Diplomarbeit

Systematischer Vergleich von Frameworks für die plattformübergreifende Entwicklung mobiler Applikationen mit Web Technologien

Ausgeführt zum Zweck der Erlangung des akademischen Grades
Dipl.-Ing. für technisch-wissenschaftliche Berufe
am Fachhochschul-Masterstudiengang „Telekommunikation und Medien“ St. Pölten

von:

Ines Schwaighofer, BSc
tm0910262559

Erstbegutachter und Betreuer:
Dipl. Ing. Markus Seidl

Zweitbegutachter:
Dipl. Ing. Grischa Schmiedl

St. Pölten, Juni 2011

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einer Begutachterin/einem Begutachter zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der vom Begutachter beurteilten Arbeit überein.

Wien, 15. Juni 2011

Ort, Datum

Unterschrift

Zusammenfassung

Die rasante Weiterentwicklung des mobilen Internet in den letzten Jahren hinsichtlich immer schneller werdender Übertragungsmöglichkeiten in Kombination mit den aufgekommenen Smartphones führt bei den Benutzerinnen/Benutzern zu einer vermehrten Nutzung des mobilen Internet. Mobile Applikationen mit reichhaltigen Funktionalitäten sind für die Benutzerinnen/Benutzer ein wichtiger Bestandteil auf den Smartphones geworden und die Verwendung dieser im Alltag nicht mehr wegzudenken. Alle Hersteller möchten vom mobilen Markt profitieren, wodurch eine Vielfalt an Plattformen und unterschiedlichen Programmiersprachen für die Entwicklung von mobilen Applikationen entstanden ist. Dies bedeutet einen hohen Kosten- und Zeitaufwand für die Entwicklung, um alle Plattformen und somit potenzielle Kundinnen/Kunden erreichen zu können.

Frameworks für die plattformübergreifende Entwicklung mobiler Applikationen mit Web Technologien sollen hierbei Abhilfe schaffen und die Entwicklerinnen/Entwickler bei der gleichzeitigen Umsetzung für alle Plattformen unterstützen. In dieser Diplomarbeit haben wir einen systematischen Vergleich von vier der bekanntesten Frameworks jQuery Mobile, Sencha Touch, PhoneGap und Titanium Mobile unter Berücksichtigung von ausgewählten Kriterien (Installation und Entwicklungsumgebung, User Interface, Geolocation, Karte, offline Applikationen, lokale Datenbank, Kamera, Test, Distribution) vorgenommen. Nach einer Literaturrecherche über die vier Frameworks zeigen wir anhand einer prototypischen Umsetzung die Unterschiede der Frameworks auf. Wir setzen eine Beispielapplikation basierend auf den Kriterien in jedem Framework um und testen diese auf vier der marktführenden Plattformen Symbian, Android, BlackBerry und iOS. Da die Empfehlung eines Frameworks von unterschiedlichen Anforderungen seitens der Entwicklerinnen/Entwickler abhängig ist, wird diese anhand von drei verschiedenen Szenarien unter Berücksichtigung der Ergebnisse aus der Umsetzung der Beispielapplikation abgegeben.

Abstract

In recent years the rapid development of the mobile Internet in terms of faster transmission possibilities and in combination with the arisen smartphones lead the users to an increasing use of the mobile Internet. Mobile applications with rich functionalities have become an important part for users on smartphones and are indispensable in their everyday life. All manufacturers want to profit from the mobile market, which results in a wide variety of platforms and different programming languages for the development of mobile applications. This means a lot of time and higher costs in development to reach all platforms and therefore potential customers.

Frameworks for the development of cross platform mobile applications with web technologies should assist developers in the simultaneous implementation for all platforms. This thesis covers a systematic comparison of four of the most well known frameworks jQuery Mobile, Sencha Touch, PhoneGap and Titanium Mobile under the consideration of selected criteria (installation and development environment, user interface, geolocation, map, offline applications, local database, camera, test, distribution). For better visualization a sample application based on the criteria has been realized in every framework and tested on four of the market leading platforms Symbian, Android, BlackBerry and iOS. Due to the reason the recommendation of one framework depends on different requirements by the developer, this will be made based on three different scenarios under the consideration of the results of the implemented sample application.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

3G	Dritte Generation
4G	Vierte Generation
ADT	Android Developer Toolkit
A-GPS	Assisted Global Positioning System
API	Application Programming Interface
AJAX	Asynchronous JavaScript And XML
CEO	Chief Executive Officer
CSS	Cascading Style Sheets
DOM	Document Object Model
EDGE	Enhanced Datarates for GSM Evolution
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HSDPA	High Speed Downlink Packet Access
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
ID	Identity
IP	Internet Protocol
IT	Information Technology
Java ME	Java Micro Edition
JDK	Java Development Toolkit
Ltd.	Limited
LTE	Long Term Evolution
MB	Megabyte
MOAP	Mobile Oriented Applications Platform
NFC	Near Field Communication
OHA	Open Handset Alliance
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
SD	Secure Digital
SDK	Software Development Kit
SMS	Short Message Service
UMTS	Universal Mobile Telecommunications System

USB	Universal Serial Bus
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WLAN	Wirless Local Area Network
WML	Wirless Markup Language
XHTML	Extensible HyperText Markup Language
XHTML MP	Extensible HyperText Markup Language Mobile Profile

1 Einleitung

Aufgrund der stetig wachsenden Zugriffe von Smartphones auf das mobile Internet und dem daraus resultierenden lukrativen Geschäft mit nativen auf Smartphones installierten Applikationen führte zu einem mobilen Markt mit einer Vielfalt an unterschiedlichen Plattformen. Da jede Plattform für die Entwicklung von nativen Applikationen auf einer anderen Programmiersprache aufbaut, bedeutet dies einen hohen Aufwand sowie hohe Kosten um alle potenziellen Kundinnen/Kunden zu erreichen. In den letzten Jahren sind einige Frameworks auf den Markt gekommen, die bei der plattformübergreifenden Entwicklung mobiler Applikationen mit Hilfe von bekannten Web Technologien unterstützen sollen. Wir nehmen dies zum Anlass, einen tiefen Einblick in vier der bekanntesten Frameworks zu nehmen, und die Frage zu beantworten, welches Framework für die plattformübergreifende Entwicklung mobiler Applikationen unter Berücksichtigung verschiedener Kriterien für die Entwicklerinnen/Entwickler geeignet ist, und bei der Umsetzung unterstützen kann.

Für die Beantwortung der Forschungsfrage führen wir eine Bestandsaufnahme anhand eines ausgewählten Kriterienkataloges für alle vier Frameworks durch. Anschließend setzen wir für den systematischen Vergleich der vier plattformübergreifenden Frameworks eine Beispielapplikation unter Berücksichtigung der Kriterien in allen Frameworks um und testen diese auf den vier Plattformen mit dem höchsten Marktanteil. Die Empfehlung eines Frameworks ergibt aufgrund von unterschiedlichen Anforderungen seitens der Entwicklerin/dem Entwickler für verschiedene Szenarien. In diese Szenarien fließen die Ergebnisse aus der Umsetzung der Beispielapplikation und die daraus resultierende Bewertung der Frameworks ein.

Kapitel 2 gibt zunächst einen geschichtlichen Abriss über das mobile Internet sowie die vermehrte Nutzung dieses durch Smartphones und mobile Applikationen wieder. Kapitel 3 gibt einen Überblick über die vier marktführenden Plattformen und Kapitel 4 geht auf die verschiedenen Applikationstypen näher ein. Kapitel 5 widmet sich der Möglichkeit der plattformübergreifenden Entwicklung mobiler Applikationen mit Web Technologien. In den Kapiteln 6 bis 9 findet die Bestandsaufnahme anhand verschiedener Kriterien für die vier ausgewählten Frameworks statt. Kapitel 10 beschäftigt sich mit der Umsetzung der Beispielapplikation und Kapitel 11 gibt eine Framework Empfehlung ab. Kapitel 12 gibt eine abschließende Zusammenfassung wieder.

2 Das mobile Internet

Die Nutzung des mobilen Internet ist in den letzten Jahren aufgrund der sich laufend verändernden Übertragungsmöglichkeiten sowie durch das Aufkommen von Smartphones rasant gestiegen. Das Smartphone ist auf dem Vormarsch und verändert das Verhalten der Benutzerinnen/Benutzer aufgrund verschiedenster neuer Funktionalitäten erheblich. Smartphones sind ein wichtiger Bestandteil für die Benutzerinnen/Benutzer geworden und ermöglichen im Zusammenhang mit mobilen Applikationen einen komplett neuen Zugang zum mobilen Internet. Dadurch wird die Entwicklung mobiler Applikationen immer bedeutsamer und repräsentiert ein neues Segment in der Software-Branche. Im folgenden Kapitel werden diese Themen, beginnend mit einem geschichtlichen Abriss bis hin zu den verschiedenen Einsatzmöglichkeiten von Smartphones, genauer erläutert.

Kaikkonen (2008, S. 2) zeigt in ihrer Studie auf, dass die Definition des mobilen Internet sehr unterschiedlich ausgelegt werden kann. Einerseits kann unter dem mobilen Internet ein für den mobilen Gebrauch angepasstes Internet verstanden werden, andererseits kann das mobile Internet als Zugriff auf das Internet über einen mobilen Webbrowser definiert werden. Kaikkonen (2008, S. 2) ist jedoch der Meinung, dass es verschiedene Möglichkeiten gibt, um auf das mobile Internet zugreifen zu können (vgl. Abbildung 1).

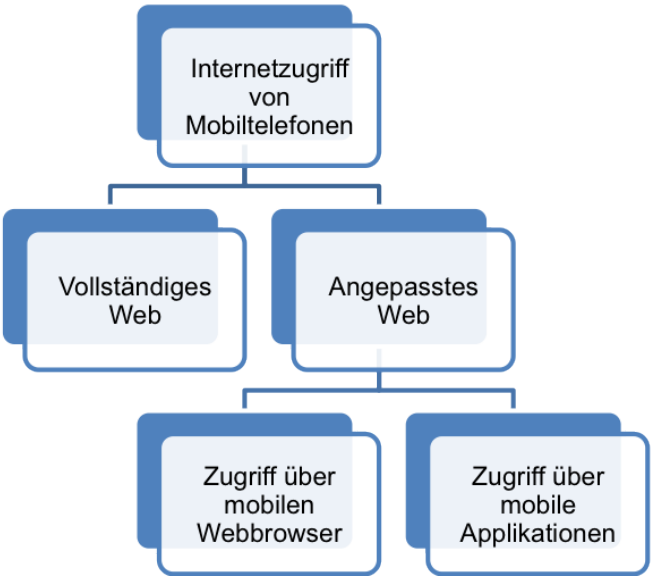


Abbildung 1: Möglichkeiten des Internetzugriffs von Mobiltelefonen (vgl. Kaikkonen, 2008, S. 2)

Zunächst kann der Internetzugriff von Mobiltelefonen auf das vollständige oder angepasste Web erfolgen. Unter dem vollständigen Web werden normale Websites, die für den Desktop Webbrowser entwickelt wurden, verstanden. Das angepasste Web beinhaltet Websites, die auf Mobiltelefone zugeschnitten sind und beispielsweise eine andere Navigationsstruktur und angepasste Inhalte aufweisen. Weiters gibt es zwei verschiedene Ansätze, um auf das angepasste Web zugreifen zu können. Einerseits von einem mobilen Webbrowser und andererseits von Applikationen, die bereits auf dem Mobiltelefon vorinstalliert sind oder heruntergeladen und installiert werden können. Diese Applikationen benötigen keinen Webbrowser, um sich zum Internet zu verbinden und verschiedenste Informationen abzurufen. Beispielsweise werden Applikationen dazu verwendet, um Fotos auf einen Blog hochzuladen oder von diesem herunterzuladen. (vgl. Kaikkonen, 2008, S. 2 - 3)

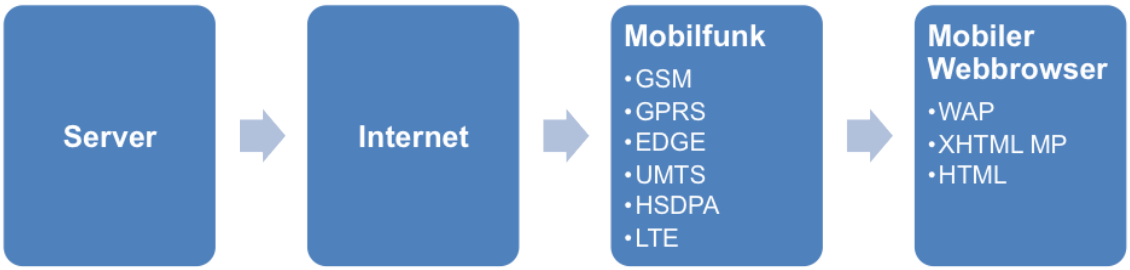


Abbildung 2: Architektur des mobilen Internet

Abbildung 2 zeigt die Architektur des mobilen Internet, die im Folgenden näher erläutert wird. Am Beginn des mobilen Webs stand das Wireless Application Protocol (WAP), das 1999 kommerziell auf dem Markt eingeführt wurde. WAP ist eine standardisierte Technologie, die es ermöglicht über ein sogenanntes WAP-Gateway Informationen zwischen einem Mobiltelefon und dem Internet zu transportieren. Zeitgleich entwickelte NTT DOCOMO das i-mode Protokoll, dieses konnte sich jedoch nur in Japan etablieren. Die Auszeichnungssprache von WAP-Seiten (WAP 1.0 und WAP 1.1) heißt Wirless Markup Language (WML), diese wird von modernen Smartphones, wie iPhone oder Android nicht mehr unterstützt. (vlg. Firtman, 2010, S. 54 - 56) 2009 lag der WML Datenverkehr der mobilen Internetnutzung unter 2 % (vlg. Firtman, 2010, S. 98).

Die damaligen WAP Browser konnten nur drei bis vier Textzeilen anzeigen, unterstützten die Anzeige von Bildern nicht und der Bildschirm war meistens schwarz-weiß. Manche bezeichneten WAP früher oft scherzhaft als „Wait And Pay“ (vgl. Alby, 2008, S. 21), da die Datenverbindung über das GSM (Global System for Mobile

Communications) Netz sehr langsam war und die Abrechnung auf Zeitbasis erfolgte. (vlg. Firtman, 2010, S. 55)

WAP 2.0 wurde 2002 veröffentlicht und ermöglicht eine direkte HTTP (HyperText Transfer Protocol) Kommunikation zwischen dem Mobiltelefon und dem Server. XHTML MP (Extensible HyperText Markup Language Mobile Profile) ersetzte die alte WML Auszeichnungssprache. (vlg. Firtman, 2010, S. 56) XHTML MP stellt eine Teilmenge von XHTML dar und ist auf die speziellen Anforderungen von Mobiltelefonen zugeschnitten (vlg. Firtman, 2010, S 109).

Das General Packet Radio Service (GPRS) ist eine Erweiterung des GSM Netzes und ermöglicht eine paketorientierte Datenübertragung. Dadurch wird der Sprachkanal nicht mehr dauerhaft blockiert, außer wenn Daten übertragen werden. Mobiltelefone können somit permanent im Internet bleiben, um beispielsweise E-Mails abzuholen. (vgl. Alby, 2008, S. 22) Die Benutzerinnen/Benutzer bezahlen nicht mehr für die Verbindungsdauer, sondern für die Menge des übertragenen Datenvolumens (vlg. Firtman, 2010, S 55). EDGE (Enhanced Datarates for GSM Evolution) ist ein weiterer Standard von GSM und ermöglicht eine schnellere Datenübertragung als GPRS mit bis zu 473,6 kbit/s (vgl. Alby, 2008, S. 23).

Bereits 2003 waren vier mobile Webbrowser (Opera von Opera Inc. (vgl. Opera, 2011), NetFront von Access Inc. (vgl. Access, 2011), WebViewer von Reqwireless Inc. (vgl. Delbrouck, 2003) und Doris von Anygraaf Inc. (vgl. Anygraaf, 2011)) für das Nokia S60 Betriebssystem verfügbar, die es ermöglichten auf alle HTML (HyperText Markup Language) Websites zugreifen zu können und nicht nur auf die speziell für Mobiltelefone angepassten Seiten. (vgl. Kaikkonen, 2008, S. 1) Heutzutage verfügen die gängigsten Mobiltelefone über einen Webbrowser, der normale Websites wie auf einem PC darstellen kann. Weiters verändert sich auch die Auszeichnungssprache für mobile Websites. Aus der Studie „Verwendbarkeit und Verwendung des mobilen Webs“ geht hervor, dass mittlerweile 75 der insgesamt 90 gefundenen mobilen Versionen der 100 am häufigsten von Österreicherinnen/Österreichern besuchten Websites in normalem HTML erstellt wurden. (vgl. Schmiedl et al., 2009a, S. 10 ff)

UMTS (Universal Mobile Telecommunications System) ist die dritte Generation (3G) der Mobilfunkübertragungstechnik (vgl. Alby, 2008, S. 24) und mobilkom austria (heute A1 Telekom Austria) führte im September 2002 in Österreich das erste UMTS Netz Europas ein (vgl. A1 Telekom Austria Presseabteilung, 2002). mobilkom austria hat die

ersten UMTS-fähigen Mobiltelefone im April 2003 angeboten (vgl. A1 Telekom Austria Presseabteilung, 2003). Im Vergleich zu GSM mit 9,6 kbit/s und GPRS mit 171,4 kbit/s kann mit UMTS eine Transferrate von 384 kbit/s erreicht werden. High Speed Downlink Packet Access (HSDPA) ist ein weiteres Datenübertragungsverfahren von UMTS, mit welchem Datenraten bis zu 7,2 Mbit/s möglich sind. (vgl. Tanner & Hofstetter, 2008, S. 10) Mit High Speed Uplink Packet Access (HSUPA) kann eine Datenrate von 5,76 Mbit/s erreicht werden (vgl. Tanner & Hofstetter, 2008, S. 26).

Die bestehenden Mobilfunkstandards reichen jedoch für die zukünftigen Bedürfnisse der Benutzerinnen/Benutzer von Mobiltelefonen nicht mehr aus. Für viele Mobiltelefon Besitzerinnen/Besitzer wird es immer selbstverständlicher E-Mails abzurufen, Videos herunterzuladen oder einfach im Internet zu surfen. Die Datenraten steigen dabei kontinuierlich an. Aus diesem Grund wird schon an der nächsten Generation (4G), dem LTE (Long Term Evolution) Standard, gearbeitet. LTE steigert die Datenraten erneut, ein Uplink von bis zu 50 Mbit/s und ein Downlink von bis zu 100 Mbit/s sind möglich. (vgl. Tanner & Hofstetter, 2008, S. 11) Im Oktober 2010 wurden in Österreich die ersten LTE Sendestationen kommerziell in Betrieb genommen (vgl. Gabriel, 2010). Tabelle 1 gibt einen Überblick über die verschiedenen Mobilfunkübertragungstechniken und ihre Datenraten.

Mobilfunkübertragungstechniken	Datenraten
GSM	9,6 kbit/s*
GPRS	171,4 kbit/s*
EDGE	473,6 kbit/s*
UMTS	384 kbit/s*
HSDPA	7,2 Mbit/s*
LTE Downlink	100 Mbit/s*
LTE Uplink	50 Mbit/s*

* theoretischer Maximalwert

Tabelle 1: Verschiedene Mobilfunkübertragungstechniken und ihre Datenraten

Die neue Bezeichnung „Mobile Web 2.0“ entstand 2007, als Smartphones, wie iPhone oder Android auf den Markt kamen (vgl. Firtman, 2010, S. 59). Ein Mobiltelefon wird als Smartphone bezeichnet, wenn dieses über ein Multi-Tasking Betriebssystem, einen vollständigen Desktop Webbrowser sowie über eine WLAN (Wirless Local Area Network) und 3G Anbindung verfügt. Weiters ist ein Smartphone mit zusätzlichen Funktionen, wie einem GPS (Global Positioning System), einem digitalen Kompass, einer Kamera und einem Beschleunigungsmesser ausgestattet. (Firtman, 2010, S. 8)

Aufgrund von Smartphones veränderte sich das mobile Web entscheidend. Diese endgerätespezifischen Funktionalitäten kreieren im Zusammenhang mit einer großen Anzahl an neu aufgekommenen sozialen Netzwerken (z.B. Facebook), Blogs und von Benutzern generierten Inhalten auf Portalen, das Mobile Web 2.0. Typische Charakteristiken von Mobile Web 2.0 Websites sind zum Beispiel: Geolocation, Offline Speicherung, Aktivitäten auf sozialen Netzwerken, HTML 4/5, CSS (Cascading Style Sheets) 2/3, JavaScript, Ajax (Asynchronous JavaScript And XML) und berührungssensitive Benutzeroberflächen (Touch/Multi-Touch). Mobile Web 2.0 möchte das Benutzererlebnis des mobilen Webs im Gegensatz zum bisherigen WAP 2.0 deutlich verbessern. (vgl. Firtman, 2010, S. 59 - 60)

Der Pressebericht „Gartner Highlights Key Predictions for IT Organizations and Users in 2010 and Beyond“ von Gartner (2010a) prognostiziert:

> *By 2013, mobile phones will overtake PCs as the most common Web access device worldwide.*

Bis 2013 wird die Anzahl der internetfähigen Mobiltelefone weltweit bei 1,82 Milliarden liegen und die Anzahl der Computer, die bei 1,78 Milliarden liegen wird, übersteigen. Daraus resultiert, dass Inhaberinnen/Inhaber einer Webseite oder Applikation diese für Mobiltelefone optimieren müssen, um Wettebewerbsnachteilen und Markteintrittsbarrieren zu entgehen. (vgl. Gartner, 2010a)

Das Ergebnis einer Studie der Fachhochschule St. Pölten zeigt klar auf, dass Österreicherinnen/Österreicher, die das mobile Internet verwenden, speziell dafür angepasste Websites bevorzugen. Gründe für eine optimierte mobile Website sind beispielsweise die automatische Anpassung der Bildschirmgröße und eine direktere Führung der Navigation zum Ziel. Weiters hat die Studie ergeben, dass nur 6 von 100 der größten österreichischen Firmen eine mobile Version ihrer Website haben. (vgl. Schmiedl et al., 2009b)

Aus dem Bericht „AdMob Mobile Metrics Highlights May 2010“ von AdMob (2010, S. 5) geht hervor, dass der mobile Internetverkehr in den letzten Jahren rasant gestiegen ist. Die mobile Internetaktivität hat sich von Mai 2008 bis Mai 2010 in den Regionen Afrika und Osteuropa vervierfacht und in den Regionen Nordamerika, Asien, Westeuropa, Lateinamerika und Ozeanien sogar versechsfacht. (vgl. AdMob, 2010, S. 5) Für 2011

liegen noch keine aktuellen Daten vor, wir gehen jedoch davon aus, dass der mobile Internetverkehr weiter gestiegen sein könnte.

Das Marktforschungs- und Beratungsunternehmen Fittkau & Maaß Consulting (2010) zeigt in ihrer W3B Studie „Das mobile Internet“ auf, dass vor allem Smartphone Benutzerinnen/Benutzer das Wachstum des mobilen Internet vorantreiben.

Die Daten der W3B Studie beziehen sich mit 96 % auf den deutschsprachigen Raum und betreffen die Länder Deutschland, Schweiz und Österreich. Die restlichen 4 % entfallen auf andere Länder und weisen eine sehr geringe Stichprobe auf. (vgl. Persönliche E-Mail von Motl, 13.10.2010) In Abbildung 3 wird ersichtlich, dass im Frühjahr 2010 Smartphone Benutzerinnen/Benutzer um 35,1 % deutlich intensiver das mobile Internet genutzt haben als Nicht-Smartphone Benutzerinnen/Benutzer (vgl. Fittkau & Maaß Consulting, 2010). Weltweit gesehen setzten sich im Mai 2010 Smartphones betreffend der mobilen Internetnutzung mit 46 % ebenfalls an die Spitze (vgl. AdMob, 2010, S. 15).

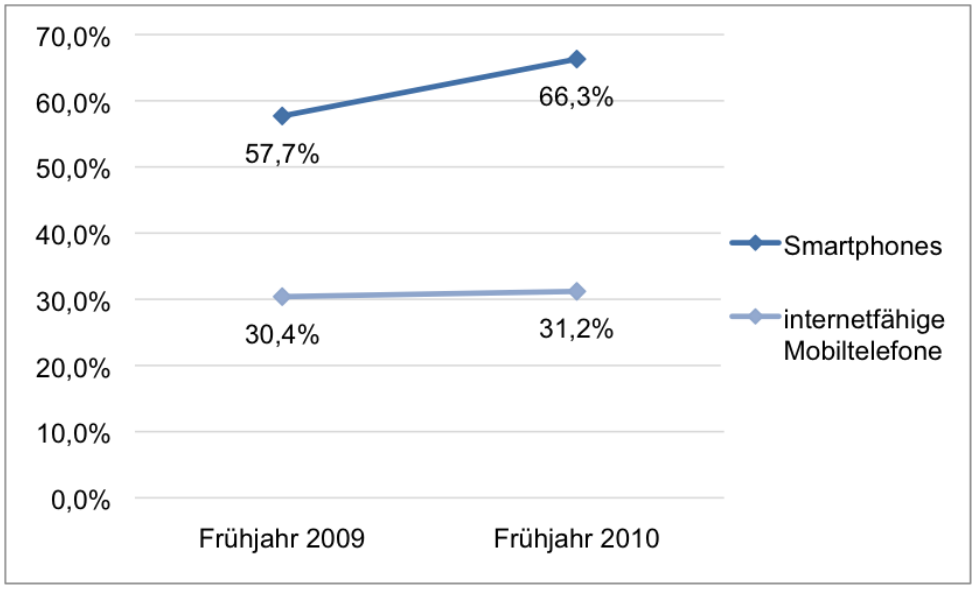


Abbildung 3: Nutzung des mobilen Internet im deutschsprachigen Raum durch Besitzerinnen/Besitzer von Smartphones und anderen internetfähigen Mobiltelefonen (vgl. Fittkau & Maaß Consulting, 2010)

Ausschlaggebend für den raschen Anstieg der mobilen Internetnutzung ist die Beliebtheit von nativen Applikationen (vgl. Abbildung 4), die von 66 % der Smartphone Besitzerinnen/Besitzer benutzt werden. Native Applikationen sind kleine Zusatzanwendungen, die auf Smartphones installiert werden können.

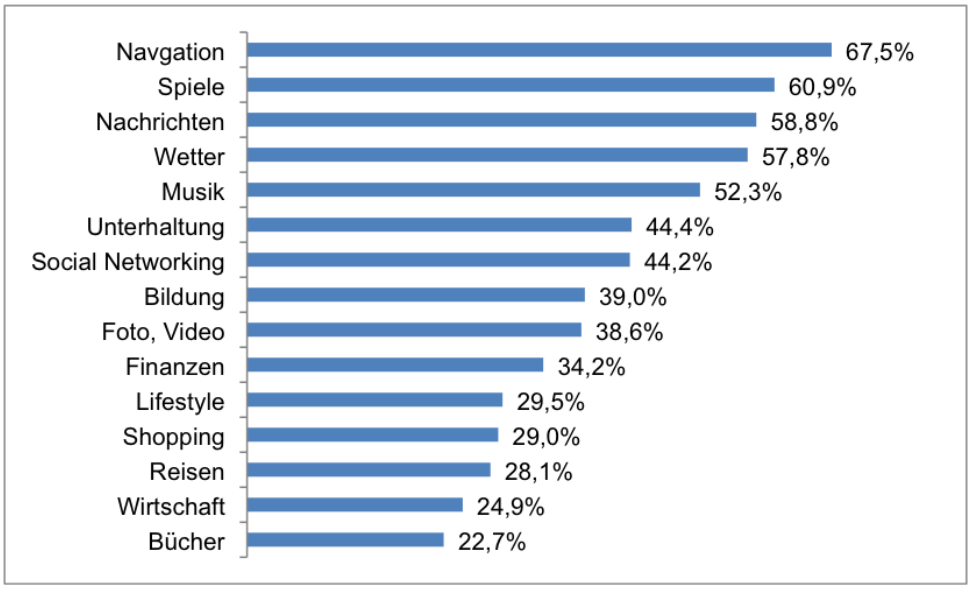


Abbildung 4: Beliebteste native Applikationen im deutschsprachigen Raum (vgl. Fittkau & Maaß Consulting, 2010)

Aus Abbildung 5 geht hervor, dass im Jahresvergleich der Besitz von Smartphones um 5,9 % gestiegen ist, hingegeben die anderen internetfähigen Mobiltelefone 4,9 % verloren haben. Daraus folgt, dass normale Mobiltelefone mit Internet-Zugang zukünftig durch Smartphones verdrängt werden. (vgl. Fittkau & Maaß Consulting, 2010)

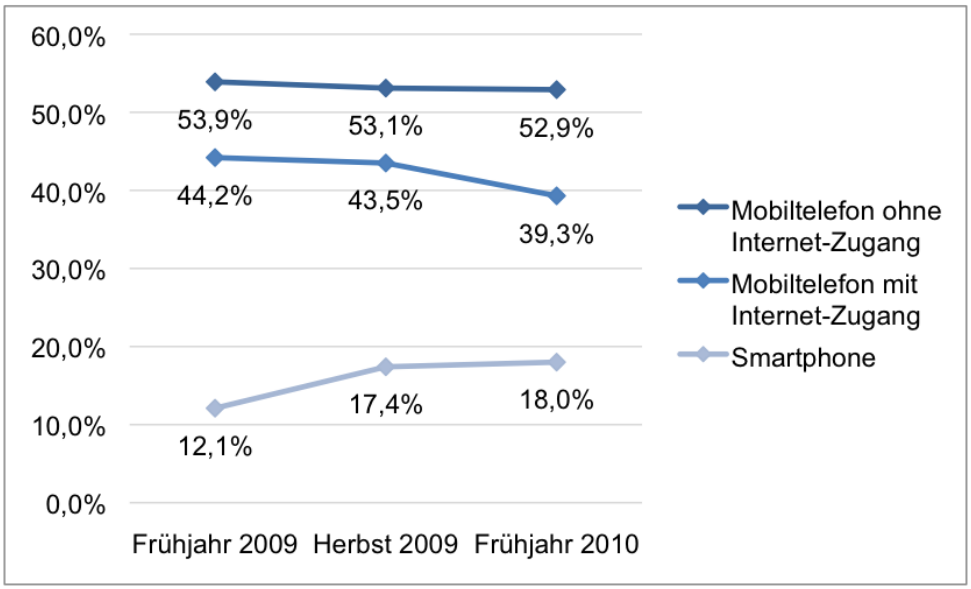


Abbildung 5: Smartphones ersetzen internetfähige Mobiltelefone im deutschsprachigen Raum (vgl. Fittkau & Maaß Consulting, 2010)

Dies wird auch am weltweiten Absatz von Smartphones in Abbildung 6 ersichtlich. Im Jahr 2010 liegt die Gesamtanzahl der verkauften Smartphones bei 296,6 Millionen.

Dies bedeutet einen Anstieg von 72 % im Vergleich zum Jahr 2009 mit 172,3 Millionen verkauften Smartphones. (vgl. Gartner, 2011a)

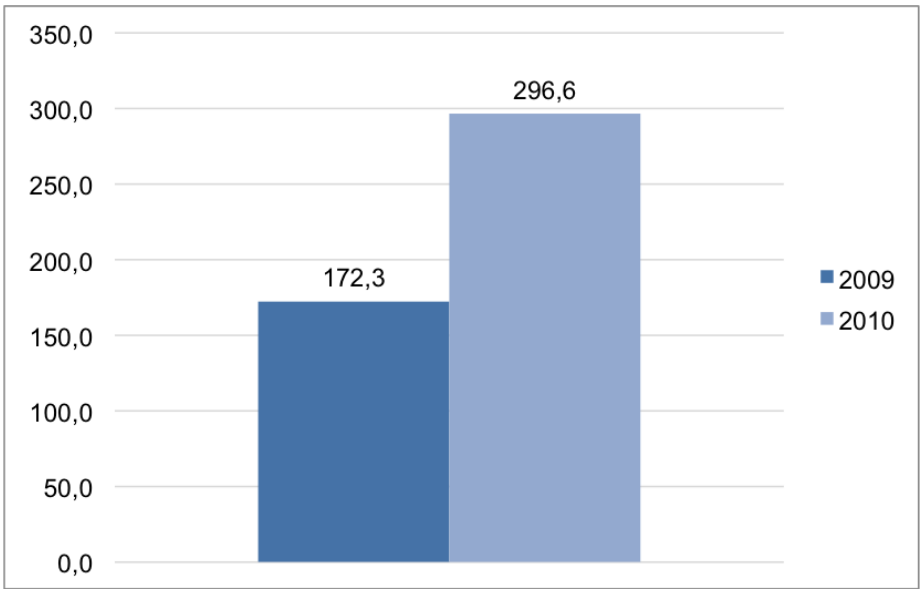


Abbildung 6: Weltweiter Absatz von Smartphones in Millionen (vgl. Gartner, 2011a)

Mobiltelefone ohne Internet-Zugang bleiben laut Abbildung 5 mit 52,9 % weiter an der Spitze. Gründe hierfür sind seitens der Benutzerinnen/Benutzer einerseits zu hohe Kosten und andererseits der mangelnde Komfort des mobilen Internet, wie Abbildung 7 verdeutlicht. (vgl. Fittkau & Maaß Consulting, 2010)

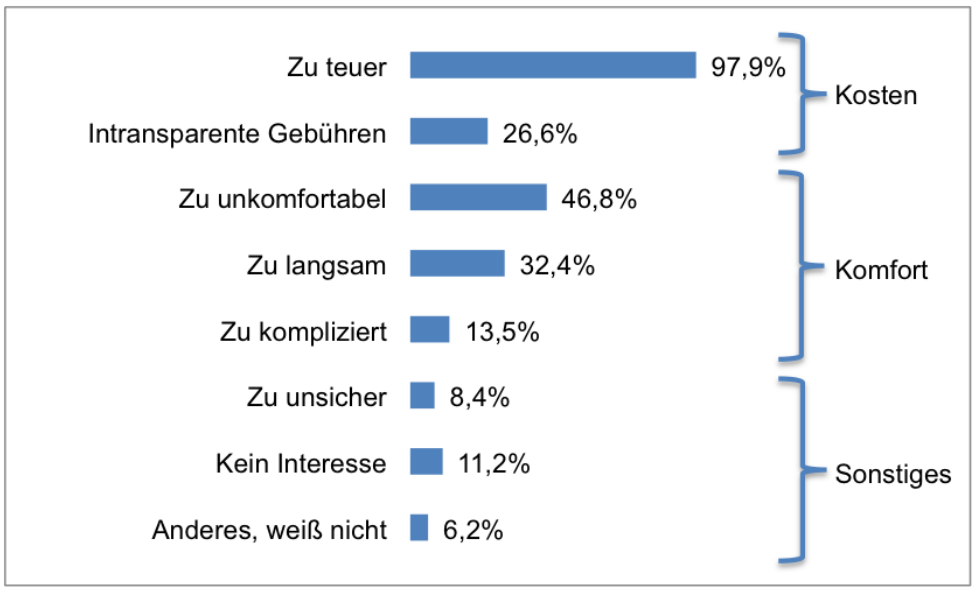


Abbildung 7: Gründe gegen Internet-Nutzung im deutschsprachigen Raum (vgl. Fittkau & Maaß Consulting, 2010)

Wie aus dem Pressebericht „Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down“ von Gartner (2010b) hervorgeht, werden Kommunikationsdienstleister ihre Datentarife für die mobile Internetnutzung zunehmend transparenter gestalten, wodurch Smartphones für verschiedene Marktsegmente zugänglich werden und den gesättigten Markt dominieren könnten. Dies bedeutet eine Reduktion der Gesamtbetriebskosten und spricht neue Benutzerinnen/Benutzer für das mobile Internet an. (vgl. Gartner, 2010b) Eine höhere mobile Internetnutzung aufgrund geringerer Kosten bestätigt auch der „Telecommunications Report 2009“ der EU-Kommission, wodurch Österreicherinnen/Österreicher EU-weit am häufigsten das mobile Internet benutzen (vgl. Schmiedl et al., 2009b, S. 1). Tanner & Hofstetter (2008. S. 12) meinen, dass zukünftig Mobilfunktechnologien IP-basiert sein werden und dadurch die Kosten pro übertragenes Bit weiter reduziert werden können. Dies ermöglicht, dass neue Dienstleistungen und Applikationen für Benutzerinnen/Benutzer kostengünstiger zur Verfügung stehen. (vgl. Tanner & Hofstetter, 2008, S. 12)

Das Benutzerverhalten hat sich aufgrund der vielfältigen Einsatzmöglichkeiten von Smartphones erheblich verändert. Eine Studie von Zokem (2010a) gibt Aufschluss darüber, dass Smartphones über den ganzen Tag verteilt kontinuierlich genutzt werden.

> *... we can confidently say that smartphone is a pervasive medium – people use it everywhere and all the time (Zokem, 2010a)*

Kommunikationsdienste, wie Telefonieren, SMS (Short Message Service), Adressbuch, Anrufliste, soziale Netzwerke und E-Mail werden vorallem am Tag in Anspruch genommen. Am Abend dominieren andere Funktionen des Smartphones, wie zum Beispiel Kamera, Musik, Webbrowser, Karten, Wetter und Spiele. (vgl. Zokem, 2010a)

Die reichhaltigen Smartphone Funktionalitäten sind dafür verantwortlich, dass neben dem Telefonieren und SMS schreiben mobile Applikationen immer mehr an Bedeutung gewinnen (vgl. Nielsen, 2010). Für Computer Anwenderinnen/Anwender wird es immer selbstverständlicher, dass verschiedenste Web Anwendungen auch mobil nutzbar und überall erreichbar sind. Mobile Anwendungen werden zu einem unverzichtbaren Bestandteil von Smartphones, wodurch die Entwicklung mobiler Applikationen immer bedeutsamer wird und ein neues Segment in der Software Branche repräsentiert. (vgl. Hence, 2010, S. 15).

3 Smartphone Plattformen

In diesem Kapitel werden die vier Smartphone Plattformen mit dem höchsten Absatz 2010 genauer erläutert. Bei jeder Plattform wird im Detail auf die Geschichte, die native Programmiersprache, den mobilen Standard Webbrowser sowie auf den Markt zum Vertrieb der mobilen Applikationen eingegangen. Diese vier Plattformen werden beim Vergleich der plattformübergreifenden Frameworks im weiteren Verlauf der Diplomarbeit herangezogen.

