. Die eigentliche App wird dann mit JAVA und XML entwickelt. (52)

Ist eine App fertig gestellt kann sie mit Hilfe der IDE zur einer .apk Datei zusammen gepackt werden. (52) Dazu wird das im SDK enthaltene Programm "dx" verwendet. Dieses dient dazu herkömmliche Java Binärdateien in das Dalvik Executeble Format umzuwandeln. Das bedeutet, dass aus ".class" Dateien ".dex" Dateien erzeugt werden. Die Umwandlung wird vorgenommen, da im Android Betriebssystem eine Dalvik Virtual Maschine implementiert ist. Der Vorteil zur Java Virtual Maschine ist die höhere Geschwindigkeit und das der Code ressourcenschonender ist. (53)

6.2 Apple Xcode

Xcode ist eine integrierte Entwicklungsumgebung. Sie dient dazu Programme für Mac OSX oder iOS zu schreiben. Sie ist nur auf dem Mac OSX Betriebssystem verfügbar. Xcode unterstützt durch seine Modularität eine Vielzahl von Programmiersprachen, der Fokus liegt aber auf den Programmiersprachen C, C++ und Objective- C mit dem Cocoa Framework. Zur Entwicklung von Programmen enthält Xcode eine Menge an Tools zum Debuggen, Kompilieren und zum GUI designen sowie zum Simulieren von iPad und iPhone Anwendungen. (54)

Die Entwicklung von iOS Apps in Xcode erfolgt mit der Programmiersprache Objektive-C und der LLVM[12]-Compiler-Suite. (55)

Die LLVM-Compiler-Suite dient dazu den Objektive-C Code in eine LLVM-Zwischensprache zu übersetzen. Diese Zwischensprache wird dann zur Laufzeit auf das aktuelle System übersetzt. Um dieses zu erreichen wird eine virtuelle Maschine eingesetzt, die in der Lage ist einen Hauptprozessor zu virtualisieren. Der große Vorteil von LLVM ist aber die Performance, dadurch ist es auch möglich das Programme auch auf langsamen Systemen laufen. (56)

[12] LLVM steht für Low Level Virtual Machine (55)

Ist die Entwicklung einer App abgeschlossen wird sie mit dem LLVM Compiler übersetzt und kann auf ein iOS Gerät übertragen werden. (55)

6.3 Microsoft Windows Phone SDK

Das Microsoft Windows Phone SDK enthält alle Programm die zum Erstellen von Windows Phone Apps nötig sind. Die wichtigsten Komponenten sind ein Compiler, ein Emulator und Quellcode Bibliotheken. Als IDE kommt beim Windows Phone SDK das Visual Studio 2012 oder Expression Blend zum Einsatz. Mit Hilfe des SDK ist die IDE in der Lage einen qngepassten GUI Editor zu Verfügung zu stellen. (25)

Windows Phone Apps werden in den Programmiersprachen C# oder Visual Basic geschrieben. Diese definieren die Funktionsweise der App. Zusätzlich wird noch XAML[13] ,welches das Aussehen der Oberfläche definierter, verwendet. Die XAML Datei kann entweder mit Hilfe des graphischen Editors generiert werden oder manuell geändert werden. (25)

Ist die App fertig geschrieben wird sie mit der IDE und den SDK Tools auf Fehler überprüft sowie kompiliert. Abschließend wird das gesamte Projekt mit allen Ressourcen wie z.B. Bilder in ein Anwendungspaket zusammen gepackt. Die Dateiendung für dieses ist "XAP". Im nächsten Schritt kann die App auf das Windows Phone oder den Emulator übertragen werden. (25)

[13] XAML steht für Extensible Application Markup Language und beschreibt hier das Aussehen der App (25)

7 Anforderung der Testanwendung

Die Implementierung der Testanwendung dient dazu herauszufinden in wieweit sich die einzelnen Cross Frameworks nutzen lassen, um die Sensoren, Kamera oder auch das Netzwerkinterface anzusprechen. Das Weiteren ist zu testen, ob die entwickelten Cross Apps auch auf Android, iPhone und Windows Phone lauffähig sind. Zusätzlich zu den Cross Frameworks wird die App noch als native App auf den drei eben genannten Plattformen implementiert, so dass ein Vergleich zwischen Cross Apps und Native Apps möglich ist. Dieser soll dann zeigen, in wieweit es möglich ist Native Apps durch Cross Apps zu ersetzen.

7.1 Aufgabe der Testanwendung

Wie in der Aufgabenstellung dieser Bachelorarbeit schon beschrieben, soll eine App entwickelt werden die Wanderer bei Ausflügen hilft. Die zu entwickelnde App bietet daher grundlegende Funktionen, um sich im freien Gelände zu orientieren. Eine dieser Funktionen ist das Anzeigen eines digitalen Kompasses. Auch das Anzeigen der Koordinaten für die aktuelle Position gehört zum Funktionsumfang dazu. Eine weitere nützliche Funktion ist das Anzeigen der Aktivitäten des Benutzers. Zum Schluss ist es dem Benutzer noch möglich Bilder aufzunehmen und zu teilen. Teilen bedeutet hier, dass der Benutzer die Bilder per Bluetooth auf ein anderes Gerät überträgt oder das er das Bild über WLAN oder UMTS an einen Server sendet.

7.2 Features der Anwendung

Im folgenden sind die einzelnen Features nochmal erfasst.

- **PF1** Kompass- Daten abrufen und ein Kompass- Icon demensprechend drehen
- **PF2** GPS- Daten abrufen und anzeigen
- **PF3** Bilder mit der Kamera aufnehmen und aus der Galerie auswählen
- **PF4** Daten übertragen mit UMTS/WLAN /Bluetooth
- **PF5** Beschleunigungssensor- Daten abrufen und anzeigen

7.3 Oberflächendesign

Die Benutzeroberfläche der Anwendung soll einfach und funktional gestaltet sein. Hierzu muss beachtet werden, dass eine Veröffentlichung der App in den jeweiligen App Stores möglich sein soll. Um das zu gewährleisten ist es wichtig die Designvorgaben der einzelnen Betriebssystemhersteller zu berücksichtigen (siehe Kapitel 3.2.2, Kapitel 3.3.2und Kapitel 3.4.2).

Zur Zeitersparnis reicht es, wenn die Anwendung mit dem Android Oberflächendesign als Standard entwickelt wird. Bei der Implementierung muss allerdings darauf hingewiesen werden, wenn die Möglichkeit besteht, die Designs vom iOS und Windows Phone umzusetzen.

Abbildung 7-1 Design des Startmenüs bei Android, iOS und Windows Phone

Bei der Menüführung ist auch darauf zu achten, dass die Vorgaben der Hersteller eingehalten werden. So navigiert der Nutzer bei Android mit dem Zurück-Button auf den vorherigen Bildschirm wohingegen bei iOS oben in der Titelleiste die Navigationsmöglichkeiten angezeigt werden. Bei der Implementierung ist dieses zu berücksichtigen.

7.4 GUI / Ressourcenzugriff

Kompass:

Hierbei soll dem Benutzer eine Grafik eines Kompasses angezeigt werden, die sich durch Berühren des Aktualisierungsbuttons ausrichtet. Das Ausrichten soll mit den Werten des im mobile Devices verbauten Kompassmodul geschehen.

Folgende Grafik soll dazu angezeigt werden.

Abbildung 7-2 Kompass Darstellung bei der Android App

GPS:

Hier soll es dem Benutzer ermöglicht werden die Längengrade und Breitengrad seiner aktuellen Position abzurufen. Die Aktualisierung der Daten soll vorzugsweise automatisch ablaufen. Ist das nicht möglich ist ein Aktualisierungsbutton zu implementieren. In der nachfolgenden Grafik ist dieses Dialogfenster auf einem Android-Gerät Dargestellt.

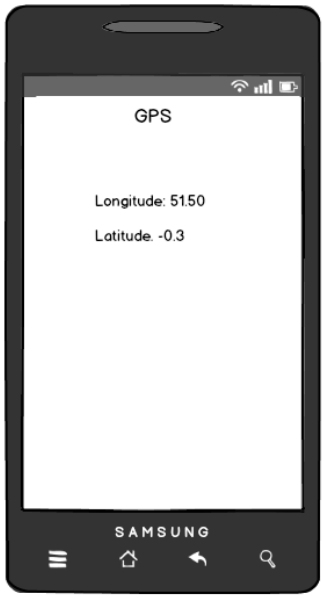


Abbildung 7-3 GPS Daten Darstellung bei der Android App

Beschleunigungssensor:

Bei diesem Punkt soll es dem Benutzer gestattet sein, seine letzten Aktivitäten zu sehen. Die Aktivitäten sind hierbei die Daten des Beschleunigungssensors. Um diese Werte übersichtlich darzustellen, soll es eine Tabelle geben, in der die Werte stehen. Optional kann die Tabelle durch ein Liniendiagramm ersetzt werden, um die Übersichtlichkeit noch zu erhöhen. Als Beispiel ist dieses Fenster in den nachfolgenden Grafiken zusehen.

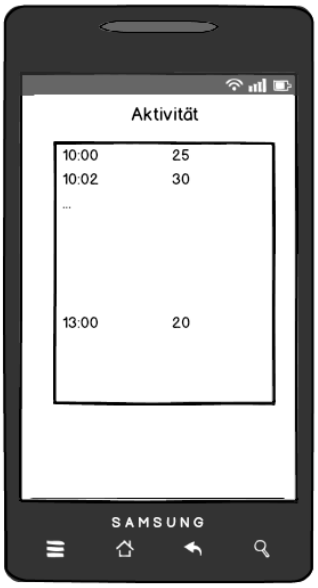


Abbildung 7-4 Darstellung der Aktivitäten in Tabellenform bei der Android App

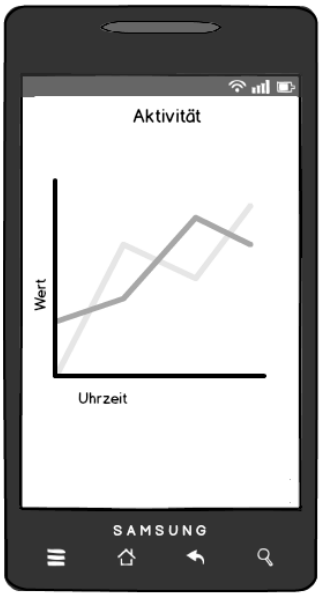


Abbildung 7-5 Darstellung der Aktivitäten in Graphenform bei der Android App

Kamera:

Zum Festhalten der Umgebung ist bei der App die Möglichkeit zum Aufnehmen von Bildern zu implementieren. Zur besseren Übersicht ist jeweils das zuletzt aufgenommene Bild anzuzeigen. Des Weiteren soll es hier schon die Möglichkeit gegeben seine Bilder zu versenden mittels Netzwerk oder Bluetooth. In der folgenden Grafik ist das Beispiel zu dieser Oberfläche bei einem Android Gerät zusehen.

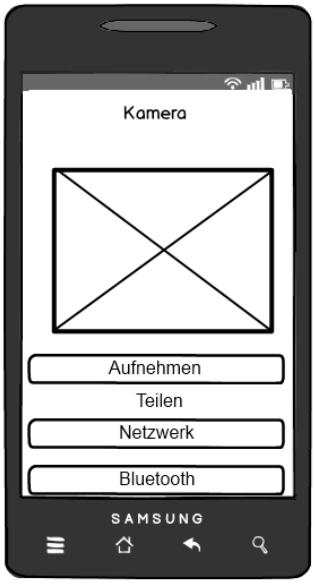


Abbildung 7-6 Darstellung des Bildschirms zum Aufnehmen eines Bildes bei der Android App

Netzwerk:

Hier soll der Nutzern nun die in der Lage sein ein aufgenommenes Bild auf einen Webserver zu laden. Dazu soll er die Adresse des Servers eingeben, sowie das zu versendende Bild auswählen. Anschließend wird das Bild automatisch zum Server gesendet. Die Übertragung des Bildes soll intern als HTTP Post Request realisiert werden. Die Dokumentation des Webservers befindet sich im Anhang. Nachfolgend ist das Fenster auf einem Android Gerät dargestellt.

Abbildung 7-7 Dialog zum Senden eines Bildes zum einem Server bei der Android App

Bluetooth:

Abbildung 7-8 Bildübertragung mittels Bluetooth bei der Android App

7.5 Performance

Da diese App hauptsächlich Sensoren abfragt, ist darauf zu achten, dass der Nutzer nicht länger als drei Sekunden warten muss, bis sich ein Zustand geändert hat.

7.6 Zielplattformen

Die fertige App ist auf den folgenden Geräten mit den jeweiligen Betriebssystem zum Laufen zu bringen.

Tabelle 7-1 Zielplattformen der Testapp

Gerät	Betriebssystem
iPhone 4S	iOS 6.1
Nexus S	Android 4.1
Nokia Lumia 620	Windows Phone 8

7.7 Lizenzkosten

Zur Entwicklung dieser App werden nur die Frameworks verwendet die komplett kostenlos zur Verfügung stehen oder eine kostenlose Testversion anbieten. In der anschließenden Evaluierung sollen aber für die Testversionen die Preise für die Komplettversion berücksichtigt werden. Es sind aber auch die Lizenzkosten für den Zugang zu den AppStores zu berücksichtigen.

7.8 Vorbereitung zur Distribution

Die Testanwendung ist so zu implementieren, dass es möglich ist, sie in den jeweiligen AppStores zu veröffentlichen.

7.9 Requirements der Anwendung

Aus den genannten Features(siehe Kapitel 7.2) der Testanwendung lassen sich folgende Requirements ableiten.

- **[Req1]** Das Auslesen der Kompass- Daten
- **[Req2]** Das Drehen eines Kompassicon
- **[Req3]** Das Auslesen der GPS- Daten
- **[Req4]** Das Anzeigen der GPS- Daten
- **[Req5]** Das Aufnehmen von Bildern mit der Kamera
- **[Req6]** Das Auswählen von Bildern aus der Gallerie
- **[Req7]** Des Versenden von Bildern mit WLAN
- **[Req8]** Des Versenden von Bildern mit UMTS
- **[Req9]** Des Versenden von Bildern mit Bluetooth
- **[Req10]** Das Auslesen der Beschleunigungssensor- Daten
- **[Req11]** Das Anzeigen der Beschleunigungssensor - Daten

Diese Requirements werden den Anforderungen der Testanwendung aus dem Kapitel 7.1 wie folgt gegenüber gestellt.

Tabelle 7-2 Zuordnung der Requirements 7.9 zu den Produktfeatures 7.2

Requirement	**PF1**	**PF2**	**PF3**	**PF4**	**PF5**
Req1	X				
Req2	X				
Req3		X			
Req4		X			
Req5			X		
Req6			X		
Req7				X	
Req8				X	
Req9				X	
Req10					X
Req11					X

7.10 Struktur der Anwendung

Durch die vorhergehenden Anforderungen ergibt sich folgende Struktur der Testanwendung.

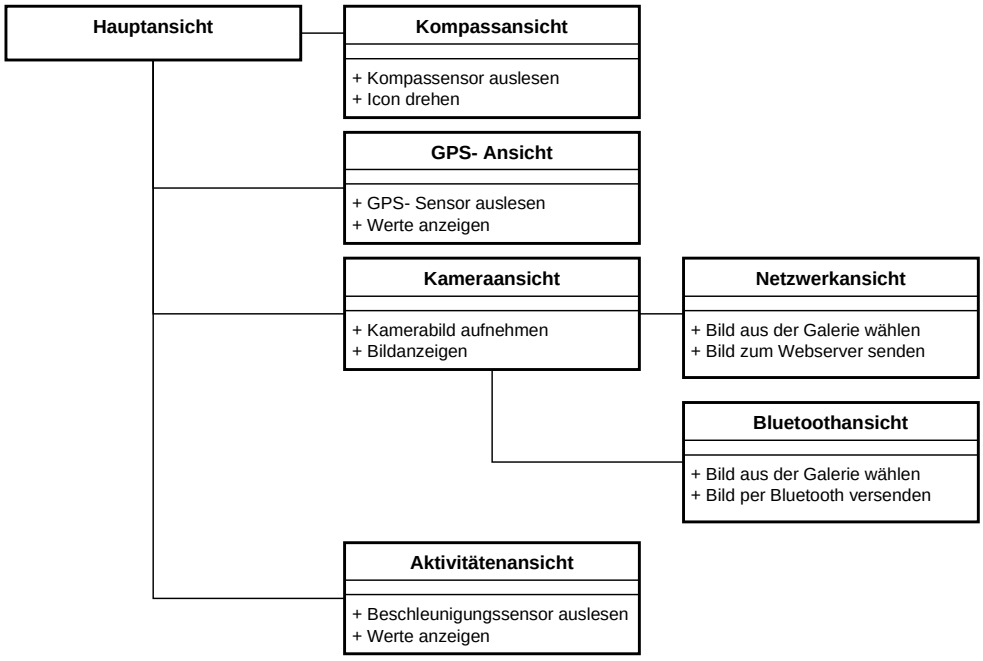


Abbildung 7-9 Struktur der Testanwendung

8 Bewertung der Frameworks

Dieses Kapitel zeigt einen Überblick über die zu bewertenden Punkte mit den dazugehörigen Noten.