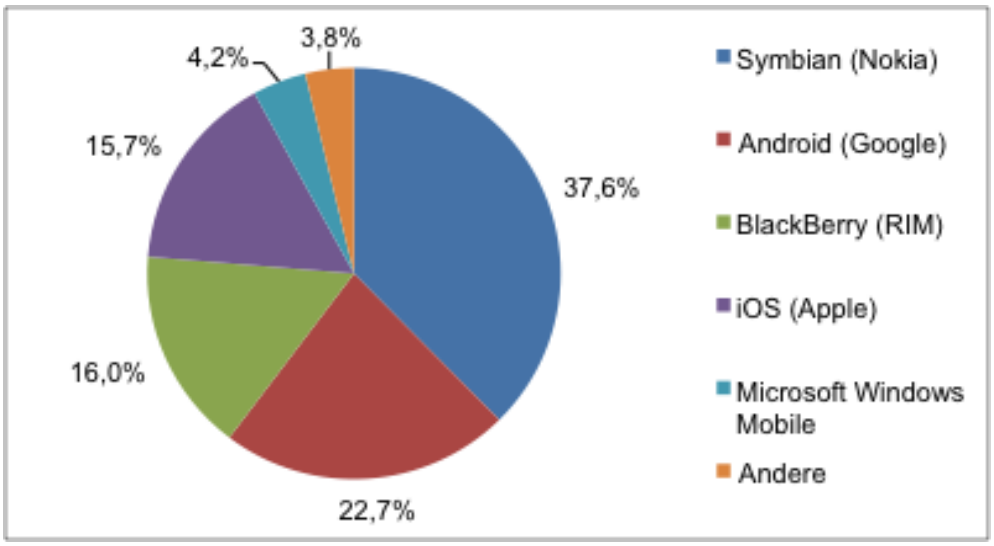


Abbildung 8: Weltweiter Absatz Smartphone Plattformen 2010 (vgl. Gartner, 2011a)

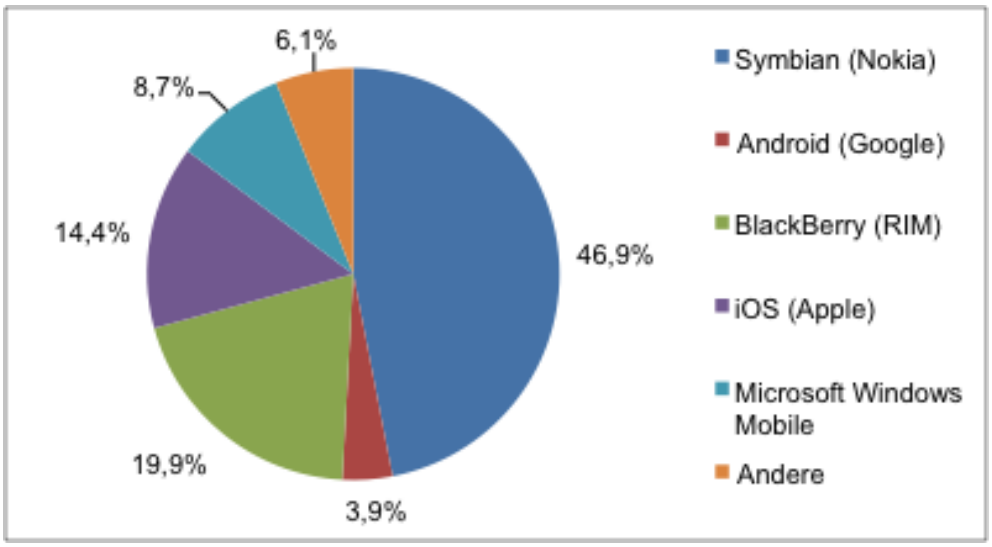


Abbildung 9: Weltweiter Absatz Smartphone Plattformen 2009 (vgl. Gartner, 2011a)

Gartner (2010c) hat den weltweiten Absatz von Smartphones im Jahr 2010 nach den Plattformen gemessen. Wonach in Abbildung 8 ersichtlich wird, dass Symbian (Nokia) mit 37,6 % an erster Stelle liegt, gefolgt von Android (Google) mit 22,7 %, BlackBerry (RIM) mit 16,0 % und iOS (Apple) mit 15,7 %. Microsoft Windows Mobile stellt mit 4,2 % einen geringen Marktanteil gegenüber den vier führenden Plattformen dar. (vgl. Gartner, 2011a). Im Vergleich zum Jahr 2009 (vgl. Abbildung 9) hat Symbian 9,3 % verloren, Android 18,8 % gewonnen, BlackBerry 3,9 % verloren und iOS 1,3 % gewonnen (vgl. Gartner, 2011a).

3.1 Symbian

Die Geschichte von Symbian kann bis in das Jahr 1980 zurückgeführt werden, als David Potter die Firma Psion gründete, die Software für Sinclair PCs entwickelte. 1984 brachte Psion weltweit den ersten Handheld Computer, den sogenannten Psion Organiser, auf den Markt. Psion begann für die PDA (Personal Digital Assistant) Produkte mit der Entwicklung des Betriebssystems EPOC im Jahre 1987. In den Jahren 1989 und 1997 wurden weitere Versionen des EPOC Betriebssystems veröffentlicht und im Juni 1998 mit der Gründung der unabhängigen Firma Symbian Ltd. (Limited) in das Symbian Betriebssystem umbenannt. Die Firma Symbian Ltd. war ein Gemeinschaftsunternehmen, an welchem Psion und große Mobiltelefonhersteller (Ericsson, Motorola und Nokia) beteiligt waren. (vgl. Symbian, 2010a) Das Unternehmen nahm später Samsung und Sony Ericsson als weitere Mitglieder auf (vgl. Firtman, 2010, S. 20). Das Unternehmen Symbian Ltd. fokussierte sich auf die Umsetzung von mobilen Plattformen für Smartphones sowie auf die Lizenzierung des Symbian OS (Operating System) für Hersteller moderner Mobiltelefone (vgl. Fitzek et al., 2010, S. 4). Im Juni 2001 konnte mit der ersten Symbian OS Version 6.0 auf dem Nokia 9210 Communicator (vgl. Abbildung 10) der mobile Markt erreicht werden (vgl. Morris, 2007, S. 15).

Abbildung 10: Nokia 9210 Communicator (vgl. Wikipedia 2005)

Der Zugriff auf verschiedenste Dienste und Technologien verbesserte sich mit jeder Symbian OS Version. Symbian OS v6.0 unterstützte Bluetooth, GSM und GPRS, gefolgt von v7.0 mit IPv4/v6 und EDGE Funktionalität, v8.0 mit Unterstützung von 3G und v9.0 mit integriertem WLAN. (vgl. Morris, 2007, S. 15 - 16)

Aufgrund der unterschiedlichen Mobiltelefon-Modelle der Hersteller und der damit verbundenen unterschiedlichen Bedienung dieser Geräte mit Tastatur oder Stift, begann Symbian mit der Entwicklung verschiedener User Interfaces (UI). Im Symbian OS v6.0 waren zwei UI inkludiert. Crystal, welches mit dem Nokia 9210 Communicator ausgeliefert wurde und später Nokia Series 80 genannt wurde sowie Quartz, das UI von Motorola und Sony Ericsson Mobiltelefonen, aus welchem sich das UIQ UI entwickelte. (vgl. Morris, 2007, S. 32) Ein weiterer UI Standard von Symbian auf dem japanischen Markt lautet MOAP (Mobile Oriented Applications Platform), entwickelt von NTT DOCOMO. Die Smartphone Linie von Nokia begann mit der S60 Plattform, die früher als Series 60 UI bekannt war. Dieses UI läuft auch auf einigen nicht-Nokia Geräten, wie beispielsweise Samsung. Alle S60 Geräte basieren auf Symbian OS, unterstützen Multitasking (mehrere Applikationen können gleichzeitig verwendet werden) und sind mit einer integrierten Kamera, einem mobilen Webbrowser sowie mit einer numerischen oder QWERTZ (deutsche Tastaturbelegung, Abbildung 10) Tastatur ausgestattet. Die 5th Edition von S60 verfügt weiters über eine berührungssensitive Oberfläche. (vgl. Firtman, 2010, S. 20 - 21)

Am 24. Juni 2008 gaben Nokia, Sony Ericsson, Motorola und NTT DOCOMO bekannt, Symbian OS und die dazugehörigen User Interfaces S60, UIQ und MOAP zu einer offenen Plattform zu vereinigen. Gemeinsam mit AT&T, LG Electronics, Samsung Electronics, STMicroelectronics, Texas Instruments and Vodafone wurde die Gründung der gemeinnützigen Symbian Foundation geplant. Um die Gründung der gemeinnützigen Symbian Foundation zu ermöglichen, entschloss sich Nokia das Unternehmen Symbian Ltd. zu 100 % zu erwerben. Ab Anfang Februar 2010 war die Plattform für alle frei verfügbar (vgl. Symbian, 2010b) und für alle Mitglieder der Foundation lizenzfrei. (vgl. Nokia, 2008) Das Ziel der Symbian Foundation war es, mit der quelloffenen Plattform eine intuitive, produktive und kreative Zusammenarbeit und Weiterentwicklung der Plattform zu fördern sowie die Anzahl der Symbian-basierten Smartphones zu steigern (vgl. Symbian, 2010a).

Am 8. November 2010 gab die Symbian Foundation bekannt, dass sie zukünftig nur mehr für die Lizenzierung der Symbian Plattform verantwortlich sein wird (vgl.

Symbian, 2010c). Am 17. Dezember 2010 wurden alle öffentlichen Symbian Foundation Websites und das damit verbundene Wiki und Source Code Repository vom Netz genommen (vgl. Symbian Developer, 2010d). Nokia wird die Entwicklung der Symbian Plattform künftig unter Ausschluss der Öffentlichkeit und mit eigenen Ressourcen fortsetzen (vgl. Nokia, 2010a).

Bevor die Symbian Foundation gegründet wurde, lieferte Symbian nur den Betriebssystemkern aus und dieser wurde von den Mobiltelefon-Herstellern mit ihrem eigenen UI erweitert. (vgl. Fitzek et al., 2010, S. 15) Symbian^1 bildet die Basis für die Plattform der Symbian Foundation und ist eine Kopie der S60 5th Edition Plattform, die auf der Symbian OS Version 9.4 aufbaut (vgl. Symbian Developer, 2010a). Symbian^2 ist die erste offene Version der Symbian Plattform ohne Lizenzgebühr und erweitert Symbian^1 mit einer Reihe neuer Funktionalitäten. Ein großer Teil des Quellcodes war allerdings nur für Symbian Foundation Mitglieder verfügbar. Nach der Auslieferung einiger japanischer Smartphones mit Symbian^2, wurde im September 2010 bekannt gegeben, dass diese Version nicht mehr weiterentwickelt wird. (vgl. Symbian Developer, 2010b) Im Juni 2010 wurde die aktuellste Version der Symbian Plattform – Symbian^3 – offiziell fertig gestellt. Nokia N8 (vgl. Abbildung 11) ist das erste Smartphone, das auf Symbian^3 basiert. (vgl. Symbian Developer, 2010c)

Abbildung 11: Nokia N8-00 (vgl. Forum Nokia, 2011b)

Native Applikationen für die Symbian^3 Plattform können mit Symbian C++ oder Qt erstellt werden. Symbian C++ ist eine speziell für die Symbian Plattform angepasste Version der C++ Programmiersprache und erfordert eine lange Lernkurve, da spezielle Techniken, beispielsweise für die Speicherfreigabe, angewendet werden müssen. (vgl. Virkus, 2011, S. 67) Java ME (Micro Edition) oder Python ermöglichen ebenfalls die Entwicklung von nativen Applikationen. Für diese sind jedoch die nativen Symbian APIs nicht verfügbar, die einen Zugriff auf verschiedene Gerätefunktionen ermöglichen (vgl. Virkus, 2010, S. 27). Am 21. Oktober 2010 kündigte Nokia an, sich für die

Erstellung von nativen Applikationen auf Qt zu fokussieren, um eine plattformübergreifende Entwicklung für Symbian und MeeGo Plattformen zu ermöglichen sowie die Applikationserstellung für Entwicklerinnen/Entwickler transparenter zu gestalten (vgl. Nokia, 2010b). MeeGo ist eine Open Source Plattform von Nokia und Intel, die auf Linux basiert (vgl. MeeGo, 2011). Qt basiert auf Standard C++ und soll laut Nokia (2010b) zukünftig das einzige Framework für die Entwicklung von nativen Applikationen sein. Qt SDK 1.0 ist derzeit die aktuellste Version (Qt SDK 1.1 beta steht ebenfalls zum Download zur Verfügung) und beinhaltet alle notwendigen Werkzeuge sowie APIs für die Realisierung von nativen Qt Applikationen (vgl. Forum Nokia, 2011a). Qt ermöglicht außerdem über das QtWebkit Modul, eine HTML Browser Engine, die Einbindung von Webinhalten in native Qt Applikationen sowie den Zugriff auf native Komponenten mit JavaScript (vgl. Qt Reference Documentation, 2011). Weiters ermöglichen die sogenannten WRT (Web Runtime) Widgets die Erstellung von installierbaren Applikationen für Symbian^3 mit ausschließlich Web Technologien. WRT Widgets können auch als MiniView auf dem Home-Bildschirm eines Symbian^3 Smartphones permanent angezeigt werden. Dies eignet sich besonders gut für die Anzeige von Informationen die ständig aktualisiert werden, wie beispielsweise Nachrichten aus Tageszeitungen oder sozialen Netzwerken. Über die Nokia JavaScript API können verschiedene Ressourcen, wie Kontakte, Kalender, Kamera oder Geolocation angesprochen werden. (vgl. Firtman, 2010, S. 403 - 406)

2005 brachte Nokia einen auf WebKit basierten Open Source Webbrowser für die S60 Plattform auf den Markt, der seit diesem Zeitpunkt auf allen Nokia Smartphones installiert wurde. Der Webbrowser ermöglicht die Darstellung von normalen Desktop Websites sowie die Verkleinerung/Vergrößerung dieser (vgl. Firtman, S. 48). Auf Symbian^3 läuft derzeit der aktuellste S60 Browser 7.2 mit der Unterstützung von HTML, WAP 2.0 und Flash Lite 4.0 (vgl. Forum Nokia, 2011b). HTML5 wird derzeit von diesem Webbrowser noch nicht unterstützt. Der HTML5 Test von Leenheer (2010) auf einem Nokia N8-00 Smartphone mit Symbian^3 erreicht nur 29 von 400 Punkte. Nokia plant jedoch in diesem Jahr ein Symbian Upgrade, das unter anderem einen neuen Webbrowser mit HTML5 Unterstützung beinhaltet (vgl. Fakhre, 2010). Der Acid3 Test vom Web Standards Project dient zur Überprüfung der Webbrowser, ob diese den Web Standards von W3C entsprechen (vg. Acid3, 2011). Der CSS3 Selektoren Test überprüft die Kompatibilität der Webbrowser auf die Unterstützung von CSS3 Selektoren (vgl. CSS3 Selectors Test, 2011). CSS3 Selektoren ermöglichen das Ansprechen von Elementen, ohne die Vergabe eines eignen Klassen- (`class=""`) oder

ID-Selektors (`id=""`) (vgl. Çelik, 2009). Tabelle 2 zeigt alle Ergebnisse der mobilen Webbrowser Tests auf einem Symbian^3 basierten Nokia N8 -00 Smartphone.

Mobile Webbrowser Tests	Ergebnisse
HTML5 Test	29 / 400
Acid3 Test	33 / 100
CSS3 Selektor Test	23 / 41

Tabelle 2: Symbian Webbrowser Tests

2009 (vgl. Eddy, 2009) führte Nokia zum Vertrieb von Series 40, S60 und Symbian^3 Applikationen den Nokia Ovi Store auf dem Markt ein. Qt, Symbian C++ und Java ME Applikationen müssen für die Installation auf dem Smartphone und den Vertrieb über Ovi Store signiert werden, dies ist für Entwicklerinnen/Entwickler seitens Nokia kostenlos. WRT Applikationen sind von der Signierung ausgenommen. Damit Entwicklerinnen/Entwickler den Ovi Store zur Signierung und zum Vertrieb ihrer Applikationen benützen können, ist eine Registrierung und einmalige Gebühr von einem Euro zu bezahlen. Wenn Benutzerinnen/Benutzer eine Applikation mit Kreditkarte bezahlen, erhalten die Entwicklerinnen/Entwickler 70 % vom Verkaufspreis, bei Kauf auf Rechnung erhalten die Entwicklerinnen/Entwickler 60 % vom Erlös. (vgl. Virkus 2011, S. 65) Applikationen können über die mobile Ovi Applikation oder von der Ovi Store Desktop Website auf Symbian Smartphones heruntergeladen und installiert werden (vgl. Ovi Store, 2011). Mit Stand 12. April 2011 sind im Ovi Store über 40.000 Applikationen verfügbar (vgl. Forum Nokia, 2011c).

3.2 Android

Die Android Open Source Plattform basiert auf dem Linux Kernel 2.6 und steht seit 2007 zur Verfügung. Android ist ein Projekt der Open Handset Alliance (OHA), die von Google gegründet wurde. Zu den Mitgliedern der OHA zählen beispielsweise NTT DOCOMO, Dell, HTC, Intel, Motorola, Samsung, Sony Ericsson, LG und T-Mobile. HTC brachte im Oktober 2008 das erste Android Smartphone namens T-Mobile G1 (vgl. Abbildung 12) auf den Markt. Weitere 12 Smartphones folgten im Jahr 2009. Da die Android Plattform von vielen Hardware Herstellern unterstützt wird, stehen mittlerweile einige Smartphones sowie auch Tabletts zur Verfügung. (vgl. Virkus, 2011, S. 17; Allen, 2010, S. 35; Open Handset Alliance, 2011)

Abbildung 12: Erstes Android Smartphone: T-Mobile G1 (vgl. HTC, 2011a)

Android ist eine Multitasking Plattform und die Smartphones sind mit umfangreichen Funktionen, wie GPS, Kompass, Beschleunigungssensor, Kamera, Multi-Touch Oberfläche und QWERTZ Tastatur ausgestattet. Weiters verfügen Android Smartphones über verschiedene Google-spezifische Anwendungen, wie Google Maps, Google Calender und Google Mail. Da es sich bei Android um Open Source Software handelt, könnten die Hersteller theoretisch die Plattform an ihre Bedürfnisse anpassen und beispielsweise alle Google Anwendungen entfernen. Die Anbieter führten dies bis jetzt noch nicht durch. (vgl. Firtman, 2010, S. 26) Am 6. Dezember 2010 veröffentlichte Android die Version 2.3 (Gingerbread), die beispielsweise NFC (Near Field Communication) als neue Funktionalität beinhaltet (vgl. Android Developers, 2010). NFC ermöglicht einen drahtlosen Datenaustausch über kurze Distanzen. Android Applikationen die über NFC verfügen, können NFC Tags, die beispielsweise in Postern, Aufkleber, Werbung oder andere Geräte eingebettet sind lesen und darauf reagieren. (vgl. Android Developers, 2011f) Die aktuellste Version 2.3.3 wurde am 9. Februar 2011 bekannt gegeben und beinhaltet kleine Verbesserungen, zum Beispiel für NFC (vgl. Android Developers, 2011a). Die Version 3.0 entwickelte Android speziell für Tablets und andere Geräte mit größeren Bildschirmen (vgl. Virkus, 2011, S. 17).

Java ist die Programmiersprache für die Erstellung von nativen Android Applikationen. Die Android Plattform unterstützt jedoch nur einen Teil der Java Bibliotheken und verfügt über viele plattformspezifische APIs. Java ME wird von Android nicht unterstützt. (vgl. Virkus, 2011, S. 18) Die Android SDK beinhaltet alle notwendigen Werkzeuge, beispielsweise ein Plug-In für Eclipse sowie einen Emulator und APIs für die Entwicklung von nativen Applikationen (vgl. Android Developers, 2011b). Android ermöglicht, wie Nokia mit ihrem QtWebkit Modul über die Klasse `WebView` die Integration von Webinhalten in native Applikationen und den Zugriff von JavaScript auf native APIs (vgl. Android Developers, 2011e).

Abbildung 13: HTC Nexus One (vgl. HTC, 2011b)

Der auf Android vorinstallierte mobiler Webbrowser ist dem Safari Browser auf der iOS Plattform sehr ähnlich. Der Android Webbrowser basiert ebenfalls auf der WebKit Browser Engine, der ein vollständiger HTML Webbrowser ist und die Darstellung von normalen Desktop Websites ermöglicht. (vgl. Firtman, 2010, S. 26) Weiter unterstützt Androids Webbrowser bereits einige HTML5 Funktionen, wie Geolocation, Offline Web Applications, Video und Web SQL Database. Der HTML5 Test von Leenheer (2010) auf einem HTC Nexus One Smartphone (vgl. Abbildung 13) mit der Android Version 2.2.1 erreichte 182 von 400 Punkte. Tabelle 3 zeigt alle Ergebnisse der mobilen Webbrowser Tests auf einem Android Version 2.2.1 basierten HTC Nexus One Smartphone.

Mobile Webbrowser Tests	Ergebnisse
HTML5 Test	182 / 400
Acid3 Test	93 / 100
CSS3 Selektor Test	40 / 41

Tabelle 3: Android Webbrowser Tests

Seit dem 22. Oktober 2008 steht der Android Market für Benutzerinnen/Benutzer von Android Smartphones zur Verfügung. Für den Vertrieb von Android Applikationen über Android Market müssen Entwicklerinnen/Entwickler diese signieren. Die Entwicklerinnen/Entwickler können nach Registrierung und Bezahlung einer einmaligen Gebühr von 25 Dollar ihre Android Applikationen über den Android Market vertreiben. Sie erhalten 70 % vom Verkaufspreis, die restlichen 30 % müssen als Transaktionsgebühr dem Android Market weiter gegeben werden. (vgl. Android Developers, 2008) Da Android eine freie Plattform ist, können die Entwicklerinnen/Entwickler ihre Applikationen auch über andere Märkte zur Verfügung stellen (vgl. Virkus, 2011, S. 22; WIP Connector, 2010). Applikationen können entweder über die Android Market Applikation, die auf vielen Android Smartphones

vorinstalliert ist, oder von jedem Desktop Webbrowser über den neuen Android Market Webstore (vgl. Android Market, 2011) heruntergeladen und installiert werden. Über Desktop Webbrowser ausgewählte Applikationen werden – nach Anmeldung mit dem Google Benutzerkonto – automatisch auf dem Android Smartphone heruntergeladen und installiert, ohne das Smartphone über ein USB Kabel an den PC anschließen zu müssen. (vgl. Savov, 2011) Mit Stand 03. März 2010 stehen laut AndroLib (2011) über 250.000 Applikationen im Android Market zum Download bereit.

3.3 BlackBerry

Das kanadischen Unternehmen Resarch in Motion (RIM) entwickelte die proprietäre Multitasking Plattform und führte diese 1999 auf dem Markt ein (vgl. Virkus, 2011, S. 27). 2002 wurde das erste auf Java ME basierte GSM/GPRS BlackBerry Smartphone 5810 (vgl. Abbildung 14) veröffentlicht (vgl. Duwe, 2002).

Abbildung 14: Erstes BlackBerry Smartphone 5810 (vgl. Duwe, 2002)

BlackBerry Smartphones gewannen durch die komfortable Eingabe auf einer vollständig vorhandenen QWERTZ Tastatur und der langen Akkulaufzeit an Bedeutung. Vorallem bei Geschäftsleuten wurden BlackBerry Smartphones aufgrund verschiedenster Applikationen für E-Mail, Kontakte, Kalender, Aufgaben und Notizen, sehr beliebt. (vgl. Virkus, 2011, S. 27) Weiters besteht die Möglichkeit diese Applikationen über den sogenannten BlackBerry Enterprise Server mit Microsoft Exchange, IBM Lotus Domino oder Novell GroupWise zu synchronisieren, wodurch BlackBerry auch in Unternehmen sehr populär wurde (vgl. BlackBerry Software, 2011a).

Das aktuellste BlackBerry Smartphone Torch 9800 basiert auf dem neuen BlackBerry Betriebssystem 6 und ist mit einer Multi-Touch Oberfläche, einer ausziehbaren QWERTZ Tastatur sowie einem Trackpad ausgestattet. Weiters verfügt das Smartphone über WLAN, eine Kamera, 3G, GPS, Beschleunigungssensoren und einen auf Webkit basierten Webbrowser. (vgl. Thornton & Schmit, 2010, S. 6 - 7)

BlackBerry bietet zwei Möglichkeiten zur Entwicklung von nativen Applikationen an: Java Application Development oder BlackBerry WebWorks Development. Das Java Application Development ermöglicht die Erstellung von nativen Applikationen mit Java ME und den BlackBerry APIs. Dafür kann das BlackBerry Java Development Environment (JDE) (vgl. BlackBerry Developers, 2011a) oder das BlackBerry Plug-In für die Eclipse IDE (vgl. BlackBerry Developers, 2011b) herangezogen werden. JDE ist eine Entwicklungsumgebung von BlackBerry, die einen BlackBerry Smartphone Simulator sowie Java ME und BlackBerry APIs enthält (vgl. BlackBerry Documentation, 2010c). Webinhalte können über das Browser Field einer nativen Java Applikation hinzugefügt werden und ermöglichen die Kommunikation mit nativen Java APIs (vgl. BlackBerry Developers, 2011c). Das BlackBerry WebWorks Development ermöglicht den Entwicklerinnen/Entwicklern die Realisierung von nativen Applikationen mit Standard Web Technologien. Hierfür kann entweder das BlackBerry WebWorks Plug-In für Eclipse, Microsoft Visual Studio oder eine andere bevorzugte IDE verwendet werden. (vgl. BlackBerry Developers, 2011d) BlackBerry WebWorks Applikationen können mit der BlackBerry WebWorks API auf native BlackBerry APIs und somit auf verschiedene gerätespezifische Informationen zugreifen (vgl. BlackBerry Developers, 2011e).

Abbildung 15: Aktuelles BlackBerry Torch 9800 (vgl. Apple, 2011b)

Der Standard Webbrowser von BlackBerry Smartphones basiert seit der Version 6.0 auf der WebKit Browser Engine und unterstützt die Anzeige von vollständigen Desktop Website (vgl. BlackBerry Documentation, 2010d). Einige HTML5 Funktionen werden ebenfalls bereits vom BlackBerry Webbrowser implementiert (vlg. BlackBerry Documentation, 2010b). Der HTML5 Test von Leenheer (2010) auf einem BlackBerry Torch 9800 (vgl. Abbildung 15) mit Version 6.0 erreichte 233 von 400 Punkte. Tabelle 4 zeigt alle Ergebnisse der mobilen Webbrowser Tests auf einem BlackBerry Torch 9800 mit der Version 6.0.

Mobile Webbrowser Tests	Ergebnisse
HTML5 Test	233 / 400
Acid3 Test	100 / 100
CSS3 Selektor Test	40 / 41

Tabelle 4: BlackBerry Webbrowser Tests

Nachdem eine native Applikation signiert wurde (vlg. BlackBerry Developers, 2011f), können diese nach kostenloser Erstellung eines Lieferantenkontos und Bestätigung dieses Kontos durch BlackBerry über BlackBerrys App World vertrieben werden. Entwicklerinnen/Entwickler erhalten 70 % vom Verkaufspreis. (vgl. BlackBerry Developers, 2011g) Native Applikationen können über die mobile Applikation App World (vgl. BlackBerry Software, 2011b) direkt auf das Smartphone heruntergeladen und installiert werden. Weiters besteht die Möglichkeit native Applikationen entweder über die App World Desktop Website (vgl. BlackBerry App World, 2011) oder BlackBerry Desktop Software (vgl. BlackBerry Services, 2011), nachdem das Smartphone über USB mit dem PC verbunden wurde, herunter zu laden und zu installieren. Mit Stand 14. Februar 2011 sind in der BlackBerry App World über 20.000 Applikationen verfügbar (vgl. Marketwire, 2011).

3.4 iOS

Die iOS Plattform von Apple basiert auf Mac OS X, ein UNIX basiertes Desktop Betriebssystem, und ist speziell auf die mobilen Geräte von Apple (iPhone, iPod touch und iPad) ausgerichtet. Bis zur Version 4.0 wurde iOS als iPhone OS bezeichnet. Derzeit gibt es vier Generationen des iPhones, die von der ersten Generation an mit WLAN, Kamera, Multi-Touch Oberfläche, einer QWERTZ Tastatur und Safari als mobiler Webbrowser ausgestattet sind. (vgl. Firtman, 2010, S. 17) Apples CEO (Chief Executive Officer) Steve Jobs präsentierte das erste GSM iPhone (vgl. Abbildung 16)

am 9. Jänner 2007, das zunächst in den USA später in Europa auf den Markt gebracht wurde (vgl. Honan, 2007).

Abbildung 16: Erstes iPhone 2007 (vgl. Mittwich, 2009)

Gefolgt von iPhone 3G mit 3G als Mobilfunkübertragungstechnik sowie GPS Unterstützung (vgl. Apple, 2008). Die nächste Generation, das iPhone 3GS (S für Speed) verfügt über einen schnelleren Prozessor und ist zusätzlich mit einem Kompass und einer Kamera mit höherer Auflösung sowie Videofunktion ausgestattet (vgl. Apple, 2011a). Die aktuellste Generation, das iPhone 4 (vgl. Abbildung 17) hat zwei Kameras für FaceTime Videotelefonate, einen LED-Blitz und ein Retina Display für eine hochauflösende Darstellung (vgl. Apple, 2011b)

Apple hat mit jeder neuen Generation auch eine neue Version der Plattform angeboten. Ab iOS 4.0 ist die Plattform Multitasking-fähig (vgl. Apple, 2011c) und ermöglicht die Erstellung von bis zu 180 Ordner mit maximal 12 Applikationen (vgl. Apple, 2011d). Die aktuellste Version iOS 4.3 ist mit weiteren Neuerungen ausgestattet. Beispielsweise enthält der Safari Webbrowser die neue Nitro JavaScript Engine, die JavaScript zweimal schneller als in iOS 4.2. ausführen kann und es besteht die Möglichkeit das iPhone als WLAN Hotspot zu nutzen (vgl. Apple, 2011e).

Der Safari Webbrowser des iPhone unterstützte zunächst nur Web Applikationen und Apple plante keine SDK für die Entwicklung von nativen Applikationen zur Verfügung zu stellen (vgl. Gonsalves, 2007).

> *The full Safari engine is inside of iPhone. And so, you can write amazing Web 2.0 and Ajax apps that look exactly and behave exactly like apps on the iPhone. And these apps can integrate perfectly with iPhone services. They can make a call, they can send an email, they can look up a location on Google Maps.*

> *And guess what? There's no SDK that you need! You've got everything you need if you know how to write apps using the most modern web standards to write amazing apps for the iPhone today. So developers, we think we've got a very sweet story for you. You can begin building your iPhone apps today. (Jobs, 2007)*

Benutzerinnern/Benutzern können Web Applikationen über die Web Apps Website von Apple zur Verfügung stellen (vgl. Apple, 2011f).

Im Oktober 2007 kündigte Apple jedoch an, dass zukünftig eine SDK für Entwicklerinnen/Entwickler zur Verfügung stehen soll (vgl. Fletcher, 2007). Native Applikationen für die iOS Plattform werden mit der Programmiersprache Objective-C entwickelt. Objective-C ist eine auf der Programmiersprache C basierte objektorientierte Sprache. (vgl. Virkus, 2011, S. 42) Für die Erstellung von nativen iOS Applikationen wird das iOS SDK benötigt, das nach kostenloser Registrierung von der Apple Developer Website (vgl. Apple Developer, 2011a) heruntergeladen werden kann. Das iOS SDK beinhaltet Xcode als IDE, einen Interface Builder zur Gestaltung des User Interface einer Applikation, verschiede Instrumente zur Überwachung der Applikation während der Ausführung, einen iOS Simulator zum Testen einer nativen Applikation und eine Vielzahl von APIs. (vgl. Virkus, 2011, S. 39) Wie bereits bei Symbian und Android erwähnt ermöglicht Objective-C ebenfalls mit der Klasse `UIWebView` die Einbindung von Webinhalten in native Applikationen und den Zugriff auf native Komponenten über JavaScript (vgl. Apple Developer, 2011d).

Abbildung 17: Aktuelles iPhone 4 (vgl. Apple, 2011b)

Der Standard Webbrowser der iOS Plattform, genannt Safari, basiert auf der WebKit Browser Engine und ermöglicht die Anzeige von normalen HTML Desktop Websites,

die durch ein doppeltes Antippen vergrößert bzw. verkleinert werden können. Safari unterstützt bereits einige HTML5 Funktionen, wie Geolocation, Offline Web Applications, Video und Web SQL Database. Der HTML5 Test von Leenheer (2010) auf einem iPhone 3GS mit iOS 4.3.2 erreichte 206 von 400 Punkte. Tabelle 5 zeigt alle Ergebnisse der mobilen Webbrowser Tests auf einem iPhone 3GS mit iOS 4.3.2.

Mobile Webbrowser Tests	Ergebnisse
HTML5 Test	206 / 400
Acid3 Test	100 / 100
CSS3 Selektor Test	41 / 41

Tabelle 5: iOS Webbrowser Tests

Apple kündigte 2008 den App Store für den Vertrieb von nativen iOS Applikationen an. Entwicklerinnen/Entwickler müssen jährlich eine Gebühr von 99 Dollar bezahlen, um Applikationen auf dem App Store zur Verfügung zu stellen. Sie erhalten 70 % vom Verkaufspreis, die restlichen 30 % müssen an Apple abgeführt werden. Die Applikationen werden von Apple überprüft und freigegeben, dies dauert ca. zwei Wochen. Native Applikationen können aus verschiedensten Gründen von Apple abgewiesen werden. AppRejections (2011) listet die häufigsten Gründe für die Ablehnung einer Applikation. Apple stellt einige Leitfäden für Entwicklerinnen/Entwickler zur Verfügung, in welchen erläutert wird, wie native Applikationen auszusehen haben um von Apples App Store akzeptiert zu werden. iOS Applikationen können über die mobile Applikation App Store direkt auf das iPhone heruntergeladen und installiert werden. Auch über Appels Mediaplayer iTunes besteht die Möglichkeit native Applikationen zu erwerben und auf das angeschlossene iPhone zu synchronisieren. Mit Stand 22. Jänner 2011 sind über 350.000 Applikationen im App Store verfügbar (vgl. Apple, 2011g). Tabelle 6 gibt einen abschließenden Überblick über alle Plattformen.

	Symbian	Android	BlackBerry	iOS
Anbieter	Nokia	Google	RIM	Apple
Aktuelle Version	^3	2.3.3	6.0	4.3.2
Native Programmiersprache	Qt C++	Java	Java ME	Objective-C
Webbrowser	WebKit	WebKit	WebKit	WebKit
Store	Ovi Store	Android Market	App World	App Store

Tabelle 6: Übersicht über alle Plattformen

4 Mobile Applikationen

> *Mobile applications or apps are compact software programs that perform specific tasks for the mobile user (mobiThinking, 2010).*

Mobile Applikationen können in drei Arten – Nativ, Web und Hybrid – unterteilt werden. Nachfolgend wird näher auf die Unterschiede dieser drei Arten von mobilen Applikationen eingegangen.

4.1 Native Applikationen

Native Applikationen sind eigenständige Programme, die für ein bestimmtes Smartphone Betriebssystem, wie Symbian, Android, BlackBerry oder iOS programmiert und kompiliert werden. Native Applikationen werden nicht in einem mobilen Webbrowser aufgerufen, sondern müssen auf das Smartphone heruntergeladen und installiert werden, sie laufen somit lokal auf dem Smartphone. Ein großer Vorteil von nativen Applikationen ist die Möglichkeit, dass diese auf verschiedenste Hardwarefunktionen eines Smartphones zugreifen können. Hierbei handelt es sich beispielsweise um die Kamera oder den Bewegungssensor eines Smartphones (vgl. Spiering & Haiges, 2010, S. 8).

Die Einführung des iPhone App Store löste den Hype um die sogenannten Apps aus und im Zuge dessen hat sich der mobile Markt stark verändert (vgl. Hausmann, 2010, S. 14; Spiering & Haiges, 2010, S. 5).

> *Internetdienste werden von einer großen Masse wie selbstverständlich genutzt, und der mobile Markt hat sich, anstatt sich auf eine oder einige wenige Plattformen zu konsolidieren, in Höchstgeschwindigkeit weiter fragmentiert (Spiering & Haiges, 2010, S. 5).*

Marktforscher aus den USA erwarten im Jahr 2010, dass Benutzerinnen/Benutzer für native Applikationen 6,8 Milliarden Franken (= 5,1 Milliarden Euro, Stand: 16.10.2010) ausgeben werden. In drei Jahren wird der Umsatz voraussichtlich auf 32,2 Milliarden Franken (= 24,0 Milliarden Euro, Stand: 16.10.2010) steigen. (vgl. Hausmann, 2010, S. 14) Weiters liegt der Umsatz der Märkte für Applikationen im Jahr 2010 bei 5,2

Milliarden Dollar und wird im Jahr 2011 auf 15,1 Milliarden Dollar ansteigen (vgl. Gartner, 2011b).

Alle Hersteller möchten von diesem lukrativen Geschäft profitieren und versuchen mit ihrem Betriebssystem und ihrem eigenen Application Store Kundinnen und Kunden für sich zu gewinnen. Daraus resultiert ein unübersichtlicher mobiler Markt mit einer Vielfalt an unterschiedlichen Plattformen, die für Firmen eine Herausforderung darstellen. Da beispielsweise native iPhone Applikationen nicht auf Android Smartphones und umgekehrt laufen, müssen Firmen ihre Applikationen für jedes Betriebssystem gesondert entwickeln, um alle potenziellen Kundinnen und Kunden zu erreichen und dies bedeutet einen großen Aufwand sowie hohe Kosten (vgl. Hausmann, 2010, S. 14).

Die Accenture „Mobile Web Watch“ - Studie 2010 (2010) zeigt, dass in Deutschland 60 % und in Österreich 63 % der Smartphone Benutzerinnen/Benutzer native Applikationen nutzen. Vorallem auf dem iPhone werden verschiedene native Applikationen von 96 % der deutschen Besitzerinnen/Besitzer verwendet. Dies liegt daran, dass Apple mit seinem App Store gegenüber anderen Anbietern, wie Android Market derzeit an der Spitze liegt. (vgl. Accenture, 2010, S. 30) Im Jahr 2011 prognostiziert Gartner (2011b), dass 17,7 Milliarden Applikationen aus den Application Stores heruntergeladen werden, das bedeutet einen Anstieg von 117 % zum Vorjahr 2010 mit 8,2 Milliarden heruntergeladenen Applikationen. Die durchschnittliche Nutzungsdauer von freien und auch bezahlten nativen iPhone Applikationen geht jedoch nach dem ersten Download rapide bergab. Weniger als 30 % kehren einen Tag nach dem Herunterladen einer Applikationen wieder zu dieser zurück. Nach 30 Tagen verwenden nur mehr weniger als 5 % der Benutzerinnen/Benutzer diese Applikation, nur 1 % verwendet die heruntergeladene Applikation langfristig. (vgl. Schonfeld, 2009)

Der Trend - Report 2010 sieht in den Apps eine Art Werkzeugkasten, mit dessen Hilfe die Benutzerinnen/Benutzer ihr Gerät nach den eigenen Bedürfnissen individuell gestalten können. Das Internet, das eine Flut an Informationen liefert, wird durch Apps für die Anwenderinnen/Anwender überschaubar und einfach verwendbar. Aufgrund der flachen Funktionsstruktur und den geringen Einstellungsmöglichkeiten sind Apps auch für nicht computer-affine Menschen leicht zugänglich (vgl. Dziemba et al., 2009, S. 41 - 45). Tißler (2010) meint, dass Nutzerinnen/Nutzer native Applikationen hochwertiger wahrnehmen und im Gegensatz zu mobilen Web Applikationen zunächst als wichtiger eingestuft werden. Entscheidend für die Nutzung von mobilen Anwendungen ist laut

der Accenture Mobile Web Watch - Studie 2010 (2010, S. 22) ein konkreter Mehrwert für unterwegs. Das Smartphone als Navigationsgerät, standortbezogenes Service, Wetter- oder Nachrichtendienst ist bei Benutzerinnen/Benutzer solcher nativer Applikationen innovativ und beliebt.

Das mobile Web wird laut Senior Analyst Beccue (2010) technisch immer ausgeklügelter, sodass Benutzerinnen/Benutzer die Funktionsvielfalt von mobilen Websites mehr verwenden werden, als gleichbedeutende native Applikationen.

> *We see two emerging trends: first, many applications (increasingly built on web standards) will migrate from app stores to regular websites, and for some sites you won't need an app at all. In addition, more and more popular applications will be preloaded on mobile devices. Social networking apps in particular will be pre-loaded on new products. (Beccue, 2010)*

Zusammengefasst sieht Fling (2010, S. 79) bei nativen Applikationen folgende Vorteil und Nachteile. Ein Vorteil einer nativen Applikation ist die nahtlose Integration in das UI der jeweiligen Plattform, wodurch sich ein unverwechselbares und einzigartiges Benutzererlebnis ergibt. Weitere Vorteile sind die Möglichkeit auf alle Software- und Hardwarefunktionen eines Smartphones zugreifen zu können sowie der Vertrieb und Verkauf einer Applikation über den jeweiligen Store einer Pattform. Für nur eine Plattform ist eine native Applikation recht einfach zu entwickeln. Der große Nachteil liegt darin, dass sich native Applikationen nicht leicht auf andere Plattformen portieren lassen. Die Entwicklung, das Testen und die Unterstützung von verschiedenen Plattformen sind mit einem hohen Aufwand und Kosten verbunden. Weiters erfordert der Vertrieb einer nativen Applikationen über einen der App Stores das Durchlaufen eines Genehmigungsprozesses, der mitunter ein paar Wochen dauern kann und nach jeder Aktualisierung der Applikation durchgeführt werden muss. Der Verkaufserlös muss mit den Betreibern der Plattformen geteilt werden.

4.2 Web Applikationen

Mobile Web Applikationen werden mit Standard Web Technologien (HTML, CSS und JavaScript) erstellt und müssen, im Gegensatz zu nativen Applikationen, nicht auf Smartphones installiert werden, sondern können direkt im mobilen Webbrowser eines Smartphones aufgerufen werden (vgl. Fling, 2010, S. 75).

> *Daher ist die Webanwendung auch nur so gut wie der Browser, in dem sie läuft (Spiering & Haiges, 2010, S. 10).*

Immer mehr mobile Webbrowser verschiedener Plattformen, wie Android, iOS oder BlackBerry basieren jedoch bereits auf der gleichen modernen WebKit Browser Engine, die neueste Web Standards (HTML5, CSS3) unterstützt und eine plattformunabhängige Anzeige von Web Applikationen auf verschiedensten Smartphones ermöglicht (vgl. Spiering & Haiges, 2010, S. 10). YouTube beispielsweise setzt auf eine mobile Web Applikation und verwendet neue Funktionalitäten von HTML5 für die Anzeige von Videos (vgl. Kincaid, 2010).

Eine Studie von Zokem (2010b) ergab, dass der mobile Webbrowser mit Abstand die beliebteste Applikation für Benutzerinnen/Benutzer von Smartphones ist. Der mobile Webbrowser wird mindestens einmal im Monat und durchschnittlich 300 Minuten lang zum Surfen im Internet verwendet. eMarketer (2010) gibt Aufschluss darüber, dass Benutzerinnen/Benutzer aus den USA Inhalte der Kategorien Produkt Rezension, Blogs, Sport, News, Video und lokale Informationen (Events, Wetter etc.) lieber über den mobilen Webbrowser als über native Applikationen abrufen. Bei sozialen Netzwerken, Musik und Spielen werden native Applikationen bevorzugt verwendet.

Hazaël-Massieux (2010), Mitarbeiter des W3C, versteht unter mobilen Web Applikationen im Gegensatz zu einer mobilen Website eine in sich geschlossene Anwendung mit einem interaktiven UI, das einer nativen Applikation ähnlich ist. Weiters ist eine Web Applikation mehr auf Aktion, als auf Information ausgerichtet und es werden verschiedene fortschrittliche Smartphone-Funktionalitäten, wie Geolocation, eingesetzt. Mobile Web Applikationen zeichnen sich auch dadurch aus, Informationen offline zur Verfügung zu stellen.

> *Combining the power of the Web with the strengths of mobile devices (W3C, 2011a).*

Mit dieser Devise arbeitet das W3C an einigen JavaScript API Standards, um zukünftig von mobilen Webbrowser auf verschiedensten Software- und Hardwarefunktionen eines Smartphones zugreifen zu können (vgl. W3C, 2011a). Diese APIs sollen beispielsweise die Aufnahme von Fotos über die Kamera oder den Zugriff auf Kontakte, Kalender und Dateien ermöglichen (vgl. W3C, 2011b). Das W3C (2011c) gibt einen Überblick über den aktuellen Status und weitere Vorgehensweise der

verschiedenen W3C Spezifikationen für mobile Web Applikationen. Weiters stellt das W3C (2010) einen Leitfaden mit den Namen „Mobile Web Application Best Practices Cards“ zur Verfügung, der kurz und prägnant Vorschläge für die Umsetzung von mobilen Web Applikationen beinhalten.

Zusammengefasst sind die Vorteile von mobilen Web Applikationen laut Fling (2010, S. 77) die Möglichkeit diese mit HTML, CSS und JavaScript leicht und mit geringen Kosten entwickeln zu können, sie plattformunabhängig auf verschiedenen Smartphones ausführen zu können und den Inhalt in jedem mobilen Webbrowser erreichbar zu machen. Mobile Web Applikationen müssen keinen Genehmigungsprozess durchlaufen, um für Benutzerinnen/Benutzer zur Verfügung zu stehen, sondern können nach Fertigstellung sofort ins Internet gestellt werden, um weltweit verfügbar zu sein. Weiters besteht die Möglichkeit die Anwendungen jederzeit ohne Neuinstallation aktualisieren zu können und es fallen keine Kosten für die Zertifizierung, wie beispielsweise bei Apples App Store an. (vgl. Spiering & Haiges, 2010, S. 10) Ein großer Nachteil ist, dass mobile Web Applikationen nicht auf alle Funktionen eines Smartphones, wie Kamera, Dateisystem, Kontakte etc. zugreifen können. Weiters ist es schwierig ein optimales Benutzererlebnis auf allen mobilen Geräten zu erreichen. (vgl. Fling, 2010, S. 77) Nachteilig ist auch, dass sich der Verkauf von mobilen Web Applikationen, im Gegensatz zu nativen Applikationen in App Stores, recht umständlich gestalten kann (vgl. Spiering & Haiges, 2010, S. 10).

4.3 Hybride Applikationen

> *A hybrid is a mix between a web and a native application, having the best of both worlds available at the same time (Firtman, 2010, S. 413).*

Die Idee hinter hybriden Applikationen ist die Erstellung eines nativen Grundgerüsts, in welchem die Inhalte mit Standard Web Technologien (HTML, CSS, JavaScript) erstellt werden können. Innerhalb dieses nativen Grundgerüsts kann mit JavaScript auf verschiedene Software- und Hardwarekomponenten (Kontakte, Kamera etc.) eines Smartphones zugegriffen werden.

Das Grundgerüst einer hybriden Applikation kann zum Beispiel aus einer nativen Webbrowser Komponente bestehen. Bei iOS ist dies die `UIWebView` Komponente und bei Android die `WebView` Komponente (vgl. Morrow, 2010). Innerhalb dieses nativen Webbrowsers besteht die Möglichkeit das UI aus nativen plattformspezifischen

Bedienelementen sowie aus plattformunspezifischen CSS Bedienelementen zu erstellen und auf native Funktionen zugreifen zu können. Eine hybride Applikation muss wie eine native Applikation kompiliert werden. Diese ist für Benutzerinnen/Benutzer von einer nativen Applikationen fast nicht zu unterscheiden, da sie auf dem Smartphone installiert wird und über die verschiedenen Application Stores vertrieben werden kann.

Das Hauptmerkmal des Modells hybrider Applikation ist eine Brücke zwischen JavaScript und nativen Gerätefunktionen zu schaffen. Ein Framework für dieses Modell muss nicht erst von Entwicklerinnen/Entwicklern erstellt werden, da bereits einige freie Frameworks auf dem Markt zur Verfügung stehen. (vgl. Shum & Cole, 2010) Beispielsweise können die Entwicklerinnen/Entwickler über die JavaScript APIs der Frameworks PhoneGap und Titanium Mobile plattformübergreifend auf verschiedene Funktionen eines Smartphones zugreifen. PhoneGap und Titanium Mobile verfolgen zwei verschiedene Ansätze. PhoneGap implementiert einen nativen Webbrowser für die Darstellung von Webinhalten sowie den Zugriff auf native Komponenten (vgl. Spiering & Haiges, 2010, S. 237). Titanium Mobile übersetzt das erstellte JavaScript Gerüst für den Zugriff auf native Bedienelemente und Hardware-/Softwarefunktionen in nativen Quellcode (vgl. Appcelerator Wiki, 2011a).

5 Plattformübergreifende Entwicklung mit Web Technologien

Aus Kapitel 2 ist hervorgegangen, dass der Zugriff auf das mobile Internet sowie der Verkauf von Smartphones stetig wachsen. Kapitel 3 ergab, dass jede der vier marktführenden Plattformen eine andere Programmiersprache für die Erstellung von nativen Applikationen verwendet und die Zahl der mobilen Applikationen kontinuierlich ansteigt. Weiters ist die Entwicklung von nativen Applikationen für verschiedene Plattformen, wie in Kapitel 4 erläutert, mit hohem Aufwand und hohen Kosten verbunden. Aufgrund der soeben genannten Faktoren wird eine plattformübergreifende Entwicklung mobiler Applikationen basierend auf Web Technologien immer bedeutsamer. Abbildung 18 veranschaulicht den unterschiedlichen Entwicklungsaufwand für die verschiedenen mobilen Applikationstypen.

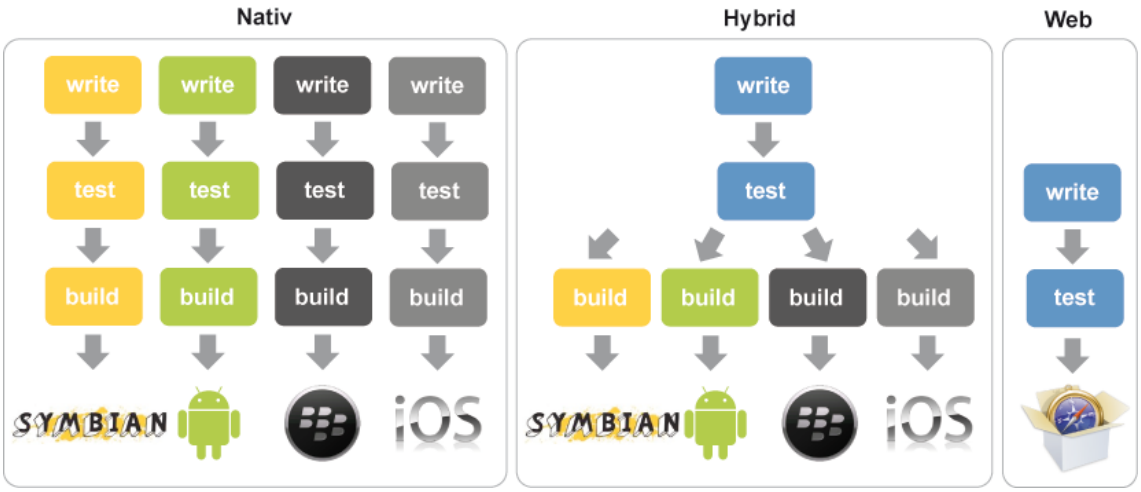


Abbildung 18: Unterschiedlicher Entwicklungsaufwand für die verschiedenen mobilen Applikationstypen (vgl. Jacobs, 2011)

Obwohl die Smartphones der verschiedenen Plattformen über die gleichen Software- und Hardwarefunktionalitäten verfügen, setzt jede Plattform unterschiedliche APIs für den Zugriff auf diese Ressourcen ein. Eine mit Java erstellte BlackBerry oder Android Applikation kann beispielsweise nicht ohne weiteres auf eine iOS Plattform übertragen werden, da verschiedene Programmiersprachen und APIs für den Zugriff auf native Komponenten verwendet werden. (vgl. Dern, 2010) Die Bereitstellung von nativen Applikationen für jede Plattform erfordert hohe Ressourcen hinsichtlich Entwicklerinnen/Entwicklern mit unterschiedlichem Programmiersprachen Know-how und daraus resultierenden hohen Kosten (vgl. Fling, 2010, S. 79).

Eine native Applikation muss für jede Plattform neu entwickelt werden, Web Technologien können hierbei Abhilfe schaffen. Der große Vorteil ist, dass diese auf allen Plattformen unterstützt werden und Entwicklerinnen/Entwickler auf bestehende Kenntnisse zurückgreifen können bzw. relativ schnell erlernbar sind. Web Technologien sind weiters sofort einsatzbereit und verfügen über eine große bereits bestehende Entwicklerinnen/Entwickler Gemeinschaft. Das ist auch der Grund warum einige Plattformen, wie Symbian mit WRT und BlackBerry mit WebWorks (Kapitel 3) für die Entwicklung von nativen Applikationen auch auf Web Technologien setzen. Die Entwicklungskosten für die Erstellung plattformübergreifender Applikationen werden durch Web Technologien ebenfalls reduziert. (vgl. Constantinou, 2010)

> *Der „kleinste gemeinsame Nenner" aller Smartphone-Betriebssysteme ist ein installierter Browser. Das macht web-basierte Applikationen mit Smartphone-optimierter Oberfläche zu einer interessanten Alternative für native Apps. (Aurelio, 2010, S. 114)*

Die Mehrheit der marktführenden Plattformen verfügen über einen mobilen Webbrowser, der, wie in Kapitel 3 bereits erwähnt, auf der sogenannten WebKit Browser Engine basiert. Der Erfolg von WebKit ist auf die exzellente Unterstützung von Web Standards und vielen neuen HTML5, JavaScript API Spezifikationen zurückzuführen. Allerdings verfügt nicht jeder auf WebKit basierte Webbrowser, wie Symbian über die neuesten Web Standards. WebKit ist eine Open Source Browser Engine, die basierend auf der Konqueror HTML Layout Engine und KJS JavaScript Engine von Apple weiterentwickelt wurde. Da bereits viele Plattformen WebKit Browser im Einsatz haben, können sehr ähnliche Darstellungen von Web Applikationen auf verschiedensten Smartphones erwartet werden. (vgl. Fling, 2010, S. 199 – 200; Firtman, 2010, S. 44)

Mit der HTML5 Standardisierung wird versucht eine einheitliche Plattform für mobile Applikationen zu schaffen, die eine komplexe Anwendungserstellung ermöglicht. HTML5 hält vorallem in der mobilen Entwicklung Einzug, da einige W3C Spezifikationen in mobilen Webbrowsern bereits implementiert sind. HTML5 gemeinsam mit den immer leistungsfähigeren mobilen Webbrowsern eröffnen bei der Entwicklung mobiler Web Applikationen neue Möglichkeiten. HTML5 definiert viele neue Elemente, wie `audio` und `video` für die Einbettung multimedialer Inhalte, `input` Elemente mit neuen `type`-Attributen für `url`, `email`, `number`, `tel`, `datetime` und `range` mit automatischer Korrektur über das `autocorrect`-Attribut. Das neue Element `canvas`

ermöglicht die Erstellung von dynamischen Bitmap Grafiken für beispielsweise Animationen oder Diagramme. Auch eine offline Speicherung von Web Applikationen ist über eine sogenannte Cache Manifest Datei möglich. Verschiedene JavaScript API Spezifikationen von W3C für zum Beispiel Geolocation oder lokale Speicherung von Daten stellen eigene Spezifikationen dar und gehören streng genommen nicht zur HTML5 Spezifikation. (vgl. Spiering & Haiges, 2010, S. 83 - 91)

Der Bedarf nach nativen Applikation ist besonders auf den lückelosen Zugriff auf geräteinterne Funktionen zurückzuführen, Global Intelligence Alliance (2010, S. 7) prognostiziert jedoch:

> *... by 2013 the majority of native device attributes are set to reach mobile/HTML5 web applications (as estimated below) while enabling user experiences that rival those of native applications:*

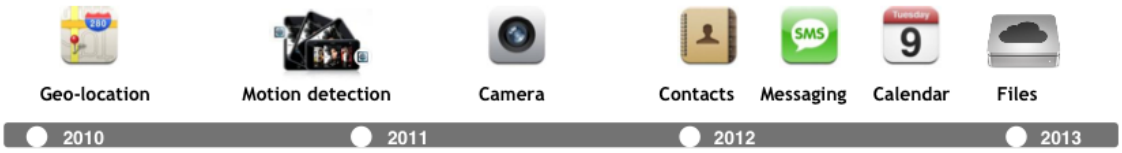


Abbildung 19: Zugriff auf die meisten Gerätefunktionen innerhalb mobiler Webbrowser in den nächsten drei Jahren (Global Intelligence Alliance, 2010, S. 7)

Weiters hat die Studie von Global Intelligence Alliance (2010, S. 7) ergeben, dass webbasierte Applikationen vorallem für kleinere Applikationen, wie abonnementbasierte Dienste (Nachrichten, Wetter, Finanzen, Shopping etc.) nützlich sind und performancelastige Applikationen, wie Spiele weiterhin im nativen Feld bleiben werden. In bestimmten Bereichen, wo eine tiefe Integration in die Systemfunktionen eines Smartphones notwendig ist, sind native Applikationen weiterhin unerlässlich (vgl. Dern, 2010).

Das W3C versucht mit den sogenannten Widgets einen plattformübergreifenden Standard für mobile Applikationen einzuführen. Widgets basieren auf Web Technologien und können über eine einheitliche API auf native Gerätefunktionen zugreifen. Sie stehen lokal zur Verfügung, da sie, wie native Applikationen auf das Smartphone heruntergeladen und installiert werden können. Dafür muss das Widget mit allen notwendigen Dateien gezippt und die Dateiendung in *.wgt* umbenannt werden. Dies ermöglicht einen einfachen plattformunabhängigen Vertrieb des Widgets.

Derzeit wird der W3C Widget Standard jedoch von keiner der vier marktführenden Plattformen unterstützt. (vgl. Koch, 2009; Brauner, 2010)

Aufgrund des rasanten Wachstums im mobilen Bereich, vorallem hinsichtlich verschiedener Plattformen, Smartphones und Applikationstypen, sind in den letzten Jahren einige Frameworks für die plattformübergreifende Entwicklung basierend auf Web Technologien auf den Markt gekommen. Diese können in zwei Kategorien eingeteilt werden: Frameworks für die Erstellung von hybriden Applikationen mit dem Einsatz von plattformübergreifenden APIs für den Zugriff auf Gerätefunktionen sowie Frameworks für die plattformübergreifende Entwicklung von mobilen Web Applikationen, die sich vorallem auf ein einheitliches UI konzentrieren und von den bereits implementierten W3C JavaScript APIs in den mobilen Webbrowsern profitieren (vgl. Dern, 2010; Allen, 2010, S. 10)

Die vier meist verbreiteten, bereits produktiv eingesetzten und aktiv weiter entwickelten plattformübergreifenden Frameworks basierend auf Web Technologien werden anhand von ausgewählten Kriterien (Installation und Entwicklungsumgebung, UI, Geolocation, Karte, offline Applikationen, lokale Datenbank, Kamera, Test, Distribution) in den nächsten Kapiteln näher analysiert. Es handelt sich bei den mobilen Web Applikationen um die Frameworks jQuery Mobile und Sencha Touch, bei den hybriden Applikationen um die Frameworks PhoneGap und Titanium Mobile.

6 jQuery Mobile

jQuery Mobile ist ein einheitliches UI Framework für die plattformübergreifende Entwicklung mobiler Web Applikationen und befindet sich derzeit im Alpha Stadium (vgl. jQuery Mobile, 2010a). jQuery Mobile baut für eine konsistente und bekannte Syntax auf dem jQuery core (vgl. jQuery, 2010) auf und ist somit sehr einfach zu erlernen. Weiters unterstützt jQuery Mobile WAI-ARIA (Accessible Rich Internet Applications) (vgl. Pappas et al., 2011) Funktionen für einen barrierefreien Zugriff von beispielsweise Screenreadern, wie VoiceOver bei iOS (vgl. jQuery Mobile, 2010aa). Derzeit werden die Plattformen iOS (3.1 – 4.2), Android (1.6 – 2.3), BlackBerry (6), Palm WebOS (1.4) sowie die mobilen Webbrowser Opera Mobile 10.1 (Android), Opera Mini 5.02 (iOS, Android) und Firefox Mobile beta (Android) unterstützt (vgl. jQuery Mobile, 2010z).

6.1 Installation und Entwicklungsumgebung

Alle notwendigen Dateien können unter jQuery Mobile (2010y) heruntergeladen werden. Wie folgender HTML Quellcode aufzeigt, sind für den Einsatz des jQuery Mobile Frameworks drei Dateien im `<head>` Bereich einer jQuery Mobile Website zu inkludieren.

```
1  <head>
2     <link rel="stylesheet" href="jquery.mobile-1.0a4.1.min.css" />
3     <script type="text/javascript" src="jquery-
4     1.5.2.min.js"></script>
5     <script type="text/javascript" src="jquery.mobile-
6     1.0a4.1.min.js"></script>
7  </head>
```

Neben der jQuery Mobile CSS Datei `jquery.mobile-1.0a4.1.min.css` (Zeile 2), wird die JavaScript Datei der aktuellen jQuery Bibliothek Version 1.5.2 `jquery-1.5.2.min.js` (Zeile 3), auf die das jQuery Mobile Framework aufbaut sowie die aktuelle jQuery Mobile Framework JavaScript Datei Version 1.0 Alpha 4.1 `jquery.mobile-1.0a4.1.min.js` (Zeile 5) eingefügt. Damit alle im jQuery Mobile CSS definierten Bilder angezeigt werden können, ist der *image* Ordner, der sich im heruntergeladenen jQuery Mobile Framework befindet, in das Projekt zu kopieren. (vgl. jQuery Mobile, 2010b)

Durch das Einbinden des Frameworks auf einer Website wird diese automatisch mit der jQuery Mobile Standardkonfiguration erweitert. Wie der folgende JavaScript Quellcode aufzeigt, kann diese Standardkonfiguration für alle Seiten über das Event `mobileinit` (Zeile 1) individuell verändert werden. Dieses wird mit dem `document` Objekt verbunden und die JavaScript Datei mit dem enthaltenen `mobileinit` Event wird vor der jQuery Mobile JavaScript Datei eingebunden. `$.mobile.defaultTransition = 'pop'` (Zeile 2) ermöglicht beispielsweise die Änderung des Seitenübergang-Effekts für alle Seiten vom Standardwert `slide` auf `pop`. (vgl. jQuery Mobile, 2010h)

```
1  $(document).bind('mobileinit', function(){
2     $.mobile.defaultTransition = 'pop';
3  });
```

Für die Entwicklung von mobilen Web Applikationen mit jQuery Mobile kann jede beliebige Entwicklungsumgebung, die HTML, CSS und JavaScript unterstützt, herangezogen werden.

6.2 User Interface

Im Folgenden wird auf die verschiedenen UI Elemente des jQuery Mobile Frameworks näher eingegangen.

6.2.1 Seitenstruktur

Folgender Quellcode stellt ein Standard Template für eine einzelne jQuery Mobile Seite dar (vgl. Abbildung 20). Der Aufbau einer jQuery Mobile Website basiert auf dem HTML5 Doctype `<!DOCTYPE html>` (Zeile 1), um alle Funktionen des Frameworks nutzen zu können. Innerhalb des `<body>` Bereichs benötigt jQuery Mobile für die Kennzeichnung einer Seite ein `<div>` Element mit einem `data-role="page"` (Zeile 9) Attribut. (vgl. jQuery Mobile, 2010b) Das `data-` Attribut ist ein neuer Bestandteil der W3C (World Wide Web Consortium) HTML5 Spezifikation, das benutzerdefinierte Attribute innerhalb von HTML Elementen ermöglicht. Der individuellen Bezeichnung, beispielsweise bei jQuery Mobile `role`, muss immer das Präfix `data-` vorangestellt werden. Der zugewiesene Wert `"page"` kann mittels JavaScript über das Attribut `dataset` mit `element.dataset.role` ermittelt werden. (vgl. Hickson, 2011a) Innerhalb des `page` Containers in Zeile 9 werden von jQuery Mobile weitere `data-role` Attribute für `header` (Zeile 10), `content` (Zeile 13) und `footer` (Zeile 16) vorgeschlagen, die ein eigenes Design und bestimmte Eigenschaften vorweisen und optional verwendet

werden können. Im Kapitel 6.2.3 wird der `header` und `footer` Bereich genauer erläutert.

```
1  <!DOCTYPE html>
2  <html>
3     <head>
4       <meta charset="utf-8">
5       <title>Seitentitel</title>
6       //jQuery Mobile Dateien
7     </head>
8  <body>
9  <div data-role="page">
10    <div data-role="header">
11          <h1>Kopfzeile</h1>
12    </div>
13    <div data-role="content">
14          <p>Seiteninhalt</p>
15    </div>
16    <div data-role="footer">
17          <h4>Fußzeile</h4>
18    </div>
19 </div>
20 </body>
21 </html>
```

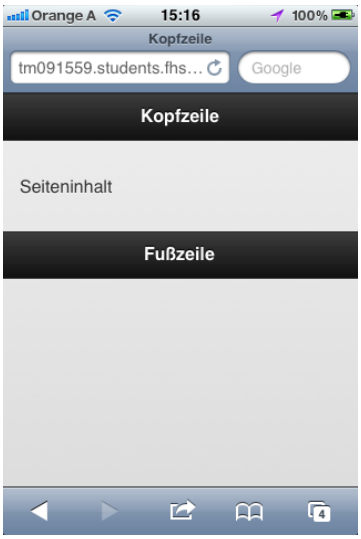


Abbildung 20: jQuery Mobile Seitenstruktur

Innerhalb einer einzelnen jQuery Mobile HTML Seite können weitere Seiten über das `data-role="page"` Attribut deklariert werden, wodurch ein schnelleres Anzeigen neuer Seiten ermöglicht wird. Die Wartezeit verkürzt sich im Vergleich zum Öffnen einer vollständig neuen Seite. Wie in folgendem Quellcode ersichtlich wird, erhält jeder `page` Container eine eindeutige `id` (Zeile 2 und Zeile 16), um intern auf die einzelnen Seiten zugreifen zu können. Beim Aufruf der Website blendet das Framework alle Seiten, bis auf die Erste mit der `id="start"` (Zeile 2), aus. Wenn beispielsweise auf `href="#artikel"` in Zeile 7 geklickt wird, sucht das Framework nach einer internen Seite mit der `id="artikel"` (Zeile 16) und zeigt diese an. Weiters besteht auch die Möglichkeit mit `href="artikel.html"` in Zeile 8 auf eine externe Seite der gleichen Domain zu verlinken. jQuery Mobile formuliert für den Aufruf aller Links innerhalb einer Domain eine Ajax Anfrage mit einem animierten Seitenübergang (Kapitel 6.2.2). Wenn die Ajax Anfrage erfolgreich war, wird der neue Inhalt der externen Seite `artikel.html` zum DOM (Document Object Model) hinzugefügt. Um eine vollständige Seitenaktualisierung ohne Ajax Navigation zu ermöglichen, muss im externen Link `rel="external"`, `data-ajax="false"` oder das Attribut `target` mit beispielsweise dem Wert `_blank` angegeben werden, damit der Ajax Hash (`#`) innerhalb der URL (Uniform Resource Locator) gelöscht wird. (vgl. jQuery Mobile, 2010b) E-Mail und Telefon Links werden ebenfalls vom jQuery Mobile Framework unterstützt (vgl. jQuery Mobile, 2010m).

```
1  <!-- Erste Seite -->
2  <div data-role="page" id="start">
3     <div data-role="header">
4           <h1>Seitentitel</h1>
5     </div>
6     <div data-role="content">
7           <p><a href="#artikel">Artikel anzeigen</a></p>
8           <p><a href="artikel.html">Externe Artikel Seite
9           anzeigen</a></p>
10    </div>
11    <div data-role="footer">
12          <h4>Fußzeile</h4>
13    </div>
14 </div>
15 <!-- Zweite Seite -->
16 <div data-role="page" id="artikel">
17    <div data-role="header">
18          <h1>Seitentitel</h1>
19    </div>
20    <div data-role="content">
21          <p><a href="#start">Startseite aufrufen</a></p>
22    </div>
23    <div data-role="footer">
24          <h4>Fußzeile</h4>
25    </div>
26 </div>
```

Das Framework verfügt neben dem Standard Seiten-Design über fünf weitere vordefinierte Seiten-Designs, die über das `data-theme` Attribut für die gesamte Seite im `page` Container mit den Werten `a`, `b`, `c`, `d` oder `e` ausgewählt werden können. Das Attribut `data-theme` kann optional auch nur auf einzelne Elemente, wie beispielsweise auf die Kopf- und Fußzeile oder auf eine Schaltfläche angewendet werden. (vgl. jQuery Mobile, 2010g)

jQuery Mobile definiert den Viewport standardmäßig wie folgt: `<meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum-scale=1.0" />`. Dieser ist mit den Standard Webbrowsern von Android, iOS und BlackBerry kompatibel, Symbian^3 unterstützt Viewport derzeit nicht (vgl. jQuery Mobile, 2010h; Firtman, 2010, S. 127). Unter Viewport ist das sichtbare Browserfenster gemeint, in welchem eine Website dargestellt wird. Es besteht die Möglichkeit dem Viewport eine Breite, Höhe sowie einen Zoomfaktor zu vergeben. Mit `width=device-width` wird die Breite des Viewports automatisch an die Bildschirmbreite des jeweiligen Smartphones angepasst. Für eine benutzerfreundliche mobile Applikation wird mit `minimum-scale=1.0` und `maximum-scale=1.0` die Möglichkeit des Zoomens unterdrückt. (vgl. Firtman, 2010, S. 125 - 126) Mit Hilfe des Events `mobileinit` (Kapitel 6.1) kann über `metaViewportContent` die Viewport Eigenschaften geändert werden (vgl. jQuery Mobile, 2010h).

6.2.2 Seitenübergang

jQuery Mobile verfügt über sechs verschiedene Seitenübergang-Effekte die auf CSS basieren. Standardmäßig verwendet das Framework den für mobile Applikationen typischen `slide` Effekt. Dieser Effekt bewegt die Seite von rechts nach links, außer bei Betätigung der Zurück-Schaltfläche wird die Seite von links nach rechts bewegt. Damit eine links nach rechts Bewegung erzwungen werden kann, muss das Attribut `data-direction="reverse"` auf dem gewünschten Link angegeben werden (vgl. jQuery Mobile, 2010b). Zum Überschreiben des Standard-Effekts kann innerhalb des Links das Attribut `data-transition` mit den Werten `slideup`, `slidedown`, `pop`, `fade` oder `flip` angegeben werden. (vgl. jQuery Mobile, 2010k) Über das Event `mobileinit` (Kapitel 6.1) wird mit der Eigenschaft `defaultTransition` der Seitenübergang-Effekt global für alle Seiten geändert (vgl. jQuery Mobile, 2010h). Jede Seite einer jQuery Mobile Website kann durch Vergabe des Attributs `data-rel="dialog"` auf einen Link als Dialog fungieren (vgl. jQuery Mobile, 2010l).

6.2.3 Kopf- und Fußzeile

Mit dem Attribut `data-role="header"` (Zeile 1) lässt sich eine Kopfzeile am Beginn einer Seite spezifizieren, die typischerweise den Seitentitel in einer `<h1>` Überschrift und maximal zwei Schaltflächen enthält. Besteht eine jQuery Mobile Website aus mehreren Seiten, fügt das Framework automatisch eine *Back* Schaltfläche links in der Kopfzeile ein. Der Text der Schaltfläche kann über das Attribut `data-back-btn-text="zurück"` im `page` Container oder einfacher für alle Seiten innerhalb des `mobileinit` Events (Kapitel 6.1) mit der Eigenschaft `$.mobile.page.prototype.options.backBtnText='zurück'` geändert werden. Durch Angabe des Attributs `data-backbtn="false"` im `header` Container kann die Zurück-Schaltfläche aus der Kopfzeile entfernt werden. (vlg. jQuery Mobile, 2010c) jQuery Mobile stellt einige Schaltflächen mit vordefinierten Symbolen zur Verfügung (vgl. jQuery Mobile, 2010i). Beispielsweise kann innerhalb des `header` Containers durch Einfügen des Quellcodes `<a href="#" data-icon="plus" class="ui-btn-right">Hinzufügen</a>` eine weitere Schaltfläche rechts vom Seitentitel hinzugefügt werden. Das Attribut `data-icon="plus"` definiert automatisch ein Plus Symbol und aufgrund der Klasse `class="ui-btn-right"` wird die Schaltfläche rechts in der Kopfzeile platziert. Durch Weglassen der Klasse `class="ui-btn-right"` wird die Schaltfläche standardmäßig links vom Seitentitel eingefügt. Wenn eine weitere Schaltfläche ohne Klasse angegeben wird, wird diese – abhängig von der Reihenfolge im HTML Quellcode – automatisch rechts in der Kopfzeile angezeigt. Im folgenden

Quellcodebeispiel und in Abbildung 21 wird dies verdeutlich. (vlg. jQuery Mobile, 2010c)

```
1  <div data-role="header">
2     <a href="#" data-
3     icon="delete">Abbrechen</a>
4     <h1>Seitentitel</h1>
5     <a href="#" data-
6     icon="plus">Hinzufügen</a>
7  </div>
```

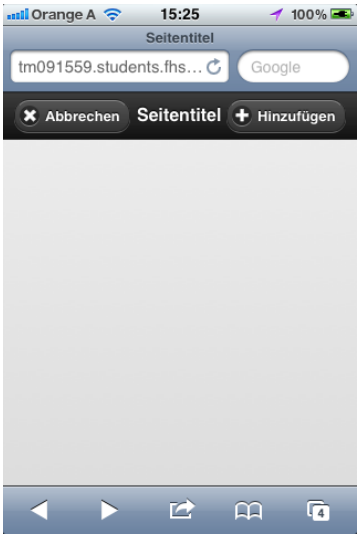


Abbildung 21: jQuery Mobile Kopfzeile

`data-role="footer"` definiert eine Fußzeile am Ende einer jQuery Mobile Seite. Im Unterscheid zur Kopfzeile werden in der Fußzeile keine Schaltfläche automatisch hinzugefügt oder je nach Reihenfolge im HTML Quellcode links oder rechts vom Seitentitel angezeigt. Alle eingefügten Schaltflächen werden vom linken Rand beginnend der Reihe nach angezeigt. Damit ein Abstand zwischen den Schaltflächen eingefügt wird, muss die Klasse `class="ui-bar"` auf dem `footer` Container vergeben werden. Weiters kann die Fußzeile als fixes Navigationselement am Ende einer Seite fungieren, indem diese an der gleichen Stelle bleibt, wenn auf eine andere Seite gewechselt wird. Dafür wird auf allen relevanten Seiten im `footer` Container das Attribut `data-id="myfooter"` hinzugefügt, wobei der Wert `myfooter` frei wählbar ist. Für eine korrekte Verarbeitung des fixen Navigationselements muss das Attribut `data-position="fixed"` im `footer` Container angegeben werden. (vlg. jQuery Mobile, 2010d)

Das Attribut `data-position="fixed"` ermöglicht eine statische Kopf- bzw. Fußzeile. Bei langen Inhalten erscheint die Kopf- und Fußzeile automatisch wieder im Sichtfeld der Benutzerinnen/Benutzer. Durch Antippen des Smartphone Bildschirms kann die Kopf- und Fußzeile aus- bzw. wieder eingeblendet werden. (vlg. jQuery Mobile, 2010e) jQuery Mobile ermöglicht weiters das Ein- und Ausblenden der Kopf- und Fußzeile durch Antippen des Smartphone Bildschirms mit der Angabe des Attributs `data-fullscreen="true"` im `page` Container. `data-position="fixed"` muss ebenfalls

wieder dem `header` und `footer` Container vergeben werden. Der einzige Unterschied zur alleinigen Angabe des Attributs `data-position="fixed"` ist, dass bei zusätzlicher Angabe von `data-fullscreen="true"` die Kopf- und Fußzeile am Beginn und am Ende einer Seite fix – ohne Ausblendungsmöglichkeit – angezeigt wird. (vlg. jQuery Mobile, 2010f)

Nachfolgend wird ein Quellcodebeispiel für eine Navigationsleiste mit zwei Schaltflächen in einer fixen `data-position="fixed"` (Zeile 1) Fußzeile angeführt (vgl. Abbildung 22) . jQuery Mobile ermöglicht die Erstellung einer Navigationsleiste mit maximal fünf Schaltflächen, die typischerweise in der Fußzeile implementiert wird und jede Schaltfläche kann optional über ein themenspezifisches Symbol verfügen. Die Navigationsleiste besteht aus einer unsortierten Liste `<ul><li></li></ul>`, die innerhalb eines `div` Elements mit dem Attribut `data-role="navbar"` (Zeile 3) platziert wird. Für den aktiven Zustand einer Schaltfläche kann die Klasse `class="ui-btn-active ui-state-persist"` (Zeile 8) innerhalb eines Links vergeben werden. Durch Hinzufügen des Attributs `data-icon` wird ein vordefiniertes Symbol, wie beispielsweise `star` (Zeile 7) oder `grid` (Zeile 13) für die jeweilige Schaltfläche definiert. (vlg. jQuery Mobile, 2010j) Es können auch eigene Symbole oder Symbole aus einer Bibliothek, wie Glyphish verwendet werden (vgl. Glyphish, 2010).

```
1  <div data-role="footer" data-
2  position="fixed">
3    <div data-role="navbar">
4      <ul>
5        <li>
6        <a href="#start"
7        data-icon="star"
8        class="ui-btn-
9        active ui-state-
10       persist">Start</a>
11       </li>
12       <li>
13       <a href="#artikel" data-
14       icon="grid">Artikel</a>
15       </li>
16     </ul>
17   </div>
18 </div>
```

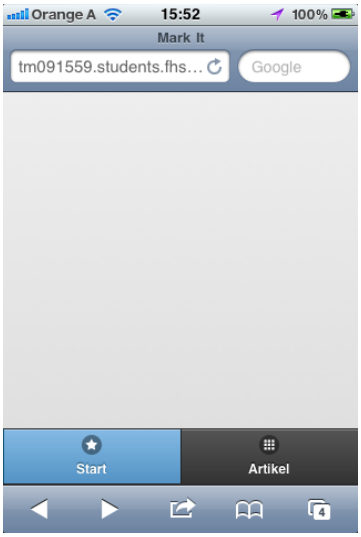


Abbildung 22: jQuery Mobile Fußzeile

6.2.4 Schaltflächen

Durch Vergabe des Attributs `data-role="button"` (Zeile 2) auf einen Link `<a href="#start" data-role="button">Start</a>` wird dieser vom Framework zu

einer Schaltfläche umgewandelt. Die Elemente `<button>Schalfäche</button>` (Zeile 3) und `<input type="submit" name="senden" value="Senden" />` (Zeile 4) werden ohne dem Attribut `data-role="button"` automatisch als Schalflächen dargestellt. (vgl. jQuery Mobile, 2010n) Alle Schalflächen können über das Attribut `data-icon` (Zeile 3) vordefinierte Symbole enthalten und diese können weiters über das Attribut `data-iconpos` nach rechts (`right`), links (`left`), oben (`top`) oder unten (`bottom`) positioniert werden. Für die Erstellung einer Schalfläche ohne Text muss das Attribut `data-iconpos` den Wert `notext` erhalten. (vgl. jQuery Mobile, 2010i) Standardmäßig füllen die Schalflächen die gesamte Bildschirmbreite aus. Mit dem Attribut `data-inline="true"` (Zeile 2) beschränkt sich die Schaltflächenbreite auf die Weite des Symbols und/oder Textes. (vgl. jQuery Mobile, 2010o) Die Schaltflächen können innerhalb eines `div` Elements mit dem Attribut `data-role="controlgroup"` (Zeile 5) vertikal gruppiert werden. Durch Hinzufügen eines weiteren Attributs `data-type="horizontal"` (Zeile 12) wird eine horizontale Gruppierung ermöglicht. (vgl. jQuery Mobile, 2010p) Das Quellcodebeispiel wird in Abbildung 23 veranschaulicht.

```
1  <div data-role="content">
2    <a href="#" data-role="button" data-inline="true">Button</a>
3    <button data-icon="delete">Delete</button>
4    <input type="submit" name="senden" value="Senden" />
5    <div data-role="controlgroup">
6      <a href="#" data-role="button">Ja</a>
7      <a href="#" data-role="button">Nein</a>
8      <a href="#" data-role="button">Vielleicht</a>
9    </div>
10 </div>
11 <div data-role="footer" data-position="fixed">
12   <div data-role="controlgroup" data-type="horizontal">
13     <a href="#" data-role="button">Ja</a>
14     <a href="#" data-role="button">Nein</a>
15     <a href="#" data-role="button">Vielleicht</a>
16   </div>
17 </div>
```

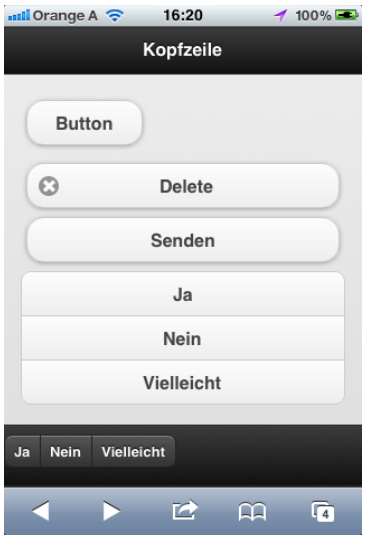


Abbildung 23: jQuery Mobile Schalflächen

6.2.5 Inhalte

Für ein individuelles Arbeiten vergibt das jQuery Mobile Framework auf normale HTML Inhalte, wie Überschriften, Paragraphen, Links, sortierte/unsortierte Listen und Tabellen keine eigenen Stile (vgl. jQuery Mobile, 2010q).