8.1 Frameworkumfang

Zur Bewertung des Implementierungsaufwandes wird das folgende drei Kategoriensystem verwendet. Zu sehen ist es in der nachfolgenden Tabelle. Hier wird für jeden Bereich der bei der Umsetzung wichtig ist eine Note vergeben.

Tabelle 8-1 Bewertung des Implementierungsaufwands

Bereich	**1(sehr gut)**	**2(gut)**	**3(befriedigend)**
Dokumentation	Quellcode Dokumentation sowie Codebeispiele	Quellcode Dokumentation	Keine Dokumentation
Installation	Automatischer Installer verfügbar, der nur ausgeführt werden muss	Automatischer Installer verfügbar, der nur ausgeführt werden muss, aber zusätzlichem Anpassen im Anschluss	Installation muss manuel vorgenommen werden
GUI Builder	Eigener GUI Builder vorhanden	Externer GUI Builder verfügbar	Kein GUI Builder verfügbar
Quellcodeeditor	Eigener Quellcodeeditor verfügbar	Externer Quellcodeeditor verfügbar	Kein Quellcodeeditor verfügbar

8.2 GUI- Design

Hier geht es nun um die Bewertung der Oberfläche der App. In der nachfolgenden Tabelle sind die

Bewertungskriterien für das Oberflächendesign angegeben.

Tabelle 8-2 Bewertung der Oberflächengestaltung

Oberfläche	**Punkte**
Automatisch angepasst für die unterstützten Plattformen	1 (sehr gut)
Manuell angepasst für die unterstützten Plattformen	2 (gut)
Nur Unterstützung für eine Oberfläche, die auf allen unterstützten Plattformen gleich ist	3 (befriedigend)

8.3 Quellcodeumfang

Die nachfolgende Tabelle zeigt die Bewertungskriterien für den Zugriff auf die Sensoren sowie auf weitere Ressourcen. Zur Bewertung werden hier die Codezeilen herangezogen.

Tabelle 8-3 Bewertung des Quellcodeumfangs

Codezeilen für den Zugriff auf einen Sensor oder Ressource	**Punkte**
5 und weniger	1 (sehr gut)
10 und weniger	2 (gut)
Mehr als 10	3 (befriedigend)

8.4 Unterstützte Plattformen

Zur Bewertung der Frameworks, in Bezug auf die Unterstützung von verschiedenen Plattformen, ist die folgende Tabelle gedacht. Es wird hier aber nur die Unterstützung für die in diesem Test verwendeten Plattformen gewertet.

Tabelle 8-4 Bewertung der unterstützten Plattformen

Unterstützte Plattformen	Punkte
3 und mehr	1 (sehr gut)
2	2 (gut)
1	3 (befriedigend)

8.5 Sensortests

In diesem Kapitel wird beschrieben, wie die einzelnen Sensoren, die bei der Testapp zu implementieren sind, getestet werden sollen. Bei der endgültigen Auswertung wird ein bestandener Test mit 1 bewertet. Ist vom Framework eine Unterstützung des jeweiligen Sensors vorgesehen, aber der Test ist fehlgeschlagen, so wird es mit 2 bewertet. Kommt es nun vor, dass das Framework keine Unterstützung anbietet, wird das mit 3 bewertet.

8.5.1 Kompass- Sensor

Zum Testen des Kompass bei der App wird dieser mit einem analogen Kompass verglichen. Dazu werden beide auf einen ebenen Untergrund gelegt und fünfmal mal nach rechts gedreht und anschließend wird dann jedesmal die Ausrichtung der Kompassnadel notiert.

Der Test ist bestanden, wenn der Kompass der App jedes mal in die gleiche Richtung zeigt, wie der analoge Kompass.

8.5.2 GPS- Sensor

Der Test des GPS- Sensors in der App wird mit Hilfe eines Navigationsgerätes im freien durchgeführt. Hierzu wird sowohl das Smartphonee als auch das Navigationsgerät auf eine Ebene Fläche nebeneinander gelegt. Anschließend werden die Koordinaten miteinander verglichen. Das Vergleichen der Koordinaten wird fünfmal wiederholt.

Der Test ist bestanden, wenn die Koordinaten alle fünf Mal übereinstimmten.

8.5.3 Bildaufnahme

Beim Test der Bildaufnahme wird ein Testbild vor die Kamera gelegt und fotografiert. Dieses wurde nun fünfmal wiederholt. Nach jeder Aufnahme wird das aufgenommene Bild mit dem Testbild

verglichen.

Der Test ist bestanden, wenn alle aufgenommenen Bilder mit dem Testbild übereinstimmen.

8.5.4 Netzwerk Bildupload

Zum Testen des Bilduploads der App werden mit der Kamera fünf verschiedenen Testbilder aufgenommen. Diese werden dann aus der Galerie ausgewählt und an den Server gesendet. Nach jedem Bildupload wird überprüft, ob das hochgeladenen Bild mit dem auf dem Smartphone übereinstimmt.

Der Test ist bestanden, wenn alle fünf Bilder erfolgreich hochgeladen wurden.

8.5.5 Bluetooth Bildtransfer

Zum Testen des Bilduploads der App werden mit der Kamera fünf verschiedenen Testbilder aufgenommen und an ein anderes Smartphone gesendet. Nach jedem Bildupload wird überprüft, ob das gesendete Bild auf beiden Smartphone übereinstimmt.

Der Test ist bestanden, wenn alle fünf Bilder erfolgreich versendet wurden.

8.5.6 Beschleunigungs-Sensor

Der Test des Beschleunigungs-Sensors erfolgt auf einem ebenen Untergrund. Bei der Durchführung wird das Smartphone erst mit der Rückseite auf den Tisch gelegt und danach mit der Vorderseite. Hierbei zeigt der Beschleunigungs-Sensor die Erdanziehungskraft jeweils an einer anderen Achse an. Der Legewechsel wird dann fünfmal wiederholt und jedesmal wird überprüft, ob auch die Anzeige gewechselt hat.

Der Test ist bestanden, wenn die Anzeige bei allen fünf Wechseln auch gewechselt hat.

8.6 Testabdeckung

In der nachfolgenden Tabelle ist zusehen, welche Anforderung der App, mit welchem Testfall abgedeckt wird.

Tabelle 8-5 Testabdeckung der App

	Tests					
Requirement	Kompass	GPS	Bildaufnahme	Bildupload	Bluetooth	Beschleunigung
Req1	X					
Req2	X					
Req3		X				
Req4		X				
Req5			X			
Req6				X		
Req7				X		
Req8				X		
Req9					X	
Req10						X
Req11						X

9 Implementierung der Testanwendung

In diesem Kapitel wird im Folgenden genau erklärt, wie die Implementierung mit Hilfe der einzelnen Frameworks funktioniert.

An dieser Stelle ist zu beachten, dass die Nativen Frameworks hier, anders als in den vorherigen Kapitel, als erstes aufgeführt werden. Der Grund ist, dass die Cross Frameworks sich bei der Implementierung auf die Native Frameworks stützen.

Des Weiteren wird sich bei der Beschreibung der Implementierung nur auf den direkten Zugriff auf die Sensoren konzentriert.

9.1 Google Android SDK

9.1.1 Vorbereitung

Lizenzsierung:

Bei der Entwicklung von Android Apps ist es nicht notwendig sich eine Lizenz von Google zu erwerben. Diese wird nur gebraucht, wenn die fertige App in den App Strore eingestellt werden soll.

Installation:

Die Anleitung zur Installation des SDK befindet sich im Anhang im Kapitel 15.1 .

9.1.2 Implementierung

Das Auslesen der Kompass- Daten [Req1] und Drehen des Kompassicon [Req2]

Zum Auslesen der Kompassdaten ist es bei Android notwendig auf zwei Sensoren zugreifen diese werden in der nachfolgenden Grafik in den Zeilen 63 und 64 zugewiesen. Zusätzlich wird noch der SensorManager von Android benötigt der die Sensoren überwacht. Um nun die Werte auszulesen, werden noch die zu den beiden Sensoren gehörenden Listener registriert. In diesem Fall ist die aktuelle Klasse selbst der Listener, so dass sie auf die Ereignisse der Sensoren reagieren kann, wie in Zeile 72 zusehen. Je nachdem welchen Sensor das Event ausgelöst hat, wird der Wert gespeichert. Sind nun für beide Sensoren Werte vorhanden, folgt die Umrechnung der Werte in das Koordinatensystem der Welt, siehe Zeile 81. Nun wird noch die Abweichung von Norden

ausgelesen und in einen Winkel umgerechnet. Als nächstes wird das Kompassicon nun gedreht, wie in Zeile 88 zusehen.

```
60
61          //Sensoren initialiesiren
62          mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
63          accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
64          magnetometer = mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
65
66          //Eventlistner registrieren
67          mSensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_UI);
68          mSensorManager.registerListener(this, magnetometer, SensorManager.SENSOR_DELAY_UI);
69        }
70
71        //Eventlistner für die Sensoren
72        public void onSensorChanged(SensorEvent event) {
73          if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
74            mGravity = event.values;
75          if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD)
76            mGeomagnetic = event.values;
77          if (mGravity != null && mGeomagnetic != null) {
78            float R[] = new float[9];
79            float I[] = new float[9];
80            //Umrechnen der Sensorwerten in das Koordinatensystem der Welt
81            boolean tFinisch = SensorManager.getRotationMatrix(R, I, mGravity, mGeomagnetic);
82            if (tFinisch) {
83              float tOrientation[] = new float[3];
84              SensorManager.getOrientation(R, tOrientation);
85              //Auslesen des Azimut(abweichung von Norden)
86              mNordview = tOrientation[0];
87              //Umrechnen in einen Rotationswinkel
88              mImageView.setRotation(-mNordview*360/(2*(float)Math.PI));
```

Abbildung 9-1 Auslesen und Auswerten der Kompassdaten bei Android

Das Auslesen [Req3] und Anzeigen [Req4] der GPS- Daten

Beim Auslesen der GPS- Daten ist es bei Android wichtig den Locationmanager zu initialisieren, wie in dem folgenden Bild in Zeile 29 zusehen. Im nächsten Schritt muss dem Locationmanager ein Ziel für Änderungen hinzugefügt werden, das ist, wie in Zeile 38 zusehen, die aktuelle Klasse. In der Zeile 43 ist nun zusehen, dass bei Änderungen die aktuelle Position ausgelesen und angezeigt wird.

```
29          LocationManager locationManager = (LocationManager)
30                  getSystemService(Context.LOCATION_SERVICE);
31
32
33          mLong = (TextView)findViewById(R.id.txtLong);
34          mLat = (TextView)findViewById(R.id.txtLat);
35
36          //GPS abruf initialiesieren
37          locationManager.requestLocationUpdates(
38          LocationManager.GPS_PROVIDER, 5000, 10, this);
39      }
40
41      //Auktuelle Position ausgeben
42      @Override
43      public void onLocationChanged(Location loc) {
44          mLat.setText(loc.getLatitude()+"");
45          mLong.setText(loc.getLongitude()+"");
46      }
47
48      @Override
49      public boolean onCreateOptionsMenu(Menu menu) {
50          // Inflate the menu; this adds items to the action bar if it is present.
51          getMenuInflater().inflate(R.menu.g, menu);
52          return true;
```

Abbildung 9-2 auslesen und anzeigen der GPS Koordinaten bei Android

Das Aufnehmen von Bildern mit der Kamera [Req5]

Zur Aufnahme von Bilder gibt es bei Android die Möglichkeit das über den systeminternen Dialog zu ermöglichen. Wie in der nachfolgenden Grafik zu sehen, wird dieser Dialog in Zeile 60 initialisiert.

```
56      //Buttonlistner der zur aufnahme eines Bildes gestartet wird
57      aufnehmen.setOnClickListener(new OnClickListener() {
58          @Override
59          public void onClick(View v) {
60              Intent tPictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
61              //Speicherort festlegen
62              File dir= Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM);
```

Abbildung 9-3 Kameraaufnahmedialog bei Android

Die folgende Grafik zeigt nun was passiert, wenn ein Foto erfolgreich aufgenommen wurde. In diesem Fall wird es eingelesen, wie in Zeile 96 zusehen. Nun wird das Bild noch in der Größe verändert, damit es angezeigt werden kann. In Zeile 99 wird es nun in der Oberfläche angezeigt.

```
89      //Funktion die ausgeführt wird nach dem aufnehmen eines Bildes
90      @Override
91      protected void onActivityResult(int requestCode, int resultCode, Intent data) {
92          if(requestCode == CAMERA_FOTO && resultCode == RESULT_OK){
93              Uri turi = Uri.fromFile(mOutputDir);
94              try {
95                  //Bild auf die richtige Größe skallieren
96                  Bitmap tBmp = Media.getBitmap(getContentResolver(), turi );
97                  int tScal = (int) ( tBmp.getHeight() * (512.0 / tBmp.getWidth()) );
98                  Bitmap tScalBmp = Bitmap.createScaledBitmap(tBmp, 512, tScal, true);
99                  mImageView.setImageBitmap(tScalBmp);
```

Abbildung 9-4 Aufgenommenes Bild bei Android anzeigen

Das Auswählen von Bildern aus der Galerie [Req6]

Beim Auswählen von Bildern aus der Galerie verhält es sich ähnlich wie beim Aufnehmen von Bilder zuvor. Der einzige Unterschied besteht in der Initialisierung des Dialogfensters, wie die folgende Grafik in Zeile 61 und 62 zeigt.

```
58          auswahl.setOnClickListener(new OnClickListener() {
59              @Override
60              public void onClick(View v) {
61                  Intent intent = new Intent();
62                  intent.setType("image/*");
```

Abbildung 9-5 Bilder aus der Gallerie auswählen bei Android

Das Versenden von Bildern mit WLAN [Req7] oder UMTS [Req8]

Damit es bei Android möglich ist Bilder per Http Post zu versenden wurden zusätzliche Bibliotheken eingebunden, wie in der folgenden Abbildung zusehen.

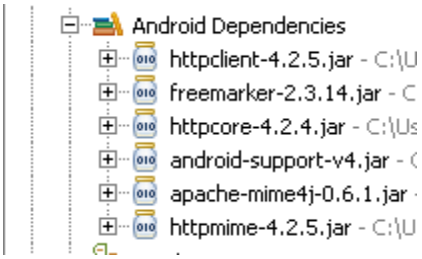


Abbildung 9-6 Zusatzbibliotheken bei Android

Nach dem Einbinden der zusätzlichen Bibliotheken ist es möglich, die in der nachfolgenden Grafik in Zeile 117 und 118 gezeigten, HttpClient und Post Objekte anzulegen. Im nächsten Schritt muss das Bild eingelesen werden, was ab Zeile 120 beginnt. Ist das Bild erfolgreich eingelesen, wird der Request zusammen gebaut, wie ab Zeile 133 zusehen. Anschließend kann er dann ausgeführt

werden.

```
115             URI uri = new URI("http://x-y-z.bplaced.net/uploadfile.php");
116             //Request anlegen
117             HttpClient httpclient = new DefaultHttpClient();
118             HttpPost post = new HttpPost(uri);
119
120             //Bild einlesen
121             File file = new File(mImagePath.toString());
122             int size = (int) file.length();
123             byte[] bytes = new byte[size];
124             try {
125                 BufferedInputStream buf = new BufferedInputStream(new FileInputStream(file));
126                 buf.read(bytes, 0, bytes.length);
127                 buf.close();
128             } catch (Exception e) {
129                 // TODO Auto-generated catch block
130                 e.printStackTrace();
131             }
132             //Request zusammenstellen
133             MultipartEntity entity = new MultipartEntity();
134             ByteArrayBody bab = new ByteArrayBody(bytes, "image2.png");
135             entity.addPart("media", bab);
136             post.setEntity(entity);
137             //Request starten
138             HttpResponse response = httpclient.execute(post);
```

Abbildung 9-7 Bildupload bei Android

Das Versenden von Bildern mit Bluetooth [Req9]

Zum Versenden von Bildern per Bluetooth bietet Android die Möglichkeit den systemeigenen Versanddialog zu benutzen. In der nachfolgenden Grafik ist die Verwendung dieses Dialoges zusehen. In Zeile 51 wird definiert, dass der Dialog zum Versenden geöffnet werden soll. Zum Versenden eines Bildes wird nun dem Dialog die Adresse des Bildes mitgeteilt. Im Anschluss daran wird der Dialog in Zeile 55 geöffnet.

```
49              Uri tImage = data.getData();
50              Intent tIntent = new Intent();
51              tIntent.setAction(Intent.ACTION_SEND);
52              tIntent.setType("text/plain");
53              tIntent.putExtra(Intent.EXTRA_STREAM, tImage);
54              //BluetoothDialog starten
55              startActivity(tIntent);
56          } catch (Exception e) {
57              e.printStackTrace();
58          }
```

Abbildung 9-8 Bilder per Bluetooth versenden bei Android

Das Auslesen [Req10] und Anzeigen [Req11] der Beschleunigungssensor- Daten

Zum Auslesen des Beschleunigungssensors ist es wichtig zunächst einen Sensormanager zu initialisieren. In der nachfolgenden Grafik passiert das in der Zeile 34. Im Anschluss wird noch der Sensor an sich festgelegt und die aktuelle Klasse als Ziel von Änderungen eingetragen, siehe Zeile 38. Ist nun eine Änderung der Sensorwerten eingetreten, lassen sich die Werte leicht auslesen wie in Zeile 44 zusehen.

```
33          // Sensor initialisieren
34          mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
35          mAccelerometer = mSensorManager
36                  .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
37          mSensorManager.registerListener(this, mAccelerometer,
38                  SensorManager.SENSOR_DELAY_UI);
39      }
40
41      // Sensorwerte auslesen
42      public void onSensorChanged(SensorEvent event) {
43          if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
44              String werte = event.values[0] + " " + event.values[1] + " "
45                      + event.values[2];
46              addRow(werte);
```

Abbildung 9-9 Auslesen und Anzeigen des Beschleunigungssensors bei Android

9.1.3 Evaluation

Die Entwicklung einer App mit dem Android SDK war sehr einfach, da die mitgelieferte Entwicklungsumgebung sich sehr leicht installieren lies. Diese brachte dann einen GUI- Builder sowie einen Quellcodeeditor mit. Damit war es sehr einfach möglich eine Android App zu erstellen. Auch die Übertragung der App auf ein Smartphone ist mit der IDE leicht möglich.