Derzeit stellt jQuery Mobile zwei Stile zur Verfügung, um den Inhalt einer Website einfacher gestalten zu können. Zunächst ermöglicht das Attribut `data-role="collapsible"` (Zeile 2, 6, 10) auf einem `div` Element einen auf- und zuklappbaren Bereich (vgl. Abbildung 24). Ohne der Vergabe des Attributs `data-collapsed="true"` (Zeile 2, 6, 10) wird der Bereich standardmäßig aufgeklappt dargestellt. Folgender Quellcode zeigt, dass durch das Platzieren mehrerer zusammenklappbarer Bereiche innerhalb eines `div` Elements mit dem Attribut `data-role="collapsible-set"` (Zeile 1) immer nur ein Bereich aufgeklappt dargestellt werden kann. (vgl. jQuery Mobile, 2010r)

```
1  <div data-role="collapsible-set">
2     <div data-role="collapsible" data-collapsed="true">
3           <h1>Eins</h1>
4           <p>Erster zusammenklappbarer Bereich</p>
5     </div>
6     <div data-role="collapsible" data-collapsed="true">
7           <h1>Zwei</h1>
8           <p>Zweiter zusammenklappbarer Bereich</p>
9     </div>
10    <div data-role="collapsible" data-collapsed="true">
11          <h1>Drei</h1>
12          <p>Dritter zusammenklappbarer Bereich</p>
13    </div>
14 </div>
```

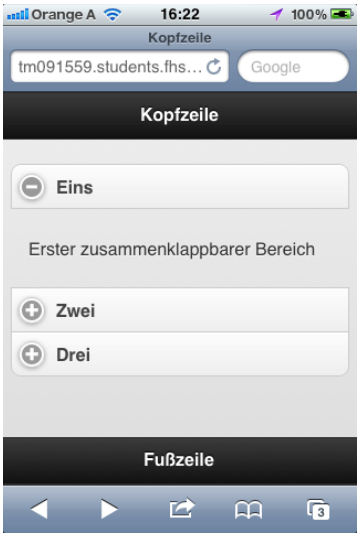


Abbildung 24: jQuery Mobile Inhalte

Der zweite Stil von jQuery Mobile ermöglicht das Organisieren von Inhalten in mehrspaltigen Rastern durch Vergabe einer CSS Klasse mit der Konvention `ui-grid-` auf beispielsweise ein `div` oder `fieldset` Element. Die Klasse `ui-grid-a` erstellt zwei Spalten, `ui-grid-b` drei Spalten und so weiter. Der folgende Quellcode zeigt auf, dass innerhalb des Elements `<div class="ui-grid-a">` (Zeile 1) – für einen zweispaltigen Raster – auf der ersten Spalte die Klasse `class="ui-block-a"` (Zeile 2) und auf der zweiten Spalte die Klasse `class="ui-block-b"` (Zeile 5) vergeben werden muss, damit die Spalten nebeneinander positioniert werden. Innerhalb der Spalten wurde mit `<button></button>` (Zeile 3, 6) jeweils eine Schaltfläche platziert. (vgl. jQuery Mobile, 2010s; Abbildung 25)

```
1  <div class="ui-grid-a">
2    <div class="ui-block-a">
3      <button>1. Spalte</button>
4    </div>
5    <div class="ui-block-b">
6      <button>2. Spalte</button>
7    </div>
8  </div>
```

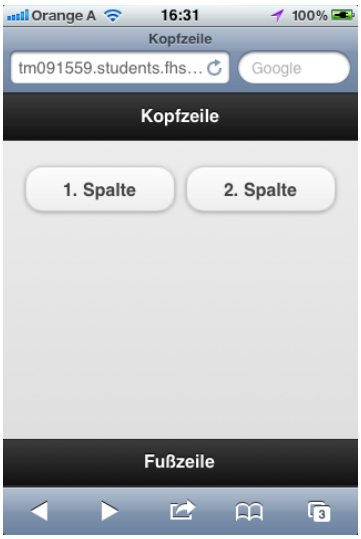


Abbildung 25: jQuery Mobile Tabelle

6.2.6 Formulare

jQuery Mobile ersetzt automatisch alle HTML Formular Elemente, wie beispielsweise Eingabefelder (Zeile 4, 6), Kontrollkästchen oder Schieberegler mit angepassten Bedienelementen speziell für berührungssensitive Smartphone-Bildschirme. Alle Elemente erhalten eine flexible Breite, die sich an die Bildschirmbreite anpasst. (vgl. jQuery Mobile, 2010t) Auf jQuery Mobile (2010u) wird ein Überblick der unterstützten Bedienelemente sowie das Aussehen dieser gezeigt.

Mit dem Framework können neue HTML5 Eingabefelder vom Typ `number` (Zahl), `email`, `url`, `tel` (Telefon), `search` (Suche) und `range` (Schieberegler) verwendet werden (vgl. jQuery Mobile, 2010v). Die Android, BlackBerry und iOS Plattform passen jeweils die Tastaturbelegung für die Eingabefeldtypen `number` und `tel` für eine schnellere Eingabemöglichkeit automatisch an. iOS passt die Tastaturbelegung auch für die Eingabefeldtypen `email` und `url` an. (vgl. Spiering & Haiges, 2010, S. 88)

Bei einem Eingabefeld des Typs `search` (Zeile 2) fügt jQuery Mobile automatisch rechts ein *x* Symbol zum schnellen Löschen des Inhalts ein. `range` (Zeile 8) ermöglicht die Erstellung eines Schiebereglers, mit welchem eine Zahl zwischen zwei Grenzwerten – durch Angabe von `min` und `max` (Zeile 8) im Eingabefeld – ausgewählt werden kann. Ein Schalter, um beispielsweise eine Option zu aktivieren bzw. deaktivieren, ist ein typisches UI Element auf einem Smartphone und kann mit dem Attribut `data-role="slider"` (Zeile 10) in einem `select` Element dargestellt werden. (vgl. jQuery Mobile, 2010u) Die im Quellcodebeispiel angeführten Bedienelemente können in Abbildung 26 betrachtet werden.

```
1  <label for="search">Search Input:</label>
2  <input type="search" name="search" id="search" value="" />
3  <label for="name">Text Input:</label>
4  <input type="text" name="name" id="name" value=""  />
5  <label for="textarea">Textarea:</label>
6  <textarea cols="40" rows="8" name="textarea"
    id="textarea"></textarea>
7  <label for="slider">Input slider:</label>
8  <input type="range" name="slider" id="slider" value="0" min="0"
    max="100"  />
9  <label for="slider">Select slider:</label>
10 <select name="slider" id="slider" data-role="slider">
11    <option value="off">Off</option>
12    <option value="on">On</option>
13 </select>
```

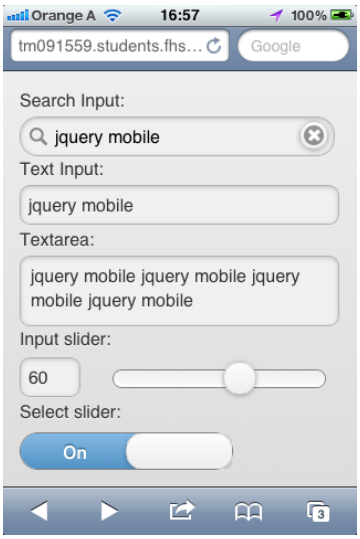


Abbildung 26: jQuery Mobile Formulare

Um eine automatische Initialisierung der Formular Elemente durch jQuery Mobile zu verhindern, kann entweder im jeweiligen Bedienelement das Attribut `data-role="none"` angegeben werden oder über das Event `mobileinit` (Kapitel 6.1) die Option `$.mobile.page.prototype.options.keepNative = 'input, textarea';` mit den gewünschten Elementen versehen werden (vgl. jQuery Mobile, 2010t).

6.2.7 Listen

jQuery Mobile stellt eine Auswahl an verschiedenen Listen für beispielsweise die Anzeige von Daten, Navigationen oder Ergebnislisten zur Verfügung. Eine normale unsortierte `<ul><li></li></ul>` oder sortiere `<ol><li></li></ol>` Liste kann durch Vergabe des Attributs `data-role="listview"` (Zeile 1) zu einer benutzerfreundlichen mobilen Liste umgewandelt werden, die die gesamte Bildschirmbreite ausfüllt. Durch Angabe eines Links innerhalb des Listen Elements `<li><a href="#start">Start</a></li>` (Zeile 3) wird auf der rechten Seite automatisch ein Pfeil Symbol eingefügt und ein Seitenübergang-Effekt bei Anklicken des Links ausgeführt. Wenn innerhalb eines `<li>` Elements eine weitere `<ul>` Liste verschachtelt wird, generiert jQuery Mobile automatisch eine neue Seite für die verschachtelte Liste, wodurch beispielsweise ein Navigationsbaum erstellt werden kann.

Innerhalb eines `<li>` Elements können zwei Links (Zeile 8) angegeben werden, um einen vertikalen Trennstrich zwischen den beiden Links zu erzeugen. Die Benutzerinnen/Benutzer können die beiden Links unabhängig von einander mit verschieden ausgeführten Aktionen anklicken. Durch Vergabe des Attributs `data-split-icon="plus"` (Zeile 7) auf das `<ul>` Element kann das Symbol des rechten

Links verändert werden. Um Gruppierungen innerhalb von Listen zu ermöglichen, können Trennlinien über das Attribut `data-role="list-divider"` (Zeile 11) auf `<li>` Elementen vergeben werden.

Durch Hinzufügen des Attributs `data-filter="true"` (Zeile 14) zu einem `<ul>` oder `<ol>` Element ermöglicht das Framework eine durchsuchbare Liste. Oberhalb der Liste wird automatisch ein Eingabefeld für die Suche eingefügt und bei Eingabe von Buchstaben oder Zahlen nur die passenden Listeneinträge angezeigt.

In jedem Listenelement kann ein Zählerstand in Form eines Kreises durch Angabe der Klasse `ui-li-count` (Zeile 4) auf beispielsweise ein `<span>` Element angezeigt werden. Weiters kann am Beginn eines `<li>` Elements ein Bild `<img />` eingefügt werden, das vom Framework zu einem 80 Pixel Breiten und Hohen Miniaturbild skaliert wird. Durch Vergabe des Attributs `data-inset="true"` (Zeile 18) auf das Element `<ul>` oder `<ol>` wird die Liste als Block mit abgerundeten Ecken und Abstand zum Bildschirmrand formatiert. Die verschiedenen Listentypen aus dem Quellcodebeispiel sind in Abbildung 27 ersichtlich.

```
1  <ul data-role="listview">
2     <li>
3           <a href="#">Listeneintrag</a>
4           <span class="ui-li-count">3</span>
5     </li>
6  </ul>
7  <ul data-role="listview" data-split-icon="plus">
8     <li><a href="#">Listeneintrag</a><a href="#"></a></li>
9  </ul>
10 <ul data-role="listview">
11    <li data-role="list-divider">L</li>
12    <li><a href="#">Listeneintrag</a></li>
13 </ul>
14 <ul data-role="listview" data-filter="true">
15    <li><a href="#">Anfang</a></li>
16    <li><a href="#">Ende</a></li>
17 </ul>
18 <ul data-role="listview" data-inset="true">
19    <li><a href="#">Listeneintrag</a></li>
20 </ul>
```

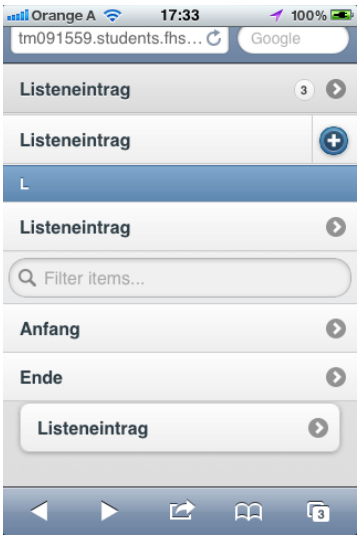


Abbildung 27: jQuery Mobile Listen

Mit der Methode `refresh()` können alle Stile einer Liste nach Hinzufügen neuer Element aktualisiert werden. (vgl. jQuery Mobile, 2010w)

6.2.8 Events

jQuery Mobile stellt einige Events zum Erkennen von getätigten Aktionen speziell auf Smartphones zur Verfügung. Beispielsweise eruiert das Event `taphold`, ob die Benutzerinnen/Benutzer den Bildschirm länger als eine Sekunde berührt haben oder das Event `swipeleft`, ob die Benutzerinnen/Benutzer den Finger von rechts nach links bewegt haben. Das nachfolgende Quellcode Beispiel zeigt auf, dass das Event `orientationchange` erkennt, ob das Smartphone in ein Hochformat `e.orientation == 'portrait'` oder Querformat `e.orientation == 'landscape'` gedreht wurde. (vgl. jQuery Mobile, 2010x)

```
1  $(document).bind('orientationchange', function(e) {
2      if(e.orientation == 'portrait')
3          alert('Hochformat');
4      if(e.orientation == 'landscape')
5          alert('Querformat' );
6  } );
```

6.3 Geolocation

> *Geolocation is the art of figuring out where you are in the world and (optionally) sharing that information with people you trust (Pilgrim, 2010, S. 24).*

Smartphones unterstützen verschiedene Techniken zur Ermittlung des aktuellen Standorts. Viele Smartphones sind mit einem eingebauten GPS Chip ausgestattet, der Daten für die Positionsermittlung von Satelliten empfangen kann und eine exakte Standortbestimmung ermöglicht. Der Empfang von GPS Signalen ist ohne Mobilfunknetz oder Internetverbindung möglich, kann nur im Freien stattfinden und nimmt viel Zeit und Akkuleistung in Anspruch. Assisted GPS (A-GPS) ist ein software-basiertes System für Smartphones, das ein Mobilfunknetz voraussetzt und bei einem schlechten oder nicht vorhandnen GPS Signal bei der Ermittlung des Standorts schneller behilflich zu sein. A-GPS stellt eine Verbindung zu Mobilfunkmasten her, um ein besseres Satellitensignal ausfindig zu machen oder vorübergehend eine weniger exakte Positionsbestimmung bereitzustellen, bis GPS wieder zur Verfügung steht. Da ein Mobilfunkbetreiber die genaue Position jedes einzelnen Sendemasten kennt, kann – falls kein GPS-Signal zur Verfügung steht – mittels Triangulation der ungefähre Standort eines Smartphones ermittelt werden. Bei der Triangulation muss das Smartphone mindestens drei Basisstationen empfangen, damit der Standort berechnet werden kann. Je mehr Sendemasten, beispielsweise in einer Großstadt, in der Nähe eines Smartphones sind, umso genauer wird die Positionsbestimmung. Weiters kann auch ohne Mobilfunknetz – bei Vorhandensein eines WLANs – über die IP Adresse eine Standortbestimmung ermöglicht werden. Für die Orientierung innerhalb eines Gebäudes kann jede WLAN Benutzerin/jeder WLAN Benutzer eine eindeutige IP Adresse erhalten, um zu Erkennen mit welchem Hotspot eine Benutzerin/ein Benutzer verbunden ist und demnach eruiert werden, in welchem Stockwerk oder Zone sich diese/dieser befindet. (vgl. Firtman, 2010, S. 370 - 372)

Für die Verwendung von Geolocation in mobilen Web Applikationen, die auf dem jQuery Mobile Framework basieren, kann die Entwicklerin/der Entwickler auf die sogenannte Geolocation API zurückgreifen. Das W3C arbeitet an einem Standard, der sogenannten Geolocation API, zur Ermittlung des geografischen Standorts über JavaScript. Die Spezifikation baut nicht nur auf einer der oben genannten Lokalisierungstechnologien auf, sondern sieht vor, dass alle Technologien für die Lokalisierung von den Webbrowsern herangezogen werden können. Von den in Kapitel 3 vorgestellten Plattformen wird die W3C Geolocation API von den vorinstallierten mobilen Webbrowsern der Plattformen Android ab Version 2.0, BlackBerry ab Version 6.0 und iOS ab Version 3.0 unterstützt. In den BlackBerry Webbrowser-Optionen muss unter *Privat&Sicherheit* Geolocation explizit aktiviert werden, um Geolocation im Webbrowser verwenden zu können (vgl. BlackBerry Documentation, 2010a). Symbian

unterstützt die W3C Geolocation API derzeit nur in ihren installierbaren Web Runtime (WRT) Widgets und nicht in ihrem Webbrowser. (vg. Firtman, 2010, S. 375)

Beim Einsatz der Geolocation API im Webbrowser sieht die Spezifikation eine ausdrückliche Zustimmung der Benutzerin/des Benutzer über die Erlangung der geografischen Daten des Standorts vor. Beim Aufruf einer Website, die Geolocation verwendet, erscheint ein Dialog, der den Benutzerinnen/den Benutzern eine Zustimmung oder Ablehnung zur Positionsbestimmung ermöglicht. (vgl. Popescu, 2010a)

Wenn die Geolocation API im Webbrowser implementiert ist, verfügt das `navigator` Objekt in JavaScript über eine neue Eigenschaft `geolocation`, die es ermöglicht mit der Geolocation API zu interagieren. Die API ermöglicht mit den Funktionen `getCurrentPosition` eine einmalige Positionsanfrage und mit `watchPosition` eine kontinuierliche Positionsanfrage. Da für die Standortbestimmung aufgrund von beispielsweise GPS etwas mehr Zeit in Anspruch genommen werden kann, baut die API für die Ermittlung des Breiten- und Längengrades auf Callback-Funktionen auf. (vgl. Popescu, 2010b)

Eine einmalige Positionsanfrage kann folgendermaßen aussehen:

```
1  var options = {enableHighAccuracy:true, timeout:60000,
   maximumAge:0};
2  navigator.geolocation.getCurrentPosition(showMap, handleError,
   options);
3  function showMap(position) {
4     var lat = position.coords.latitude;
5     var lng = position.coords.longitude;
6  }
7  function handleError(error) {
8     switch(error.code) {
9           case error.PERMISSION_DENIED: alert('Permission denied');
10          break;
11          case error.POSITION_UNAVAILABLE: alert('Position
12          unavailable');
13          break;
14          case error.TIMEOUT: alert('Timeout');
15          break;
16    }
17 }
```

Der Funktion `getCurrentPosition` in Zeile 2 können drei Argumente `showMap`, `handleError`, `options` übergeben werden, die letzten beiden Argumente sind optional. Wenn die Benutzerin/der Benutzer der Positionsbestimmung über den Dialog zugestimmt hat und der geografische Standort bestimmt werden konnte, wird die

Funktion `showMap` in Zeile 3 aufgerufen. Die erfolgreiche Standortbestimmung steht auch in Abhängigkeit mit dem dritten Argument `options`. Die Funktion `handleError` in Zeile 7 wird aufgerufen, wenn aufgrund des Arguments `options` keine Standortbestimmung möglich ist. (vgl. Popescu, 2010b)

Dem `PositionOptions` Objekt `options` in Zeile 1 können drei Attribute übergeben werden. Das erste Attribut `boolean enableHighAccuracy` auf `true` gesetzt, forciert eine exakte Positionsbestimmung über beispielsweise GPS. Bei Smartphones führt dies jedoch zu längeren Wartezeiten und einem hohen Akkuverbrauch. Standardmäßig ist `enableHighAccuracy` auf `false` gesetzt. Das zweite Attribut `long timeout` gibt in Millisekunden (60000ms = 6s) an, wie lange für die Positionsbestimmung benötig werden darf. Das dritte Attribut `long maximumAge` gibt an, ob eine zuvor eruierte Position, die nicht älter als der angegebene Millisekunden Bereich ist, verwendet werden darf. Wird `maximumAge` auf `0` gesetzt, wird nur eine neue Positionsermittlung akzeptiert. (vgl. Popescu, 2010c)

Die Funktion `showMap` in Zeile 3 erhält ein `position` Objekt mit verschiedenen `coords` Eigenschaften. Die Eigenschaft `coords` kann laut W3C Standard sieben Attribute haben. Die Attribute `double latitude` (Breitengrad, Zeile 4) und `double longitude` (Längengrad, Zeile 5) geben die geografischen Koordinaten als Dezimalzahl laut dem World Geodetic System 84 (vgl. DoD WGS 84, 2000) aus. Das Attribut `double accuracy` präzisiert die `latitude` und `longitude` Koordinaten in Meter. Beispielsweise bedeutet der Wert 500, dass die Koordinaten um bis zu 500 Meter von der wirklichen Position abweichen. Die folgenden Attribute sind optional und werden nicht von allen Webbrowser Implementierungen unterstützt. Bei Nicht-Unterstützung wir der Wert `null` zurück gegeben. Das Attribut `double altitude` gibt die Höhe der Position in Meter an. `double altitudeAccuracy` präzisiert, wie das Attribut `double accuracy` die Höhe in Meter. `double heading` gibt die Richtung im Uhrzeigersinn relativ zum Norden in Grad an. Das Attribut `double speed` zeigt die Geschwindigkeit in Meter pro Sekunde an. (vgl. Popescu, 2010d; Spiering & Haiges, 2010, S. 161)

Wenn ein Fehler auftritt, wird ein `error` Objekt an die Funktion `handleError` in Zeile 7 übergeben. Das `error` Objekt verfügt über drei Fehlermeldungen. `PERMISSION_DENIED` gibt an, dass die Benutzerin/der Benutzer der Standortbestimmung nicht zugestimmt hat. Bei `POSITION_UNAVAILABLE` konnte die Position, beispielsweise aufgrund eins fehlenden GPS Signals, nicht ermittelt werden.

`TIMOUT` wird als Fehlermeldung ausgegeben, wenn die angegebene Zeit des Attributs `timeout` im `PositionOptions` Objekt überschritten wurde. (vgl. Popescu, 2010e)

Das folgende Quellcodebeispiel in Zeile 1 zeigt, dass die Funktion `watchPosition` für eine kontinuierliche Positionsbestimmung genau gleich wie die soeben beschriebene `getCurrentPosition` aufgebaut werden kann. Der einzige Unterschied ist, dass die Funktion `watchPosition` eine numerische ID `watchID` zurück gibt, damit die ständige Standortbestimmung mit der Funktion `clearWatch` in Zeile 2 beendet werden kann. (vgl. Popescu, 2010b)

```
1  var watchId = navigator.geolocation.watchPosition(showMap,
   handleError, options);
2  navigator.geolocation.clearWatch(watchId);
```

6.4 Karte

Für die Einbindung von Karten in mobilen Web Applikationen, die auf dem jQuery Mobile Framework basieren, kann die Google Maps JavaScript API Version 3 (V3) eingesetzt werden. Die neue V3 ist wesentlich schneller als die alte V2 und deshalb speziell auf mobile Geräte ausgerichtet. Derzeit wird die V3 von den Webbrowsern der Plattformen Android, BlackBerry und iOS unterstützt. Auf der Symbian Plattform wird ein fixer Ausschnitt der Karte angezeigt, diese kann mit Berührungen nicht bewegt werden. Eine weitere Erneuerung ist, dass für die Einbindung von Google Maps kein API-Schlüssel mehr notwendig ist. (vgl. Google Maps, 2011a)

Folgender Quellcode zeigt, wie Google Maps auf einer Website eingebunden werden kann.

```
1  <html>
2  <head>
3  <script type="text/javascript"
    src="http://maps.google.com/maps/api/js?sensor=true"></script>
4  <script type="text/javascript">
5  function initialize() {
6      var latlng = new google.maps.LatLng(48.2138999, 15.6318955);
7      var myOptions = {
8        zoom: 8,
9        center: latlng,
10       mapTypeId: google.maps.MapTypeId.ROADMAP
11     };
12     var map = new
       google.maps.Map(document.getElementById('map_canvas'),
       myOptions);
13 }
14 </script>
15 </head>
```

```
16 <body onload="initialize()">
17   <div id="map_canvas" style="width:100%; height:100%"></div>
18 </body>
19 </html>
```

Die Google Maps JavaScript API (Zeile 3) wird im `head` Bereich einer Website angeben. Der `sensor` Parameter ist ebenfalls neu in V3 und gibt mit dem Wert `true` an, dass die Standortbestimmung über ein GPS-fähiges Smartphone erfolgt. (vgl. Google Maps, 2011b) In Zeile 6 wird für die Anzeige einer bestimmten Position auf einer Karte an das `google.maps.LatLng` Objekt die Parameter für Breiten- und Längengrad `(-34.397, 150.644)` übergeben und in der Variable `latlng` gespeichert. Zwischen Zeile 7 und 11 werden die Kartenoptionen in der Variable `myOptions` definiert. `zoom` definiert die Zoomstufe beim Laden der Karte, `center` gibt an, dass die Karte um den Wert `latlng` zentriert werden soll und `mapTypeId` ermöglicht die Auswahl verschiedener Kartentypen. `ROADMAP` zeigt eine Straßenkarte, `SATELLITE` zeigt eine Karte mit Sattelitenbildern, `HYBRID` ist eine Mischform aus `ROADMAP` und `SATELLITE` und `TERRAIN` zeigt eine Karte mit Geländeinformationen, wie Gewässer und Erhebungen. In Zeile 12 wird mit dem `google.maps.Map` Objekt eine Karte erstellt. Das Objekt platziert mit `document.getElementById('map_canvas')` die Karte in den `div` Container mit der `id='map_canvas'` (Zeile 17) und weiters werden die Kartenoptionen `myOptions` angegeben. (vgl. Google Maps, 2011c)

Mit der umgekehrten Geocodierung (Reverse Geocoding) kann die Google Maps API den ermittelten Breiten- und Längengrad in Zeile 1 in eine Adresse umwandeln. Mit dem Objekt `google.maps.Geocoder` in Zeile 2 kann auf das Geocodierungsservice zugegriffen werden. Mit Hilfe der Methode `geocode()` in Zeile 3 wird eine Anfrage mit den zuvor ermittelten Koordinaten über `{"latLng": latlng}` an das Service gestellt und erhält als Ergebnis `results[0]` die Adresse zu diesen Koordinaten. Mit `results[0].formatted_address` in Zeile 6 kann die Adresse ausgegeben werden. (vgl. Google Maps, 2011d)

```
1  var latlng = new google.maps.LatLng(48.2138999, 15.6318955);
2  var geocoder = new google.maps.Geocoder();
3  geocoder.geocode({"latLng": latlng}, function(results, status) {
4     if (status == google.maps.GeocoderStatus.OK) {
5           if (results[0]) {
6                 alert(results[0].formatted_address);
7           }
8     } else {
9           alert("Reverse Geocoding nicht moeglich : " +
      status);
10    }
```

```
11 });
```

6.5 Offline Applikationen

Die W3C HTML5 Spezifikation ermöglich mit einer sogenannten Cache Manifest Datei den Zugriff auf eine mobile Web Applikation, auch wenn die Benutzerin/der Benutzer nicht mit dem Internet verbunden ist (vgl. Hickson, 2011b). Diese Spezifikation kann für mit jQuery Mobile realisierten mobilen Web Applikationen herangezogen werden. Die Offline Speicherung wird derzeit von den Webbrowsern der Plattformen Android, iOS (vgl. Firtman, 2010, S. 311) und BlackBerry unterstützt (vgl. BlackBerry Documentation, 2010b).

> *Ein Cache Manifest ist eine normale Textdatei, in der die URLs der zu speichernden Ressourcen aufgelistet werden. Sie muss UTF-8-kodiert sein und ihr MIME-Type muss text/cache-manifest lauten (Kröner, 2010, S. 154).*

Eine Cache Manifest Datei könnte beispielsweise folgendermaßen aussehen:

```
1  CACHE MANIFEST
2  # Kommentar
3  index.html
4  js/script.js
5  css/styles.css
6  images/logo.png
7  http://www.demo.at/bild.png
```

Jede Cache Manifest Datei muss, wie in Zeile 1 mit `CACHE MANIFEST` beginnen und jede zu speichernde Datei oder URL (Zeile 3 - 7) muss in einer eigenen Zeile stehen. Ein Kommentar wird mit beginnender Raute `#` angeführt werden. Die Dateierweiterung der Cache Manifest Datei kann individuell gewählt werden. Wenn die Datei beispielsweise unter `demo.manifest` abgespeichert wurde, wird diese über das Attribut `manifest` im `<html>` Tag eingebunden, `<html manifest="demo.manifest">`. Der MIME-Type ist in die `.htaccess` Datei mit `AddType text/cache-manifest .manifest` anzugeben. (vgl. Kröner, 2010, S. 154 - 155)

Wenn eine Website zum ersten Mal aufgerufen wird, werden alle Dateien die in der Cache Manifest Datei angeführt sind heruntergeladen. Wird die Website erneut aufgerufen, überprüft der Webbrowser den Inhalt der Cache Manifest Datei auf Änderungen und lädt bei etwaigen Änderungen die Dateien erneut herunter. Inhalte, die sich in den eingetragenen Dateien geändert haben, führen noch nicht zur

Aktualisierung des Caches. Es muss explizit der Inhalt in der Cache Manifest Datei verändert werden, um eine Aktualisierung herbeizuführen. Beispielsweise durch Hochzählen einer Versionsnummer in einem Kommentar. (vgl. Spiering & Haiges, 2010, S. 227)

Die Spezifikation für Offline Web Applications ermöglicht die Gliederung der Cache Manifest Datei in verschiedene Abschnitte.

```
1  CACHE MANIFEST
2
3  CACHE:
4  index.html
5  offline.html
6  js/script.js
7  css/styles.css
8
9  NETWORK:
10 *
11
12 FALLBACK:
13 / offline.html
```

Alle Dateien die unter dem Abschnitt `CACHE:` in Zeile 2 angegeben werden, stehen im Offline-Betrieb zur Verfügung. `CACHE:` ist optional und kann auch weggelassen werden, da alle Dateien die direkt unter `CACHE MANIFEST` stehen gespeichert werden. Dateien die unter `NETWORK:` in Zeile 9 angegeben werden, werden vom Caching ausgeschlossen und benötigen eine Internetverbindung. Damit der Zugriff auf externe Ressourcen, wie beispielsweise auf Google Maps im online Betrieb einwandfrei funktioniert, können unter `NETWORK:` mit der Angabe eines Sterns `*` alle Ressourcen einer Website, die eine Internetverbindung benötigen, erfasst werden. Der `FALLBACK:` Abschnitt ermöglicht die Angabe einer gecachten Datei, (beispielsweise `offline.html`) die als Ersatz angezeigt wird, wenn auf eine Ressource (`/` kennzeichnet alle Ressourcen) nicht zugegriffen werden kann. Alle Leerzeilen in einer Cache Manifest Datei werden ignoriert. (vgl. Spiering & Haiges, 2010, S. 229 - 230)

Zur manuellen Überprüfung und Manipulation der Cache Manifest Datei kann das globale Objekt `applicationCache` (Zeile 1) eingesetzt werden. Die Eigenschaft `status` verfügt über verschiedene Konstanten, um den aktuellen Application Cache Status abzufragen.

```
1  var cache = window.applicationCache;
2  alert(cache.status);
```

`cache.status` gibt eine Zahl von `0` bis `5` zurück, die für verschiedene Zustände stehen.

`0` (`UNCACHED`) bedeutet, dass für die aufgerufene Website keine Cache Manifest Datei vorhanden ist und die Website vom Netzwerk geladen wird.

`1` (`IDLE`) gibt an, dass der Cache auf dem neuesten Stand ist.

`2` (`CHECKING`) vergleicht die lokale Cache Manifest Datei mit der auf dem Server.

`3` (`DOWNLOADING`) zeigt an, dass Dateien heruntergeladen werden.

`4` (`UPDATEREADY`) gibt an, dass der Cache aktualisiert wurde.

`5` (`OBSOLETE`) bedeutet, dass der Cache nicht mehr aktuell ist, beispielsweise weil keine Cache Manifest Datei mehr auf dem Server vorhanden ist.

(vgl. Hickson, 2011c; Kröner, 2010, S. 163)

Die Application Cache API stellt einige Events zur Verfügung, mit welchen die Cache Manifest Datei auf ihre Funktionalität überprüft werden kann. Die Events werden bei jedem Aufruf einer Website mit einer angeführten Cache Manifest Datei aufgerufen.

`checking`: Ist immer das erste aufgerufene Event. Der Webbrowser versucht die Dateien in der Cache Manifest Datei zum ersten Mal herunterzuladen oder überprüft das Manifest auf Änderungen.

`noupdate`: Gibt an, dass sich die Cache Manifest Datei nicht geändert hat.

`downloading`: Zeigt an, dass die in der Cache Manifest Datei angegebenen Ressourcen erstmalig oder aufgrund einer Aktualisierung erneut heruntergeladen werden.

`progress`: Die Dateien werden heruntergeladen.

`cached`: Die Dateien wurden heruntergeladen.

`updateready`: Die Ressourcen in der Cache Manifest Datei wurden erneute heruntergeladen und sind jetzt aktualisiert. Mit der Methode `swapCache()` kann der neue Cache angesteuert werden.

`obsolete`: Die Anfrage zur Cache Manifest Datei verursachte Fehler, weshalb der Application Cache gelöscht wird.

`error`: Es sind Fehler beim Herunterladen der Dateien aufgetreten. Die Cache Manifest Datei hat sich während des Herunterladens verändert oder die Website/Cache Manifest Datei konnte nicht geladen werden.

(vgl. Hickson, 2011d; Kröner, 2010, S. 155 - 156)

Da bei einer Aktualisierung des Caches, die Website nicht sofort neu geladen wird, kann die Methode `swapCache()` im Zusammenhang mit dem Event `updateready` angewendet werden.

```
1  window.applicationCache.addEventListener('updateready', function(e){
2     window.applicationCache.swapCache();
3     if (confirm('Eine neue Version der Website ist vorhanden. Soll
      diese geladen werden?')) {
4           window.location.reload();
5     }
6  }, false);
```

Wenn die Ressourcen aktualisiert wurden, wird das Event `updateready` in Zeile 1 aufgerufen. In Zeile 2 `window.applicationCache.swapCache();` kann dem Webbrowser mitgeteilt werden, dass ein neuer Cache zur Verfügung steht. In Zeile 3 wird der Benutzerin/dem Benutzer ein Dialog angezeigt und darauf hingewiesen, dass eine neue Version der Website vorhanden ist. Stimmt die Benutzerin/der Benutzer dem Neuladen der Website zu, wird `window.location.reload();` in Zeile 4 aufgerufen. (vgl. Kröner, 2010, S. 163 - 164)

6.6 Lokale Datenbank

Die W3C Spezifikation Web SQL Database definiert eine API, die eine lokale Speicherung von Daten in einer SQL Datenbank ermöglicht. Mit jQuery Mobile realisierte mobile Web Applikationen können diese Spezifikation einsetzen. W3C hat jedoch die Weiterentwicklung dieses Standardisierungsentwurfs aus folgendem Grund eingestellt:

> *... all interested implementors have used the same SQL backend (Sqlite), but we need multiple independent implementations to proceed along a standardisation path (Hickson, 2010a).*

Mit der Indexed Database API arbeitet W3C an einem neuen Standard für die clientseitige Speicherung von Daten (vgl. Mehta et al., 2011). Da Web SQL Database, im Gegensatz zu Indexed Database, bereits in den Webbrowser der Plattformen

Android, iOS und BlackBerry implementiert ist (vgl. Leenheer, 2010), wird der Einsatz dieser lokalen Speichermöglichkeit genauer erläutert. Die Symbian Plattform unterstützt Web SQL Database derzeit nicht in ihrem Webbrowser (vgl. Leenheer, 2010).