Die folgende Tabelle stellt die Bewertung dieses Frameworks da. Auffällig ist hier, dass die Dokumentation etwas schlechter abgeschnitten hat. Das liegt daran, dass ich nicht gleich Quellcodebeispiele gefunden habe. Ein weiterer kritischer Punkt ist, dass schlechte Abschneiden bei den unterstützten Plattformen. Das liegt aber daran, dass es ein Native Framework ist und dadurch nur für Android gedacht ist.

Tabelle 9-1 Evaluation Android Allgemein

	Bewertung
Herstellerdokumentation	2
Installationsaufwand	1
GUI- Builder	1
Quellcodeeditor	1
Unterstützte Plattformen	3
GUI- Designe	1

In der nun folgenden Tabelle zeigt sich wie einfach es ist mit dem Framework bestimmte Ressourcen anzusprechen und wie es beim Test abgeschnitten hat. Die Bewertung des Kompass-Sensors ist beim Quellcodeumfang bei nur einer 3, weil zum Auslesen kompliziert mit zwei Sensoren gearbeitet werden muss. Bei dem Test ist es dann auch nur auf eine 2 gekommen, weil der Kompass nicht immer in die richtige Richtung gezeigt hat. Bei der Kamera, der Bildergalerie und dem Bluetooth Bildversand sind beim Quellcodeumfang nur 3 Punkte erreicht worden, weil hier jeweils ein systemeigenes Dialogfenster geöffnet wurde. Des Weiteren hat es beim Bildupload zum Server auch nur 3 Punkte gegeben. Der Grund hierfür sind die externen Bibliotheken, die noch eingebunden werden mussten.

Tabelle 9-2 Evaluation XCODE Ressourcenzugriff

	Bewertung / Punkte	
Ressource	**Quellcodeumfang**	**Durchgeführter Test**
Kompass- Sensor	3	2
GPS- Sensor	1	1
Bild mit der Kamera aufnehmen	3	1
Bild aus der Gallerie auswählen	3	1
Bildupload zum Server	3	1

Bild per Bluetooth versenden	3	1
Beschleunigungssensor	1	1

9.2 Apple Xcode

9.2.1 Vorbereitung

Lizenzsierung:

Bei der Entwicklung von iOS Apps ist es notwendig eine Lizenz von Apple zur erwerben. Diese wird gebraucht, um die fertige App auf das Smartphone zu übertragen oder um die App in den AppStore einzustellen.

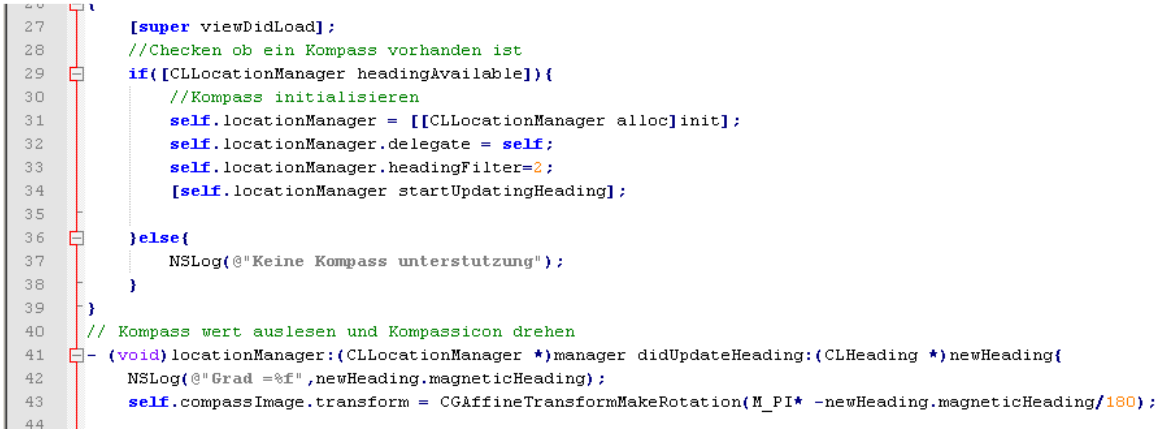
Installation:

Die Anleitung zur Installation des SDK befindet sich im Anhang im Kapitel 15.2 .

9.2.2 Implementierung

Das Auslesen der Kompass- Daten [Req1] und Drehen des Kompassicon [Req2]

Bei iOS ist zum Auslesen der Kompass- Daten eine Locationmanager zu initialisieren, was in der folgenden Grafik in der Zeile 31 zusehen ist. Des Weiteren wird die aktuelle Klasse als Ziel von Ereignissen hinzugefügt. Ab der Zeile 41 ist nun zusehen, was passiert, wenn sich die Kompass-Daten ändern. In der Zeile 43 wird nun die Drehung des Kompasses berechnet und das Bild entsprechen gedreht.

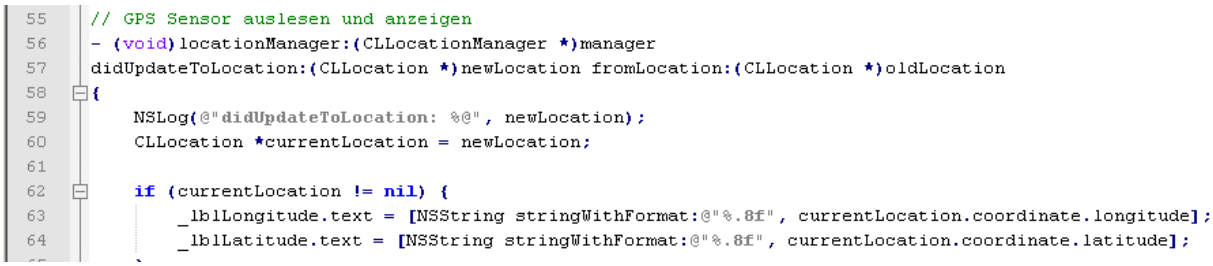


```
27      [super viewDidLoad];
28      //Checken ob ein Kompass vorhanden ist
29      if([CLLocationManager headingAvailable]){
30          //Kompass initialisieren
31          self.locationManager = [[CLLocationManager alloc]init];
32          self.locationManager.delegate = self;
33          self.locationManager.headingFilter=2;
34          [self.locationManager startUpdatingHeading];
35
36      }else{
37          NSLog(@"Keine Kompass unterstutzung");
38      }
39  }
40  // Kompass wert auslesen und Kompassicon drehen
41  - (void)locationManager:(CLLocationManager *)manager didUpdateHeading:(CLHeading *)newHeading{
42      NSLog(@"Grad =%f",newHeading.magneticHeading);
43      self.compassImage.transform = CGAffineTransformMakeRotation(M_PI* -newHeading.magneticHeading/180);
```

Abbildung 9-10 Kompass-Sensor auswerten bei iOS

Das Auslesen [Req3] und Anzeigen [Req4] der GPS- Daten

Beim Auslesen der GPS- Daten ist es wichtig zunächst einen LocationManager zu initialisieren. Ist das geschehen, muss im nächsten Schritt ein Ziel für Events hinzugefügt werden. Im folgenden Bild ist die aktuelle Klasse das Ziel. Tritt nun eine Änderung der Werte ein, wird der Code ab Zeile 56 ausgeführt. Die Koordinaten können nun ausgelesen und angezeigt werden, wie in Zeile 63 und 64 zusehen.



```
55  // GPS Sensor auslesen und anzeigen
56  - (void)locationManager:(CLLocationManager *)manager
57  didUpdateToLocation:(CLLocation *)newLocation fromLocation:(CLLocation *)oldLocation
58  {
59      NSLog(@"didUpdateToLocation: %@", newLocation);
60      CLLocation *currentLocation = newLocation;
61
62      if (currentLocation != nil) {
63          _lblLongitude.text = [NSString stringWithFormat:@"%.8f", currentLocation.coordinate.longitude];
64          _lblLatitude.text = [NSString stringWithFormat:@"%.8f", currentLocation.coordinate.latitude];
```

Abbildung 9-11 GPS- Sensor auswerten bei iOS

Das Aufnehmen von Bildern mit der Kamera [Req5] und das Auswählen von Bildern aus der Galerie [Req6]

Wie auch bei Android bietet iOS die Möglichkeit ein systemeigenen Dialog zum Aufnehmen von Bildern zu starten. Um das zu ermöglichen ist es notwendig einen "ImagePicker" zu definieren, wie im folgenden Bild in Zeile 25 zusehen. In der Zeile 27 kann nun definiert werden, ob das Bild aufgenommen werden soll oder aus der Galerie ausgewählt werden soll. Ist nun ein Bild aufgenommen oder ausgewählt, wird der Code ab Zeile 33 ausgeführt. Das führt dazu, dass das

Bild nun in der Oberfläche angezeigt wird.

```
25      imagePicker.delegate = self;
26      imagePicker.allowsEditing =NO;
27      imagePicker.sourceType = //UIImagePickerControllerSourceTypePhotoLibrary;
28      UIImagePickerControllerSourceTypeCamera;
29      [self presentViewController:imagePicker animated:YES completion:nil];
30  }
31
32  //Kamerabild darstellen
33  -(void)imagePickerController:(UIImagePickerController *)picker
34  didFinishPickingMediaWithInfo:(NSDictionary *)info
35  {
36      UIImage *image;
37      image =(UIImage *)[info valueForKey:UIImagePickerControllerOriginalImage];
38      self.kameraImage.image = image;
```

Abbildung 9-12 Kamerabild aufnehmen bei iOS

Das Versenden von Bildern mit WLAN [Req7] oder UMTS [Req8]

iOS bietet zwar die Funktion Bilder zu versenden. Es war aber nicht möglich, dieses umzusetzen. Der Grund hierfür ist die Komplexität des Aufrufes in iOS.

Das Versenden von Bildern mit Bluetooth [Req9]

iOS bietet zwar die Möglichkeit auf die Bluetooth- Schnittstelle zuzugreifen, aber der Austausch von Bildern war leider nicht zu realisieren.

Das Auslesen [Req10] und Anzeigen [Req11] der Beschleunigungssensor- Daten

Auch diese Funktionalität bringt iOS mit. Leider war es auch hier nicht möglich die Vorgaben umzusetzen. Der Grund hierfür sind Probleme mit dem Anzeigen der einzelnen Beschleunigungsdaten in einer Tabelle.

9.2.3 Evaluation

Apple bietet für die Entwicklung mit iOS eine sehr guet Dokumentation an. Auch die Installation von XCODE ist problemlos abgelaufen. Was besonders gut gelungen ist, ist der GUI- Builder, der die Navigation von einer Oberfläche zur nächsten sehr gut darstellt. Insgesamt ist die Programmierung mit dem Xcode Framework sehr gut gemacht.

Die nachfolgende Tabelle gibt nun einen genaueren Überblick über die Bewertung dieses Frameworks. Auffällig sind hier die 3 Punkte bei unterstützten Plattformen. Dieses liegt aber nur daran, weil XCODE in diesem Fall nur die Entwicklung von iOS Apps unterstützt.

Tabelle 9-3 Evaluation XCODE Allgemein

	Bewertung / Punkte
Herstellerdokumentation	1
Installationsaufwand	1
GUI- Builder	1
Quellcodeeditor	1
Unterstützte Plattformen	3
GUI- Designe	1

Die folgende Tabelle zeigt wie einfach es ist mit dem XCODE Framewerk bestimmte Ressourcen anzusprechen und wie es beim Test abgeschnitten hat. Bei den ersten 4 Punkten in der Tabelle ist der Quellcode sehr lang, dieses kommt durch die Umsetzung mit Objektiv- C zustande. Bei den letzten drei Punkten ist kein Quellcodeumfang vorhanden, da hier die Implementierung nicht abgeschlossen werden konnte. Eine genauere Erklärung findet sich bei der Implementierung der einzelnen Punkte. Bei der Durchführung der Tests ist aufgefallen, das der Kompass nicht immer gleich in die richtige Richtung zeigt, daher bekam er nur eine 2. Bei den letzten 3 Punkten wurde beim Test auch immer 2 Punkte vergeben. Das liegt daran, da diese Funktionen zwar unterstützt werden, aber nicht funktioniert haben.

Tabelle 9-4 Evaluation XCODE Ressourcenzugriff

	Bewertung	
Ressource	**Quellcodeumfang**	**Durchgeführter Test**
Kompass- Sensor	2	2
GPS- Sensor	3	1

Bild mit der Kamera aufnehmen	2	1
Bild aus der Gallerie auswählen	2	1
Bildupload zum Server	-	2
Bild per Bluetooth versenden	-	2
Beschleunigungssensor	-	2

9.3 Windows Phone SDK

9.3.1 Vorbereitung

Lizenzsierung:

Bei der Entwicklung von Windows Phone Apps ist es notwendig sich für eine Lizenz bei Mircosoft zu registrieren. Das ist notwendig, um mit der Entwicklung überhaupt zu beginnen. Genauer gesagt wird sie benötigt, um ein Smartphone überhaupt als Entwickler Phone zu registrieren und um die Apps in den Store von Microsoft einzustellen.

Installation:

Die Anleitung zur Installation des SDK befindet sich im Anhang im Kapitel 15.3 .