Die WebKit Browser implementieren SQLite für die clientseitige Datenbank API (vgl. Rogers, 2010; SQLite, 2011). Mit der Methode `openDatabase()` kann eine neue Datenbank erstellt oder eine bereits bestehende Datenbank geöffnet werden.

```
var db = openDatabase('demo', '1.0', 'Demo Datenbank', 5*1024*1024);
```

Der Methode können vier Argumente übergeben werden. Der Name der Datenbank `'demo'`, die Version der Datenbank `'1.0'`, eine Beschreibung der Datenbank `'Demo Datenbank'` und die Größe der Datenbank wird in Bytes angegeben, `5*1024*1024` entspricht 5 MB (Megabyte). (vgl. Van Kesteren, 2008) In der Spezifikation werden 5 MB als Datenbankgröße empfohlen, da bis 5 MB in den meisten mobilen Webbrowsern kein Dialog erscheint, ob für eine mobile Web Applikation lokaler Speicher verwendet werden darf.

Mit der Methode `changeVersion()` kann die Versionsnummer einer Datenbank verifiziert und geändert werden sowie zur gleichen Zeit das Schema einer Datenbank aktualisiert werden. Mit `db.version()` kann die aktuelle Version der Datenbank abgefragt werden. (vgl. Hickson, 2010c)

Nachdem die Datenbank erstellt wurde, kann auf diese die `transaction()` Methode angewandt werden, um SQL Befehle auszuführen. (vgl. Spiering & Haiges, 2010, S. 206 - 207)

> *Eine Transaktion stellt eine atomare Einheit dar und besteht aus mehreren Befehlen, die entweder alle erfolgreich ausgeführt werden können oder alle zurückgenommen werden (Spiering & Haiges, 2010, S. 206).*

```
1  db.transaction(transactionCallback, errorCallback, successCallback);
2  function transactionCallback(t) {
3     t.executeSql('CREATE TABLE IF NOT EXISTS demo (id INTEGER NOT
   NULL PRIMARY KEY AUTOINCREMENT, title TEXT)');
4  }
5  function errorCallback(error) {
6     alert('Message: ' + error.message + ' Code: ' + error.code);
7  }
```

```
8  function successCallback() {
9     alert('successCallback!');
10 }
```

Der `transaction()` Methode in Zeile 1 können drei Argumente übergeben werden, die alle eine Callback Funktion darstellen. An das erste Argument `transactionCallback` (Zeile 2) wird ein SQL Transaktionsobjekt `t` übergeben, auf das die Methode `executeSql()` in Zeile 3 zum Ausführen von SQL Befehlen, beispielsweise zum Erstellen einer neuen Tabelle, angewendet werden kann. (vgl. Van Kesteren, 2008) Das zweite Argument `errorCallback` (Zeile 5) wird bei einem Transaktionsfehler aufgerufen und ein `SQLError` Objekt `error` übergeben, das über die Attribute `code` und `message` verfügt. `message` beschreibt den Fehler auf Englisch und `code` kann zwischen acht verschiedenen Fehlern unterscheiden. (vgl. Hickson, 2010b)

`0 (UNKNOWN_ERROR)`: Ein Fehler, der nicht im Zusammenhang mit der Datenbank steht, ist aufgetreten.

`1 (DATABASE_ERR)`: Der SQL Befehl ist möglicherweise fehlerhaft und wird nicht von der Datenbank verstanden.

`2 (VERSION_ERR)`: Die aktuelle Datenbank Version entspricht nicht der erwarteten Version, weshalb ein Fehler aufgetreten ist.

`3 (TOO_LARGE_ERR)`: Die von der Datenbank zurück gelieferten Daten sind zu groß.

`4 (QUOTA_ERR)`: Der maximale Speicherplatz der Datenbank wurde erreicht und die Benutzerin/der Benutzer hat einer Speicherplatzerhöhung nicht zugestimmt.

`5 (SYNTAX_ERR)`: Die Syntax des SQL Befehls ist fehlerhaft oder die Anzahl der Argumente stimmt nicht mit der Anzahl der Platzhalter `?` überein.

`6 (CONSTRAINT_ERR)`: Ein `INSERT`, `UPDATE` oder `REPLACE` SQL Befehl konnte aufgrund eines Constraint Fehlers nicht ausgeführt werden. Beispielsweise wird eine Zeile mit einem `PRIMARY KEY` eingefügt, der bereits in der Tabelle vorhanden ist.

`7 (TIMEOUT_ERR)`: Das Objekt für die Transaktion konnte innerhalb einer bestimmten Zeit nicht erzeugt werden.

Das dritte Argument `successCallback` (Zeile 8) wird nach einer erfolgreichen Transaktion aufgerufen. (vgl. Hickson, 2010b; Spiering & Haiges, 2010, S. 208 - 209)

```
1  db.transaction(function(t) {
2     t.executeSql('INSERT INTO demo (title) VALUES (?)', [title],
   sqlStatementCallback, sqlStatementErrorCallback);
3  });
4  function sqlStatementCallback(t, result) {
5     refreshData();
6  }
7  function sqlStatementErrorCallback(t, error) {
8     alert('Message: ' + error.message + ' Code: ' + error.code);
9  }
```

Der Methode `executeSql()` in Zeile 2 können vier Argumente übergeben werden. Zunächst wird ein SQL Statement, beispielsweise `'INSERT INTO demo (title) VALUES (?)'`, angegeben. Zur Vermeidung von SQL Injections werden Platzhalter `?` verwendet, die vom zweiten angegebenen Argument, einem Array `[title]`, ersetzt werden. (vgl. Van Kesteren, 2008) Ohne die Verwendung von Platzhaltern `"INSERT INTO demo (title) VALUES ('" + title + " ')"`, kann die Benutzerin/der Benutzer zum Beispiel im Eingabefeld `'; DROP TABLE demo; --` eingeben, was dazu führt, dass die Tabelle `demo` gelöscht wird. SQLite ignoriert den restlichen SQL Befehl, da durch Angabe der beiden Bindestriche `--` dieser als Kommentar angesehen wird. (vgl. Spiering & Haiges, 2010, S. 210 - 211) Das dritte Argument `sqlStatementCallback` stellt eine Callback Funktion dar, die aufgerufen wird, wenn das SQL Statement erfolgreich ausgeführt wurde. An die Funktion (Zeile 4) wird ein Transaktionsobjekt `t` und ein `SQLResultSet` Objekt `result`, das die Daten einer SQL Abfrage enthält, übergeben. In diesem Beispiel ist das `result` Objekt ohne Bedeutung, da keine `SELECT` Abfrage auf die Datenbank ausgeführt wurde. Das vierte Argument `sqlStatementErrorCallback` wird bei nicht erfolgreicher Ausführung des SQL Statements aufgerufen (Zeile 7). (vgl. Van Kesteren, 2008)

Folgender JavaScript Quellcode zeigt ein kleines Beispiel zum `SQLResultSet` Objekt `result` in Zeile 4, das die Daten aus der `SELECT` Abfrage in Zeile 2 enthält und mit einer `for` Schleife (Zeile 5 - 8) ausgegeben wird. (vgl. Rogers, 2010)

```
1  db.transaction(function(t) {
2     t.executeSql("SELECT * FROM demo", [], sqlStatementCallback,
   sqlStatementErrorCallback);
3  });
4  function sqlStatementCallback(t, result) {
5     for(var i = 0; i < result.rows.length; i++) {
6           var row = result.rows.item(i);
7           alert("ID: " + row.id + " Titel: " + row.title);
8     }
9  }
```

6.7 Kamera

W3C arbeitet in der HTML Media Capture Spezifikation an einem Standard für den Zugriff auf Bild, Audio und Video Inhalte über ein HTML Eingabefeld. Mit folgendem Quellcode kann eine Benutzerin/ein Benutzer ein Bild über die Kamera hochladen.

```
<input type="file" accept="image/*;capture=camera">
```

Dem `input` Eingabefeld wird ein neues Attribut `accept` mit den Werten `image/*`, `audio/*` oder `video/*` hinzugefügt. Der Parameter `capture` beschreibt den Medientyp mit `camera`, `camcorder`, `microphone`, `filesystem`. (vgl. Oksanen & Hazaël-Massieux, 2010)

Ein weiterer W3C Standard, die Media Capture API, ermöglicht den Zugriff auf die Kamera über JavaScript.

```
1  navigator.device.capture.captureImage(success, error, {limit: 2});
2  function success(data) {
3    var container = document.createElement('div');
4    for (var i in data) {
5      var img = document.createElement('img');
6      img.src = data[i].url;
7      container.appendChild(img);
8    }
9    document.body.appendChild(container);
10 }
11 function error(err) {
12   if (err.code === err.CAPTURE_INTERNAL_ERR) {
13     alert('The capture failed due to an internal error.');
14   }
15   else {
16     alert('Other error occured.');
17   }
18 }
```

In Zeile 1 werden der Methode `navigator.device.capture.captureImage()` drei Parameter `(success, error, {limit: 2})` übergeben. Nachdem die Kamera Applikation eines Smartphones erfolgreich Bilder aufgenommen hat, wird die Funktion `success` mit dem Argument `data` in Zeile 2 aufgerufen, in welcher alle Bilder in einem `div` Container ausgegeben werden. Bei einem aufgetretenen Fehler wir die Funktion `error` in Zeile 11 aufgerufen. Der dritte Parameter kann verschiede Optionen enthalten, beispielsweise wird über das Attribut `limit` die maximale Anzahl der aufnehmbaren Bilder definiert. (vgl. Tran et al., 2010)

Derzeit ist keiner dieser Standards in einem der mobilen Webbrowser der vier Plattformen Symbian, Android, BlackBerry und iOS implementiert (vgl. Leenheer, 2010).

6.8 Test

Zum Testen einer mit jQuery Mobile erstellten mobilen Web Applikation können Desktop Webbrowser oder Simulatoren/Emulatoren herangezogen werden. Der Safari und Chrome Desktop Webbrowser eignen sich sehr gut zum Testen, da diese viele HTML5 Spezifikationen (z.B. Geolocation, Web SQL Database, Offline Web Applications) unterstützen. In Firefox ist beispielsweise Web SQL Database nicht implementiert. Weiters besteht die Möglichkeit über einen sogenannten User Agent Switcher die mobilen Webbrowser verschiedener Plattformen innerhalb von Desktop Webbrowser (z.B. Safari, Chrome, Firefox) zu simulieren. Jeder mobile Webbrowser verfügt über einen eigenen User Agent String (vgl. User Agent String, 2011), der über verschiedene Add-Ons (vgl. Pederick, 2011) den User Agent String der Desktop Webbrowser überschreiben kann. (vgl. Firtman, 2010, S. 319 - 320)

Zusätzlich können die jeweiligen Simulatoren/Emulatoren der Plattformen installiert werden. Ein Emulator emuliert die Hardware und das Betriebssystem mobiler Geräte. Ein Simulator ist ein einfacheres Werkzeug, das im Gegensatz zum Emulator keine Hardware emuliert und nicht auf dem echten Betriebssystem aufbaut. (vgl. Firtman, 2010, S. 75 - 76) Nokia bietet über den Dienst Remote device access den Zugriff auf reale Smartphones zum Testen an (vgl. Forum Nokia, 2011d). Der Android Emulator ist im SDK beinhaltet und kann von Android Developers (2011c) für Mac OS X, Linux und Windows heruntergeladen werden. Die Simulatoren der BlackBerry Plattform können separat unter BlackBerry Developers (2011h) nur für Windows heruntergeladen werden. Damit ein Zugriff auf das Internet und das Verschicken von E-Mails möglich ist, muss zusätzlich das BlackBerry Email and MDS Services Simulator Packet heruntergeladen werden. Der Simulator für die iOS Plattform ist im Xcode enthalten und kann von Apple Developer (2011a) nur für Mac OS X heruntergeladen werden.

6.9 Distribution

Mobile Web Applikationen basierend auf dem jQuery Mobile Framework können nach Fertigstellung sofort ins Internet gestellt werden, um weltweit über mobile Webbrowser abgerufen werden zu können. Da mobile Web Applikationen technisch gesehen eine Art Website sind, können diese auch in der Google Suchmaschine aufgenommen

werden. (vgl. Spiering & Haiges, 2010, S. 337 - 338) Auf den Plattformen Symbian, Android, BlackBerry und iOS kann eine mobile Web Applikation als Lesezeichen mit Icon zum Home-Bildschirm hinzugefügt werden und sieht somit wie eine native Applikation aus. Dafür wird im `<head>` Bereich folgendes Link Element `<link rel="apple-touch-icon" href="images/apple-icon.png" />` eingefügt. Auf der iOS Plattform wird das Icon automatisch mit runden Ecken und einem Glanzeffekt versehen. Soll dies verhindert werden, muss das Attribut `rel="apple-touch-icon-precomposed"` im `link` Element verwendet werden. Bei den Plattformen Symbian und Android muss zunächst ein Lesezeichen erstellt werden, um dieses anschließend zum Home-Bildschirm hinzufügen zu können. iOS und BlackBerry ermöglichen ein direktes Hinzufügen. Einzig auf der Symbian Plattform wird das Icon nicht angezeigt und das Lesezeichen kann auch nicht direkt über den mobilen Webbrowser, sondern nur vom Home-Bildschirm aus, zum Home-Bildschirm hinzugefügt werden.

Damit sich eine mobile Web Applikation noch mehr wie eine native Applikation anfühlt, stellt die iOS Plattform weitere Konfigurationen zur Verfügung. Mit `<link rel="apple-touch-startup-image" href="images/apple-startup.png" />` kann ein Bild definiert werden, das solange angezeigt wird, bis die mobile Web Applikation fertig geladen ist. Die Angabe `<meta name="apple-mobile-web-app-capable" content="yes" />` versteckt die Adress- und Fußzeile des Safari Webbrowsers und stellt die Web Applikation im Vollbild-Modus dar. Zusätzlich kann mit dem JavaScript Quellcode `window.navigator.standalone` ermittelt werden, ob sich die Web Applikation im Vollbild-Modus befindet. Mit `<meta name="apple-mobile-web-app-status-bar-style" content="black" />` kann das Aussehen der Statuszeile zum Beispiel auf Schwarz geändert werden. `content="black-translucent"` macht die Statuszeile transparent. (vgl. Apple Developer, 2011e) Da viele Benutzerinnen/Benutzer nicht über die Möglichkeit Bescheid wissen, mobile Web Applikationen zum Home-Bildschirm hinzufügen zu können, kann mit Hilfe der JavaScript Bibliothek Mobile Bookmark Bubble am Ende der mobilen Web Applikation in einem Dialog darauf hingewiesen werden. Diese JavaScript Bibliothek ist derzeit speziell auf den Safari Webbrowser der iOS Plattform ausgerichtet. (vgl. Google, 2011)

Es gibt bereits ein paar Stores über die mobile Web Applikationen vertrieben werden können. Dazu gehören zum Beispiel der Web Apps Store von Apple (vgl. Apple, 2011f), OpenAppMkt (2010) oder AppStoreHQ (2010).

7 Sencha Touch

Die Begründer des Ext JS Frameworks für browserübergreifende Desktop Web Applikationen haben sich mit den Entwicklerinnen/Entwicklern von jQTouch, ein jQuery Plug-In für die Erstellung von mobilen iPhone Web Applikationen und Raphaël, eine JavaScript Bibliothek für Vektorgrafiken im Web, zusammengeschlossen und die Firma Sencha Inc. gegründet (vgl. Ihlenfeld, 2010). Sencha Touch, das erste Produkt von Sencha Inc., baut auf der selben Grundlage wie Ext JS auf und ist ein JavaScript Framework für die Entwicklung von plattformübergreifenden mobilen Web Applikationen speziell für Smartphones basierend auf den Plattformen iOS, Android und BlackBerry. Das Framework basiert auf HTML5 für die Bereitstellung von HTML5 spezifischen Komponenten, wie Geolocation, Audio und Video und CSS3 für vordefinierte UI Elemente. Mit Sencha Touch realisierte Applikationen werden mit reinem JavaScript Quellcode erstellt. (vgl. Sencha, 2011a) Die kommerzielle Software Lizenz sowie die Open Source Lizenz von Sencha Touch ist für Entwicklerinnen/Entwickler kostenlos. Die kommerzielle OEM (Original Equipment Manufacturer) Lizenz, das heißt die Vermarktung und Weiterentwicklung des Sencha Touch SDKs unter einem anderen Namen, ist für Herstellerinnen/Hersteller mit Kosten verbunden. (vgl. Sencha, 2011b)

7.1 Installation und Entwicklungsumgebung

Das Sencha Touch Framework inklusive Beispiele kann von Sencha (2011c) heruntergeladen werden.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4     <meta charset="utf-8">
5     <title>Sencha Touch</title>
6     <link rel="stylesheet" href="sencha-touch.css" type="text/css">
7     <link rel="stylesheet" href="mycss.css" type="text/css">
8     <script type="text/javascript" src="sencha-touch.js"></script>
9     <script type="text/javascript" src="myjs.js"></script>
10 </head>
11 <body></body>
12 </html>
```

Der erste Schritt für die Entwicklung einer Sencha Touch Applikation ist die Erstellung einer HTML Datei, die auf dem HTML5 Doctype (Zeile 1) basiert und die CSS `sencha-`

touch.css (Zeile 6) sowie JavaScript sencha-touch.js (Zeile 8) Datei des Sencha Touch Frameworks im <head> Bereich beinhaltet. Das Sencha Touch Framework verfügt neben dem Standard Seiten-Design (sencha-touch.css) über drei weitere vordefinierte Seiten-Designs, die dem UI von iOS, Android und BlackBerry angepasst sind und ebenfalls als CSS Dateien im Framework enthalten sind (vgl. Sencha, 2011d). Sencha Touch ermöglicht auch die individuelle Anpassung des Seiten-Designs. Die im Sencha Touch CSS definierten Bilder sind data:image/png;base64 kodiert und müssen somit nicht, wie bei jQuery Mobile separat in das Projekt kopiert werden. Weiters werden auch eigene CSS mycss.css (Zeile 7) und JavaScript myjs.js (Zeile 9) Dateien, in denen die eigentliche Entwicklung statt findet, inkludiert. Sencha Touch stellt auch eine Debug JavaScript Version sencha-touch-debug.js zur Verfügung, die während der Entwicklung der mobilen Applikation zum Eruieren von Fehlern hilfreich sein kann. Das <body> Element bleibt leer, da das Framework den Seiteninhalt automatisch via JavaScript generiert. (vgl. Sencha, 2011e)

Für die Entwicklung von mobilen Web Applikationen mit Sencha Touch kann jede beliebige Entwicklungsumgebung, die HTML, CSS und JavaScript unterstützt, herangezogen werden.

7.2 User Interface

Sencha Touch verfügt über eine Vielzahl vorgefertigter UI Komponenten, die einer nativen Applikation sehr ähnlich sind.

7.2.1 Seitenstruktur

Die gesamte Applikation wird mit JavaScript Quellcode, der sich innerhalb der eigens erstellten JavaScript Datei befindet, aufgebaut.

```
1  Ext.setup({
2      phoneStartupScreen: 'images/apple-startup.png',
3      phoneIcon: 'images/apple-icon.png',
4      glossOnIcon: true,
5      statusBarStyle: 'black',
6      onReady: function() {
7           var toolbar = new Ext.Toolbar({
8                title: 'Kopfzeile'
9           });
10          new Ext.Panel({
11               fullscreen: true,
12               layout: 'hbox',
13               dockedItems: [ toolbar,
14                    {
```

```
15                              xtype: 'toolbar',
16                              ui: 'light',
17                              items: [{
18                                    text: 'Button'
19                              }]
20                        }, {
21                              xtype: 'toolbar',
22                              dock: 'bottom',
23                              title: 'Fußzeile'
24                        }
25                  ],
26                  defaults: {
27                        xtype: 'button'
28                  },
29                  items: [{
30                        text: 'Button 1'
31                  },{
32                        text: 'Button 2',
33                  },{
34                        text: 'Button 3'
35                  }]
36            });
37      }
38 });
```

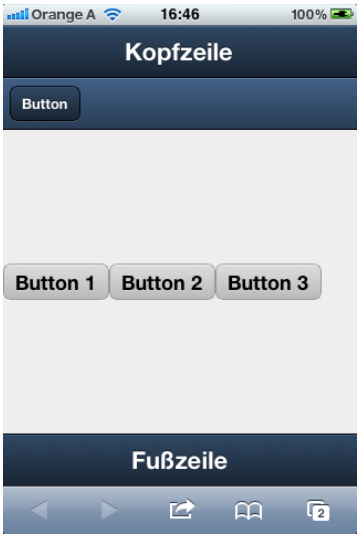


Abbildung 28: Sencha Touch Seitenstruktur

Das Quellcodebeispiel wird in Abbildung 28 veranschaulicht. Mit der `Ext.setup()` Methode in Zeile 1 wird eine Seite für ein Smartphone erstellt, die verschiedene Eigenschaften beinhalten kann. Die folgenden beschriebenen Eigenschaften (Zeile 2 - 5) werden wirksam, wenn die mobile Web Applikation auf den Home-Bildschirm eines Smartphones gelegt wird. Legt die Benutzerin/der Benutzer die mobile Web Applikation auf den Home-Bildschirm eines iPhones, wird diese automatisch im Vollbild-Modus ohne die Adress- und Fußzeile des Safari Webbrowsers gestartet.

`String phoneStartupScreen`: Über die Eigenschaft `phoneStartupScreen` kann ein Bild (320x460) definiert werden, das solange angezeigt wird, bis das iPhone die mobile Web Applikation fertig geladen hat.

`String phoneIcon`: Über die Eigenschaft `phoneIcon` wird ein Icon (57x57), das auf dem Home-Bildschirm eines Smartphones angezeigt wird, definiert.

`Boolean glossOnIcon`: Die Eigenschaft `glossOnIcon` gibt an, ob dem Icon auf einem iPhone ein Glanz Effekt hinzugefügt werden soll oder nicht.

`String statusBarStyle`: Mit der Eigenschaft `statusBarStyle` kann das Aussehen der Statuszeile auf einem iPhone geändert werden.

(vgl. Sencha Touch API, 2011a)

Der JavaScript Quellcode innerhalb der Methode `onReady()` (Zeile 6) wird ausgeführt, wenn die Seite vollständig geladen wurde. Innerhalb von `onReady()` wird ein Panel mit `new Ext.Panel()` (Zeile 10) erstellt, das den Hauptcontainer darstellt und unterschiedliche UI Komponenten enthalten kann. Abgeleitet vom Haupt-Panel stellt Sencha Touch spezielle Panels, wie beispielsweise `Carousel` (Kapitel 7.2.5), `TabPanel` (Kapitel 7.2.3), `FormPanel` (Kapitel 7.2.6) oder `NestedList` (Kapitel 7.2.7) zur Verfügung. Im Panel können verschiedene Eigenschaften definiert werden. (vgl. Sencha, 2011e; Neil, 2010a)

`Boolean fullscreen`: Wenn die Eigenschaft `fullscreen` (Zeile 11) auf `true` gesetzt wird, erhält das Panel eine Breite und Höhe von 100 Prozent.

`String layout`: Die Eigenschaft `layout` (Zeile 12) gibt an, wie die hinzugefügten Komponenten zur Eigenschaft `items` (Zeile 29) angeordnet werden sollen. Der Standardwert von `layout` ist `auto`. Dieser Layouttyp hat eine Breite von 100 Prozent, die Höhe wird dem Inhalt automatisch angepasst. Ein Layout vom Typ `fit` füllt die gesamte vorhandene Fläche in vertikaler und horizontaler Richtung aus. Dieses Layout akzeptiert nur eine dem Panel hinzugefügte Komponente. Das `card` Layout funktioniert auf die gleiche Weise wie `fit`, ermöglicht jedoch das Hinzufügen von mehreren Komponenten, da immer nur eine Komponente angezeigt wird. Solange das `card` Layout über keine Navigationsmöglichkeit verfügt, kann über die Methode `setActiveItem(1)` die entsprechende Komponente in den Vordergrund gebracht werden. Sencha Touch verfügt über zwei spezielle Panels, `Carousel` (Kapitel 7.2.5) und `TabPanel` (Kapitel 7.2.3), die dem `card` Layout ein UI für die Navigation zwischen den verschiedenen Komponenten bereitstellen. Weiters besteht die Möglichkeit die

Komponenten über die Layouttypen `vbox` und `hbox` (Zeile 12) vertikal bzw. horizontal auszurichten. Diese Layouttypen passen die Höhe und Breite automatisch dem Inhalt an.

`Array dockedItems`: Der Eigenschaft `dockedItems` (Zeile 13) können eine Komponente oder mehrere Komponenten als Array übergeben werden. `dockedItems` ermöglicht über den Parameter `dock` (Zeile 22) die Platzierung einer Komponente oben (`top`), rechts (`right`), unten (`bottom`) oder links (`left`) innerhalb eines Panels und wird beispielsweise bei Kopf- und Fußzeilen angewendet. Ohne Angabe von `dock` wird eine Toolbar standardmäßig am Beginn einer Web Applikation angezeigt.

`Array items`: Der Eigenschaft `items` (Zeile 29) können eine Komponente oder mehrere Komponenten als Array übergeben werden. Diese werden abhängig vom angegebenen `layout` (Zeile 12) Typ arrangiert.

`defaults`: Innerhalb der Eigenschaft `defaults` (Zeile 26) können Standardeinstellungen für hinzugefügte Komponenten in `items` vorgenommen werden. In Zeile 27 wird beispielsweise definiert, dass alle hinzugefügten Komponenten als Schaltflächen `xtype: 'button'` aufscheinen sollen.

Sencha Touch Komponenten können auf zwei Arten erstellt werden. Beispielsweise kann eine Kopf- oder Fußzeile entweder, wie in Zeile 7 über die Klasse `new Ext.Toolbar()` oder über die Eigenschaft `xtype: 'toolbar'` (Zeile 15) erstellt werden. Über die Eigenschaft `ui` (Zeile 16) kann das Aussehen der Komponenten mit vordefinierten Stilen, wie `'dark'` und `'light'` für Toolbars verändert werden. Weiters besteht die Möglichkeit die Eigenschaften `dockedItems` und `items` zu verschachteln. Beispielsweise wird in Zeile 17 der über `dockedItems` hinzugefügten Toolbar eine Schaltfläche über `items` hinzugefügt. Komponenten die zu einer Toolbar hinzugefügt werden sind standardmäßig Schaltflächen, weshalb die Eigenschaft `xtype: 'button'`, wie in Zeile 27 nicht gesetzt werden muss. (vgl. Sencha, 2011e; Sencha Touch API, 2011b; Neil, 2010b)

7.2.2 Seitenübergang

Sencha Touch verfügt über verschiedene Seitenübergang-Effekte, wie `fade`, `slide`, `flip`, `cube`, `pop`, und `wipe` die auf den Panels `TabPanel` (Zeile 1) oder `Carousel` mit einem Layout des Typs `card` (Zeile 4) angewendet werden können. Über die

Eigenschaft `cardSwitchAnimation` in Zeile 3 kann der Seitenübergang-Effekt definiert werden.

```
1  new Ext.TabPanel({
2     fullscreen: true,
3     cardSwitchAnimation: {type: 'cube'},
4     layout: 'card',
5     items: [item1, item2, item3]
6  });
```

(vgl. Neil, 2010b; Sencha Touch API, 2011c)

Sencha Touch stellt vordefinierte Dialoge zur Verfügung, die über `Ext.Msg()` definiert werden können. Beispielsweise wird über `Ext.Msg.alert('Alert Message!');` ein Alert-Dialog und über `Ext.Msg.confirm('Confirm Message!',function(){});` ein Confirm-Dialog aufgerufen. (vgl. Sencha Touch API, 2011d)

7.2.3 Kopf- und Fußzeile

Eine Kopf- bzw. Fußzeile mit Navigationsleiste kann über die spezielle Panel Komponente `Ext.TabPanel()` (Zeile 1) hinzugefügt werden. Innerhalb der Eigenschaft `items` (Zeile 4) wird die gewünschte Anzahl an Panels in einem Array definiert und die Namen der Schaltflächen innerhalb der Navigationsleiste werden über `title` (Zeile 5) vergeben. Standardmäßig wird die Navigationsleiste am Beginn einer Web Applikation angezeigt (vgl. Abbildung 29). Die Eigenschaft `sortable` (Zeile 3) ermöglicht die Sortierung der Tabs durch Berühren, Halten und Verschieben eines Tabs an die gewünschte Stelle.

```
1  new Ext.TabPanel({
2     fullscreen: true,
3     sortable: true,
4     items: [{
5           title: 'Panel 1'
6     },{
7           title: 'Panel 2'
8     },{
9           title: 'Panel 3'
10    }]
11 });
```

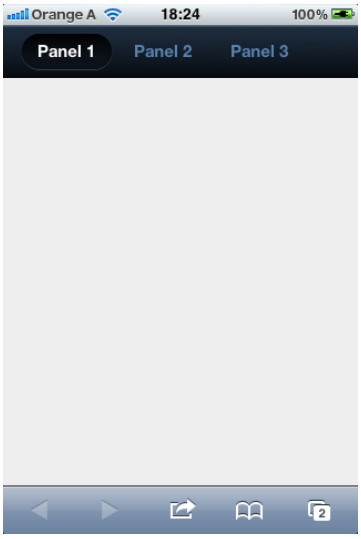


Abbildung 29: Sencha Touch TabPanel

Über die Eigenschaft `tabBar` in Zeile 3 kann die Navigationsleiste am Ende einer Web Applikation mit `dock: 'bottom'` (Zeile 4) hinzugefügt und die Schaltflächen mit `pack: 'center'` (Zeile 6) zentriert werden. Das Layout der Navigationsleiste verändert sich durch die Angabe von `dock: 'bottom'` automatisch. Die Schrift der Schaltflächen-Titel wird kleiner und oberhalb eines Titels besteht die Möglichkeit ein Icon über `iconCls` (Zeile 11) zu definieren. Sencha Touch verfügt über mehr als 300 Icons, die im heruntergeladenen Framework unter *resources/themes/images/default/pictos* zu finden sind. Unter Sencha (2011f) werden die bereits im CSS vordefinierten Icons veranschaulicht. `badgeText` (Zeile 12) legt eine Kennzeichnung in Form eines roten Kreises über die gewünschte Schalfläche. Zusätzlich kann jedem Panel eine Kopfzeile mit `xtype: 'toolbar'` (Zeile 14) über die Eigenschaft `dockedItems` (Zeile 13) hinzugefügt werden. Dieser Toolbar können wiederum Schaltflächen über die Eigenschaft `items` (Zeile 16) hinzugefügt werden. `xtype: 'spacer'` in Zeile 19 fügt einen Abstand zwischen den Schaltflächen ein und platziert die zweite Schaltfläche automatisch im linken Bereich der Kopfzeile. (vgl. Neil, 2011a; Abbildung 30)

```
1  new Ext.TabPanel({
2     fullscreen: true,
3     tabBar: {
4           dock: 'bottom',
5           layout: {
6                 pack: 'center'
7           }
8     },
9     items: [{
10          title: 'Panel 1',
11          iconCls: 'locate',
12          badgeText: '1',
13          dockedItems: [{
14               xtype: 'toolbar',
15               title: 'Titel 1',
16               items: [{
17                 text: 'Button'
18               }, {
19                 xtype: 'spacer'
20               },{
21                 text: 'Button'
22               }]
23          }]
24    },{
25          title: 'Panel 2',
26          iconCls: 'favorites'
27    },{
28          title: 'Panel 3',
29          iconCls: 'info'
30    }]
31 });
```

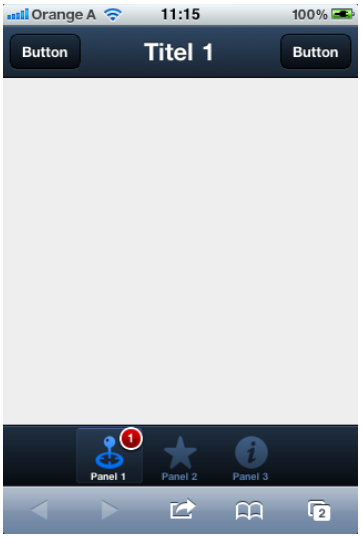


Abbildung 30: Sencha Touch tabBar

7.2.4 Schaltflächen

Eine Schaltfläche im Sencha Touch Framework kann über `xtype: 'button'` (Zeile 26) oder `Ext.Button()` definiert und über die Eigenschaft `ui` (Zeile 7) hinsichtlich Farbe und Form verändert werden. Der Stil einer Schaltfläche ist auch davon abhängig, ob diese in einer Toolbar (Zeile 5) oder innerhalb eines Panels (Zeile 28) platziert wurde. Weiters können einer Schalfläche mit `iconCls: 'star'` und `iconMask: true` (Zeile 20) ein Icon hinzugefügt werden. Im folgenden Beispiel und Abbildung 31 werden die verschiedenen vordefinierten Stile angeführt. (vgl. Neil, 2011a)

```
1  new Ext.Panel({
2     fullscreen: true,
3     layout: 'vbox',
4     dockedItems: [{
5           xtype: 'toolbar',
6                 items: [
7                       {text: 'back', ui: 'back'},
8                       {text: 'normal', ui: 'normal'},
9                       {text: 'small', ui: 'small'},
10                      {text: 'round', ui: 'round'},
11                      {text: 'forward', ui: 'forward'}
12                ]
13          }, {
14          xtype: 'toolbar',
15          dock: 'bottom',
16                items: [
17                      {text: 'action', ui: 'action'},
18                      {text: 'confirm', ui: 'confirm'},
19                      {text: 'decline', ui: 'decline'},
20                      {text: 'icon', ui: 'action', iconCls: 'star',
                  iconMask: true}
21                ]
22          }],
23    defaults: {
24          layout: 'hbox',
25          flex: 1,
26          defaults: {xtype: 'button'}
27    },
28    items: [{
29                items: [
30                      {text: 'back', ui: 'back'},
31                      {text: 'normal', ui: 'normal'},
32                      {text: 'icon', ui: 'normal', iconCls: 'star',
                  iconMask: true},
33                      {text: 'small', ui: 'small'},
34                      {text: 'round', ui: 'round'},
35                      {text: 'forward', ui: 'forward'},
36                ]}, {
37                items: [
38                      {text: 'action', ui: 'action'},
39                      {text: 'confirm', ui: 'confirm'},
40                      {text: 'decline', ui: 'decline'}
41                ]}, {
42                items: [
```

```
43                      {text: 'action round', ui: 'action-round'},
44                      {text: 'action small', ui: 'action-small'}
45                  ]}, {
46                  items: [
47                      {text: 'confirm round', ui: 'confirm-round'},
48                      {text: 'confirm small', ui: 'confirm-small'}
49                  ]}, {
50                  items: [
51                      {text: 'decline round', ui: 'decline-round'},
52                      {text: 'decline small', ui: 'decline-small'}
53                  ]}
54              ]
55 });
```

Abbildung 31: Sencha Touch Schaltflächen

7.2.5 Inhalte

Sencha Touch stellt mit `Ext.Carousel()` (Zeile 1) oder `xtype: 'carousel'` ein spezielles Panel zur Verfügung, in welchem die Möglichkeit besteht zwischen mehreren Panels, die über `items` (Zeile 3) hinzugefügt werden, zu wechseln (vgl. Abbildung 32). Carousel fügt automatisch eine Navigationsleiste aus Punkten hinzu, die je nach angegebener Richtung unten oder rechts platziert wird. Die Standardrichtung ist horizontal und kann über die Eigenschaft `direction='vertical'` in vertikal geändert werden. Wenn ein Panel aktiv ist, wird der Navigationspunkt dunkel dargestellt. (vgl. Sencha Touch API, 2011e)

```
1  new Ext.Carousel({
2      fullscreen: true,
3      items: [{
4              style: 'background-color:
#5E99CC'
5      },{
6              style: 'background-   color:
#759E60'
7      }]
8  });
```

Abbildung 32: Sencha Touch Carousel

Das Framework ermöglicht weiters die Einbindung von HTML5 Video Inhalten mit `Ext.Video()` oder `xtype: 'video'` (Zeile 4) und Audio Inhalten mit `Ext.Audio()` oder `xtype: 'audio'` (Zeile 7). Über die Eigenschaft `url` (Zeile 5) wird die gewünschte Audio- oder Videoquelle hinzugefügt. (vgl. Sencha Touch API, 2011f; Sencha Touch API, 2011g; Abbildung 33)

```
1  new Ext.Panel({
2      fullscreen: true,
3      items: [{
4              xtype: 'video',
5              url: "space.mp4"
6          },{
7              xtype: 'audio',
8              url: "crash.mp3"
9          }
10     ]
11 });
```

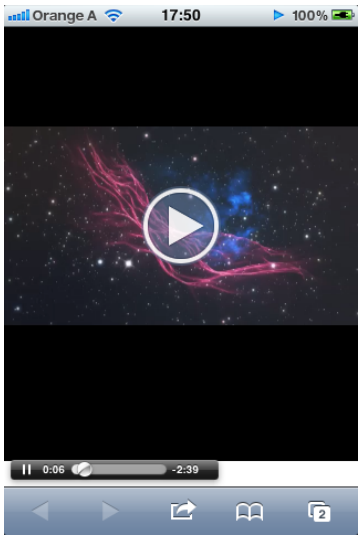


Abbildung 33: Sencha Touch Video und Audio

DOM Elemente können in Sencha Touch mit `Ext.util.Draggable` bewegbar und mit `Ext.util.Sortable` sortierbar gemacht werden (vgl. Sencha Touch API, 2011h; Sencha Touch API, 2011i).

7.2.6 Formulare

Formulare können über das spezielle Panel `Ext.form.FormPanel()` (Zeile 1) oder `xtype: 'form'` definiert werden. Sencha Touch erstellt für alle Formular Elemente, wie beispielsweise Eingabefelder, Kontrollkästchen oder Schieberegler speziell für berührungssensitive Smartphone-Bildschirme angepasste Bedienelemente. Alle Elemente erhalten eine flexible Breite, die sich an die Bildschirmbreite anpasst. Das Framework verwendet neue HTML5 Eingabefelder vom Typ `number` (Zahl), `email`, `url` (vgl. Abbildung 34), `tel` (Telefon), `search` (Suche) und `range` (Schieberegler), die über die Eigenschaft `xtype` (Zeile 4) definiert werden können. Die Android, BlackBerry und iOS Plattform passen jeweils die Tastaturbelegung für die Eingabefeldtypen `number` und `tel` für eine schnellere Eingabemöglichkeit automatisch an. iOS passt die Tastaturbelegung auch für die Eingabefeldtypen `email` und `url` (vgl. Abbildung 34) an. (vgl. Spiering & Haiges, 2010, S. 88) Auf Sencha (2011g) werden die verschiedenen vordefinierten Bedienelemente veranschaulicht. (vgl. Sencha Touch API, 2011j)

```
1  new Ext.form.FormPanel({
2     fullscreen: true,
3     items: [{
4           xtype: 'urlfield',
5           name: 'url',
6           label: 'Url'
7     }]
8  });
```

Abbildung 34: Sencha Touch Formulare

Ein Kalender zur Auswahl eines Datums (vgl. Abbildung 35) kann über die `Ext.DatePicker()` Komponente hinzugefügt werden (vgl. Sencha Touch API, 2011k).

Abbildung 35: Sencha Touch DatePicker

7.2.7 Listen

Eine Liste in Sencha Touch kann über `Ext.List()` (Zeile 22) erstellt werden. Diese verwendet die Eigenschaft `store` (Zeile 24), in welcher die Daten `data` (Zeile 10) für die Liste über `new Ext.data.Store()` (Zeile 4) angebunden werden. `Ext.data.Store()` benötigt die Registrierung eines Models `mit Ext.regModel()` (Zeile 1) mit den entsprechenden Feldern `fields` (Zeile 2), das über die Eigenschaft `model` (Zeile 5) zugewiesen wird. Die Eigenschaft `itemTpl` (Zeile 25) dient zur Angabe des anzuzeigenden Listenelements, im konkreten Beispiel wird der Vorname und Nachname `'{firstName} {lastName}'` angezeigt. Sencha Touch ermöglicht die Gruppierung von Listeneinträgen über die Eigenschaft `grouped` (Zeile 26) in Abhängigkeit mit der Methode `getGroupString` (Zeile 7). Diese Methode gibt den ersten Buchstaben des Nachnamens für die Gruppierung zurück. Die alphabetische Sortierung der Liste nach beispielsweise Nachnamen erfolg über `sorters` (Zeile 6) und mit `indexBar` (Zeile 27) wird rechts eine alphabetische Zeile zur schnelleren Durchsuchung der Liste hinzugefügt. (vgl. Sencha Touch API, 2011l; Sencha Touch API, 2011m; Neil, 2011b) Das Quellcodebeispiel wird in Abbildung 36 veranschaulicht.

```
1  Ext.regModel('Contact', {
2      fields: ['firstName', 'lastName']
3  });
4  var store = new Ext.data.Store({
5     model: 'Contact',
6     sorters: 'lastName',
7     getGroupString : function(record) {
8           return record.get('lastName')[0];
9      },
10    data: [
11          {firstName: 'Tommy', lastName: 'Maintz'},
```

```
12          {firstName: 'Rob', lastName: 'Dougan'},
13          {firstName: 'Ed', lastName: 'Spencer'},
14          {firstName: 'Jamie', lastName: 'Avins'},
15          {firstName: 'Aaron', lastName: 'Conran'},
16          {firstName: 'Dave', lastName: 'Kaneda'},
17          {firstName: 'Michael', lastName: 'Mullany'},
18          {firstName: 'Abraham', lastName: 'Elias'},
19          {firstName: 'Jay', lastName: 'Robinson'}
20      ]
21 });
22  new Ext.List({
23    fullscreen: true,
24    store: store,
25    itemTpl: '{firstName} {lastName}',
26    grouped: true,
27    indexBar: true
28 });
```

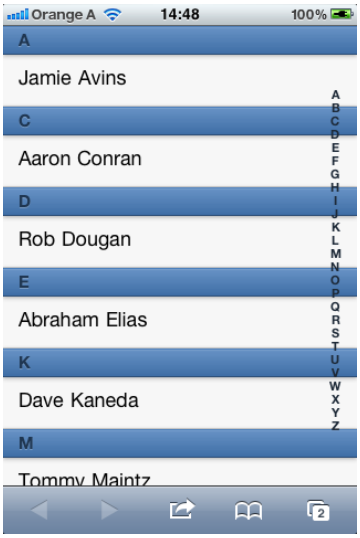


Abbildung 36: Sencha Touch Listen

Sencha Touch ermöglicht auch die Erstellung von verschachtelten Listen mit `Ext.NestedList` (vgl. Sencha Touch API, 2011n).

Listen können über die sogenannte Pull to Refresh Funktion, die von nativen Applikationen bekannt ist, aktualisiert werden. Benutzerinnen/Benutzer können eine Liste nach unten ziehen (vgl. Abbildung 37/1, 2) und führen somit eine automatische Aktualisierung (vgl. Abbildung 37/3) der Listeninhalte herbei. Sencha Touch stellt für diese Funktionalität das Plug-In `new Ext.ux.touch.ListPullRefresh` zur Verfügung. (vgl. Sencha, 2011h)

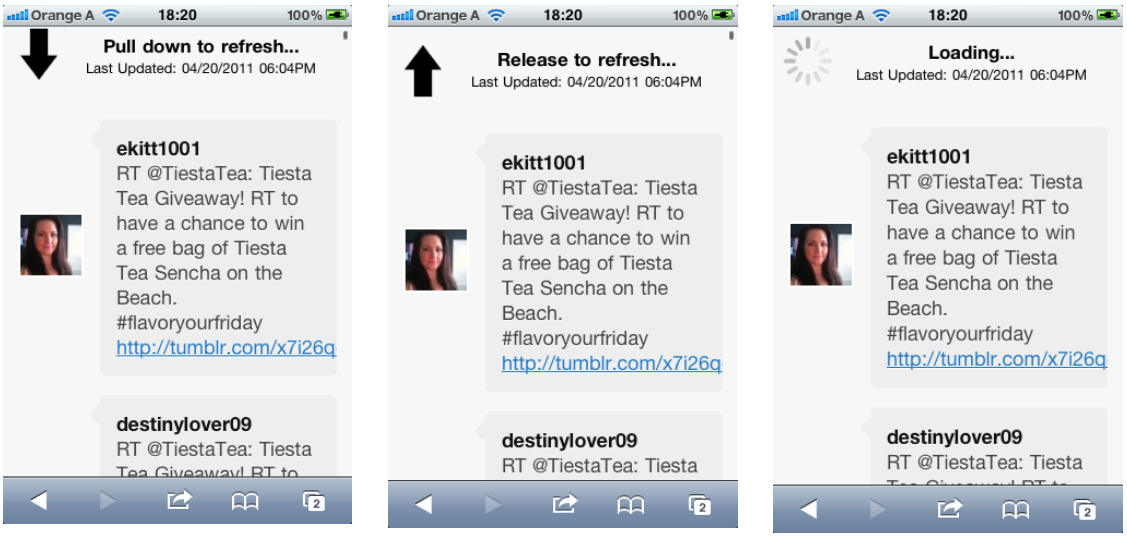


Abbildung 37: Sencha Touch Pull to Refresh (vgl. Sencha, 2011i)

7.2.8 Events

Sencha Touch verfügt über eine Vielzahl an Events (`touchstart`, `touchmove`, `touchend`, `touchdown`, `tap`, `tapstart`, `tapcancel`, `taphold`, `singletap`, `doubletap`, `swipe`, `swipeleft`, `swiperight`, `drag`, `dragstart`, `dragend`, `pinch`, `pinchstart`, `pinchend`), die speziell auf berührungssensitive Smartphones abgestimmt sind. Das Verhalten von Events kann auf verschiedenste Weise beeinflusst werden. Beispielsweise kann die Registrierung eines Events durch die Angabe einer Zeit in Millisekunden verzögert werden. (vgl. Neil, 2010c)

7.3 Geolocation

Die Klasse `Ext.util.GeoLocation()` (Zeile 1) basiert auf der W3C Geolocation API und ermöglicht die Bestimmung des aktuellen Standorts einer Benutzerin/eines Benutzers. Folgendes Beispiel zeigt, wie Geolocation in eine Sencha Touch Applikation eingebunden werden kann. `Boolean autoUpdate: false` (Zeile 2) verhindert die ständige Neubestimmung des Standorts. Mit `locationupdate` (Zeile 4) wird die aktuelle Position bestimmt. Breiten- und Längengrad können mit `geo.latitude` und `geo.longitude` (Zeile 5) abgerufen werden. Das `locationerror` (Zeile 7) Ereignis wird bei einem aufgetretenen Fehler aufgerufen. Mit `geo.updateLocation()` (Zeile 21) kann erneut eine Positionsbestimmung angefordert werden. (vgl. Sencha Touch API, 2011o)

```
1  var geo = new Ext.util.GeoLocation({
2      autoUpdate: false,
3      listeners: {
4          locationupdate: function (geo) {
```

```
5               alert(latitude: ' + geo.latitude + longitude: ' + geo.
                longitude);
6           },
7           locationerror: function (   geo,
8                                       bTimeout,
9                                       bPermissionDenied,
10                                      bLocationUnavailable,
11                                      message) {
12              if(bTimeout){
13                  alert('Timeout occurred.');
14              }
15              else{
16                  alert('Error occurred.');
17              }
18          }
19      }
20 });
21 geo.updateLocation();
```

7.4 Karte

Sencha Touch ermöglicht über die Komponente `Ext.Map()` (Zeile 1) die Einbindung einer Google Maps Karte in einer mobilen Web Applikation. Voraussetzung ist, dass im `<head>` Bereich der HTML Datei die Google Maps Bibliothek inkludiert ist. Genauere Details zu Google Maps finden sich im Kapitel 6.4. Folgendes Beispiel zeigt, wie eine Karte eingebunden werden kann. Mit `Boolean useCurrentLocation: true` (Zeile 3) wird der aktuelle Standort der Benutzerin/des Benutzers auf der Karte ohne Marker zentriert angezeigt und mit `zoom` der Zoombereich festgelegt. (vgl. Sencha Touch API, 2011p)

```
1  var map = new Ext.Map({
2     title: 'Map',
3     useCurrentLocation: true,
4     mapOptions: {
5       zoom: 16
6     }
7  });
```

7.5 Offline Applikationen

Wie in Kapitel 6.5 des jQuery Frameworks beschrieben kann auch für mit Sencha Touch erstellte mobile Web Applikationen eine Cache Manifest Datei für die offline Speicherung der Applikation zur Verfügung gestellt werden.

7.6 Lokale Datenbank

Die W3C Spezifikation Web SQL Database, in Kapitel 6.6 bereits erläutert, kann auch in mit Sencha Touch erstellten mobilen Web Applikationen angewendet werden.

7.7 Kamera

Einen möglichen Zugriff auf die Kamera beschreiben wir in Kapitel 6.7.

7.8 Test

Der Test von mit Sencha Touch erstellten mobilen Web Applikationen läuft, wie in Kapitel 6.8 des jQuery Mobile Frameworks beschrieben ab.

7.9 Distribution

Für die Distribution von mit Sencha Touch erstellten mobilen Web Applikationen gilt das Gleiche, wie in Kapitel 6.9 des jQuery Mobile Frameworks beschrieben. Das Icon für den Home-Bildschirm sowie die iOS spezifischen Konfigurationen für eine nativ wirkende Applikation werden im JavaScript Quellcode (Kapitel 7.2.1) angegeben.

8 PhoneGap

PhoneGap ist ein Open Source (MIT Lizenz) Framework der Firma Nitobi Software, das die Entwicklung von hybriden Applikationen mit HTML, CSS und JavaScript für die Plattformen iOS, Android, BlackBerry, Symbian, Palm webOS und Windows Mobile plattformübergreifend ermöglicht. Das fertige Endprodukt kann über verschiedene Stores, wie Android Market, Apple App Store, BlackBerry App World oder Nokia Ovi Store vertrieben werden. (vgl. PhoneGap, 2011a)

> *PhoneGap applications are native applications that open a full-screen embedded browser with our mobile web code running inside. This framework provides a bridge between JavaScript and the native runtime, providing support for additional features not available in JavaScript. (Firtman, 2010, S. 401)*

Mit PhoneGap realisierte Applikationen sind demnach Web Applikationen, die innerhalb eines nativen Webbrowser Fensters laufen, um auf die native API einer Plattform zugreifen und verschiedene native Software- und Hardwarefunktionen verwenden zu können. (vgl. Spiering & Haiges, 2010, S. 239) PhoneGap verwendet als nativen Webbrowser die `UIWebView` Komponente des Objective-C SDKs für die iOS Plattform, die `WebView` Komponente des Java SDKs für die Android Plattform und die `BrowserField` Komponente des WebWorks SDKs für die BlackBerry Plattform, um von JavaScript auf native Funktionen und umgekehrt zugreifen zu können (vgl. Morrow, 2010; BlackBerry Developers, 2011c). Bei der Symbian Plattform greift PhoneGap auf das WRT Widget zurück, da dieses auf Web Technologien und nicht auf nativen Quellcode aufbaut. (vgl. Johnson, 2010).

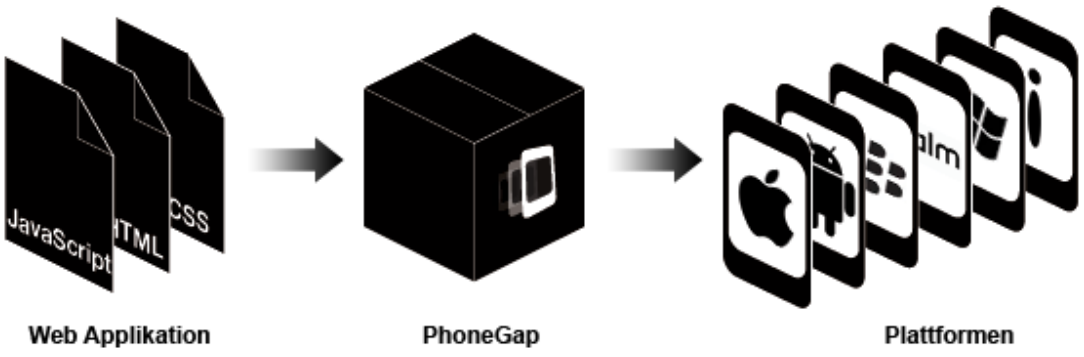


Abbildung 38: PhoneGap Architektur (vgl. PhoneGap, 2011a)

PhoneGaps Architektur (vgl. Abbildung 38) bietet also eine Brücke zwischen Web Technologien und nativen Funktionen, indem PhoneGap eine JavaScript API zur Verfügung stellt, um auf Smartphone-spezifische Funktionen (Beschleunigungssensor, Kamera, Kompass, Kontakte, Dateisystem, Geolocation, Audio, Benachrichtigung, lokale Datenbank), die derzeit noch nicht von mobilen Webbrowsern unterstützt werden, zugreifen zu können. PhoneGap unterstützt jedoch nicht alle Plattformen mit dem vollen Funktionsumfang. (vgl. PhoneGap, 2011d; Spiering & Haiges, 2010, S. 240) PhoneGap ermöglicht über das globale Objekt `navigator` den Zugriff auf verschiedene native Hardware- und Softwarekomponenten. Beispielsweise können die Entwicklerinnen/Entwickler auf die Kamera mit `navigator.camera.getPicture()` oder auf die Kontakte mit `navigator.service.contacts()` zugreifen. (vgl. Pearce, 2011) Vorteilhaft ist, dass PhoneGap die JavaScript API abstrahiert und somit der gleiche JavaScript Quellcode für viele der unterstützten Plattformen, ohne oder mit geringen Veränderungen, verwendet werden kann. (vgl. Spiering & Haiges, 240)

8.1 Installation und Entwicklungsumgebung

Das aktuelle PhoneGap Framework kann von PhoneGap (2011b) heruntergeladen werden und beinhaltet Projektvorlagen für die verschiedenen Plattformen. Jede Plattform verfügt über eine eigene PhoneGap JavaScript Datei, die sich im jeweiligen Ordner des heruntergeladenen Frameworks befindet. In jeder mobilen Applikation, die auf PhoneGap aufbaut, kann das Event `deviceready` eingebaut werden. Das Event wird ausgelöst, wenn das PhoneGap Framework vollständig geladen wurde und ermöglich somit den sicheren Zugriff auf die Funktionen der PhoneGap JavaScript API. (vgl. PhoneGap Documentation, 2011d) PhoneGap benötigt für die Kompilierung der mobilen Applikationen die entsprechenden SDKs der Plattformen, an die unterschiedliche Anforderungen geknüpft sind.

8.1.1 Symbian

PhoneGap greift für die Erstellung einer Symbian Applikation auf WRT zurück. Eine Symbian WRT Vorlage mit den notwendigen Dateien befindet sich im Ordner *Symbian/framework/www* des heruntergeladenen PhoneGap Frameworks (vgl. PhoneGap, 2011b). Diesem Ordner können weitere Dateien, die für die Entwicklung notwendig sind, hinzugefügt werden. Die im Ordner *www* befindliche *index.html* Datei kann in jeder beliebigen IDE oder in der Aptana Studio Entwicklungsumgebung (vgl. Aptana Studie) mit dem installierten Nokia WRT Plug-In mit den gewünschten PhoneGap API Funktionen versehen werden. (vgl. PhoneGap, 2011f) PhoneGap

arbeitet auch bereits an der Unterstützung von Symbian Qt Applikationen (vgl. Van Beelen, 2011).

8.1.2 Android

Für die Entwicklung von Android Applikationen benötigt das PhoneGap Framework die Eclipse Classic Entwicklungsumgebung (vgl. Eclipse, 2011), das Android SDK (vgl. Android Developers, 2011c) sowie das ADT (Android Developer Toolkit) Plug-In für Eclipse (vgl. Android Developers, 2011d). Die gesamte benötige Software steht für Windows, Mac OS X und Linux zur Verfügung. In Eclipse kann mit der Auswahl *File → New → Project* ein neues Android Projekt angelegt werden, in welchem ein *Project name*, *Application name*, *Package name* beginnend mit *com.phonegap.* und *Create Activity* Name vergeben sowie die gewünschte SDK Version ausgewählt werden muss. Im Root dieses Projektes müssen zwei Ordner *libs* und *assets/www* angelegt werden. Das heruntergeladene PhoneGap Framework (vgl. PhoneGap, 2011b) beinhaltet den Ordner Android, in welchem die *phonegap.jar* Datei in den Ordner *libs* und die *phongap.js* Datei in den Ordner *assets/www* kopiert werden muss. Im Android Projekt müssen noch ein paar Änderungen vorgenommen werden, die auf PhoneGap (2011e) genau beschrieben werden. Anschließend kann unter *assets/www* eine *index.html* mit den gewünschten PhoneGap API Aufrufen angelegt werden. (vgl. PhoneGap, 2011e)

8.1.3 BlackBerry

Wenn Applikationen mit dem PhoneGap Framework für die BlackBerry Plattform entwickelt werden möchten, ist Windows XP oder Windows 7 notwendig. Das PhoneGap Framework baut für die Erstellung von BlackBerry Applikationen auf BlackBerry WebWorks auf (vgl. BlackBerry Developers, 2011d). Eine BlackBerry WebWorks Applikation kann mit dem Kommandozeilentool Apache Ant (vgl. Apache Ant, 2011) oder mit der Eclipse Entwicklungsumgebung (vgl. Eclipse, 2011) erstellt werden. Für die Erstellung mit Apache Ant muss zunächst das Sun Java Development Toolkit 32-bit (vgl. Oracle, 2011) und Apache Ant (vgl. Apache Ant, 2011) installiert und in die *PATH* Systemvariable auf dem Windows PC hinzugefügt werden. Weiters wird das BlackBerry Widget SDK (vgl. BlackBerry Developers, 2011i) sowie das PhoneGap Framework benötigt (vgl. PhoneGap, 2011b).

Anschließend kann im Kommandozeilen Fenster direkt im Ordner *BlackBerry-WebWorks*, der sich im heruntergeladenen PhoneGap Framework befindet, mit dem Befehl `ant create -Dproject.path=C:\Dev\phonegap\BlackBerry-`

`WebWorks\sample` ein neues PhoneGap BlackBerry WebWorks Projekt erstellt werden. In der Datei *project.properties* kann der Pfad zum BlackBerry WebWorks SDK unter *bbwp.dir*= geändert werden sowie der Pfad zu einem eigens heruntergeladenen BlackBerry Simulator. In der Datei *config.xml* ist darauf zu achten, dass die Zeile `<access subdomains="true" uri="*" />` eingetragen ist, damit Zugriffe auf externe URLs (z.B. Google Maps API) möglich sind. Im Ordner *www*, der sich im neu erstellten *sample* Projekt befindet, kann die *index.html* Datei nach den gewünschten Anforderungen angepasst und weitere notwendige Dateien hinzugefügt werden. Durch Eingabe von `ant build` im Kommandozeilen Fenster im Projektordner kann eine BlackBerry WebWorks Applikation erstellt werden. BlackBerry WebWorks Dateinamen dürfen keine Bindestriche enthalten, da ansonsten die Erstellung der Applikation fehlschlägt. (vg. PhoneGap, 2011g)

Für ein übersichtlicheres Arbeiten kann zusätzlich die Eclipse Classic Entwicklungsumgebung 32-bit (vgl. Eclipse, 2011) installiert werden. Hierfür wird das Sun Java Development Toolkit 32-bit (vgl. Oracle, 2011) und das BlackBerry Web und WebWorks SDK Plug-In für Eclipse benötigt. Anschließend kann in Eclipse mit der Auswahl *File → New → Project* ein neues BlackBerry WebWorks Projekt angelegt werden und der Ordner *www*, aus dem vorhin mit Apache Ant erstellten *sample* Projekt, importiert werden. Über *Project → Build BlackBerry WebWorks Project* wird eine BlackBerry WebWorks Applikation erstellt. (vgl. Tyberg, 2011)

8.1.4 iOS

Wenn Applikationen mit PhoneGap für die iOS Plattform entwickelt werden möchten, ist ein Intel-basierter Computer mit Mac OS X Snow Leopard (10.6) und die Installation der Xcode Entwicklungsumgebung mit der iOS SDK (vgl. Apple Developer, 2011a) Voraussetzung. Zum Herunterladen von Xcode mit der iOS SDK ist eine Apple ID notwendig, die kostenlos beantragt werden kann oder aufgrund der Verwendung von iTunes bereits vorhanden ist. Weiters muss das Framework von PhoneGap (vgl. PhoneGap, 2011b) heruntergeladen und die Datei *PhoneGapLibInstaller.pkg,* die sich im Ordner *iOS* befindet, installiert werden. Dadurch wird ein Ordner *PhoneGapLib* im Verzeichnis *Dokumente* und eine PhoneGap Xcode Vorlage in Xcode erstellt, um ein PhoneGap Projekt in der Xcode Entwicklungsumgebung ausführen zu können. Nachdem Xcode gestartet wurde, kann unter *File → New Project* ein neues PhoneGap Projekt angelegt werden. PhoneGap legt automatisch eine *index.html* Datei im Ordner *www* an, in welcher die PhoneGap JavaScript Datei im `<head>` Bereich inkludiert ist. (vgl. PhoneGap, 2011c)

8.2 User Interface

Das PhoneGap Framework verfügt über keine UI API. Es können jedoch beispielsweise jQuery Mobile oder Sencha Touch für ein einheitliches plattformübergreifendes UI herangezogen werden.

8.3 Geolocation

Die Geolocation API von PhoneGap basiert, wie in Kapitel 6.3 beschrieben auf der W3C Geolocation API Spezifikation. Plattformen, wie iOS, Android und BlackBerry die diese Spezifikation in ihrer nativen Webbrowser Komponente bereits implementiert haben, greifen nicht auf die PhoneGap Geolocation API zurück. (vgl. PhoneGap Documentation, 2011a) In Symbian WRT ist der Zugriff auf Geolocation ebenfalls möglich (vgl. PhoneGap, 2011d)

8.4 Karte

PhoneGap ermöglicht die Einbindung einer Google Maps Karte auf die gleiche Weise wie in Kapitel 6.4 erklärt.

8.5 Offline Applikationen

Alle Dateien einer PhoneGap Applikation stehen nach der Installation auf dem Smartphone offline zur Verfügung, weshalb keine Cache Manifest Datei, wie in Kapitel 6.5 beschrieben benötigt wird.

8.6 Lokale Datenbank

Die API von PhoneGap basiert bei der Speicherung von Daten in eine lokale Dankenbank auf der W3C Web SQL Database Spezifikation. Diese erläutern wir im Kapitel 6.6 genauer. iOS, Android und BlackBerry unterstützen diese Spezifikation. (vgl. PhoneGap Documentation, 2011b) In Symbian WRT ist keine Web SQL Database implementiert (vgl. PhoneGap, 2011d).

8.7 Kamera

> *The* `camera` *object provides access to the device's default camera application (PhoneGap Documentation, 2011c).*

Der Zugriff auf die eingebaute Kamera wird von den Plattformen Android, iOS, BlackBerry und Symbian unterstützt (vgl. PhoneGap, 2010d).

```
1  navigator.camera.getPicture(cameraSuccess, cameraError,
   {quality:20});
2  function cameraSuccess(imageData) {
3     var image = document.getElementById("addImage");
4     image.src = "" + imageData;
5  }
6  function cameraError(message) {
7     alert("Failed because: " + message);
8  }
```

Mit der Funktion `camera.getPicture()` in Zeile 1 wird standardmäßig die Kamera Applikation eines Smartphones aufgerufen, die drei Parameter enthält. Nachdem die Benutzerin/der Benutzer ein Foto aufgenommen hat, schließt sich die Kamera Applikation automatisch und kehrt zur ursprünglichen Applikation zurück. War der Kameraaufruf erfolgreich, wird die Funktion `cameraSuccess()` (Zeile 2) aufgerufen, in welcher die Bilddaten in der Variable `imageData` vorhanden sind. Standardmäßig wird das Bild als Base64 codierter String retourniert, also als normale Zeichenfolge, die beispielsweise in einer lokalen Web SQL Datenbank gespeichert oder direkt in der Applikation in einem `img` Element ausgegeben werden kann. Dem `src` Parameter des `img` Elements in Zeile 4 wird das Datenformat `data:image/jpeg;base64` sowie der Bilddaten String `imageData` übergeben. Die Funktion `cameraError()` in Zeile 6 wird bei einem nicht erfolgreichen Aufruf der Kamera API aufgerufen. Der dritte Parameter kann verschiede Optionen enthalten, unter anderem die Qualität des gespeicherten Bildes zwischen 0 und 100 mit `quality:20`. Mit `destinationType: Camera.DestinationType` kann das Format des Rückgabewerts definiert werden. `Camera.DestinationType.DATA_URL` gibt das Bild als base64 codierten String zurück. `Camera.DestinationType.FILE_URI` gibt den Pfad der Bilddatei retour. Optional kann eine weitere Variable `sourceType = Camera.PictureSourceType.PHOTOLIBRARY` für die Auswahl eines Bildes aus der Bibliothek angegeben werden. (vgl. PhoneGap Documentation, 2011c; Spiering & Haiges, 2010, S. 307 - 308)

8.8 Test

Das Testen von mit PhoneGap realisierten mobilen Applikationen ist auf Simulatoren/Emulatoren oder direkt auf den Smartphones möglich. Einige Software- und Hardwarefunktionen, z.B. der Zugriff auf die Kamera, kann nicht auf allen Simulatoren/Emulatoren, sondern nur direkt auf den Smartphones getestet werden.

8.8.1 Symbian

Um eine WRT Applikation zu erstellen, können aus den im *www* Ordner befindlichen Dateien eine *zip* Datei erstellt werden, die anschließend in *wgz* umbenannt wird. *wgz* stellt die Dateiendung von Symbian WRT Applikationen dar. Weiters besteht die Möglichkeit in der Aptana Studio IDE eine Symbian WRT Applikation zu erstellten, die im mitgelieferten Emulator des WRT Plug-In getestet werden kann. Die erstellte *wgz* Datei kann auch per E-Mail oder Bluetooth direkt an ein echtes Symbian Smartphone gesendet und zum Testen installiert werden. (vgl. PhoneGap, 2011f)

8.8.2 Android

Mit PhoneGap erstellte Android Applikationen können entweder im Emulator oder direkt auf einem Android Smartphone getestet werden. Für das Testen auf einem Emulator muss zunächst ein neuer Android Emulator in Eclipse unter *Window → Android SDK and AVD Manager* angelegt werden. Anschließend kann die Applikation mit *Run → Run As → Android Application* auf dem Android Emulator installiert werden. Zum Testen von beispielsweise der Kamera auf einem Android Smartphones muss zunächst Debugging auf dem Smartphone unter *Einstellungen → Anwendungen → Entwicklung* aktiviert werden. Anschließend kann das Android Smartphone mittels USB an den PC angeschlossen werden und die Applikation über Eclipse direkt auf dem Smartphone installiert und getestet werden. (vgl. PhoneGap, 2011e)

8.8.3 BlackBerry

Der Aufruf von `ant load-simulator` im Kommandozeilen Fenster direkt im Pfad des Projektordners ermöglich das Ausführen der Applikation in einem BlackBerry Simulator. In Eclipse kann die Applikation über *Run → Run As → BlackBerry Simulator* im Simulator aufgerufen werden. Bevor eine BlackBerry WebWorks Applikation direkt auf dem BlackBerry Smartphone ausgeführt werden kann, muss die Applikation signiert werden (vgl. BlackBerry Developers, 2011f). Anschließend kann das BlackBerry Smartphone über USB an den PC angeschlossen werden und das `ant load-device` Kommando ermöglich das Ausführen der Applikation direkt auf dem BlackBerry Smartphone. (vgl. PhoneGap, 2011g; Tyberg, 2011**)**

8.8.4 iOS

Eine mit PhoneGap erstellte iOS Applikation kann auf dem iPhone Simulator getestet werden, indem in der Xcode Entwicklungsumgebung aus dem linken oberen Menü der Simulator mit der gewünschten iOS Version ausgewählt wird (vgl. PhoneGap, 2011c).

Zum Testen der Applikation auf einem echten iPhone Smartphone ist eine Anmeldung beim iOS Developer Program von Apple notwendig, um ein sogenanntes Development Certificate zu erhalten. Es kann zwischen zwei iOS Developer Programs unterschieden werden, das normale iOS Developer Program (vgl. Apple Developer, 2011b) ist für Einzelpersonen und kostet 99 Dollar pro Jahr, das iOS Developer Enterprise Program (vgl. Apple Developer, 2011c) ist für Firmen und kostet 299 Dollar pro Jahr.

Bevor das Development Certificate von Apple angefragt werden kann, muss über das Mac OS X Programm Schlüsselbundverwaltung eine Certificate Signing Request Datei erstellt werden. In der Schlüsselbundverwaltung wird *Zertifikatsassistent → Ein Zertifikat einer Zertifizierungsinstanz anfordern* ausgewählt, die gewünschten Daten eingegeben und auf der Festplatte gesichert. Anschließend kann im iOS Developer Center (vgl. Apple Developer, 2011a) mit Hilfe eines Assistenten der Prozess zum Erlangen des Developer Certificate durchlaufen werden. Zunächst muss eine App ID für die eindeutige Identifizierung der mobilen Applikation angegeben werden. Danach ist die eindeutige ID des iPhones – auf welchem getestet wird – einzugeben. Anschließend kann die vorhin erstellte Certificate Signing Request Datei an Apple übermittel werden. Im nächsten Schritt wird ein Provisioning Profile erstellt, das heruntergeladen und über den Organizer im Xcode auf dem iPhone installiert werden kann. Zurück im Assistenten kann im weiteren Schritte das Development Certificate heruntergeladen und in der Schlüsselbundverwaltung installiert werden. Im letzten Schritt des Assistenten muss die erfolgreiche Installation des Zertifikats bestätigt werden. (vgl. Wei-Meng, 2009)

In der Xcode Entwicklungsumgebung kann nun ein PhoneGap Projekt geöffnet werden und in der Datei mit der Endung *.plist* muss unter *BundleIdentifier* die vorher im Assistenten angegebene App ID eingegeben werden. Anschließend kann im aus dem linken oberen Menü in der Xcode Entwicklungsumgebung *Device* ausgewählt werden, um die Applikation direkt auf dem iPhone installieren und testen zu können. (vgl. PhoneGap, 2011c)

8.9 Distribution

Die mit dem Framework PhoneGap erstellten mobilen Applikationen können über den jeweiligen Application Store der Plattformen Symbian, Android, BlackBerry und iOS vertrieben werden. In Kapitel 3 erläutern wir die einzelnen Stores sowie die damit verbunden Anforderungen für den Vertrieb von Applikationen näher.

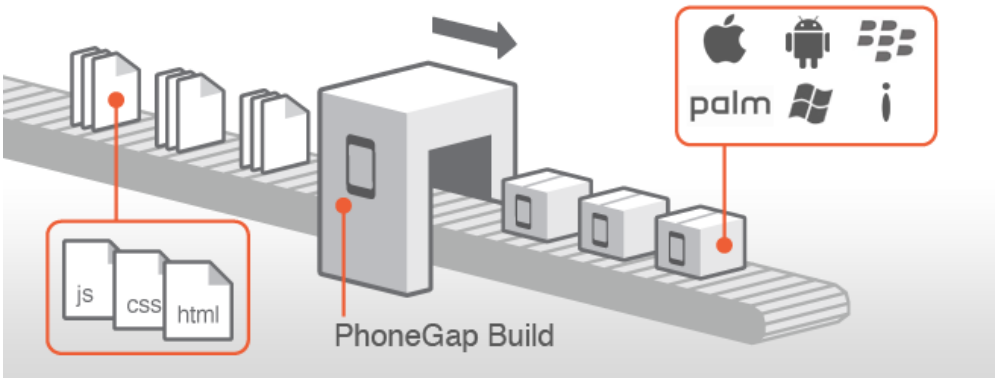


Abbildung 39: PhoneGap Build (vgl. PhoneGap, 2010b)

Das PhoneGap Build Service (vgl. Abbildung 39) befindet sich derzeit im Beta Stadium und übernimmt für die Entwicklerinnen/Entwickler die Kompilierung von Applikationen für Symbian, Android, BlackBerry, iOS und Palm Web OS. Die Entwicklerinnen/Entwickler können alle HTML, CSS und JavaScript Dateien der mobilen Applikation sowie die notwendigen Signaturen beim PhoneGap Build Service zum Kompilieren hochladen. Anschließend erstellt das PhoneGap Build Service eine vertriebsbereite Applikationen mit der jeweiligen PhoneGap JavaScript Datei für alle Plattformen und kann von den Entwicklerinnen/Entwicklern heruntergeladen werden. (vgl. PhoneGap Build, 2011)

9 Titanium Mobile

Titanium Mobile ist ein Open Source Framework des Unternehmens Appcelerator für eine plattformübergreifende Entwicklung hybrider iOS und Android Applikationen. An der Unterstützung von BlackBerry wird derzeit gearbeitet. Titanium Mobile setzt bei der Erstellung von Applikationen ganzheitlich auf die Programmiersprache JavaScript. Im Gegensatz zu PhoneGap baut Titanium Mobile nicht auf native Webbrowser Komponenten auf, sondern das erstellte JavaScript Gerüst wird in nativen Quellcode übersetzt und mit den darunterliegenden nativen iOS und Android SDKs in eine vertriebsbereite Applikation kompiliert. Titanium Mobile stellt jedoch auch eine `WebView` Komponente zur Verfügung, in welcher, wie in PhoneGap HTML Inhalte angezeigt werden können.

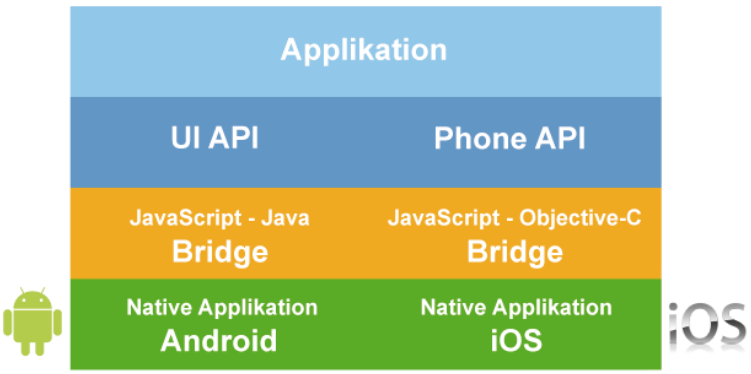


Abbildung 40: Titanium Mobile Architektur (vgl. Appcelerator Wiki, 2011a)

Die Titanium Mobile Architektur (vgl. Abbildung 40) baut auf dem Titanium Mobile SDK auf, das über eine plattformunabhängige JavaScript API verfügt, die auf native UI Komponenten und Software-/Hardwarefunktionen (Beschleunigungssensor, Kamera, Kompass, Kontakte, Dateisystem, Geolocation, Audio, Video, Benachrichtigung, lokale Datenbank) der jeweiligen Plattform SDK zurückgreift. Ein Teil der UI Elemente steht für alle unterstützten Plattformen zur Verfügung und zusätzlich kann auf systemspezifische iOS und Android Module zugegriffen werden. Außerdem verwendet Titanium Mobile für den Zugriff auf Hardware- und Softwarefunktionen eines Smartphones die native API der Plattformen. Weiters enthält das Titanium Mobile Framework eine Entwicklungsumbebung namens Titanium Developer. Die IDE ermöglicht die Erstellung von Projekte und Kompilierung zu mobilen Applikationen. Diese können entweder im iOS Simulator/Android Emulator oder direkt auf physischen Geräten getestet werden. Titanium Developer übernimmt auch die finale Paketierung

der Applikationen zum Vertrieb über Apples App Store und Android Market. Titanium Developer stellt allerdings keinen Quellcodeeditor für die Entwicklung zur Verfügung, sonder gibt nur Log Nachrichten aus. (vgl. Appcelerator Wiki, 2011a; Aurelio, 2010, S. 118)

9.1 Installation und Entwicklungsumgebung

Das Framework für Titanium Mobile kann unter Appcelerator (2011a) heruntergeladen werden und beinhaltet die Installationsdatei für die Titanium Developer IDE. Titanium Developer steht für Mac OS X, Windows und Linux zur Verfügung. iOS Applikationen können allerdings nur auf Mac OS X erstellt werden. Wenn Titanium Developer zum Ersten mal gestartet wird, werden automatisch die aktuellsten Titanium SDKs heruntergeladen und extrahiert. Für die Entwicklung von mobilen Applikationen werden zusätzlich die nativen SDKs von iOS (vgl. Apple Developer, 2011a) und Android (vgl. Android Developers, 2011c) benötigt.

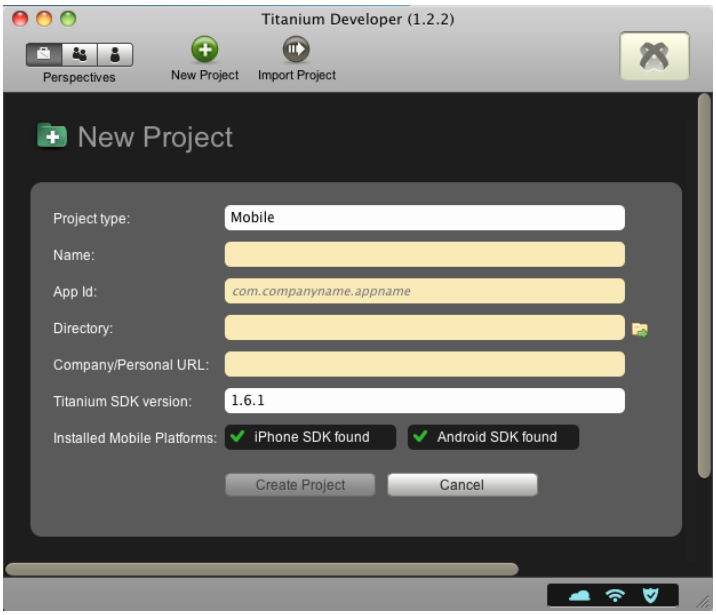


Abbildung 41: Titanium Developer

Wenn ein neues Projekt angelegt wird (vgl. Abbildung 41), erkennt Titanium Developer die SDKs von iOS und Android entweder automatisch oder es muss der Pfad zu den SDKs angeführt werden. Die gelb markierten Eingebefelder sind Pflichtfelder und können nachträglich geändert werden. Nachdem das Projekt erstellt wurde, wird von Titanium Developer eine Beispielapplikation im angegebenen Projektverzeichnis angelegt. Das Verzeichnis beinhaltet einen *build* Ordner, in dem alle notwendigen Dateien von Titanium automatisch angelegt werden, damit eine Applikation auf den

Zielplattformen lauffähig ist. Der *Resources* Ordner beinhaltet alle Dateien (JavaScript, HTML, Bilder etc.) der entwickelten Applikation. Die *tiapp.xml* Datei enthält Informationen über die Applikation. Weiters kann die Beispielapplikation *Kitchen Sink* als Projekt angelegt werden. Diese enthält zahlreiche Beispiele zu den verschiedenen UI Modulen und software-/hardwarespezifischen Funktionen. (vgl. Appcelerator Wiki, 2011a)

9.2 User Interface

Die UI Elemente von Titanium Mobile bauen auf native Komponenten auf und stützen sich nicht auf Webbrowser Bedienelemente. Nachdem die Applikation kompiliert wurde, werden die nativen UI Elemente aus den zugrunde liegenden iOS und Android SDKs herangezogen.

9.2.1 Seitenstruktur

Innerhalb des *Resources* Ordners im Projektverzeichnis befindet sich die *app.js* JavaScript Datei, die als Ausgangspunkt einer Applikation dient. Die Entwicklung kann entweder nur in dieser einen Datei stattfinden oder für eine bessere Übersichtlichkeit auf mehrere JavaScript Dateien aufgeteilt werden. Plattformspezifische Ressourcen befinden sich in den von Titanium angelegten Ordnern *android* und *iphone*. Diese beinhalten beispielsweise das Icon und Startbild einer Applikation für die jeweilige Plattform. Alle in diesen Ordnern befindlichen Dateien ersetzen gleichnamige Dateien innerhalb des *Resources* Ordners. (vgl. Appcelerator Wiki, 2011b)

Die Titanium Mobile API ist in verschiedene Module, wie `Titanium.UI` oder `Titanium.Geolocation` aufgeteilt. Im JavaScript Quellcode können die verschiedenen Module mit `Titanium` oder als Kurzform mit `Ti` angesprochen werden. Die drei Hauptkomponenten des Titanium Mobile UI sind Windows, Views und Widgets. Ein Window (`Titanium.UI.Window`) stellt das Basiselement einer Titanium Mobile Applikation dar und beherbergt alle anderen Elemente, wie beispielsweise Views (`Titanium.UI.View`). Jede Applikation muss zumindest ein Window vorweisen und kann im Gegensatz zu Views eigenständig existieren. Grundsätzlich ist eine View ein Rechteck, in welchem verschiedene Elemente und Inhalte ausgegeben werden können. Views sind hierarchisch aufgebaut, das heißt mehrere Views können übereinander gestapelt werden. Im Gegensatz zu Windows können Views nicht eigenständig existieren, sie müssen beispielsweise in ein Window eingebettet werden. Widgets sind spezielle Typen von Views, beispielsweise Schaltflächen, die spezifische

Aktionen ausführen und keine Kindelemente haben. (vgl. Appcelerator Wiki, 2011c; Tamas, 2010a)

Folgendes Beispiel zeigt die Erstellung eines Windows `Ti.UI.createWindow()`, einer Views `Ti.UI.createView()` und eines Widgets `Ti.UI.createButton()`. Das Quellcodebeispiel wird für die Android Plattform in Abbildung 42 und für die iOS Plattform in Abbildung 43 veranschaulicht.