9.3.2 Implementierung

Das Auslesen der Kompass- Daten [Req1]

Um die Daten aus dem Kompass- Sensor auszulesen ist der Quellcode in der folgenden Grafik da. Ab Zeile 41 wird der Sensor initialisiert und in Zeile 43 wird das Event festgelegt was ausgeführt wird, wenn sich die Werte ändern. Im Anschluss daran wird der Kompass in Zeile 47 gestartet. Hat sich nun der Wert des Kompasses geändert, wird der Code ab Zeile 61 ausgeführt, wo dann auch das Auslesen des aktuellen Wertes in Zeile 64 stattfindet. Dieser Wert wird dann gespeichert, so das die Kompassdarstellung gedreht werden kann.

```
41              mCompass = new Compass();
42              mCompass.TimeBetweenUpdates = TimeSpan.FromMilliseconds(20);
43              mCompass.CurrentValueChanged += mCompass_CurrentValueChanged;
44              try
45              {
46                  //Kompass Starten
47                  mCompass.Start();
48                  mTimer.Start();
49              }
50              catch (Exception ex)
51              {
52                  MessageBox.Show("Fehler beim Starten des Kompass.");
53              }
54          }
55          else
56          {
57              MessageBox.Show("Das Gerät unterstützt keinen Kompass");
58          }
59      }
60
61      void mCompass_CurrentValueChanged(object sender, SensorReadingEventArgs<CompassReading> e)
62      {
63          //Drehwinkel berechnen
64          mAngle = -e.SensorReading.MagneticHeading;
```

Abbildung 9-13 Auslesen der Kompass- Daten beim Windows Phone

Das Drehen des Kompassicon [Req2]

Wie im vorherigen Abschnitt ist in der nachfolgenden Grafik die Speicherung der Kompassdaten in einer Variablen in Zeile 64 zusehen. Die letztendliche Drehung des Kompassicons wird nun nach einer bestimmten Zeit mit einem Timerevent ausgeführt. Dieses Event ruft dann den Code ab Zeile 67 auf. Die eigentliche Drehung passiert dann in Zeile 71.

```
61      void mCompass_CurrentValueChanged(object sender, SensorReadingEventArgs<CompassReading> e)
62      {
63          //Drehwinkel berechnen
64          mAngle = -e.SensorReading.MagneticHeading;
65      }
66
67      private void mTimer_Tick(object sender, EventArgs e)
68      {
69          //Kompass drehen
70          rt.Angle = mAngle;
71          imageCompass.RenderTransform = rt;
72      }
```

Abbildung 9-14 Drehen des Kompassicons beim Windows Phone

Das Auslesen [Req3] und Anzeigen [Req4] der GPS- Daten

Beim Auslesen der GPS- Daten ist es bei Visual Studio wichtig den sogenannten Geolocator zu initialisieren. Das ist in der nachfolgenden Grafik in Zeile 31 zusehen. Nach der Initialisierung besteht nun die Möglichkeit die aktuelle Position auszulesen. Hier ist auch noch möglich einige Optionen einzustellen. Das eigentliche Anzeigen der Koordinaten passiert dann in den Zeilen 41 und 42.

```
30          //GPS initialisieren
31          Geolocator geolocator = new Geolocator();
32          geolocator.DesiredAccuracyInMeters = 50;
33
34          try
35          {
36              //Position auslesen
37              Geoposition geoposition = await geolocator.GetGeopositionAsync(
38                  maximumAge: TimeSpan.FromMinutes(5),
39                  timeout: TimeSpan.FromSeconds(10));
40              //Position anzeigen
41              txtlatitude.Text = geoposition.Coordinate.Latitude.ToString();
42              txtlongitude.Text = geoposition.Coordinate.Longitude.ToString();
43          }
44          catch (Exception ex)
45          {
46              MessageBox.Show("Fehler bei lesen der GPS Position."+ex);
47          }
```

Abbildung 9-15 Anzeigen der Koordinaten Daten beim Windows Phone

Das Aufnehmen von Bildern mit der Kamera [Req5]

Das Aufnehmen von Bildern ist bei Visual Studio mit Hilfe eines systemeigenen Dialogfenster möglich. Dazu muss dieses nur initialisiert werden, wie in der nachfolgenden Grafik in ab Zeile 40 zusehen. Nach dem Aufnehmen eines Bildes werden dann die Zeilen 44, 45 und 46 ausgeführt, die dazu dienen, das Bild in der Oberfläche anzuzeigen.

```
39          //Kamera Dialog initalisieren
40          CameraCaptureTask task = new CameraCaptureTask();
41          task.Completed += delegate(object sendern, PhotoResult photoresult)
42          {
43              //Bild anzeigen
44              BitmapImage bitmapimage = new BitmapImage();
45              bitmapimage.SetSource(photoresult.ChosenPhoto);
46              Imagebox.Source = bitmapimage;
47          };
48          //Dialog öffnen
49          task.Show();
```

Abbildung 9-16 Kamerabilder aufnehmen Daten beim Windows Phone

Das Auswählen von Bildern aus der Gallerie [Req6] und das Versenden von Bildern mit WLAN [Req7] oder UMTS [Req8]

Das Auswählen von Bildern verhält sich ähnlich wie das Aufnehmen von Bildern. Denn auch hier wird ein systemeigenes Dialogfester benutzt. Zusehen ist das in der nahfolgenden Grafik. Ab Zeile 33 wird das Dialogfenster initialisiert. Zum Versenden des Bildes werden nun zwei Erweiterungen für Visual Studio benutzt. Das ist zum einen das Json.Net Packet(57) und das RestSharp(58) Packet. Beide Packete können leicht über die Packetverwaltung installiert werden. Die Benutzung

beider Packete erleichtert den Versand des Bildes. Nach der Initialisierung des ResrRequest in Zeile 38 wird das Bild letztendlich mit dem Befahl in Zeile 41 zum Server gesendet.

```
32          //Dialog zum auswählen des Bildes aus der Galerie
33          PhotoChooserTask task = new PhotoChooserTask();
34          task.Completed += delegate(object sendern, PhotoResult photoresult)
35          {
36              //Netzwerk upload initialisieren
37              var client = new RestClient(texturl.Text);
38              var request = new RestRequest("", Method.POST);
39              request.AddFile("media", ReadBytes(photoresult.ChosenPhoto), "image2.png");
40              //upload starten
41              client.ExecuteAsync(request, r =>
```

Abbildung 9-17 Bild an einen Server versenden beim Windows Phone

Des Versenden von Bildern mit Bluetooth [Req9]

Auch hier wird Gebrauch von systemeigenen Dialogen gemacht. Die Funktion, die in der folgenden Grafik in Zeile 27 zusehen ist, wird mit einem Klick auf einen Button gestartet. Innerhalb dieser Funktion wird dann der Dialog zur Auswahl eines Bildes initialisiert und gestartet. Ist ein Bild ausgewählt wird der Dialog zum Versenden von Dateien gestartet. Zusehen ist das ab Zeile 38.

```
27      private void senden_Click(object sender, RoutedEventArgs e)
28      {
29          //Dialog zum auswählen des Bildes aus der Galerie
30          PhotoChooserTask task = new PhotoChooserTask();
31          task.Completed += delegate(object sendern, PhotoResult photoresult)
32          {
33              //Ist ein Foto ausgewählt kann es versendet werden
34              sendImage(photoresult.OriginalFileName);
35          };
36          task.Show();
37      }
38      private void sendImage(string pfad)
39      {
40          //Dialog zum versenden starten
41          ShareMediaTask sharetask = new ShareMediaTask();
42          sharetask.FilePath = pfad;
43          sharetask.Show();
44      }
```

Abbildung 9-18 Bluetooth Bildversand beim Windows Phone

Das Auslesen [Req10] und Anzeigen [Req11] der Beschleunigungssensor- Daten

Um mit Visual Studio auf den Beschleunigungssensor zuzugreifen muss dieser zu erst initialisiert werden. Dieses wird in der nachfolgenden Grafik in den Zeilen 31 bis 33 gemacht. Während der Initialisierung wird auch das Event angegeben, was nach einer bestimmten Zeit aufgerufen wird. Der Start zum Auslesen erfolgt dann in Zeile 34. Das Event ist dann ab Zeile 38 zusehen. In Zeile

40 ist dann das letztendliche Auslesen der Werte zusehen.

```
30          //Sensor initialisieren
31          accelerometer = new Accelerometer();
32          accelerometer.TimeBetweenUpdates = TimeSpan.FromMilliseconds(500);
33          accelerometer.CurrentValueChanged += accelerometer_CurrentValueChanged;
34          accelerometer.Start();
35
36      }
37      //Funktion zum auslesen der Sensorwerte
38      void accelerometer_CurrentValueChanged(object sender, SensorReadingEventArgs<AccelerometerReading> e)
39      {
40          this.Dispatcher.BeginInvoke(new Action<string>(addValue), e.SensorReading.Acceleration.X + " " +
41              e.SensorReading.Acceleration.Y + " " +
42              e.SensorReading.Acceleration.Z);
43      }
```

Abbildung 9-19 Auslesen der Beschleunigungswerte beim Windows Phone

9.3.3 Evaluation

Bevor es möglich ist Apps mit dem Visual Studio zu entwickeln, dauert es sehr lange bis die Installation abgeschlossen ist. Was auch noch wichtig ist bei der Auswahl des Entwicklungsrechner ist, dass dieser in der CPU Hardware- Virtualisierung unterstützt, da sonst der Emulator nicht funktioniert. Ist das Visual Studio dann erfolgreich installiert, ist es nicht schwer eine App zu entwickeln, da in dieser Entwicklungsumgebung ein GUI- Builder und einen Quellcodeeditor enthalten sind. Auch die Dokumentation von Microsoft ist sehr gut aufbereitet. In der nachfolgenden Tabelle ist die Bewertung nochmal im einzelnen aufgeführt. Auffällig ist hier die 3 bei den unterstützten Plattformen. Diese ist aber damit begründet, dass dies ein Native Framework ist und in diesem Umfeld nur eine Plattform unterstützt.

Tabelle 9-5 Evaluation Visual Studio Allgemein

	Bewertung
Herstellerdokumentation	1
Installationsaufwand	2
GUI- Builder	1
Quellcodeeditor	1
Unterstützte Plattformen	3
GUI- Design	1

Die nächste Tabelle beschreibt nun die Bewertung der App. Insbesondere geht es hier um den Zugriff auf die einzelnen Ressourcen. Die meisten Komponenten ließen sich leicht umsetzten. Auffällig ist hier der Kompass-Sensor bei dem der Quellcode eine wenig länger war. Bei den Tests hingegen hat der Kompass- Sensor schlechter abgeschnitten, da hier die Kompassnadel nicht immer in die richtige Richtung gezeigt hat. Ein weiterer auffälliger Punkt ist der Bildupload. Hier wurde der Quellcodeumfang nur mit 3 bewertet, da hier externe Bibliotheken zum Einsatz gekommen sind. Wäre das nicht in die Bewertung mit einbezogen worden, wäre hier eine 1 erreicht worden.

Tabelle 9-6 Evaluation Visual Studio Ressourcenzugriff

	Bewertung	
Ressource	**Quellcodeumfang**	**Durchgeführter Test**
Kompass- Sensor	2	2
GPS- Sensor	1	1
Bild mit der Kamera aufnehmen	1	1
Bild aus der Gallerie auswählen	1	1
Bildupload zum Server	3	1
Bild per Bluetooth versenden	1	1
Beschleunigungssensor	1	1

9.4 Mosync

9.4.1 Vorbereitung

Lizenzsierung:

Es ist keine Mosync eigene Lizenz nötig. Für die unterstützten Plattformen ist aber eine Lizenz erforderlich. Da Mosync die Entwicklung von Android und iOS App unterstützt gelten die Vorbereitungen aus dem Android Kapitel 9.1.1 und dem iOS Kapitel 9.2.1.

Installation:

Die Anleitung zur Installation des SDK befindet sich im Anhang im Kapitel15.4.

9.4.2 Implementierung

Das Auslesen der Kompass- Daten [Req1] und das Drehen des Kompassicon [Req2]

Bei Mosync ist es möglich mit den Zeilen 31 und 32 in der folgenden Grafik ein Event zu erstellen, was nach einer bestimmten Zeit ausgeführt wird.

```
29  // Gerät ist bereit
30  function deviceReady() {
31      var options = { frequency: 2000 }; //Stellt die Aktualisierungsrate des Kompass auf 2 Sek.
32      var watchID = navigator.compass.watchHeading(startCompass, error, options);
33  }
```

Abbildung 9-20 Auslesen der Kompass- Daten bei Mosync

Hat sich der Wert des Kompasses geändert, wird der Code aus dem folgenden Bild ausgeführt. Hier ist nun in Zeile 40 die Berechnung des Winkels für den Kompass zusehen. In der Zeile 42 erfolgt nun die Drehung des Kompassicons.

```
35   // Wird gestatet wenn der Kompass erfolgreich gestartet ist
36  function startCompass(heading) {
37      //Kompass Bild suchen
38      var compassimage = $('#imgCompass');
39      // Winkel des Kompass berechnen
40      var compassheading = 360 - heading.magneticHeading;
41      //Kompass drehen
42      compassimage.css(
43          '-webkit-transform','rotate(' + compassheading + 'deg)'
44      );
45  }
```

Abbildung 9-21 Drehen des Kompassicons bei Mosync

Das Auslesen [Req3] und Anzeigen [Req4] der GPS- Daten

Ähnlich wie bei dem Kompass- Sensor bietet Mosync auch für den GPS- Sensor die Möglichkeit ein Event zu definieren, was nach einer bestimmten Zeit ausgeführt wird. In der nachfolgenden Grafik ist das in der Zeile 32 zusehen. Wird dieses Event nun ausgeführt ist es leicht möglich die Koordinaten auszulesen, wie in denn Zeilen 37 und 38 zusehen.

```
29  // Gerät ist bereit
30  function deviceReady() {
31      var options = { frequency: 2000 };  //Stellt die Aktualisierungsrate des GPS-Sensors auf 2 Sek.
32      navigator.geolocation.getCurrentPosition(gpsready, error,options);
33
34  }
35  // Wird gestatet wenn der GPS- Sensor erfolgreich gestartet ist
36   function gpsready(position) {
37      document.getElementById('latitude').innerHTML = position.coords.latitude;
38      document.getElementById('longitude').innerHTML =position.coords.longitude;
39   };
```

Abbildung 9-22 Auslesen und Anzeigen der GPS- Daten bei Mosync

Das Aufnehmen von Bildern mit der Kamera [Req5]

Wie auch bei Native Frameworks bietet Mosync die Möglichkeit einen Dialog zum Aufnehmen von Fotos zu öffnen, wie in der folgenden Grafik in Zeile 34 zu sehen. Die Funktion in Zeile 38 liest nun das Bild ein und zeigt es mit der Funktion in Zeile 52 an.

```
30   function getImage()
31   {
32        //Starten des Foto auswahlDialoges
33        //limit:1 bedeutet das es nur möglich ist ein Foto auszuwählen
34       navigator.device.capture.captureImage(imagecapture, error, {limit:1});
35    };
36
37    //liest die aufgenomenen Bilder aus
38    function imagecapture(mediaFiles) {
39        var i, path, len;
40        for (i = 0, len = mediaFiles.length; i < 1; i += 1) {
41            path = mediaFiles[i].fullPath;
42            alert('url' + path);
43            showPhoto(path);
44        }
45    };
46    // wird ausgeführt wenn es beim aufnehmen der Bilder Fehler gibt
47    function error(message) {
48      alert('Fehler beim laden des Bildes');
49    };
50
51    //Bringt das aufgenommene Foto zur ansicht
52    function showPhoto(imageData) {
53      var smallImage = document.getElementById('smallImage');
54      smallImage.style.display = 'block';
55      smallImage.src = "" + "file://"+imageData;
56    };
57
```

Abbildung 9-23 Bilder aufnehmen mit Mosync

Das Auswählen von Bildern aus der Galerie [Req6]

Das Auswählen der Bilder aus einer Galerie war mit Mosync in dieser Umsetzung nicht möglich.

Das Versenden von Bildern mit WLAN [Req7] oder UMTS [Req8]

Um mit Mosync ein Bild über das Netzwerk zu versenden benötigt es nur den Pfad zur Bilddatei. Dieser ist aber mit der Bildauswahl über die Galerie verfügbar. Im nächsten Schritt wird nun der Code aus dem folgenden Bild ausgeführt, welcher das Post Request durchführt. Hierzu wird bei Mosync ein Filetransfer initialisiert, wie ab Zeile 53 zusehen. In Zeile 62 wird dieser dann gestartet.

```
51      //Hier wird nun das Foto zum Server Hochgeleden
52      function uploadPhoto(imageURI) {
53          var options = new FileUploadOptions();
54          options.fileKey="media";
55          options.fileName=imageURI.substr(imageURI.lastIndexOf('/')+1);
56          options.fileName="image.png";
57          options.mimeType="image/jpeg";
58          options.chunkedMode = false;
59          options.params = null;
60          var ft = new FileTransfer();
61          alert('url#' + imageURI);
62          ft.upload("file://"+imageURI, encodeURI(document.getElementById('urladdress').value),
63          function(result)
64          {
65              alert("Foto hochgeladen"+ result.responseCode+"#"+result.response+"#"+result.bytesSent);
66          },
67          function(error)
68          {
69              alert("Fehler beim hochladen. Fehlercode: " + error.code);
70          },
71          options,true);
72      }
```

Abbildung 9-24 Bildupload mit Mosync

Das Versenden von Bildern mit Bluetooth [Req9]

Die Umsetzung dieser Anforderung war im aktuellen Umfeld mit Javascript nicht möglich.

Das Auslesen [Req10] und Anzeigen [Req11] der Beschleunigungssensor- Daten

Wie schon bei dem Kompass- Sensor und dem GPS- Sensor gibt es auch für den Beschleunigungssensor ein Event, was definiert werden kann. Dieses wird dann auch nach einer bestimmten Zeit ausgeführt, wie in der folgenden Grafik in der Zeile 42 zusehen. Wird das Event

gestartet ist es nicht schwer die Werte auszulesen und anzuzeigen. In der Zeile 48 ist das zusehen.

```
39      //startet Beschleinigungssensor
40      function startAcceleration() {
41          var options = { frequency: 2000 }; //Stellt die Aktualisierungsrate auf 2 Sek.
42          watchID = navigator.accelerometer.watchAcceleration(accelerometer, error, options);
43      }
44
45      //wird ausgeführt wenn der Beschleunigungssensor gestartet wurde
46      function accelerometer(acceleration) {
47          var element = document.getElementById('accelerometer');
48          var value ='# '+acceleration.x+'#'+acceleration.y+'#'+acceleration.z;
49          addRow(''+value);
50      }
```

Abbildung 9-25 Beschleunigungssensor- Daten auslesen bei Mosync

9.4.3 Evaluation

Auch Mosync ließ sich leicht mit einem Installer installieren. Leider bringt Mosync aber kein GUI-Builder mit, daher muss die GUI im Code erstellt werden. Ein weiterer Punkt ist, dass die GUI für jedes Betriebssystem angepasst werden muss. Mosync unterstützt zwar eine große Anzahl an Betriebssystemen. In diesem Zusammenhang leider Windows Phone nicht, daher gibt es bei den unterstützten Plattformen auch nur eine 2. Alle Bewertungen sind in der Tabelle im Anschluss nochmal zusammengefasst.