```
1  var win = Ti.UI.createWindow();
2  var view = Ti.UI.createView({backgroundColor:"#ccc"});
3  var button = Ti.UI.createButton({
4    title:"Button",
5    width:80,
6    height:40
7  });
8  win.add(view);
9  win.add(button);
10 win.open();
```

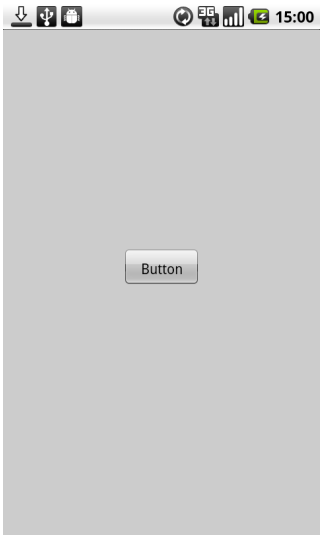


Abbildung 42: Titanium Mobile Seitenstruktur Android

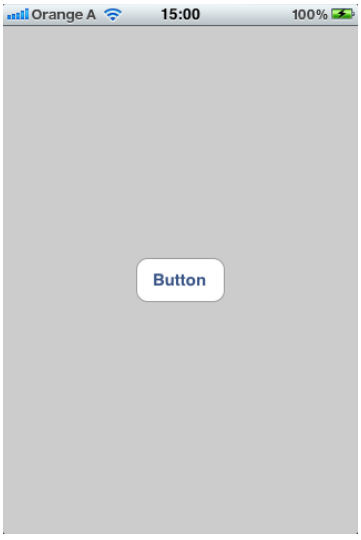


Abbildung 43: Titanium Mobile Seitenstruktur iOS

9.2.2 Animation

Titanium Mobile ermöglicht verschiedenste Animationen von Windows, Views und Widgets. Der Methode `animate()` können verschiedenste Eigenschaften, beispielsweise Veränderung der Größe, Farbe etc. betreffend der Animation übergeben werden. Weiters können in Titanium Mobile auch 2D oder 3D Transformationen vorgenommen werden. (vgl. Appcelerator Wiki, 2011d) Mit `Titanium.UI.createAlertDialog()` (vgl. Appcelerator Mobile API, 2011b) und

`Titanium.UI.createOptionDialog()` (vgl. Appcelerator Mobile API, 2011c) können verschiedene native Dialoge für die Android (vgl. Abbildung 44) und iOS (vgl. Abbildung 45) Plattform erstellt werden.

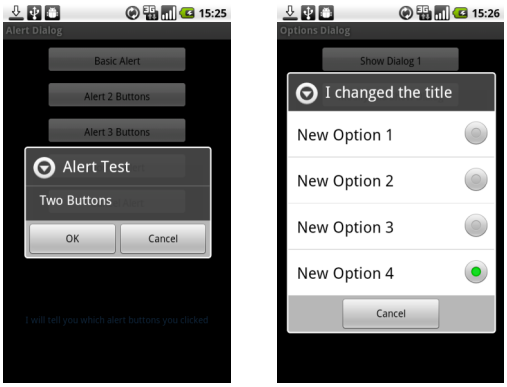


Abbildung 44: Titanium Mobile Dialoge Android

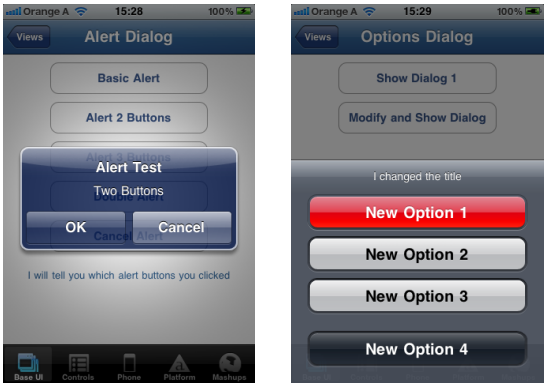


Abbildung 45: Titanium Mobile Dialoge iOS

9.2.3 Kopf- und Fußzeile

Das Hauptnavigationselement ist die sogenannte TabGroup (`Titanium.UI.TabGroup`) und die Tabs (`Titanium.UI.Tab`). Eine TabGroup kann mehrere Tabs beinhalten und jedes Tab öffnet ein anderes Window (`Titanium.UI.Window`).

```
1  var tabGroup = Titanium.UI.createTabGroup();
2  var win1 = Titanium.UI.createWindow({
3      title:'Tab 1',
4      backgroundColor:'#fff'
5  });
6  var tab1 = Titanium.UI.createTab({
7      icon:'KS_nav_views.png',
8      title:'Tab 1',
9      window:win1
10 });
11 tabGroup.addTab(tab1);
12 tabGroup.open();
```

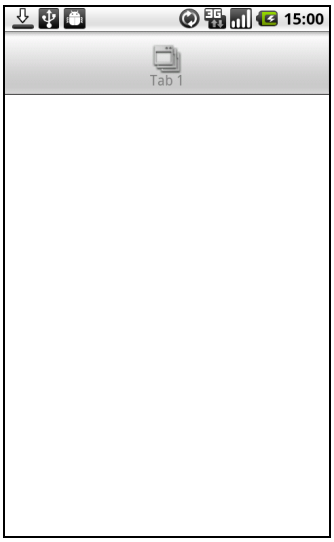


Abbildung 46: Titanium Mobile Kopfzeile Android

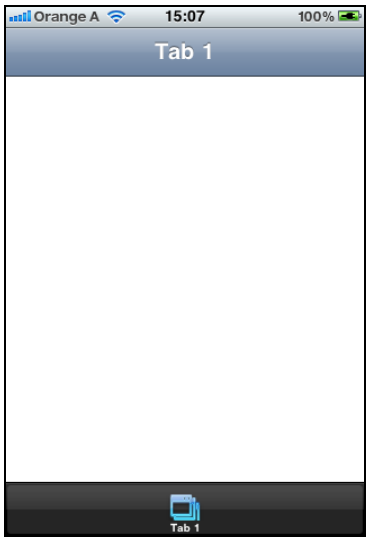


Abbildung 47: Titanium Mobile Kopf- und Fußzeile iOS

In Zeile 1 wird mit `Titanium.UI.createTabGroup()` eine TabGroup erstellt, von der die gesamte Applikation ausgeht. Sie kann in kein anderes UI Element inkludiert werden. Anschließend wird ein Window (Zeile 2) und in Zeile 6 mit `Titanium.UI.createTab()` ein Tab erstellt, das über die Eigenschaft `window` dem vorhin erstellten Window zugeordnet wird. Auf die gleiche Art und Weise können weitere Windows und Tabs kreiert werden, die alle, wie in Zeile 11 der `tabGroup` mit `addTab()` hinzugefügt und mit `tabGroup.open()` geöffnet werden. (vgl. Tamas, 2010a) Das native UI auf der iOS Plattform verfügt über eine Navigationsleiste in der Fußzeile sowie über eine Kopfzeile mit dem vergebenen Titel `title` in Zeile 3 (vgl. Abbildung 47). Androids natives UI sieht die Navigationsleiste in der Kopfzeile vor und verfügt über keine Fußzeile (vgl. Abbildung 46).

9.2.4 Schaltflächen

Schaltflächen (vgl. Abbildungen 48 und 49) können mit der Methode `Titanium.UI.createButton()` erstellt werden. (vgl. Appcelerator Mobile API, 2011e)

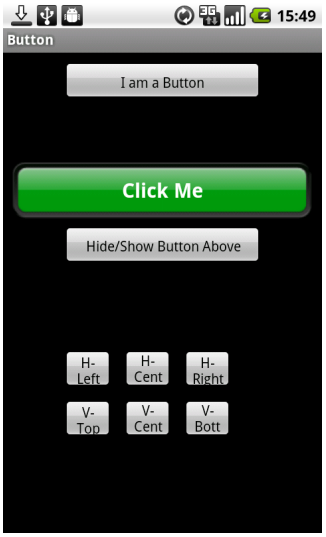


Abbildung 48: Titanium Mobile Schaltflächen Android

Abbildung 49: Titanium Mobile Schaltflächen iOS

9.2.5 Formulare

Titanium Mobile verfügt über verschiedenste native UI Elemente für Formulare. Beispielsweise stehen Schieberegler, Switcher, Schaltflächen, Eingabefelder und sogenannte Picker für die Auswahl von Daten zur Verfügung. (vgl. Appcelerator Mobile API, 2011d)

9.2.6 Listen

Die `Titanium.UI.TableView` stellt eine der bekanntesten Listenansichten dar (vgl. Abbildungen 50 und 51). Mit `Titanium.UI.createTableView()` (Zeile 4) wird eine Tabellenansicht erstellt. Über die Eigenschaft `data` (Zeile 5) können die Zeilen der Tabelle als Array hinzugefügt werden. Jede Zeile kann über einen Titel und weitere Eigenschaften, wie `hasChild` (Zeile 7), `hasDetail` (Zeile 8) oder `hasCheck` (Zeile 9) verfügen. Diese Eigenschaften fügen rechts in der Zeile ein passendes Icon hinzu. Mit der Methode `Titanium.UI.createTableViewSection()` könnte die Tabelle zusätzlich in Abschnitte unterteilt werden. (vgl. Tamas, 2010b)

```
1  var win1 = Titanium.UI.createWindow({
2     backgroundColor:"#fff"
3  });
4  var table1 =  Titanium.UI.createTableView({
5     data:[
6           {title:"Row 1 - simple row"},
7           {title:"Row 2 -  with child", hasChild:true},
8           {title:"Row 3 -  with detail", hasDetail:true},
9           {title:"Row 4 -  with check", hasCheck:true},
10    ]
11 });
12 win1.add(table1);
```

```
13 win1.open();
```

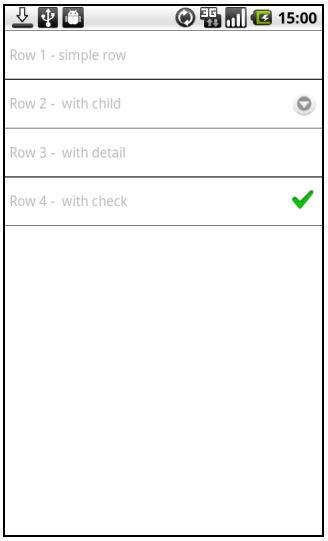


Abbildung 50: Titanium Mobile Listen Android

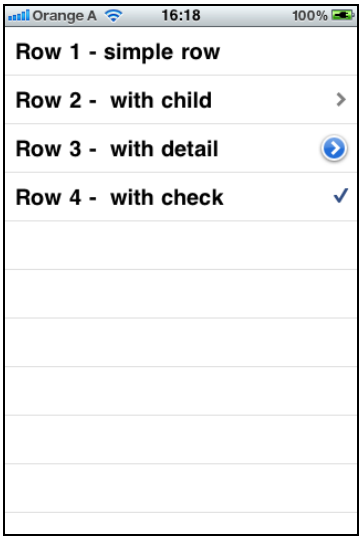


Abbildung 51: Titanium Mobile Listen iOS

9.2.7 Events

Titanium Mobile verfügt über verschiedene Events (`touchstart`, `touchmove`, `touchend`, `touchcancel`, `singletap`, `doubletap`, `twofingertap`, `swipe`, `click`, `dblclick`), die speziell auf berührungssensitive Smartphones abgestimmt sind. (vgl. Appcelerator Mobile API, 2011f)

9.3 Geolocation

Das `Titanium.Geolocation` Modul ermöglicht die Bestimmung des Standorts einer Benutzerin/eines Benutzers. Auf der iOS Version 3.2 muss zusätzlich mit der Eigenschaft `Ti.Geolocation.purpose = "GPS";` für Apple der Grund für die Verwendung von Geolocation angeführt werden. Mit der Methode `Titanium.Geolocation.getCurrentPosition()` (Zeile 1) kann die aktuelle Position eines Smartphones ermittelt werden. Die Positionsbestimmung in Titanium Mobile ist ähnlich der W3C Geolocation Spezifikation (Kapitel 6.3). War die Standortbestimmung erfolgreich wird mit `e.coords.latitude` (Zeile 6) und `e.coords.longitude` (Zeile 7) der Breiten- und Längengrad ausgelesen. Ansonsten wird mit `e.error` (Zeile 3) eine Fehlermeldung ausgegeben.

```
1  Titanium.Geolocation.getCurrentPosition(function(e) {
2    if (e.error) {
3      Ti.API.error('geo - current position' + e.error);
4      return;
```

```
5     }
6     var latitude = e.coords.latitude;
7     var longitude = e.coords.longitude;
8   });
```

Titanium Mobile stellt auch eine umgekehrte Geocodierung über die Methode `Ti.Geolocation.reverseGeocoder()` zur Verfügung. Diese funktioniert jedoch nicht im europäischen Raum. (vgl. Appcelerator Wiki, 2011e)

9.4 Karte

Eine native Karte wird über das Modul `Titanium.Map` bereit gestellt.

```
1  var mapview = Titanium.Map.createView({
2     mapType: Titanium.Map.STANDARD_TYPE,
3     region: {latitude:33.74511, longitude:-84.38993,
      latitudeDelta:0.5, longitudeDelta:0.5},
4     regionFit: true
5  });
```

Mit der Methode `Titanium.Map.createView()` (Zeile 1) wird eine neue Karte erstellt. Diese kann verschiedene Eigenschaften enthalten. Beispielsweise kann über `mapType` (Zeile 2) zwischen drei verschiedenen Kartentypen ausgewählt werden. `STANDARD_TYPE` zeigt eine Straßenkarte, `SATELLITE_TYPE` zeigt eine Karte mit Sattelitenbildern und `HYBRID_TYPE` ist eine Mischform aus `STANDARD_TYPE` und `SATELLITE_TYPE`. Der Eigenschaft `region` (Zeile 3) wird der gewünschte Breiten- und Längengrad (`latitude` und `longitude`) übergeben. `regionFit` (Zeile 4) passt die Größe der Karte an die Eigenschaft `region` an.

Ein Marker kann mit der Methode `Ti.Map.createAnnotation()` (Zeile 1) erstellt werden. Diese erhält den gewünschte Breiten- und Längengrad (Zeile 2 - 3) sowie einen Titel `title` (Zeile 4) und der Marker kann über die Eigenschaft `animate` (Zeile 5) animiert dargestellt werden. Der erstellte Marker kann innerhalb der Karte mit der Eigenschaft `annotations:[markerMap]` angezeigt werden. (vgl. Appcelerator Wiki, 2011e)

```
1  var markerMap = Ti.Map.createAnnotation({
2     latitude:lat,
3     longitude:lng,
4     title:"Titel",
5     animate:true
6  });
```

9.5 Offline Applikationen

Alle Dateien einer Titanium Mobile Applikation stehen nach der Installation auf dem Smartphone offline zur Verfügung, weshalb keine Cache Manifest Datei, wie in Kapitel 6.5 beschrieben benötigt wird.

9.6 Lokale Datenbank

Titanium Mobile ermöglicht den Zugriff auf eine SQLite Datenbank über das Objekt `Titanium.Database`, das ähnlich wie die W3C Web SQL Database (Kapitel 6.6) aufgebaut ist.

```
1  var conn = Titanium.Database.open('todos');
2  conn.execute('CREATE TABLE IF NOT EXISTS todos (id INTEGER PRIMARY
   KEY, todo TEXT)');
3  var results = [];
4  var resultSet = conn.execute('SELECT * FROM todos');
5    while (resultSet.isValidRow()) {
6      results.push({
7        notice: resultSet.fieldByName('todo')
8      });
9      resultSet.next();
10   }
11 resultSet.close();
```

Es besteht die Möglichkeit mit `Titanium.Database.install()` eine bereits bestehende SQLite Datenbank – die sich im *Ressouces* Ordner befindet – auf dem Smartphone zu installieren oder eine neue Datenbank über `Titanium.Database.open()`, wie in Zeile 1 anzulegen. In Zeile 2 wird eine neue Tabelle `todos` mit zwei Spalten `id` und `todo` angelegt. Die Methode `execute()` (Zeile 2) führt ein SQL Statement aus und liefert ein Ergebnis (`resultSet`) zurück. Um Änderungen in der Datenbank vorzunehmen, können die Statements `CREATE`, `SELECT`, `UPDATE`, `INSERT` und `DELETE` angewendet werden. Zwischen Zeile 5 und Zeile 10 wird das Ergebnis des `SELECT` Statements (Zeile 4) in das Array `results` geschrieben. Mit der Methode `fieldByName()` (Zeile 7) wird der Inhalt aus der entsprechenden Spalte der Datenbank ausgelesen und kann später über `notice` (Zeile 7) abgerufen werden. In Zeile 11 wird das `resultSet` mit `close()` geschlossen. (vgl. Whinnery, 2010; Stever, 2010)

9.7 Kamera

Das Objekt `Titanium.Media` ermöglicht den Zugriff auf verschiedene multimediale Hardwarefunktionen eines Smartphones.