Tabelle 9-7 Evaluation Mosync Allgemein

	Bewertung / Punkte
Herstellerdokumentation	1
Installationsaufwand	1
GUI- Builder	3
Quellcodeeditor	1
Unterstützte Plattformen	2
GUI- Designe	2

Bei der Umsetzung der App war es sehr einfach auf die Sensoren zuzugreifen, weshalb sie beim Quellcodeumfang in der folgenden Tabelle mit 1 bewertet wurden. Allerdings ist es beim Testen des Kompass- und GPS- Sensors zu Problemen gekommen, da diese nicht immer die richtigen

Werte angezeigt haben oder auf einer unterstützten Plattform nicht funktionierten. Das ansprechen der Kamera und der Bildupload erforderten etwas mehr Code, daher wurden sie auch nur mit 2 bewertet. Auffällig ist hier das Ansprechen der Galerie und der Bluetooth Bildversand. Dieses wird in dieser Konstellation von Mosync nicht unterstützt, daher gibt es hier auch keine Quellcodebewertung und der Test wurde nur mit 3 bewertet.

Tabelle 9-8 Evaluation Mosync Resourcenzugriff

	Bewertung	
Ressource	**Quellcodeumfang**	**Durchgeführter Test**
Kompass- Sensor	1	2
GPS- Sensor	1	2
Bild mit der Kamera aufnehmen	2	1
Bild aus der Galerie auswählen	-	3
Bildupload zum Server	2	1
Bild per Bluetooth versenden	-	3
Beschleunigungssensor	1	1

9.5 Sencha

9.5.1 Vorbereitung

Lizenzsierung:

Es ist keine Sencha eigene Lizenz nötig. Für die unterstützten Plattformen ist aber eine Lizenz erforderlich. Da Sencha die Entwicklung von Android und iOS App unterstützt gelten in diesem Kapitel die Vorbereitungen aus Android Kapitel 9.1.1 und iOS Kapitel 9.2.1.

Installation:

Die Anleitung zur Installation des SDK befindet sich im Anhang im Kapitel15.8.

9.5.2 Implementierung

Es gibt bei Sencha leider keine Möglichkeit die Sensoren anzusprechen. Eine Möglichkeit es doch noch mit Sencha zu realisieren, wäre die Oberfläche mit Sencha zu gestalten und den Zugriff auf die Sensoren mit Phonegap zu realisieren. Dazu wird dann der Code von Phonegap in Sencha eingebunden. Dies wurde aber an dieser Stelle nicht vorgenommen, da es hier um die einzelnen Frameworks geht und nicht um eine Kombination.

9.5.3 Evaluation

Da die Umsetzung nicht möglich war, wird der Code an dieser Stelle nicht bewertet sondern nur die allgemeinen Punkte.

Die für Sencha verwendete Oberfläche zur Entwicklung ist der Sencha Architect der sowohl einen Quellcodeeditor als auch einen GUI- Builder mit sich bringt. Auch die Installation lief problemlos ab. Die erstellte Oberfläche muss aber an die Oberfläche jedes Betriebssystem angepasst werden.
Die einzelnen Bewertungen sind in der nachfolgenden Tabelle zusehen.

Tabelle 9-9 Evaluation Sencha Allgemein

	Bewertung / Punkte
Herstellerdokumentation	1
Installationsaufwand	1
GUI- Builder	1
Quellcodeeditor	1
Unterstützte Plattformen	2
GUI- Design	2

9.6 Corona

9.6.1 Vorbereitung

Lizenzsierung:

Es ist keine Corona eigene Lizenz nötig. Für die unterstützten Plattformen ist aber eine Lizenz

erforderlich. Da Corona die Entwicklung von Android und iOS App unterstützt gelten in diesem Kapitel die Vorbereitungen aus Android Kapitel 9.1.1 und iOS Kapitel 9.2.1.

Installation:

Die Anleitung zur Installation des SDK befindet sich im Anhang im Kapitel15.5.

9.6.2 Implementierung

Das Auslesen der Kompass- Daten [Req1]

Auch Corona bietet die Möglichkeit ein Event für den Kompass- Sensor zu definieren. Dieser wird wie in der folgenden Grafik hinzugefügt, siehe Zeile 47.

```
44      -- Start Event zum einstellen des Kompass hinzu fügen
45      Runtime:addEventListener( "location", tInitNorth )
46      -- Kompass event zum abfragen der Daten hinzufügen
47      Runtime:addEventListener( "heading", updateCompass )
48      -- Kompass animation event hinzufügen
49      Runtime:addEventListener( "enterFrame", animateCompassImage )
```

Abbildung 9-26 Event zum Abfragen der Kompass- Daten bei Corona

Wird das Event ausgelöst ist es sehr einfach die Werte auszulesen, wie im folgenden Bild zu sehen.

```
 8  local updateCompass = function( event )
 9      -- Android unterstützt den geographischen Kompass nicht daher t
10      -- der magnetisch Kompass genutzt
11      if system.getInfo( "platformName" ) ~= "Android" then
12              mDestinationAngel = -event.geographic
13      else
14              mDestinationAngel = -event.magnetic
15      end
16  end
```

Abbildung 9-27 Kompass- Daten auslesen bei Corona

Zu beachten ist aber, das einmal am Anfang eine Kalibrierung vorgenommen werden muss, wie im folgenden Bild zusehen.

```
32      -- Start event zum einstellen des Kompass auf den Norden
33      local mCalibrationNorth = false
34      local tInitNorth = function( event )
35          if ( not mCalibrationNorth ) then
36              mCalibrationNorth = true
37              local tInitNorthFinish = function( event )
38                  Runtime:removeEventListener( "location", tInitNorth )
39              end
40              timer.performWithDelay( 1000, tInitNorthFinish )
41          end
42      end
```

Abbildung 9-28 Kalibrierung des Kompasssensors bei Corona

Das Drehen des Kompassicon [Req2]

Zum Drehen des Kompassicons wird bei Corona einfach der Unterschied vom aktuellen Winkel zum Zielwinkel gebildet, wie in der folgenden Grafik zusehen. Im Anschluss daran wird der Kompass nun gedreht.

```
18      local animateCompassImage = function( event )
19          local mCurrentAngel = mCompassImage.rotation
20          local tDeltaAngel = mDestinationAngel - mCurrentAngel
```

Abbildung 9-29 drehen des Kompassicons bei Corona

Das Auslesen [Req3] und Anzeigen [Req4] der GPS- Daten

Beim GPS- Sensor wird wieder ein Event definiert was auf Änderung des Sensors reagiert. Dazu wird es wie im folgenden Bild in Zeile 74 hinzugefügt. Wird nun das Event ausgelöst, wird der Code ab Zeile 62 ausgeführt. Ist kein Fehler aufgetreten, können die Koordinaten in Zeile 68 und 69 ausgelesen werden.

```
60
61      -- Event zum abfragen der Koordinaten erstellen
62      local mGetGps = function( event )
63          native.showAlert("gps event", event.errorMessage, {"OK"})
64          if event.errorCode then
65              native.showAlert( "Fehler bei abfragen der GPS Position",
66              event.errorMessage, {"OK"} )
67          else
68              txtLatitudet.text = string.format( '%.4f', event.latitude )
69              txtLongitude.text = string.format( '%.4f', event.longitude )
70
71          end
72      end
73      -- Event zum abfragen der Koordinaten aktivieren
74      Runtime:addEventListener( "location", mGetGps )
```

Abbildung 9-30 GPS- Daten auslesen und anzeigen bei Corona

Das Aufnehmen von Bildern mit der Kamera [Req5] und das Auswählen von Bildern aus der Galerie [Req6]

Sowie andere Frameworks bietet auch Corona die Möglichkeit einen Dialog zum Aufnehmen oder zur Auswahl von Bildern zu öffnen. Um das zu erreichen wird der Code im folgenden Bild ausgeführt. Als aller erstes wird hier überprüft, ob der iOS Simulator gestartet wurden, das ist wichtig da hier der Dialog nicht funktioniert. Anschließend wird definiert, das bei erfolgreich ausgewählt oder aufgenommenen Bildern, dieses angezeigt wird, siehe dazu Zeile 56 bis 61. Nun wird noch überprüft, ob das Gerät eine Kamera hat. Wenn ja, startet der Aufnahmedialog wie in Zeile 64 zusehen. Um nicht den Aufnahmedialog zu erhalten sondern den Galerieauswahldialog ist es nur nötig in Zeile 64 "media.Camera" durch " media.PhotoLibrary" zu ersetzten.

```
47
48      local buttonAufnahmeRelease = function( event )
49          -- Die Kamera wird im Simulator nicht unterstützt
50          local isXcodeSimulator = "iPhone Simulator" == system.getInfo("model")
51          if (isXcodeSimulator) then
52              local alert = native.showAlert( "Information",
53              "Die Kamera wird im Simulator nicht unterst\195\188tzt.", { "OK"})
54          end
55
56          local onComplete = function(event)
57          local photo = event.target
58          photo.x = 150
59          photo.y = 150
60              --print( "Foto w,h = " .. photo.width .. "," .. photo.height )
61          end
62
63          if media.hasSource( media.Camera ) then
64             media.show( media.Camera, onComplete )
65          else
66              local isAndroid = "Android" == system.getInfo("platformName")
67              if(isAndroid) then
68                  local alert = native.showAlert( "Information",
69                  "Die Kamera unter Android wird nicht unterst\195\188tzt", { "OK"})
70              else
71              native.showAlert( "Information", "Dieses Ger\195\164t hat keine Kamera", { "OK" } )
72              end
73          end
74      end
```

Abbildung 9-31 Kamerabild aufnehmen bei Corona

Das Versenden von Bildern mit WLAN [Req7] oder UMTS [Req8]

Um nun ein Foto an einen Server zu versenden ist es nur nötig das Foto auszuwählen, wie in dem vorherigen Punkt. Anschließend wird der Code im folgenden Bild benötigt. Hier wird nun der Post Request zusammen gesetzt und in Zeile 54 ausgeführt.

```
47          local multipart = MultipartFormData.new()
48          multipart:addFile("media", system.pathForFile( "testimage.jpg",
49          system.ResourceDirectory  ), "image/jpeg", "media.jpg")
50
51          local params = {}
52          params.body = multipart:getBody()
53          params.headers = multipart:getHeaders()
54          network.request(textfield.text, "POST", networkListener, params)
```

Abbildung 9-32 Bildupload zu einem Server bei Corona

Das Versenden von Bildern mit Bluetooth [Req9]

Wird von diesem Framework nicht unterstützt.

Das Auslesen [Req10] und Anzeigen [Req11] der Beschleunigungssensor- Daten

Auch hier bietet Corona die Möglichkeit ein Event anzulegen, welches ausgelöst wird, wenn sich die Daten beim Sensor ändern. Im nachfolgenden Bild ist in Zeile 59 das Festlegen des Events zu erkennen. Ab der Zeile 51 ist nun zusehen, was passiert, wenn das Event ausgelöst wird. Hier ist es dann sehr einfach die einzelnen Werte auszulesen.

```
50      -- Funktion zum auslesen der Sensorwerte
51      local function getAccelorometer(event)
52      list:insertRow{
53          id = "x" .. event.xGravity .. " y" .. event.yGravity .. " z" .. event.zGravity,
54          onRender = onRowRender
55      }
56      end
57
58      -- Event Anmelden
59      Runtime:addEventListener("accelerometer", getAccelorometer)
60
```

Abbildung 9-33 Auslesen und anzeigen der Beschleunigungssensor- Daten bei Corona

9.6.3 Evaluation

Die Entwicklung mit Corona ist nicht so unkompliziert, da der Kern von Corona ein Simulator ist, der die Apps auf dem Rechner simulieren kann. Der große Nachteil besteht darin, dass es kein GUI- Builder gibt sowie keinen eigenen Quellcodeeditor. Beim Quellcodeeditor gibt es aber die Möglichkeit auf externe zurückzugreifen. Ein weiterer wichtiger Kritikpunkt ist, dass die Oberfläche für jedes Betriebssystem angepasst werden muss. Daher gab es beim GUI- Design in der folgenden Tabelle auch nur eine 2.

Tabelle 9-10 Evaluation Corona Allgemein

	Bewertung / Punkte
Herstellerdokumentation	1
Installationsaufwand	1
GUI- Builder	3
Quellcodeeditor	2
Unterstützte Plattformen	2
GUI- Design	2

Bei der Bewertung der Umsetzung sowie dem Zugriff auf die einzelnen Ressourcen ist die folgende Tabelle entstanden. Was am auffällgsten ist, ist das Corona keine Unterstützung für

Bluetooth mitbringt, weshalb hier der Codeumfang nicht gewertet werden konnte. Im Gegensatz dazu war die Unterstützung des Kompass-, GPS- und Beschleunigungssensors sehr gut. Lediglich der Kompass- Sensor hat etwas mehr Quellcode, was aber daran liegt, das beim ersten Start eine Kalibrierung vorgenommen wird. Bei den Tests ist auffällig, dass es hier viele Bewertungen mit 2 gibt. Das liegt daran, dass es zwar vom Framework die Unterstützung dafür gibt, aber bei der Implementierung gab es Probleme das umzusetzen.

Tabelle 9-11 Evaluation Corona Ressourcenzugriff

	Bewertung	
Ressource	**Quellcodeumfang**	**Durchgeführter Test**
Kompass- Sensor	3	1
GPS- Sensor	1	2
Bild mit der Kamera aufnehmen	3	2
Bild aus der Gallerie auswählen	3	2
Bildupload zum Server	3	2
Bild per Bluetooth versenden	-	3
Beschleunigungssensor	1	1

9.7 Phonegap

9.7.1 Vorbereitung

Lizenzsierung:

Es ist keine Phonegap eigene Lizenz nötig. Für die unterstützten Plattformen ist aber eine Lizenz erforderlich. Da Phonegap die Entwicklung von Android , Windows Phone und iOS App unterstützt gelten in diesem Kapitel die Vorbereitungen aus Android Kapitel 9.1.1, Windows Phone 9.3 und iOS Kapitel 9.2.1.

Installation:

Die Anleitung zur Installation des SDK befindet sich im Anhang im Kapitel15.6.

9.7.2 Implementierung

Das Auslesen der Kompass- Daten [Req1]und das Drehen des Kompassicon [Req2]

Auch bei Phonegap gibt es wieder die Möglichkeit ein Event für das Auslesen der Kompass- Daten zu erstellen. Im nachfolgenden Bild ist das in Zeile 22 zu sehen. In der Zeile 21 wird noch die Häufigkeit des Auslesens festgelegt. Ist das Event eingetreten, werden die Daten dann in Zeile 29 ausgelesen und der Winkel berechnet. Ab Zeile 30 folgt dann die Drehung des Kompassicons.

```
19  // PhoneGap ist bereit
20  function onDeviceReady() {
21      var options = { frequency: 2000 };
22      var watchID = navigator.compass.watchHeading(onStartSuccess, onError, options);
23
24  }
25
26   // onStartSuccess
27  function onStartSuccess(heading) {
28      var arrow = $('#imgCompass');
29      var arrowOrientation = 360 - heading.magneticHeading;
30      arrow.css(
31          '-webkit-transform','rotate(' + arrowOrientation + 'deg)'
32      );
33  }
```

Abbildung 9-34 Kompass- Sensor auswerten bei Phonegap

Das Auslesen [Req3] und Anzeigen [Req4] der GPS- Daten

Wie beim Kompasssensor gibt es auch beim GPS- Sensor ein Event was eingestellt werden kann, um die Daten auszulesen. In der nachfolgenden Grafik ist das in der Zeile 20 zusehen. Ab der Zeile 23 ist dann zusehen was passiert, wenn das Event eingetreten ist. Das Auslesen der Koordinaten erfolgt dann in den Zeilen 24 und 25.

```
17  // PhoneGap ist bereit
18  function onDeviceReady() {
19      var options = { frequency: 2000 };
20      navigator.geolocation.getCurrentPosition(onSuccess, onError,options);
21
22  }
23   var onSuccess = function(position) {
24      document.getElementById('latitude').innerHTML = "'"+position.coords.latitude;
25      document.getElementById('longitude').innerHTML =position.coords.longitude;
```

Abbildung 9-35 GPS- Sensor auswerten bei Phonegap

Das Aufnehmen von Bildern mit der Kamera [Req5]

Um ein Bild mit Phonegap aufzunehmen ist der Quellcode aus der folgenden Grafik auszuführen. Zuerst wird mit dem Aufruf in Zeile 25 die Aufnahme gestartet. Anschließend wird die Funktion aus Zeile 28 aufgerufen, welche das Bild einliest und zur Anzeige bringt.

```
25          navigator.camera.getPicture(onImage, onFail, { quality: 50 });
26      }
27       //Bild anzeigen
28       function onImage(imageData) {
29          var image = document.getElementById('image');
30          image.style.display = 'block';
31          image.src = "" + imageData;
32      }
```

Abbildung 9-36 Bild aufnehmen mit Phonegap

Das Auswählen von Bildern aus der Gallerie [Req6]

Das Auswählen eines Bildes aus der Galerie ist ähnlich wie bei der Bildaufnahme. Der einzige Unterschied ist, das bevor der Aufruf in Zeile 27 gemacht wird, noch die Datenquelle geändert wird. Zusehen ist das in der nachfolgenden Grafik in der Zeile 26.

```
24      //Bildergalerie öffnen
25       function selectImage() {
26       Camera.sourceType = Camera.PictureSourceType.PHOTOLIBRARY
27        navigator.camera.getPicture(onImageSuccess, onImageFail, { quality: 50,
28           destinationType: Camera.DestinationType.FILE_URI,
29           sourceType: navigator.camera.PictureSourceType.PHOTOLIBRARY     });
30      }
```

Abbildung 9-37 Bild aus der Galerie auswählen bei Phonegap

Das Versenden von Bildern mit WLAN [Req7] oder UMTS [Req8]

Zum Hochladen einer Datei oder in diesem Zusammenhang eines Bildes bietet Phonegap einige Funktionen an. In der nachfolgenden Grafik ist die erste wichtige in Zeile 38 zusehen. Diese initialisiert den Request mit Parametern. Sind alle Parameter eingestellt wird in Zeile 44 der Dateitransfer initialisiert und in Zeile 46 gestartet.

```
31      // Bild an einen Server Versenden
32      function onImageSuccess(imageURI) {
33          var server = document.getElementById('urlstring').value;
34          if(!server){
35              alert('Keine Serveradresse vorhanden.');
36              return;
37          }
38          var uploadOptions = new FileUploadOptions();
39          uploadOptions.fileKey="media";
40          uploadOptions.fileName=imageURI.substr(imageURI.lastIndexOf('/')+1);
41          uploadOptions.mimeType="image/png";
42          uploadOptions.chunkedMode = false;
43
44          var fileTransfer = new FileTransfer();
45
46          fileTransfer.upload(imageURI, server, function(r) {
```

Abbildung 9-38 Bild zum Server senden bei Phonegap

Das Versenden von Bildern mit Bluetooth [Req9]

Bluetooth wird von Phonegap nicht unterstützt, kann aber durch eigene Erweiterungen eingebaut werden. Da das aber nicht mehr zum Framework gehört wurde an dieser Stelle darauf verzichtet.

Das Auslesen [Req10] und Anzeigen [Req11] der Beschleunigungssensor- Daten

Wie beim Kompasssensor und GPS- Sensor gibt es auch beim Beschleunigungssensor ein Event was eingestellt werden kann, um die Daten auszulesen. In der nachfolgenden Grafik ist das in der Zeile 22 zusehen. Ab der Zeile 25 ist dann zusehen was passiert, wenn das Event eingetreten ist. Das Auslesen der Beschleunigungswerte erfolgt dann in der Zeile 27.

```
19      //Auslesen des Beschleunigungssensors starten
20      function startWatch() {
21          var options = { frequency: 2000 };
22          watchID = navigator.accelerometer.watchAcceleration(onSuccess, onError, options);
23      }
24      //Werte auslesen und anzeigen
25      function onSuccess(acceleration) {
26          var element = document.getElementById('accelerometer');
27          var action ='# '+acceleration.x+'#'+acceleration.y+'#'+acceleration.z;
28          addRow(''+action);
29      }
```

Abbildung 9-39 Beschleunigungssensor auswerten bei Phonegap

9.7.3 Evaluation

Bevor es möglich ist mit Phonegap zu arbeiten ist es nötig ein Projekt zu erstellen. Dieses fügt sich

dann in die einzelnen Entwicklungsumgebungen ein. Das bedeutet, das für das Android SDK, XCODE und für Visual Studio ein eigenes Projekt erstellt werden muss. Ist das erledigt, kann das Phonegap Projekt in den jeweiligen Native Frameworks bearbeitet und auch erstellt werden. Durch diesen Aufwand bei der Erstellung eines Projektes wurde die Installation auch nur mit 3 bewertet, wie die nachfolgende Tabelle zeigt. Weitere kritische Punkte sind das Fehlen von eigenen GUI-Buildern so wie das Fehlen eines eigenen Quellcodeeditor.

Wichtig ist aber hervorzuheben, das Phongap alle geforderten Betriebssysteme unterstützt.

Tabelle 9-12 Evaluation Phonegap Allgemein

	Bewertung / Punkte
Herstellerdokumentation	1
Installationsaufwand	3
GUI- Builder	3
Quellcodeeditor	2
Unterstützte Plattformen	1
GUI- Designe	2

Das Ansprechen der Sensoren geht bei Phonegap auch mit einem sehr geringen Aufwand, was die folgenden Tabelle zeigt. Auffällig ist hier nur der etwas höhere Aufwand beim ansprechen der Kamera, der Bildergalerie und des Netzwerkes und dem damit verbundenen Bildupload. Die Bluetooth- Schnittstelle konnte nicht getestet werden, da hier für eine extra Erweiterung programmiert werden müsste und das nicht mehr zum eigentlichen Framework gehört. Das zeigt sich auch beim Test, wo hier eine 2 vergeben wurde, da es zwar möglich ist, aber nicht zum Framework gehört. Beim Kompass- und GPS- Sensor wurde beim Test auch eine 2 vergeben, da das Auslesen dieser Sensoren nicht immer bei allen Frameworks funktioniert hat.

Tabelle 9-13 Evaluation Phonegap Ressourcenzugriff

	Bewertung	
Ressource	**Quellcodeumfang**	**Durchgeführter Test**
Kompass- Sensor	1	2

GPS- Sensor	1	2
Bild mit der Kamera aufnehmen	2	1
Bild aus der Galerie auswählen	2	1
Bildupload zum Server	2	1
Bild per Bluetooth versenden	-	2
Beschleunigungssensor	1	1

9.8 Titanium

9.8.1 Vorbereitung

Lizenzsierung:

Es ist keine Titanium eigene Lizenz nötig. Für die unterstützten Plattformen ist aber eine Lizenz erforderlich. Da Titanium die Entwicklung von Android und iOS App unterstützt gelten in diesem Kapitel die Vorbereitungen aus Android Kapitel 9.1.1 und iOS Kapitel 9.2.1.

Installation:

Die Anleitung zur Installation des SDK befindet sich im Anhang im Kapitel15.7.

9.8.2 Implementierung

Das Auslesen der Kompass- Daten [Req1] und das Drehen des Kompassicon [Req2]

Wie andere Frameworks auch, bietet Titanium die Möglichkeit für den Kompass- Sensor einen Eventlistner anzulegen. Das ist in der nachfolgenden Grafik in der Zeile 39 zusehen. In den nachfolgenden Zeilen wird noch überprüft, ob das Gerät einen Kompass- Sensor hat. Wenn ja, wird dieser dann initialisiert. Im Anschluss daran wird dann in der Zeile 49 noch festgelegt was passieren soll, wenn neue Kompasswerte kommen. In den Zeilen 58 bis 63 wird dann der Drehwinkel berechnet und das Kompassicon gedreht.

```
38  //Eventlistner der beim Start ausgelöst wird
39  self.addEventListener('open',function(e)
40  {
41      //Prüfen ob eine Kompass vorhanden ist
42      if (Titanium.Geolocation.hasCompass)
43      {
44          //Kompass initialisieren
45          Titanium.Geolocation.showCalibration = false;
46          Titanium.Geolocation.headingFilter = 90;
47
48          //Kompass auslesen
49          Ti.Geolocation.getCurrentHeading(function(e)
50          {
51              //Fehler abfangen
52              if (e.error)
53              {
54                  currentHeading.text = 'error: ' + e.error;
55                  return;
56              }
57              //Wert in einen Winkel umrechnen und den Kompass drehen
58              var tmatrix = Ti.UI.create2DMatrix();
59              tmatrix = tmatrix.rotate(360-e.heading.magneticHeading);
60              var  tanimation= Ti.UI.createAnimation();
61              tanimation.transform = tmatrix;
62              tanimation.duration = 1;
63              imageView.animate(tanimation);
64
65          });
```

Abbildung 9-40 Kompasssensorauswertung bei Titanium

Das Auslesen [Req3] und Anzeigen [Req4] der GPS- Daten

Zum Auslesen des GPS- Sensor bietet Titanium eine einfache Funktion an, wie in dem nachfolgenden Bild in Zeile 75 zusehen. Tritt kein Fehler beim Auslesen auf, können die Koordinaten wie in den Zeilen 83 und 84 ausgelesen werden.

```
74          //Auslesen des GPS-Sensor starten
75          Titanium.Geolocation.getCurrentPosition(function(e)
76          {
77              //Fehler abfangen
78              if (e.error)
79              {
80                  return;
81              }
82              //GPS Koordinaten anzeigen
83              lblLongvalue.text = e.coords.longitude;
84              lblLatvalue.text = e.coords.latitude;
85          });
```

Abbildung 9-41 GPS- Daten auslesen und anzeigen bei Titanium

Das Aufnehmen von Bildern mit der Kamera [Req5]

Um eine Bild mit der Kamera aufzunehmen bietet Titanium ein Dialogfenster an was gestartet werden kann. In der nachfolgenden Grafik ist die Definition des Dialogfensters in Zeile 43 zusehen. Ist ein Bild erfolgreich aufgenommen, wird es mit dem Befehl in Zeile 49 in der Oberfläche angezeigt. Tritt ein Fehler auf, wird darauf in den Zeilen 51 bis 56 reagiert.

```
42          //Kamera Aufnahmedialog anzeigen
43          Titanium.Media.showCamera({
44              //aufgenommenes Bild anzeigen
45              success:function(event)
46              {
47                  var cropRect = event.cropRect;
48                  var image = event.media;
49                 imageView.image=image;
50              },
51              cancel:function()
52              {
53              },
54              //Fehler abfangen
55              error:function(error)
56              {
```

Abbildung 9-42 Kamerabilder aufnehmen mit Titanium

Das Auswählen von Bildern aus der Gallerie [Req6] und das Versenden von Bildern mit WLAN [Req7] oder UMTS [Req8]

Das Auswählen eines Fotos aus der systemeigenen Galerie ist bei Titanium leicht möglich mit den Zeilen 40 bis 44 in der folgenden Grafik. In Zeile 47 erfolgt dann der Versand des Bildes. Zum Versand wird als erstes ein Client definiert und initialisiert. Anschließend wird alles mit dem Befehl aus Zeile 51 an den Server gesendet.

```
40          Titanium.Media.openPhotoGallery({
41              mediaTypes:[Ti.Media.MEDIA_TYPE_PHOTO],
42
43              success:function(event) {
44                  var image = event.media;
45                  Titanium.API.info('image url: ' + image.nativePath);
46
47                  var httpclient = Ti.Network.createHTTPClient ( );
48                  httpclient.open ( "POST", textField.value );
49
50                  httpclient.setTimeout ( 20000 );
51                  httpclient.send (
52                  {
53                      "CommandType"    : "media",
54                      "file"           : event.media,
55                      "name"           : "image1.png"
56                  });
```

Abbildung 9-43 Bild an einen Server senden mit Titanium

Das Versenden von Bildern mit Bluetooth [Req9]

Nur mit kostenpflichtiger Erweiterung möglich.

Das Auslesen [Req10] und Anzeigen [Req11] der Beschleunigungssensor- Daten

Um die Werte des Beschleunigungssensor auszulesen ist es bei Titanium möglich ein Event zu definieren, wie in der nachfolgenden Grafik in der Zeile 51 zusehen. Tritt nun das Event ein, werden die Daten wie in der Zeile 39 ausgelesen.

```
36      //Auslesen der Beschleunigungswerte und eintragen in eine Tabelle
37      var accelerometerCallback = function(e) {
38         var row = Ti.UI.createTableViewRow({
39             title:'x: ' + e.x +' y: ' + e.y+' z: ' + e.z
40          });
41        tableView.appendRow(row);
42      };
43
44      //Biginne mit dem auslesen des Beschleunigungssensordaten
45      buttonGetDeviceRefresh.addEventListener('click',function(e)
46      {
47          if (Ti.Platform.model === 'Simulator' || Ti.Platform.model.indexOf('sdk') !== -1 ){
48           alert('Der Beschleunigungssensor funktioniert nicht im Simulator');
49          } else {
50          //Eventlistner zum Auslesen hinzufügen
51           Ti.Accelerometer.addEventListener('update', accelerometerCallback);
```

Abbildung 9-44 Auslesen und Anzeigen der Beschleunigungswerte bei Titanium

9.8.3 Evaluation

Die Installation des Titanium Frameworks war mit dem mitgelieferten Installer problemlos möglich, daher wurde die Installation auch mit 1 bewertet, wie die nachfolgende Tabelle zeigt. Die Entwicklung einer App mit Titanium ist etwas umständlich, da es zwar einen Quellcodeeditor gibt, aber keinen GUI- Builder. Der Grund hierfür ist aber, das Titanium die programmierte App am Ende an die jeweilige Betriebssystemoberfläche anpasst. Das bedeutet, dasa die Oberfläche bei Android aussieht wie eine typische Android App und bei iOS sieht sie so aus, als ob sie mit XCODE erstellt wurde. In der Praxis war das leider nicht so. Hier sah die Oberfläche zwar dem systemeigenen ähnlich, auffällig war aber die unterschiedliche Skalierung der Button in der Oberfläche.

Tabelle 9-14 Evaluation Titanium Allgemein

	Bewertung / Punkte
Herstellerdokumentation	1
Installationsaufwand	1
GUI- Builder	3
Quellcodeeditor	1
Unterstützte Plattformen	2
GUI- Design	2

Bei der Implementierung war es ein leichtes auf die Sensoren zuzugreifen, daher ist hier der Codeumfang in der folgenden Tabelle auch mit 1 bewertet. Etwas umständlicher war das mit dem Aufnehmen oder Auswählen von Bildern, daher gab es hier auch nur eine 2. Da Bluetooth nur über kostenpflichtige Erweiterungen zu realisieren war, wurde das an dieser Stelle nicht umgesetzt. Da es aber möglich ist, gibt es beim Test eine 2. Beim Versand von Bildern zum Server gab es und gibt es Probleme, daher gab es auch hier nur eine 2. Der Kompass hat auch nicht immer richtig funktioniert, daher gibt es auch hier beim Test nur eine 2.

Tabelle 9-15 Evaluation Titanium Ressourcenzugriff

	Bewertung	
Ressource	**Quellcodeumfang**	**Durchgeführter Test**

Kompass- Sensor	1	2
GPS- Sensor	1	1
Bild mit der Kamera aufnehmen	2	1
Bild aus der Gallerie auswählen	2	1
Bildupload zum Server	3	2
Bild per Bluetooth versenden	-	2
Beschleunigungssensor	1	1

10 Auswertung der Frameworks

Hier folgt nun ein genauerer Überblick über die gesamten Frameworks. Sprich welches Framework hat welchen Teil der Testanwendung umgesetzt und wie gut war die Entwicklungsumgebung. Hierzu werden die einzelnen Evaluationen aus dem Kapitel 9, die zu jedem Framework gemacht wurden, zu Rate gezogen.

In den folgenden Beschreibungen ist Sencha der Vollständigkeit noch mit aufgeführt, da es aber nicht möglich war mit diesem Framework die Testapp umzusetzen, wird es bei den Zusammenfassungen nicht berücksichtigt.

10.1 Entwicklungsumgebung allgemein

In der Tabelle im Anschluss findet sich die Übersicht der einzelnen Bewertungen zur Entwicklungsumgebung der Frameworks. Es ist zusehen, dass die besten Entwicklungsumgebungen (abgesehen von Sencha) die Nativen Entwicklungsumgebungen sind. Das liegt daran, das sie speziell zugeschnitten sind und die Cross Frameworks nicht. Anders betrachtet liegen die Cross Frameworks Mosync und Titanium nicht weit abgeschlagen.

Tabelle 10-1 Bewertung der Entwicklungsumgebung aller Frameworks

Framework	Dokumentation	Installation	GUI- Builder	Quellcode-editor	Gesamt
Mosync	1	1	3	1	1,5
Sencha	1	1	1	1	1
Phonegap	1	3	3	2	2,3
Corona	1	1	3	2	1,8
Titanium	1	1	3	1	1,5
Android SDK	2	1	1	1	1,25
Xcode	1	1	1	1	1
Visual	1	2	1	1	1,25

Studio

10.2 Unterstützte Plattformen

Bei dem Unterstützen der 3 Testplattformen liegt Phonegab klar vorne, da es alle 3 unterstützt. Die anderen Cross Frameworks unterstützen in diesem Test nur die Betriebssysteme iOS und Android, was wahrscheinlich daher kommt, dass dies die beiden am weitesten verbreiteten Betriebssysteme sind. Die Native Frameworks unterstützen hier nur die Entwicklung für das eigene Betriebssystem.

Tabelle 10-2 Betriebssystemunterstützung der Frameworks

Framework	**iOS**	**Android**	**WP8**	**Gesamt unterstützte Frameworks**
Mosync	X	X		2
Sencha	X	X		2
Phonegap	X	X	X	3
Corona	X	X		2
Titanium	X	X		2
Android SDK		X		1
Xcode	X			1
Visual Studio			X	1

10.3 Vergleich des Oberflächendesigns

Die nachfolgenden Tabelle zeigt den Vergleich der Oberflächen zwischen Native- und Cross-Frameworks. Hier ist ganz klar zusehen das nur die Native Frameworks von Anfang an die Oberfläche der App so gestalten können, dass sie zum Gerät passt. Bei den Cross Frameworks ist das zwar auch möglich, nur muss hier die Oberfläche für jedes Gerät einzeln angepasst werden. Eine Ausnahme stellt hier Titanium da, wo versucht wird die Oberfläche auch nur einmal zu programmieren und dann für jedes unterstützte Betriebssystem zu generieren. Das funktioniert,

wie oben bei Titanium schon beschrieben, in der Praxis nicht ganz.

Tabelle 10-3 Vergleich der Oberflächendesigns

Framework	Bewertung
Mosync	2
Sencha	2
Phonegap	2
Corona	2
Titanium	2
Android SDK	1
Xcode	1
Visual Studio	1

10.4 Gegenüberstellung der Testergebnisse der Frameworks

Bei den Testergebnissen ist zusehen, dass die beiden Cross Frameworks Phonegap und Titanium mit dem Ergebnissen der Native Frameworks mithalten können. Sie übertreffen sogar das Native Framework XCODE. Der Grund hierfür ist, dass bei XCODE nicht alle möglichen Funktionen in Code umgesetzt werden konnten. Das bedeutet, dass bei dieser Gegenüberstellung die Cross Frameworks mit den Native Frameworks konkurrieren können.

Tabelle 10-4 Vergleich der Testergebnisse der Frameworks

Framework	Kompass	GPS	Bild Aufnahme	Galerie zugriff	Netzwerk	Bluetooth	Beschleunigung	Ø
Mosync	**2**	**2**	**1**	**3**	**1**	**3**	**1**	**1,9**
Sencha	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
Phonegap	**2**	**2**	**1**	**1**	**1**	**2**	**1**	**1,4**

Corona	**1**	**2**	**2**	**2**	**2**	**3**	**1**	**1,9**
Titanium	**2**	**1**	**1**	**1**	**2**	**2**	**1**	**1,4**
Android SDK	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**1,1**
Xcode	**2**	**1**	**1**	**1**	**2**	**2**	**2**	**1,6**
Visual Studio	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**1,1**

10.5 Quellcodeumfang

Bei dem Vergleich der Codezeilen, die nötig waren, um die Systemressourcen anzusprechen, zeigt die nachfolgende Tabelle, dass die Cross Frameworks häufig besser sind als die Native Frameworks. Das ist besonders deutlich beim Zugriff auf die Kompass- Daten zusehen. Hier sind fast alle Cross Frameworks vor den Native Frameworks. Beachtet werden muss hier allerdings das die Cross Frameworks meist nur den Winkel des Kompasses auslesen können, wo hingegen die Native Frameworks meist einen tiefgehenderen Zugriff auf den Kompass- Sensor haben.

Tabelle 10-5 Vergleich des Quellcodeumfangs beim Zugriff auf die Ressourcen

Framework	Kompass	GPS	Bild Aufnahme	Galerie zugriff	Netzwerk	Bluetooth	Beschleunigung	Ø
Mosync	**1**	**1**	**2**	**-**	**2**	**-**	**1**	**1,4**
Sencha	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
Phonegap	**1**	**1**	**2**	**2**	**2**	**-**	**1**	**1,5**
Corona	**3**	**1**	**3**	**3**	**3**	**-**	**1**	**2,3**
Titanium	**1**	**1**	**2**	**2**	**3**	**-**	**1**	**1,6**
Android SDK	**3**	**1**	**3**	**3**	**3**	**3**	**1**	**2,4**

Xcode	**2**	**3**	**2**	**2**	**-**	**-**	**-**	**2,3**
Visual Studio	**2**	**1**	**1**	**1**	**3**	**1**	**1**	**1,4**

10.6 Aufwandsbeispiel Kompass App

Die nachfolgenden Tabelle gibt einen Überblick darüber, welche Ersparnis mit Cross Framework im Bezug auf Native Framework besteht. Als Cross Framework wird hier Phonegap verwendet, da es als einziges Framework alle drei Betriebssysteme unterstützt. Als Beispiel: Angenommen die Aufgabe besteht darin eine App zu entwickeln, welche die Daten des Kompass- Sensors abgreift und anzeigt. Des Weiteren soll diese Beispiel-App nun auf allen drei Betriebssystemen laufen. Dann ist es bei den Native Frameworks nötig sie dreimal zu entwickeln. Wird nun aber Phonegap benutzt ist es nur nötig die Oberfläch dreimal zur entwickeln, wie die folgende Tabelle zeigt. Dieses kurze Beispiel soll nur nochmal verdeutlichen, dass die Entwicklung mit einem Cross Framework von Vorteil sein kann.

Tabelle 10-6 Beispiel Kompass App

	Phonegap	**Android SDK**	**Xcode**	**Visual Studio**
Bereich				
Entwicklungsumgebung allgemein	1	1	1	1,
Oberflächendesign	3	1	1	1
Quellcode für Kompass	1	1	1	1
Gesamt	4	3 + 3 + 3 = 6		

11 Fazit

Die Auswertung im vorherigen Kapitel hat ja schon gezeigt, dass die Cross Frameworks in manchen Fällen einen Vorteil gegenüber den Native Frameworks haben. Im folgenden werden nun noch einmal die Einsatzmöglichkeiten der Cross- und Native Frameworks erwähnt.

11.1 Cross Frameworks

11.1.1 Eine App für mehrere Betriebssysteme

Wenn es darum geht eine App auf mehreren System zum Laufen zu bringen ist es klar von Vorteil ein Cross Framework einzusetzen. Hierbei ist es im Vorfeld aber wichtig, sich das passende Cross Framework rauszusuchen. Kriterien für die Suche sind zum Beispiel die zu verwendenden Ressourcen und die verwendeten Betriebssysteme.

11.1.2 Geringe Anforderungen an die Hardwareschnittstelle

Sind die Anforderungen an die Hardwareschnittstelle gering kann auch die Verwendung eines Cross Frameworks nützlich sein. Das kann zum Beispiel beim Auslesen der GPS- Daten sein, wo es meist nur um die Koordinaten geht. Dann kann es von Vorteil sein, ein Cross Framework für die Erstellung der App zu benutzen, auch wenn es nur eine App für ein Betriebssystem ist.

11.2 Native Frameworks

11.2.1 Nur eine App entwickeln

Native Frameworks sind bei der Erstellung einer App klar im Vorteil, mit kleinen Ausnahmen, wie in Kapitel 11.1.2 zusehen. Sie beinhalten alles was für die Entwicklung nötig ist, von GUI- Builder bis Quellcodeeditor ist alles enthalten. Die erstellten Apps fügen sich vom Aussehen her direkt in das Betriebssystem ein und müssen nicht extra angepasst werden.

11.2.2 Hohe Anforderungen an die Hardwareschnittstelle

Es kann aber auch vorkommen, dass eine App, die auf mehreren Betriebssystem laufen soll, so hohe Anforderung an die Hardwareschnittstelle stellt, dass es nicht möglich ist ein Cross Framework zu benutzen. Dann bleibt kein anderer Ausweg, als die App dreimal zu entwickeln.

12 Abbildungsverzeichnis

13 Tabellenverzeichnis

14 Literaturverzeichnis

1. Wikipedia. [Online] 10. 06 2013. http://en.wikipedia.org/wiki/Mobile_device.

2. mobilfunk-talk. [Online] http://www.mobilfunk-talk.de/news/lexikon/was-ist-ios/.

3. **Erdle, Frank.** connect. [Online] 02. 04 2013. http://www.connect.de/ratgeber/android-geschichte-des-erfolgs-1491130.html.

4. wikipedia. [Online] 20. 05 2013. http://de.wikipedia.org/wiki/Microsoft_Windows_Phone_8.

5. itwissen. [Online] http://www.itwissen.info/definition/lexikon/Blackberry-OS-Blackberry-operating-system.html.

6. chip. [Online] 31. 01 2013. http://www.chip.de/news/BlackBerry-10-OS-Alles-zum-neuen-Handy-System_57802586.html.

7. wikipedia. [Online] 27. 05 2013. http://de.wikipedia.org/wiki/Symbian-Plattform.

8. **Pols, Dr. Axel.** EITO. [Online] 13. 02 2013. http://www.eito.com/press/Press-Releases-2013/Smartphones-sorgen-fuer-96-Prozent-des-Handy-Umsatzes.

9. **Pakalski, Ingo.** golem. [Online] 08. 02 2013. http://www.golem.de/news/smartphone-apple-mit-hoeherem-marktanteil-android-dominanz-ungebrochen-1302-97470.html.

10. **Mayerhofer, Josef.** *Apps Erfolgreich Verkaufen.* München : Carl Hanser Verlag, 2012. ISBN 978-3-446-43028-0.

11. **Usadel, Norbert.** *Titanium Mobile Apps für iPhone und Android.* Haar bei München : Franzis Verlag GmbH, 2012. ISBN 978-3-645-60160-3.

12. Apple. [Online] https://developer.apple.com/support/ios/enrollment.html.

13. Apple. [Online] https://developer.apple.com/programs/which-program/.

14. Apple Developer Guidelines. [Online] https://developer.apple.com/appstore/guidelines.html.

15. **Immler, Christian.** *Der App-Entwickler-Crashkurs.* Haar bei München : Franzis Verlag GmbH, 2012. ISBN 978-3-645-60161-0.

16. Brightcove. [Online] [Zitat vom: 21. 06 2013.] http://support.brightcove.com/de/video-cloud/dokumente/vermarktung-der-ios-app-mit-iads.

17. wikipedia. [Online] 26. 05 2013. http://de.wikipedia.org/wiki/App_Store_%28iOS%29.

18. Android. [Online] http://developer.android.com/sdk/index.html.

19. Google. [Online] https://support.google.com/googleplay/android-developer/answer/113468?hl=en.

20. Android. [Online] [Zitat vom: 21. 06 2013.] http://developer.android.com/training/monetization/ads-and-ux.html.

21. Google. [Online] https://support.google.com/googleplay/android-

developer/answer/112622?hl=en.

22. Microsoft. [Online] http://www.microsoft.com/de-de/download/details.aspx?id=35471.

23. Windows Phone. [Online] https://dev.windowsphone.com/en-us/join.

24. WP 7 Connect. [Online] 11. 07 2011. http://www.wp7connect.com/2011/07/16/microsoft-publishes-my-home-server-on-wp7-marketplace-windows-home-server-2011/.

25. **Ehlert, Ralf, Woiwode, Gregor und Debus, Jörg.** *Windows Phone 8 Grundlagen und Praxis der App- Entwicklung.* Heidelberg : dpunkt.verlag GmbH, 2013. ISBN 978-3-86490-068-6.

26. Wikipedia. [Online] 04. 04 2013. http://de.wikipedia.org/wiki/Windows_Phone_Store.

27. **Laußmann, Jan.** Slideshare. [Online] iks Gesellschaft für Informations- und Kommunikationssysteme mbH, 20. 11 2012. http://de.slideshare.net/iksgmbh/mobile-applikationen-crossplattformentwicklung.

28. **Heise, Doug.** Zdnet. [Online] 30. 08 2011. http://www.zdnet.de/41555607/web-apps-und-native-apps-ein-vergleich/.

29. Heise. [Online] 05. 02 2013. http://www.heise.de/developer/meldung/Gartner-Die-Zukunft-der-App-Entwicklung-ist-hybrid-1797387.html.

30. **Zoosk, Doug Wehmeier.** http://mobiledevdesign.com/learning-resources/html5-cross-platform-compiled-or-native-choose-wisely-app-development. [Online] 22. 10 2012.

31. **Jones, Seth.** Vision Mobile. [Online] http://www.visionmobile.com/blog/2012/02/crossplatformtools/.

32. Phonegap. [Online] http://phonegap.com/about/.

33. Sencha Touch Support. [Online] [Zitat vom: 20. 06 2013.] https://www.sencha.com/store/touch/.

34. Xamarin. [Online] https://store.xamarin.com/.

35. Appcelerator. [Online] http://www.appcelerator.com/plans-pricing/.

36. Adobe. [Online] http://www.adobe.com/de/products/flex.html.

37. Unity3d. [Online] https://store.unity3d.com/.

38. Coronalabs. [Online] http://www.coronalabs.com/store/.

39. MoSync. [Online] [Zitat vom: 19. 06 2013.] http://www.mosync.com/mosync-dual-licence-model.

40. Motorola solutions. [Online] http://www.motorolasolutions.com/US-EN/RhoMobile+Suite/Rhodes.

41. Phonegap Features. [Online] http://phonegap.com/about/feature/.

42. Wikipedia Sencha Touch. [Online] [Zitat vom: 20. 06 2013.] http://en.wikipedia.org/wiki/Sencha_Touch#Features.

43. Xamarin. [Online] [Zitat vom: 20. 06 2013.] http://xamarin.com/tour.

44. Appcelerator Titanium. [Online] http://www.appcelerator.com/platform/titanium-platform/.
45. Unity 3D Überblick. [Online] [Zitat vom: 20. 06 2013.] http://unity3d.com/unity/multiplatform/.
46. Coronalabs. [Online] [Zitat vom: 20. 06 2013.] http://www.coronalabs.com/products/corona-sdk/.
47. MoSync. [Online] http://www.mosync.com/sdk.
48. Wikipedia. [Online] 15. 08 2013. http://en.wikipedia.org/wiki/MoSync.
49. Wikipedia. [Online] 06. 08 2013. http://en.wikipedia.org/wiki/Sencha_Touch.
50. **Wyllie, Diego.** Computerwoche. [Online] 29. 05 2013. http://www.computerwoche.de/a/corona-sdk-plattformuebergreifendes-app-framework,2538959.
51. **Steyer, Ralph.** *Apps mit PhoneGap Entwickeln.* München : Carl Hanser Verlag, 2013. ISBN 978-3-446-43510-0.
52. Wikipedia. [Online] 10. 08 2013. http://en.wikipedia.org/wiki/Android_software_development#Android_SDK.
53. Wikipedia. [Online] 04. 09 2013. http://de.wikipedia.org/wiki/Dalvik_Virtual_Machine.
54. Wikipedia. [Online] 07. 07 2013. http://de.wikipedia.org/wiki/Xcode.
55. **Ebert, Ralf.** Ralf Ebert. [Online] 26. 08 2013. http://www.ralfebert.de/ios/ueberblick-ios-xcode/.
56. Wikipedia. [Online] 28. 06 2013. http://de.wikipedia.org/wiki/LLVM.
57. **Newton-King, James.** Json.Net. 2013.
58. **Sheehan, John.** RestSharp. 2013.
59. Microsoft. [Online] http://www.microsoft.com/en-us/download/details.aspx?id=35471.
60. **Budde, Lars.** t3n. [Online] 04. 09 2012. http://t3n.de/news/smartphones-trend-geht-android-412431/.
61. **Beiersmann, Stefan.** zdnet. [Online] 08. 06 2012. http://www.zdnet.de/41562716/idc-windows-phone-uebertrifft-2016-den-marktanteil-von-ios/.
62. Sencha. [Online] https://www.sencha.com/store/architect/.
63. Go 2 Android. [Online] 11. 01 2013. http://www.go2android.de/kampf-der-app-stores-google-play-erreicht-800-000-apps/.
64. **Conconi, Alex.** design mind. [Online] http://designmind.frogdesign.com/blog/unraveling-html5-vs-native.html.

15 Anhang: Installation und Start der verwendeten Frameworks

15.1 Google Android SDK

15.1.1 Windows

Um das Android SDK auf einem Windows PC auszuführen, sind folgende Voraussetzungen zu erfüllen.

Systemvoraussetzungen :

Betriebssystem:

- Windows XP (32-bit)
- Vista (32- or 64-bit)
- Windows 7 (32- or 64-bit)

Herunterladen :

Zur Installation des SDK muss das ADT Bundle heruntergeladen werden. Es ist unter folgendem Link zu finden.

http://developer.android.com/sdk/index.html

Installation:

Nach dem Herunterladen kann das Packet in einen Ordner auf der Festplatte entpackt werden. In dem nun enpackten Ordner befindet sich schon die Entwicklungsumgebung sowie der Packetmanager zum Installieren weiterer Dateien.

Zur Installation der entsprechenden Android Quellcodestände ist zuerst die Entwicklungsumgebung zu starten. Anschließend ist der Packetmanager um weitere Packete zu installieren.

(20)

Abbildung 15-1 Android Packetmanager

(20)

Starten des Projekts:

Um nun ein vorhandenes Projekt auszuführen, muss es zuerst importiert werden. Dazu muss die Entwicklungsumgebung gestartet sein. Anschließend kann ein vorhandenes Projekt über den Menüpunkt Datei mit dem Untermenü Importieren eingefügt werden.

Als nächstes sollte ein Android Smartphone an den PC angeschlossen werden. Nun genügt es in der Entwicklungsumgebung auf Play zu Klicken, um das Projekt zu starten. Das Projekt wird nun auf das Smartphone übertragen und auch gleich gestartet.

15.1.2 OSX

Um das Android SDK auf einem Apple Computer auszuführen sind folgende Voraussetzungen zu erfüllen.

Systemvoraussetzungen:

Betriebssystem:

- Mac OS X 10.5.8 oder neuer

CPU:

- Intel x86

Nachfolgende Schritte siehe Kapitel 15.1.1

15.2 Apple Xcode

Xcode ist nur lauffähig auf einem Apple Computer daher wird auch nur die Applespezifische Installation erläutert.

Folgende Voraussetzungen sind zu erfüllen, um Xcode auf einem Mac zu installieren.

Systemvoraussetzungen:

Betriebssystem: OSX 10.7

CPU:

Intel x86

Herunterladen / Installieren:

Xcode kann bequem über den App Store auf dem Mac heruntergeladen und installiert werden.

Starten des Projekts:
Um ein vorhandenes Projekt mit Xcode zu starten ist diese beim Startdialog von XCode Auszuwählen. Anschließend ist ein iPhone mit dem Mac zu verbinden. Nun wird das Projekt mit einem Klick auf Play zum iPhone übertragen und gestartet.

15.3 Windows Phone SDK

Das Windows Phone SDK ist nur lauffähig auf einem Windows Computer daher wird auch nur die Windowsspezifische Installation erläutert.
Folgende Voraussetzungen sind zu erfüllen um das Windows Phone SDK auf einem Mac zu installieren.

Systemvoraussetzungen:
Betriebssystem: Windows 8 x64
CPU: x64
Festplatte: 6,5 GB freien Speicher
Arbeitsspeicher: 4GB RAM
(59)

Herunterladen / Installieren:
Als aller erstes muss Visual Studio 2012 installiert werden. Anschließend ist das Windows Phone SDK zu installieren. Im nächsten Schritt ist das aktuelle Update für Visual Studio 2012 zu installieren.

Starten des Projekts:
Um ein vorhandenes Projekt mit Visual Studio 2012 zu starten ist dieses beim Startdialog auszuwählen. Anschließend ist das freigeschaltete iPhone mit dem PC zu verbinden. Nun wird das Projekt mit einem Klick auf Play zum Windows Phone übertragen und gestartet.

15.4 Mosync

15.4.1 Windows

Um das Mosync SDK auf einem PC auszuführen sind folgenden Voraussetzungen zu erfüllen.

Systemvoraussetzungen:

Betriebssystem:

Microsoft Windows XP
Vista
Windows 7
Software:
Android SDK(siehe Kapitel15.1)
Java SE Runtime Environment (JRE) 6
Festplatte: 300MB freier Speicher

Mosync runterladen:

Das Mosync SDK kann unter folgendem Link heruntergeladen werden.

http://www.mosync.com/download

Anschließend kann das Setup ausgeführt werden. Eine Beschreibung zum Setup findet sich unter folgendem Link.

http://www.mosync.com/docs/sdk/tools/guides/ide/installing-mosync/index.html

Starten des Projekts:

Um ein vorhandenes Projekt mit Mosync zu starten ist diese zu importieren. Dazu wird unter dem Menüpunkt File der Untermenüpunkt Import ausgewählt. Im nun folgenden Dialog kann ein vorhandenes Projekt importiert werden. Anschließend ist das freigeschaltete Android Phone mit dem PC zu verbinden. Nun muss nach dem angeschlossenen Geräten gesucht werden. Dazu genügt eine Klick auf das Icon mit dem Smartphone und der Lupe wie in der folgenden Grafik zusehen.

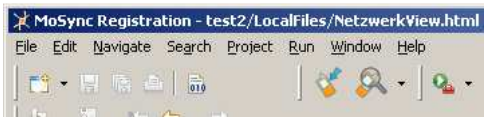


Abbildung 15-2 Mosync Smartphone Suche

Hat Mosync ein Gerät gefunden, kann es ausgewählt werden. Mit einem Klick auf das Icon mit dem Smartphone und dem Blauen Pfeil wird das Projekt nun an das Smartphone übertragen.

15.4.2 OSX

Um das Mosync SDK auf einem PC auszuführen sind folgenden Voraussetzungen zu erfüllen.

Systemvoraussetzungen:

Betriebssystem:

OSX Snow Leopard (10.6.6 oder neuer) 64-bit
Software:
Android SDK(siehe Kapitel15.1)
Java SE Runtime Environment (JRE) 6, 64-bit
Festplatte: 250MB freier Speicher

Mosync runterladen:

Das Mosync SDK kann unterfolgendem Link heruntergeladen werden.

http://www.mosync.com/download

Anschließend kann das Setup ausgeführt werden. Eine Beschreibung zum Setup findet sich unter folgendem Link.

http://www.mosync.com/docs/sdk/tools/guides/ide/installing-mosync-os-x/index.html

Starten des Projekts:
siehe Kapitel 15.4.1

15.5 Corona

15.5.1 Windows

Um das Corona SDK auf einem PC auszuführen sind folgenden Voraussetzungen zu erfüllen.

Systemvoraussetzungen:

Herunterladen / Installieren:
Zum Herunterladen von Corona ist eine Registrierung bei Coronalabs notwendig. Nach erfolgreicher Registrierung ist Corona dann unter folgendem Link herunterladbar.
http://coronalabs.com/products/corona-sdk/starter/
Anschließend kann Corona mit Hilfe des Setups auf dem PC installiert werden.

Erstellen des Projekts:

Zum Erstellen eines vorhandenen Projektes genügt es nach dem Start vom Corona- Simulator die Main Datei aus dem Projekt Verzeichnis auszuwählen. Nun gibt es die Möglichkeit, beim Corona Simulator unter dem Menüpunkt File und dem Untermenüpunkt Build for das Projekt für ein Endgerät zu erstellen.

Starten des Projekts:

Um die App auf dem Smartphone zu starten muss z.B. bei Android, die erstellte APK Datei auf das Gerät kopiert und installiert werden.

15.5.2 OSX

Um das Corona SDK auf einem Mac auszuführen sind folgenden Voraussetzungen zu erfüllen.

Herunterladen / Installieren:

siehe Kapitel 15.5.1

Erstellen des Projekts:

siehe Kapitel 15.5.1

Starten des Projekts:

Um die App auf dem Smartphone zu starten muss bei iTunes die erstellte App auf das Gerät kopiert werden.

15.6 Phonegap

Da Phongap speziell an die einzelnen Betriebssysteme angepasst wird, wird im folgenden die Installation für die hier verwendeten 3 Betriebssysteme erklärt.

15.6.1 Android

Die Installationsanleitung ist unter folgendem Link zu finden.

http://docs.phonegap.com/en/2.8.0/guide_getting-

started_android_index.md.html#Getting%20Started%20with%20Android

15.6.2 Windows Phone 8

Die Installationsanleitung ist unter folgendem Link zu finden.

http://docs.phonegap.com/en/2.8.0/guide_getting-started_windows-phone-8_index.md.html#Getting%20Started%20with%20Windows%20Phone%208

15.6.3 iOS

Die Installationsanleitung ist unter folgendem Link zu finden.

http://docs.phonegap.com/en/2.8.0/guide_getting-started_ios_index.md.html#Getting%20Started%20with%20iOS

15.7 Titanium

15.7.1 Windows

Um das Titanium Framework auf einem PC auszuführen sind folgenden Voraussetzungen zu erfüllen.

Herunterladen / Installieren:
Bevor das Titanium Framework heruntergeladen werden kann, wird ein Account bei Appcelerator benötigt. Um nun mit dem Download zu beginnen, genügt es, sich mit dem vorhandenen Account anzumelden und unter folgender Adresse das Framework herunterzuladen.
http://www.appcelerator.com/platform/titanium-platform/
Ist das Framework heruntergeladen kann es ganz normal über das Setup installiert werden. Nach erfolgreicher Installation kann Titanium nun gestartet werden. Auf der Startseite ist nun das passende Android SDK auszuwählen oder zu installieren.

Starten des Projekts:

Nach dem Start von Titanium gibt es auch hier die Möglichkeit ein vorhandenes Projekt zui importieren. Dazu wählt der Nutzer hier den Menüpunkt File sowie den Untermenüpunkt Import. Im nun folgenden Dialog lässt sich ein vorhandenes Projekt importieren.

Abbildung 15-3 Titanium Projekt Start

Als nächstes muss das iPhone mit dem PC verbunden werden. Zum Start der App genügt nun ein Klick auf des Icon mit dem grünweißen Play und dem Ordner. Nun wird das Projekt automatisch auf das Smartphone übertragen. Auf dem Smartphone kann es nun angeklickt werden, um es zu starten.

15.7.2 OSX

Um das Titanium Framework auf einem PC auszuführen sind folgenden Voraussetzungen zu erfüllen.

Herunterladen / Installieren:
siehe Kapitel 15.7.1
Starten des Projekts:

siehe Kapitel 15.7.1

15.8 Sencha Architekt

15.8.1 Windows

Um das Sencha Architekt auf einem PC auszuführen sind folgenden Voraussetzungen zu erfüllen.

Herunterladen / Installieren:
Unter folgendem Link kann der Installer heruntergeladen werden.
https://www.sencha.com/products/architect/download/
Ist der Download erfolgreich abgeschlossen kann der Installer ausgeführt werden.
Starten des Projekts:

Da Sencha nicht geeignet ist für die mit der TestApp Entwicklung ist auch kein Projekt zum ausführen verfügbar.

15.8.2 OSX

Siehe Kapitel 15.8.1

16 Anhang: Webserver

Zum hochladen der Bilder dient der Webserver mit den nachfolgenden Daten.

URL zum hochladen des Bildes:

http://x-y-z.bplaced.net/uploadfile.php

URL zum Ansehen des Bildes:

http://x-y-z.bplaced.net/uploadimages/image.png

Quellcode zum Hochladen:

Der nachfolgende PHP Code dient zum entgegennehmen des Bildes auf dem Server.

```php
<?php
    $name = 'image.png';
    $target = 'uploadimages/';
    $target = $target . $name;
    if(move_uploaded_file($_FILES['media']['tmp_name'], $target))
    {
        echo "true";
    }
    else
    {
        echo "false";
    }
?>
```

Abbildung 16-1Script zum hochladen eines Bildes

17 Anhang: Testprotokolle der Frameworks

Testprotokoll

Tester: Markus Domin Datum: 01.10.2013 Framework: Corona

Funktioniert der Kompass (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Funktioniert der GPS- Sensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert das aufnehmen von Bildern mit der Kamera (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	√	√	√	√	√	X	X	X	X	X
Gesamt	X					√					X				

Funktioniert das auswählen von Bildern aus der Gallerie (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	√	√	√	√	√	X	X	X	X	X
Gesamt	X					√					X				

Funktioniert der Versand von Bilder zum Webserver (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der Versand von Bilder über Bluetooth (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert Beschleunigungssensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Testprotokoll

Tester: Markus Domin Datum: 01.10.2013 Framework: Titanium

Funktioniert der Kompass (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	√	√	√	√	X	√	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der GPS- Sensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Funktioniert das aufnehmen von Bildern mit der Kamera (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√									

Funktioniert das auswählen von Bildern aus der Gallerie (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Funktioniert der Versand von Bilder zum Webserver (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der Versand von Bilder über Bluetooth (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert Beschleunigungssensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Testprotokoll

Tester: Markus Domin Datum: 01.10.2013 Framework: Mosync

Funktioniert der Kompass (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	√	X	√	√	X	√	√	X	√	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der GPS- Sensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	√	√	√	√	√	X	X	X	X	X
Gesamt	X					√					X				

Funktioniert das aufnehmen von Bildern mit der Kamera (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Funktioniert das auswählen von Bildern aus der Gallerie (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der Versand von Bilder zum Webserver (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Funktioniert der Versand von Bilder über Bluetooth (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert Beschleunigungssensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Testprotokoll

Tester: Markus Domin Datum: 01.10.2013 Framework: Phonegap

Funktioniert der Kompass (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	X	X	X	X	X
Gesamt	√					√					X				

Funktioniert der GPS- Sensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	√	√	√	√	√	√	√	√	√	√
Gesamt	X					√					√				

Funktioniert das aufnehmen von Bildern mit der Kamera (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Gesamt	√					√					√				

Funktioniert das auswählen von Bildern aus der Gallerie (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Gesamt	√					√					√				

Funktioniert der Versand von Bilder zum Webserver (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Gesamt	√					√					√				

Funktioniert der Versand von Bilder über Bluetooth (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert Beschleunigungssensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Gesamt	√					√					√				

Testprotokoll

Tester: Markus Domin Datum: 01.10.2013 Framework: Sencha

Funktioniert der Kompass (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der GPS- Sensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert das aufnehmen von Bildern mit der Kamera (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert das auswählen von Bildern aus der Gallerie (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der Versand von Bilder zum Webserver (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der Versand von Bilder über Bluetooth (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
Gesamt	**X**					**X**					**X**				

Funktioniert Beschleunigungssensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
Gesamt	**X**					**X**					**X**				

Testprotokoll

Tester: Markus Domin Datum: 01.10.2013 Framework: Android SDK

Funktioniert der Kompass (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	X	√	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der GPS- Sensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	X	X	X	X	X	X	X	X	X	X
Gesamt	√					X					X				

Funktioniert das aufnehmen von Bildern mit der Kamera (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	X	X	X	X	X	X	X	X	X	X
Gesamt	√					X					X				

Funktioniert das auswählen von Bildern aus der Gallerie (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	X	X	X	X	X	X	X	X	X	X
Gesamt	√					X					X				

Funktioniert der Versand von Bilder zum Webserver (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	X	X	X	X	X	X	X	X	X	X
Gesamt	√					X					X				

Funktioniert der Versand von Bilder über Bluetooth (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	X	X	X	X	X	X	X	X	X	X
Gesamt	√					X					X				

Funktioniert Beschleunigungssensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	√	√	√	√	√	X	X	X	X	X	X	X	X	X	X
Gesamt	√					√					√				

Testprotokoll

Tester: Markus Domin Datum: 01.10.2013 Framework: XCODE

Funktioniert der Kompass (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	√	√	√	√	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der GPS- Sensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	√	√	√	√	√	X	X	X	X	X
Gesamt	X					√					X				

Funktioniert das aufnehmen von Bildern mit der Kamera (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	√	√	√	√	√	X	X	X	X	X
Gesamt	X					√					X				

Funktioniert das auswählen von Bildern aus der Gallerie (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	√	√	√	√	√	X	X	X	X	X
Gesamt	X					√					X				

Funktioniert der Versand von Bilder zum Webserver (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert der Versand von Bilder über Bluetooth (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Funktioniert Beschleunigungssensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Gesamt	X					X					X				

Testprotokoll

Tester: Markus Domin Datum: 01.10.2013 Framework: Visual Studio

Funktioniert der Kompass (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	√	√	√	√	X
Gesamt	X					X					X				

Funktioniert der GPS- Sensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	√	√	√	√	√
Gesamt	X					X					√				

Funktioniert das aufnehmen von Bildern mit der Kamera (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	√	√	√	√	√
Gesamt	X					X					√				

Funktioniert das auswählen von Bildern aus der Gallerie (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	√	√	√	√	√
Gesamt	X					X					√				

Funktioniert der Versand von Bilder zum Webserver (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	√	√	√	√	√
Gesamt	X					X					√				

Funktioniert der Versand von Bilder über Bluetooth (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	√	√	√	√	√
Gesamt	X					X					√				

Funktioniert Beschleunigungssensor (X für Nein und √ für Ja)?

Framework	Android					iOS					Windows Phone 8				
Tests	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Bestanden	X	X	X	X	X	X	X	X	X	X	√	√	√	√	√
Gesamt	X					X					√				