```
1  var win = Titanium.UI.currentWindow;
2  Titanium.Media.showCamera({
3     success:function(event) {
4           var imageView = Ti.UI.createImageView({image:event.media});
5           win.add(imageView);
6     },
7     cancel:function() {
8                 alert('You canceled the action.');
9     },
10    error:function(error) {
11          if (error.code == Titanium.Media.NO_CAMERA) {
12                alert('Please run this test on device');
13          }
14          else {
15                alert('Unexpected error: ' + error.code);
16          }
17    },
18    saveToPhotoGallery:true,
19    allowEditing:true
20 });
```

Mit der Methode `Titanium.Media.showCamera()` in Zeile 2 wird die Kamera aktiviert, um Fotos aufnehmen zu können. Diese Methode verfügt über drei Ereignisse, `success`, `cancel` und `error`. War die Fotoaufnahme erfolgreich, wird die Funktion `success` aufgerufen und das Bild in einer View `Ti.UI.createImageView()` (Zeile 4) angezeigt. Wenn die Benutzerin/der Benutzer die Schließen Schaltfläche innerhalb der Kamera Applikation betätigt hat, wir die Funktion `cancel` (Zeile 7) aufgerufen. Bei einem aufgetretenen Fehler, wird die Funktion `error` (Zeile 10) aufgerufen. Weiters können noch zusätzliche Eigenschaften, wie `saveToPhotoGallery` (Zeile 18) zum Speichern des Fotos in der Bibliothek oder `allowEditing` (Zeile 19) zum nachträglichen Editieren des Fotos, vergeben werden. (vgl. Dan, 2010c; Appcelerator Mobile API, 2011a)

9.8 Test

Die Entwicklungsumgebung Titanium Developer ermöglicht das Testen von mobilen Applikationen auf dem iOS Simulator/Android Emulator oder direkt auf den Smartphones. In Abbildung 52 ist zu erkennen, dass die IDE zum Testen einer Applikation über zwei Ansichten verfügt. Die Ansicht *Run Emulator* dient zur Installation der Applikation auf den Simulator/Emulator. Die Entwicklerinnen/Entwickler müssen

keinen Android Emulator anlegen, da Titanium Developer dies selbständig macht. Die Ansicht *Run on Device* ermöglicht die Installation der Applikation direkt auf dem Smartphone. Wie bereits in Kapitel 8.1.4 und 8.8.4 näher erläutert, ist für eine Installation auf dem iPhone eine Anmeldung beim iOS Developer Program von Apple notwendig. Wenn ein Android Smartphone über USB an den PC angeschlossen ist, kann eine Titanium Mobile Applikation auf diesem installiert und getestet werden. (vgl. Appcelerator Wiki, 2011a)

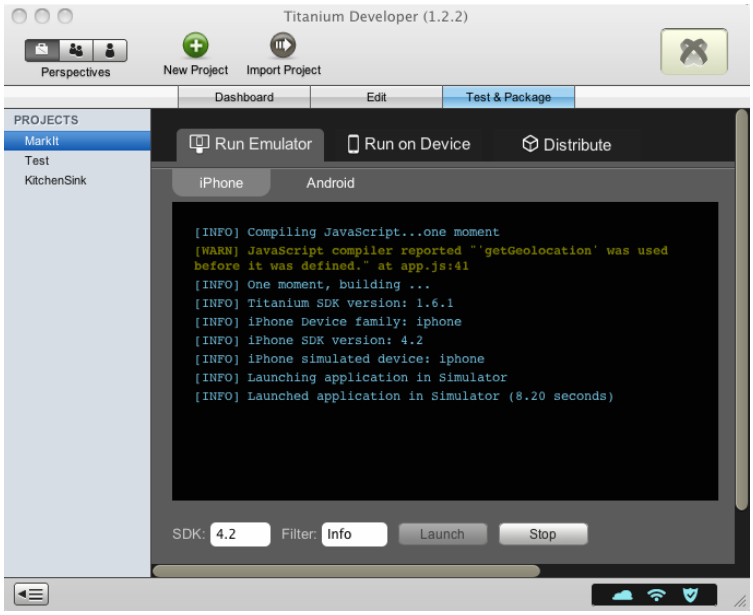


Abbildung 52: Titanium Developer Emulator

9.9 Distribution

Die mit dem Framework Titanium Mobile erstellten mobilen Applikationen können über den jeweiligen Application Store der Plattformen Android und iOS vertrieben werden. In Kapitel 3 erläutern wir die einzelnen Stores sowie die damit verbunden Anforderungen für den Vertrieb von Applikationen näher. In Abbildung 52 ist neben der Testansicht der Punkt *Distribute* zu sehen. In dieser Ansicht kümmert sich die Titanium Developer IDE um die vertriebsbereite Paketierung der Applikation.

10 Praktischer Test mit einer Beispielapplikation

Für den systematischen Vergleich der vier plattformübergreifenden Frameworks jQuery Mobile, Sencha Touch, PhoneGap und Titanium Mobile setzen wir eine Beispielapplikation, unter Berücksichtigung der beschriebenen Kriterien (Installation und Entwicklungsumgebung, User Interface, Geolocation, Karte, offline Applikationen, lokale Datenbank, Kamera, Test, Distribution) in allen Frameworks um und testen diese auf den Plattformen Symbian, Android, BlackBerry und iOS. Die mobilen Web Applikationen, umgesetzt mit jQuery Mobile und Sencha Touch, werden in den jeweiligen Standard Webbrowsern der Plattformen getestet. Der systematische Vergleich der Frameworks führt noch zu keiner endgültigen Gesamtbewertung, da dieser auch von unterschiedlichen Anforderungen seitens der Entwicklerinnen/Entwickler abhängig ist. Die endgültige Bewertung findet unter Berücksichtigung der Ergebnisse aus der Umsetzung der Beispielapplikation in Kapitel 11 in Zusammenhang mit drei verschiedenen Szenarien statt.

Die Beispielapplikation namens *Mark It* ermöglicht mit Geolocation die Anzeige des aktuellen Standorts einer Benutzerin/eines Benutzers als Adresse auf einer Google Karte. Somit können Standorte, die für die Benutzerin/den Benutzer interessant sind markiert werden. Der Standort kann mit einem Titel und einer Beschreibung sowie mit einem Foto, aufgenommen über die integrierte Smartphone Kamera, versehen werden. Die Standorte werden clientseitig in eine Datenbank gespeichert, können angezeigt, geändert und gelöscht werden und stehen auch ohne Netzverbindung offline zur Verfügung. Für das UI der Beispielapplikation sind fixe Kopf- und Fußzeilen mit Navigation vorgesehen, Formulare, Schaltflächen und Dialoge für die Speicherung und Änderung von Markern sowie eine Liste mit allen Markern.

Wir testen die mobile Applikation *Markt It* auf folgenden Smartphones (vgl. Tabelle 7):

Smartphone	Plattform	Version
Nokia N8	Symbian	^3
HTC Nexus One	Android	2.2.1
Torch 9800	BlackBerry	6.0
iPhone 3GS	iOS	4.3.2

Tabelle 7: Übersicht Smartphones zum Testen der Beispielapplikation

Die einzelnen Kriterien werden für jede Plattform nach dem Schulnotensystem von 1 bis 5 bewertet (vgl. Tabelle 8). Wir beurteilen die Kriterien UI, Geolocation, Karte, offline Applikationen, lokale Datenbank und Kamera vorallem nach aufgetretenen Fehlern und Funktionsfähigkeit. Die Bewertung der Kriterien Installation und Entwicklungsumgebung, Test und Distribution sind von der Einfachheit, Funktionsfähigkeit sowie der Bestandsaufnahme und den daraus resultierenden Vor- und Nachteilen abhängig.

Note	Bewertung
1	Sehr Gut: keine Fehler, voll funktionsfähig, sehr einfach
2	Gut: ein Fehler, einfach
3	Befriedigend: zwei bis drei Fehler, mittelmäßig
4	Genügend: vier bis fünf Fehler, schwierig
5	Nicht Genügend: mehr als fünf Fehler, nicht funktionsfähig, sehr schwierig

Tabelle 8: Bewertung nach Schulnotensystem

10.1 jQuery Mobile

Wir setzten die Beispielapplikation mit der jQuery Mobile Alpha Version 1.0a4.1 um (jQuery Mobile, 2010y). Das Framework inklusive dem jQuery core (Version 1.5.2) hat eine Größe von 225 KB. Die gesamte Beispielapplikation hat eine Größe von 295 KB. Die Installation des jQuery Mobile Frameworks ist sehr einfach zu handhaben, da lediglich die notwendigen JavaScript und CSS Dateien in die HTML Datei eingebunden werden müssen. Vorteilhaft ist, dass jQuery Mobile auf dem jQuery core aufbaut, wodurch sich für Entwicklerinnen/Entwickler mit jQuery Vorkenntnissen ein einfaches Arbeiten aufgrund der konsistenten und bekannten Syntax ergibt. jQuery Mobile ist an keine bestimmte Entwicklungsumgebung gebunden, sie kann je nach Vorliebe frei von der Entwicklerin/dem Entwickler gewählt werden.

Die Entwicklung des UI von jQuery Mobile bedurfte aufgrund der einfachen und klaren HTML Struktur sowie der ausführlichen Dokumentation eine geringe Einarbeitungszeit, wodurch wird dieses sehr schnell umsetzen konnten. Weiters sind alle für die Beispielapplikation notwendigen UI Komponenten im Framework enthalten. Da es sich bei jQuery Mobile derzeit um ein reines UI Framework handelt, verfügt dieses über keine Hilfsfunktionen zum Einbinden der Kriterien Geolocation, Karte, offline Applikationen, lokale Datenbank und Kamera. Deshalb griffen wir für die Implementierung der Karte auf die Google Maps API und für die anderen Kriterien auf bestehende W3C Spezifikationen zurück. Für die Umsetzung der genannten Kriterien

war einerseits eine längere Einarbeitungszeit notwendig, andererseits konnten die Kriterien ohne Probleme gemeinsam mit dem jQuery Mobile Framework realisiert werden. Der Zugriff auf die Kamera wird derzeit von keinem der mobilen Webbrowser unterstützt und konnte deshalb in der Beispielapplikation nicht umgesetzt werden. Die mit jQuery Mobile realisierte Beispielapplikation besteht aus einer Quellcode-Basis, die plattformunabhängig eingesetzt werden. Ein weiterer Vorteil für Entwicklerinnen/Entwickler ist, dass Änderungen in nur einer Quellcode-Basis erfolgen müssen.

Die Möglichkeit die Beispielapplikation zunächst ohne Smartphones in einem Desktop Webbrowser testen zu können, stellt einen großen Vorteil dar. Jede kleinste Änderung oder Erweiterung an der mobilen Web Applikation konnte somit sehr rasch getestet werden. Weiters kann die Applikation auch in den jeweiligen Simulatoren/Emulatoren, die von jeder Plattform kostenlos zur Verfügung stehen, getestet werden. Allerdings kann das wirkliche Verhalten der mobilen Web Applikation, beispielsweise bzgl. der UI Performance, nur direkt auf den Smartphones getestet werden. Mit jQuery Mobile entwickelte mobile Web Applikationen können, im Gegensatz zum Vertrieb über Application Stores, sofort nach Fertigstellung ins Internet gestellt werden, um weltweit über mobile Webbrowser abgerufen werden zu können. Eine große Erleichterung für Entwicklerinnen/Entwickler ist, dass die mobile Applikation jederzeit problemlos ohne Genehmigungsprozess aktualisiert werden kann. Nachteilig ist, dass die Beispielapplikation in keinem der bekannten App Stores aufscheint und sich der Verkauf der Applikation ohne App Store recht umständlich gestaltet.

10.1.1 Symbian

Die Symbian Plattform wird derzeit vom jQuery Mobile Framework noch nicht unterstützt (jQuery Mobile, 2010z), weshalb das UI nur in Textform im mobilen Webbrowser angezeigt wird. Weiters können auch die Kriterien Geolocation, Karte, offline Applikationen, lokale Datenbank und Kamera aufgrund der fehlenden HTML5 Unterstützung im mobilen Webbrowser nicht eingesetzt werden. Da die mobile Web Applikation auf der Symbian Plattform nicht funktionsfähig ist, haben wir alle Kriterien mit *Nicht Genügend* bewertet (vgl. Tabelle 9).

Kriterium	Symbian
Installation und Entwicklungsumgebung	5
UI	5
Geolocation	5

Karte	5
Offline Applikationen	5
Lokale Datenbank	5
Kamera	5
Test	5
Distribution	5
Durchschnitt	**5**

Tabelle 9: Bewertung jQuery Mobile für Symbian

10.1.2 Android

Das UI von jQuery Mobile wird auf der Android Plattform mit *Genügend* bewertet, da die Seitenübergänge nicht flüssig sind, sondern flackern und ruckeln. Weiters funktioniert die Ausblendung der Adresszeile (Abbildung 53/1) im mobilen Webbrowser und somit die Darstellung der Applikation im Vollbild Modus nicht einwandfrei. Es gibt auch immer wieder Darstellungsprobleme bei der statische Kopf- und Fußzeile (Abbildung 53/2). Ein weiteres großes Problem ist, dass manchmal ohne zu erkennenden Grund, das jQuery Mobile Framework beim erstmaligen Aufruf nicht korrekt geladen und nur in Textform dargestellt wird (Abbildung 53/3).

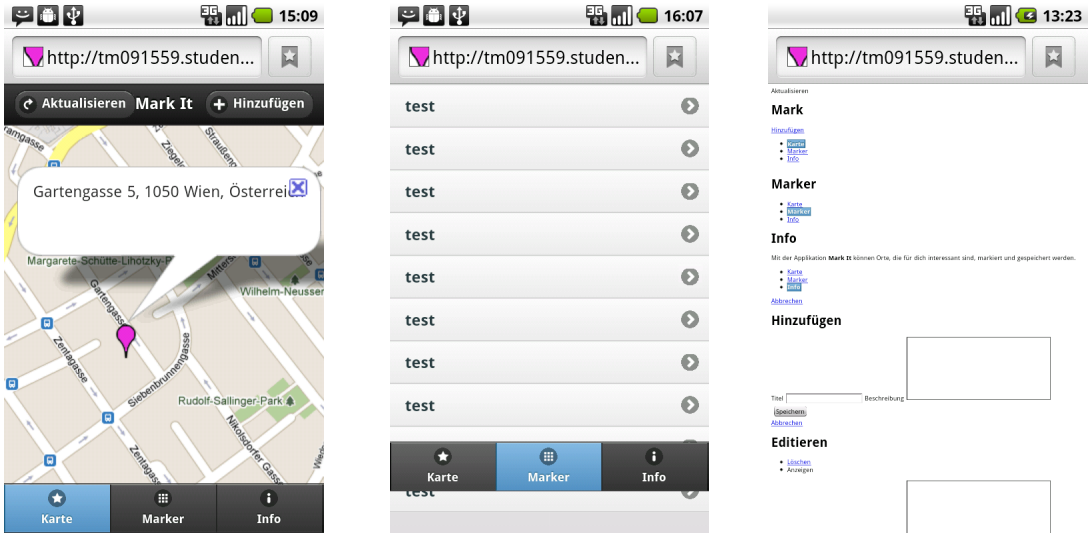


Abbildung 53: jQuery Mobile UI Android

Alle anderen UI Elemente, wie Schaltflächen, Formulare (Abbildung 54/1, 2), Listen und Dialoge (Abbildung 54/3) funktionieren einwandfrei und werden richtig dargestellt. Weiters wird das UI im Quer- und Hochformat automatisch angepasst.

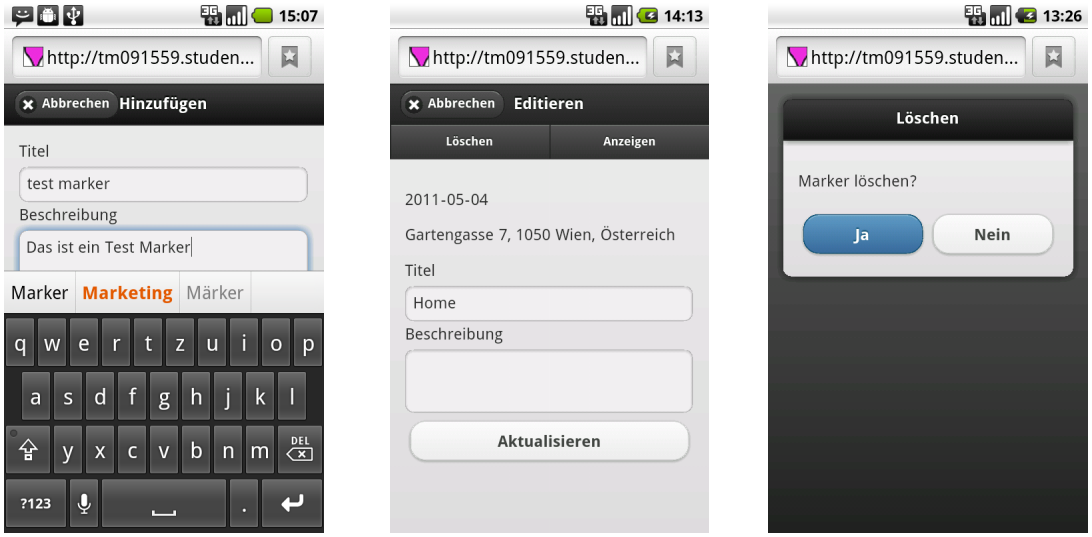


Abbildung 54: jQuery Mobile UI Android

Die Kriterien Geolocation (Abbildung 55/1), Karte (Abbildung 55/2), offline Applikationen (Abbildung 55/3) und lokale Datenbank in Kombination mit jQuery Mobile funktionieren einwandfrei und erhalten somit die Benotung *Sehr Gut*.

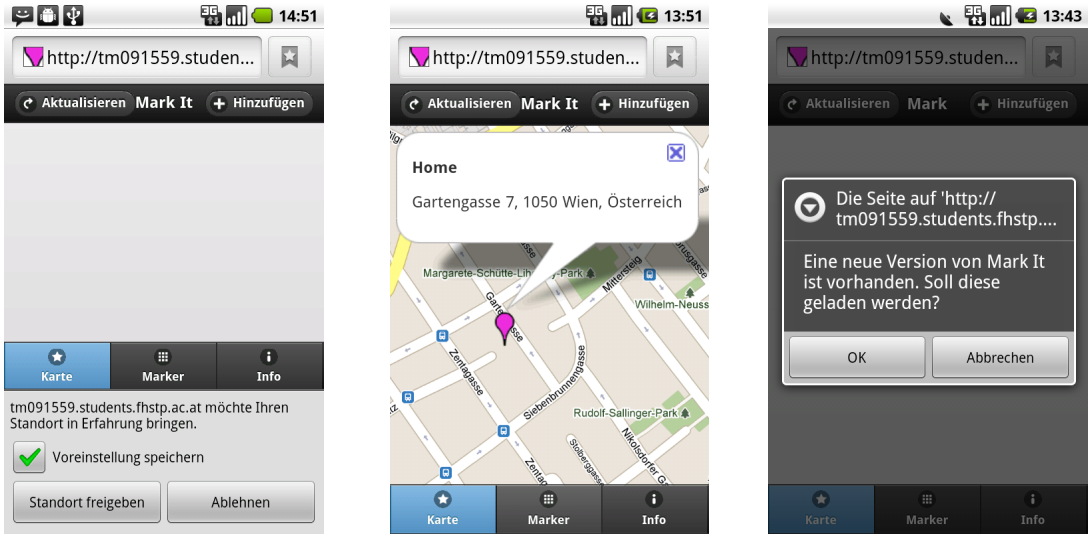


Abbildung 55: jQuery Mobile Android Kriterien

Der Zugriff auf die Kamera wird von Androids mobilen Webbrowser derzeit noch nicht unterstützt, deshalb die Bewertung *Nicht Genügend*. Das Kriterium Test wird mit *Gut* bewertet, da das wirkliche Verhalten des UI nur auf dem Smartphone getestet werden kann. Da die Vorteile und Nachteile des Kriteriums Distribution hinsichtlich Application Store recht ausgeglichen sind und die Android Plattform keine Konfigurationen für eine nativ wirkende Applikation zur Verfügung stellt, wird dieses mit *Befriedigend* bewertet. Die Bewertung aller Kriterien des jQuery Mobile Frameworks für die Android Plattform ist in Tabelle 10 ersichtlich.

Kriterium	Android
Installation und Entwicklungsumgebung	1
UI	4
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1
Kamera	5
Test	2
Distribution	3
Durchschnitt	**2,1**

Tabelle 10: Bewertung jQuery Mobile für Android

10.1.3 BlackBerry

Das UI auf der BlackBerry Plattform wird mit *Nicht Genügend* bewertet, da die Seitenübergänge sowie das Öffnen von Dialogen sehr langsam sind. Ab und zu wird nach Betätigung einer Schaltfläche die neue Seite nicht angezeigt. Die Ausblendung der Adressleiste funktioniert nicht einwandfrei und bei der automatischen Anpassung der Kopf- und Fußzeile (Abbildung 56/1, 2) auf die gesamte Breite und Höhe im Hoch- und Querformat gibt es teilweise Probleme. Ein weiteres großes Problem ist, dass manchmal ohne zu erkennenden Grund, das jQuery Mobile Framework beim erstmaligen Aufruf, wie auf der Android Plattform nicht korrekt geladen und nur in Textform dargestellt wird (Abbildung 56/3).

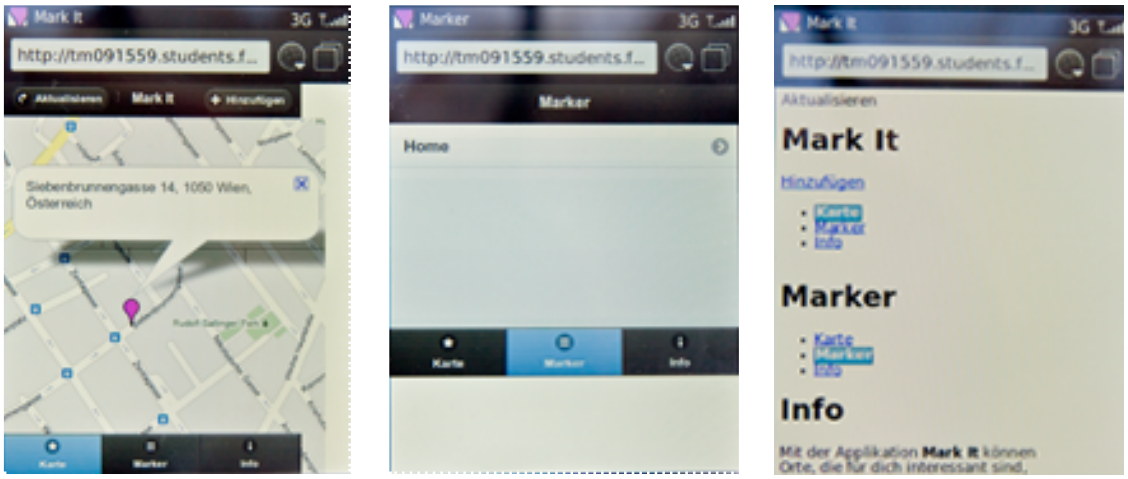


Abbildung 56: jQuery Mobile UI BlackBerry

Alle anderen UI Elemente, wie Schaltflächen, Formulare (Abbildung 57/1, 2), Listen und Dialoge (Abbildung 57/3) funktionieren einwandfrei und werden richtig dargestellt.

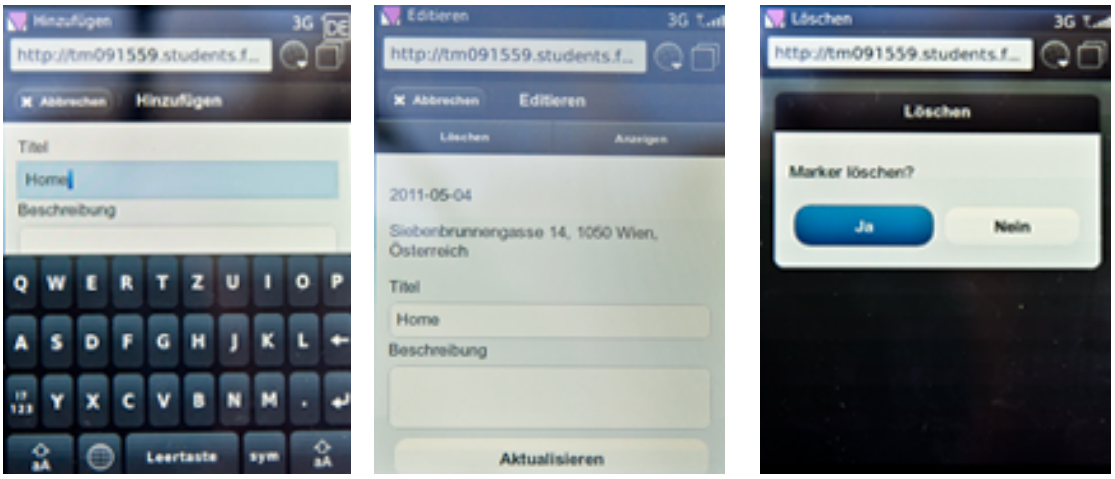


Abbildung 57: jQuery Mobile UI BlackBerry

Die Kriterien Geolocation (Abbildung 58/1), Karte (Abbildung 58/2) und lokale Datenbank funktionieren in Kombination mit jQuery Mobile und erhalten die Bewertung *Sehr Gut*. Anzumerken ist, dass die Ermittlung des Standorts auf dem BlackBerry Torch 9800 Smartphone recht lange dauert und in Innenräumen funktioniert die Positionsbestimmung teilweise gar nicht.

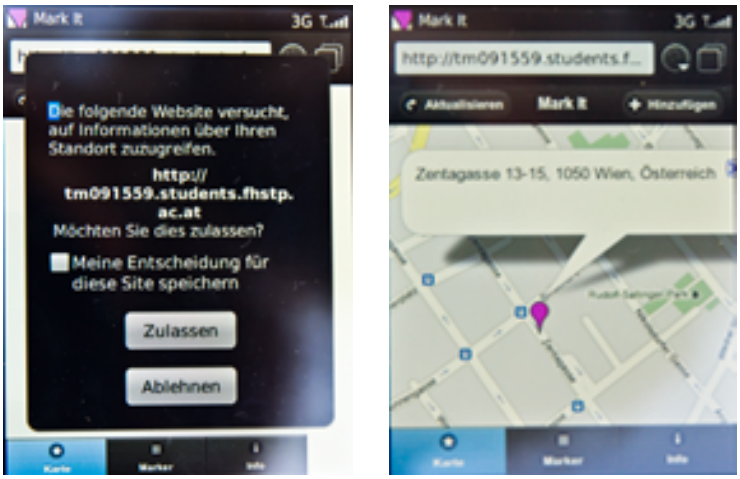


Abbildung 58: jQuery Mobile Kriterien BlackBerry

Das Kriterium offline Applikationen wird nur beim ersten Aufruf der Applikation unterstützt. Das Herunterladen von neuen Dateien bei Aktualisierungen funktioniert jedoch nicht einwandfrei. Dies hat zur Folge, dass die CSS und JavaScript Dateien des Frameworks nicht immer korrekt geladen werden und somit die mobile Web Applikation nur in Textform ohne UI dargestellt wird, aus diesem Grund haben wir die Bewertung *Nicht Genügend* vergeben. Der Zugriff auf die Kamera wird von BlackBerrys mobilen Webbrowser derzeit noch nicht unterstützt, deshalb die Bewertung *Nicht Genügend*. Der Test der mobilen Web Applikation wird mit *Befriedigend* bewertet, da das reale Verhalten der UI Performance, Geolocation und offline Speicherung nur direkt auf dem

Smartphone getestet werden kann. Da die Vorteile und Nachteile des Kriteriums Distribution hinsichtlich Application Store recht ausgeglichen sind und die BlackBerry Plattform keine Konfigurationen für eine nativ wirkende Applikation zur Verfügung stellt, wird dieses mit *Befriedigend* bewertet. Die Bewertung aller Kriterien des jQuery Mobile Frameworks für die BlackBerry Plattform ist in Tabelle 11 ersichtlich.

Kriterium	BlackBerry
Installation und Entwicklungsumgebung	1
UI	5
Geolocation	1
Karte	1
Offline Applikationen	5
Lokale Datenbank	1
Kamera	5
Test	3
Distribution	3
Durchschnitt	**2,7**

Tabelle 11: Bewertung jQuery Mobile für BlackBerry

10.1.4 iOS

Das UI auf der iOS Plattform wird mit *Befriedigend* bewertet, da die Ausblendung der Adressleiste nicht immer funktioniert, es Darstellungsprobleme bei der statischen Kopf- und Fußzeile (Abbildung 59/1, 2) gibt und die mobile Web Applikation nicht immer auf die richtige Höhe und Breite angepasst wird (Abbildung 59/3).

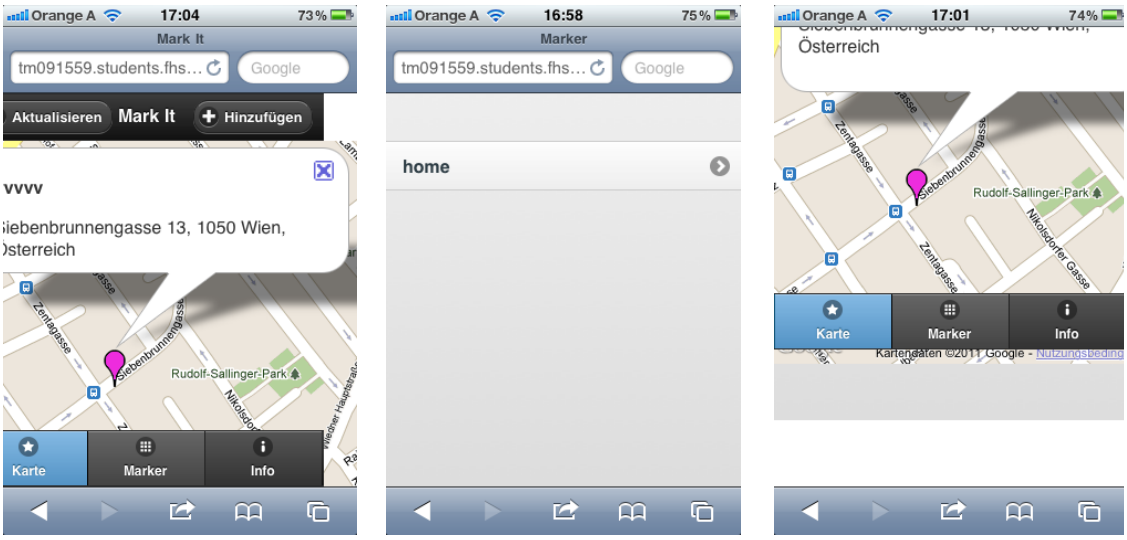


Abbildung 59: jQuery Mobile UI iOS

Die Anzeige der Applikation ist im Hoch- und Querformat möglich. Alle anderen UI Elemente, wie Schaltflächen, Formulare (Abbildung 60/1, 2), Listen und Dialoge (Abbildung 60/3) funktionieren einwandfrei und werden richtig dargestellt.

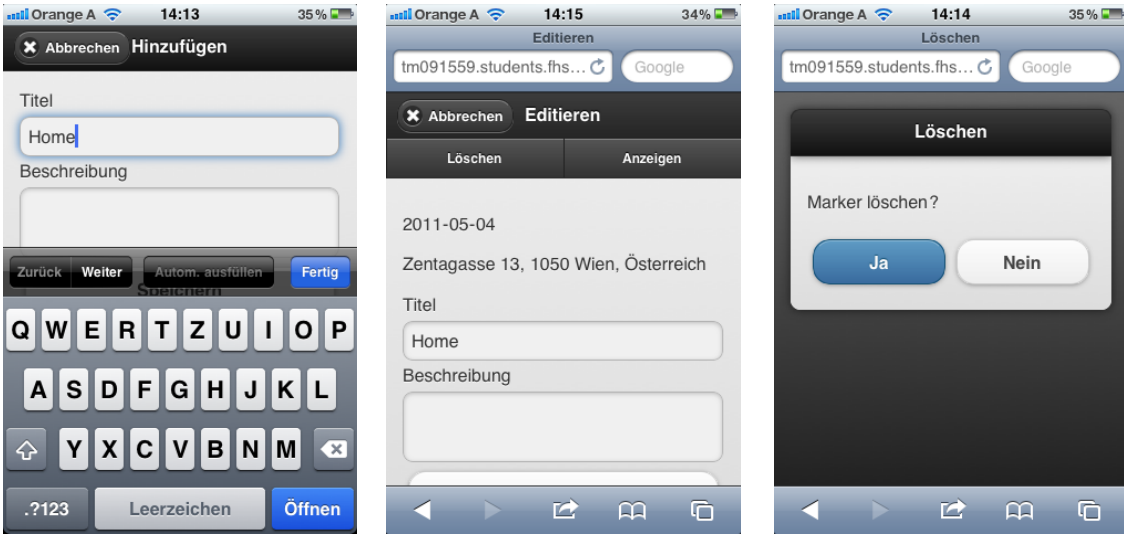


Abbildung 60: jQuery Mobile UI iOS

Die Kriterien Geolocation (Abbildung 61/1), Karte, offline Applikationen (Abbildung 61/2) und lokale Datenbank (Abbildung 61/3) funktionieren in Kombination mit jQuery Mobile einwandfrei und erhalten somit die Benotung *Sehr Gut*.

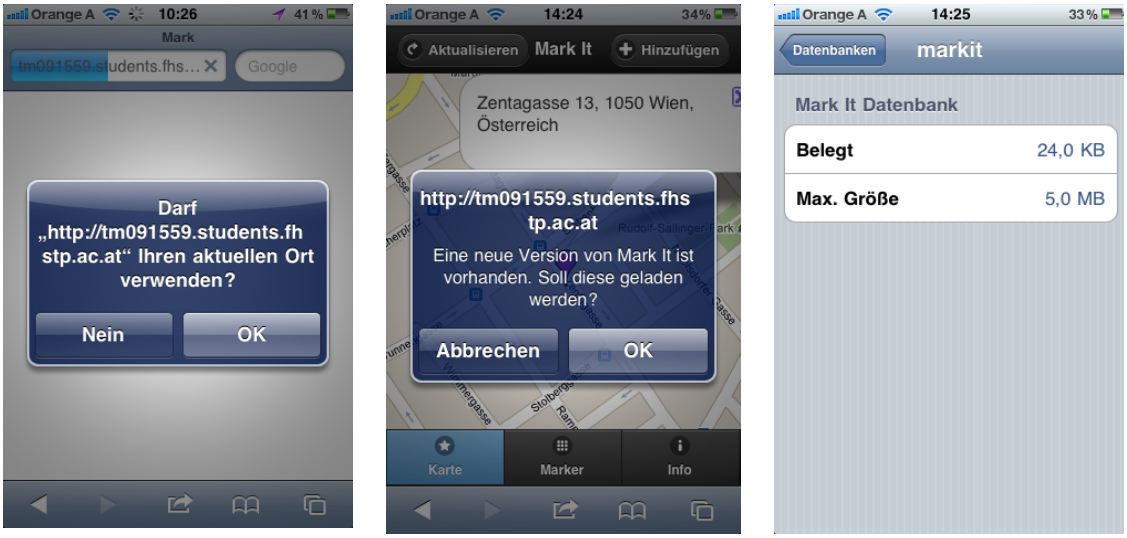


Abbildung 61: jQuery Mobile Kriterien iOS

Der Zugriff auf die Kamera wird von iOS mobilen Webbrowser derzeit noch nicht unterstützt, deshalb die Bewertung *Nicht Genügend*. Da beim Test der mobilen Web Applikation im Desktop Browser sowie im Simulator gegenüber dem Test auf einem echten Smartphone nichts gefehlt oder anders war, haben wir das Kriterium mit *Sehr Gut* benotet. Da die Vorteile und Nachteile des Kriteriums Distribution recht

ausgeglichen sind, die iOS Plattform jedoch einige Konfigurationen für eine nativ wirkende Applikation zur Verfügung stellt, wird dieses mit *Gut* bewertet. Die Bewertung aller Kriterien des jQuery Mobile Frameworks für die iOS Plattform ist in Tabelle 12 ersichtlich.

Kriterium	iOS
Installation und Entwicklungsumgebung	1
UI	3
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1
Kamera	5
Test	1
Distribution	2
Durchschnitt	**1,7**

Tabelle 12: Bewertung jQuery Mobile für iOS

10.1.5 Bewertung jQuery Mobile für alle Plattformen

Tabelle 13 zeigt die Bewertung der Kriterien des jQuery Mobile Frameworks für alle Plattformen.

Kriterium	Symbian	Android	BlackBerry	iOS	Durchschnitt je Kriterium
Installation und Entwicklungsumgebung	5	1	1	1	**2**
UI	5	4	5	3	**4,25**
Geolocation	5	1	1	1	**2**
Karte	5	1	1	1	**2**
Offline Applikationen	5	1	5	1	**3**
Lokale Datenbank	5	1	1	1	**2**
Kamera	5	5	5	5	**5**
Test	5	2	3	1	**2,75**
Distribution	5	3	3	2	**3,25**
Durchschnitt je Plattform	**5**	**2,1**	**2,7**	**1,7**	**2,9**

Tabelle 13: Bewertung jQuery Mobile für alle Plattformen

10.2 Sencha Touch

Wir setzten die Beispielapplikation mit der Sencha Touch Version 1.1.0 um (Sencha, 2011c). Das Framework hat eine Größe von 524 KB. Die gesamte Beispielapplikation hat eine Größe von 602 KB. Die Installation des Sencha Touch Frameworks ist sehr einfach zu handhaben, da lediglich die notwendigen JavaScript und CSS Dateien in die HTML Datei eingebunden werden müssen. Sencha Touch ist an keine bestimmte Entwicklungsumgebung gebunden, sie kann je nach Vorliebe frei von der Entwicklerin/dem Entwickler gewählt werden.

Die Entwicklung des UI mit dem Sencha Touch Framework bedurfte aufgrund der eigenen JavaScript Syntax eine längere Einarbeitungszeit, konnte jedoch durch ausführliche Dokumentationen und Beispiele einfach erlernt werden. Alle für die Beispielapplikation notwendigen UI Komponenten sind im Framework enthalten. Sencha Touch verfügt über eigene Komponenten zur Einbindung von beispielsweise Karten mit integrierter Geolocation. Die Implementierung einer Karte und Geolocation kann auch problemlos mit Google Maps und der W3C Geolocation Spezifikation, wie bei jQuery Mobile erfolgen. Da die Einbindung der eigenen Sencha Touch Komponente mehr Zeit und Nerven in Anspruch genommen hat und diese auf der BlackBerry Plattform nicht funktioniert, haben wir bei der Beispielapplikation auf die zweite Variante zurückgegriffen. Für die Einbindung der Kriterien offline Applikationen und lokale Datenbank haben wir, wie bei jQuery Mobile auf W3C Spezifikationen zurückgegriffen. Der Zugriff auf die Kamera wird derzeit von keinem der mobilen Webbrowser unterstützt und konnte deshalb in der Beispielapplikation nicht umgesetzt werden. Die mit Sencha Touch realisierte Beispielapplikation besteht aus einer Quellcode-Basis, die plattformunabhängig eingesetzt werden. Ein weiterer Vorteil für Entwicklerinnen/Entwickler ist, dass Änderungen in nur einer Quellcode-Basis erfolgen müssen.

Die Möglichkeit die Beispielapplikation zunächst ohne Smartphones in einem Desktop Webbrowser testen zu können, stellt einen großen Vorteil dar. Jede kleinste Änderung oder Erweiterung an der mobilen Web Applikation konnte somit sehr rasch getestet werden. Weiters kann die Applikation auch in den jeweiligen Simulatoren/Emulatoren, die von jeder Plattform kostenlos zur Verfügung stehen, getestet werden. Mit Sencha Touch entwickelte mobile Web Applikationen können, im Gegensatz zum Vertrieb über Application Stores, sofort nach Fertigstellung ins Internet gestellt werden, um weltweit über mobile Webbrowser abgerufen werden zu können. Eine große Erleichterung für Entwicklerinnen/Entwickler ist, dass die mobile Applikation jederzeit problemlos ohne

Genehmigungsprozess aktualisiert werden kann. Nachteilig ist, dass die Beispielapplikation in keinem der bekannten App Stores aufscheint und sich der Verkauf der Applikation ohne App Store recht umständlich gestaltet.

10.2.1 Symbian

Die Symbian Plattform wird vom Sencha Touch Framework nicht unterstützt, weshalb im mobilen Webbrowser nur eine weiße leere Seite angezeigt wird. Weiters können auch die Kriterien Geolocation, Karte, offline Applikationen, lokale Datenbank und Kamera aufgrund der fehlenden HTML5 Unterstützung im mobilen Webbrowser nicht eingesetzt werden. Da die mobile Web Applikation auf der Symbian Plattform nicht funktionsfähig ist, haben wir alle Kriterien mit *Nicht Genügend* bewertet (vgl. Tabelle 14).

Kriterium	Symbian
Installation und Entwicklungsumgebung	5
UI	5
Geolocation	5
Karte	5
Offline Applikationen	5
Lokale Datenbank	5
Kamera	5
Test	5
Distribution	5
Durchschnitt	**5**

Tabelle 14: Bewertung Sencha Touch für Symbian

10.2.2 Android

Das UI von Sencha Touch wird auf der Android Plattform mit *Sehr Gut* bewertet, da die Seitenübergänge flüssig sind, die Ausblendung der Adressleiste (Abbildung 62/1) funktioniert und auch alle anderen UI Komponenten, wie Kopf- und Fußzeile, Schalflächen, Formulare (Abbildung 62/2, 4), Listen (Abbildung 62/3) und Dialoge (Abbildung 62/5) im Hoch- und Querformat einwandfrei funktionieren und dargestellt werden.

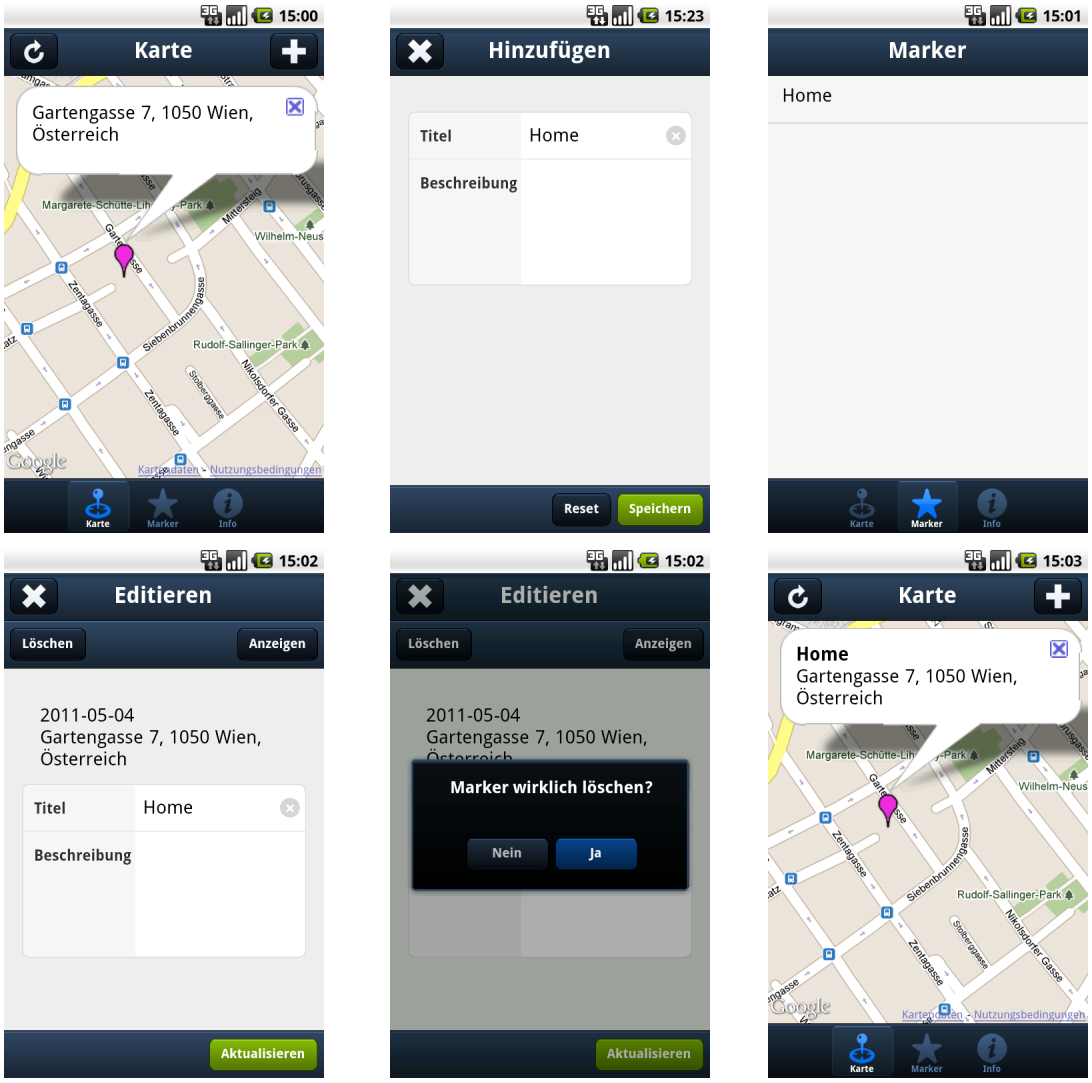


Abbildung 62: Sencha Touch UI Android

Die Kriterien Geolocation (Abbildung 63/1) und Karte (Abbildung 63/2) erhalten die Benotung *Sehr Gut*, da das von Sencha Touch zur Verfügung gestellte Karten Modul mit integrierter Geolocation mit voller Funktionsfähigkeit implementiert werden konnte und auch die Implementierung der beiden Kriterien ohne Sencha Touch Modul sehr einfach möglich ist. Die Kriterien offline Applikationen (Abbildung 63/3) und lokale Datenbank in Kombination mit Sencha Touch funktionieren einwandfrei und erhalten somit die Benotung *Sehr Gut*.

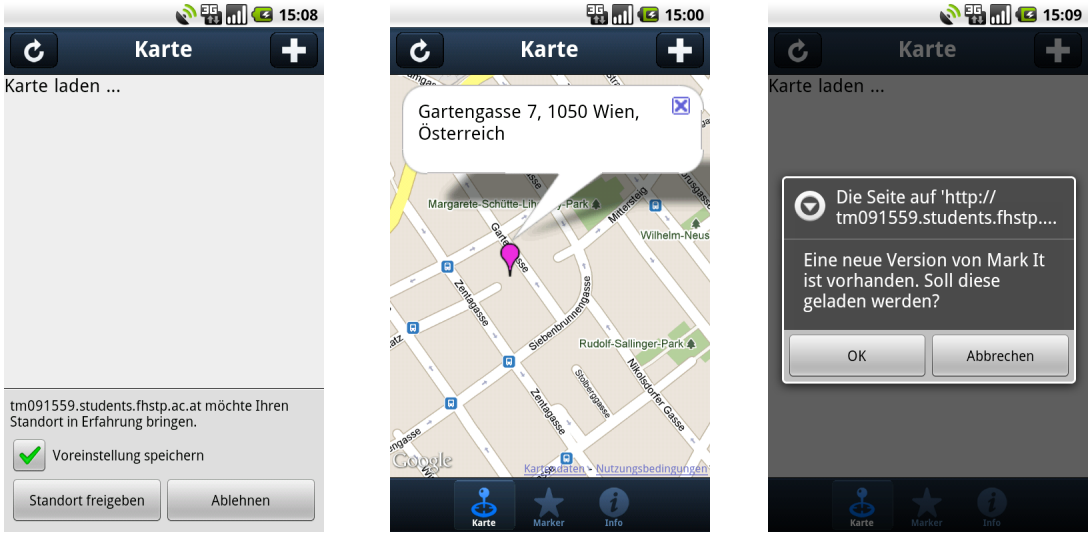


Abbildung 63: Sencha Touch Kriterien Android

Der Zugriff auf die Kamera wird von Androids mobilen Webbrowser derzeit noch nicht unterstützt, deshalb die Bewertung *Nicht Genügend*. Das Kriterium Test wird mit *Sehr Gut* bewertet, da die mobile Web Applikation im Desktop Webbrowser das gleiche Verhalten wie auf dem Smartphone aufweist. Da die Vorteile und Nachteile des Kriteriums Distribution hinsichtlich Application Store recht ausgeglichen sind und die Android Plattform keine Konfigurationen für eine nativ wirkende Applikation zur Verfügung stellt, wird dieses mit *Befriedigend* bewertet. Die Bewertung aller Kriterien des Sencha Touch Frameworks für die Android Plattform ist in Tabelle 15 ersichtlich.

Kriterium	Android
Installation und Entwicklungsumgebung	1
UI	1
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1
Kamera	5
Test	1
Distribution	3
Durchschnitt	**1,6**

Tabelle 15: Bewertung Sencha Touch für Android

10.2.3 BlackBerry

Das UI von Sencha Touch wird auf der BlackBerry Plattform mit *Sehr Gut* bewertet, da die Seitenübergänge flüssig sind, die Ausblendung der Adressleiste (Abbildung 64/1)

funktioniert und auch alle anderen UI Komponenten, wie Kopf- und Fußzeile, Schalflächen, Formulare (Abbildung 64/2, 4), Listen (Abbildung 64/3) und Dialoge (Abbildung 64/5) im Hoch- und Querformat einwandfrei funktionieren und dargestellt werden.

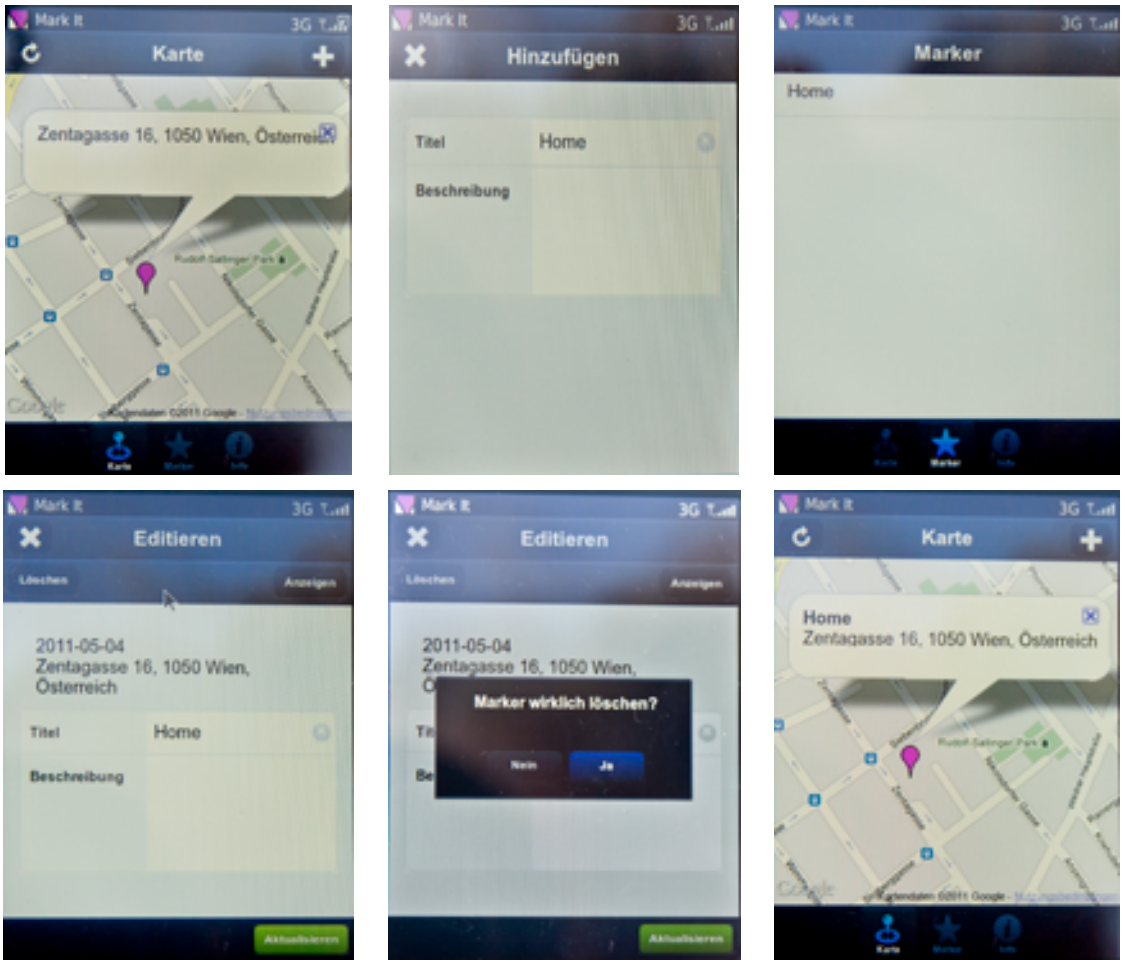


Abbildung 64: Sencha Touch UI BlackBerry

Die Kriterien Geolocation (Abbildung 65/1) und Karte (Abbildung 65/2) erhalten die Benotung *Gut*, da das von Sencha Touch zur Verfügung gestellte Karten Modul mit integrierter Geolocation auf der BlackBerry Plattform nicht funktioniert, hier scheint es noch einen Bug zu geben. Die Implementierung der beiden Kriterien ohne Sencha Touch Modul ist jedoch sehr einfach und auch schneller möglich. Anzumerken ist, dass die Ermittlung des Standorts auf dem BlackBerry Torch 9800 Smartphone recht lange dauert und in Innenräumen funktioniert die Positionsbestimmung teilweise gar nicht.

Abbildung 65: Sencha Touch Kriterien BlackBerry

Das Kriterium offline Applikationen wird nur beim ersten Aufruf unterstützt. Das Herunterladen von neuen Dateien bei Aktualisierungen funktioniert jedoch nicht korrekt, weshalb wir die Benotung *Nicht Genügend* vergeben haben. Die lokale Datenbank wird unterstützt und erhält somit die Note *Sehr Gut*. Der Zugriff auf die Kamera wird von BlackBerrys mobilen Webbrowser derzeit noch nicht unterstützt, deshalb die Bewertung *Nicht Genügend*. Der Test der mobilen Web Applikation wird mit *Befriedigend* bewertet, da das reale Verhalten der UI Performance, Geolocation und offline Applikationen nur direkt auf dem Smartphone getestet werden kann. Da die Vorteile und Nachteile des Kriteriums Distribution hinsichtlich Application Store recht ausgeglichen sind und die BlackBerry Plattform keine Konfigurationen für eine nativ wirkende Applikation zur Verfügung stellt, wird dieses mit *Befriedigend* bewertet. Die Bewertung aller Kriterien des Sencha Touch Frameworks für die BlackBerry Plattform ist in Tabelle 16 ersichtlich.

Kriterium	BlackBerry
Installation und Entwicklungsumgebung	1
UI	1
Geolocation	2
Karte	2
Offline Applikationen	5
Lokale Datenbank	1
Kamera	5
Test	3
Distribution	3
Durchschnitt	**2,5**

Tabelle 16: Bewertung Sencha Touch für BlackBerry

10.2.4 iOS

Das UI von Sencha Touch wird auf der iOS Plattform mit *Sehr Gut* bewertet, da die Seitenübergänge flüssig sind, die Ausblendung der Adressleiste (Abbildung 66/1) funktioniert und auch alle anderen UI Komponenten, wie Kopf- und Fußzeile, Schalflächen, Formulare (Abbildung 66/2, 4), Listen (Abbildung 66/3) und Dialoge (Abbildung 66/5) im Hoch- und Querformat einwandfrei funktionieren und dargestellt werden.

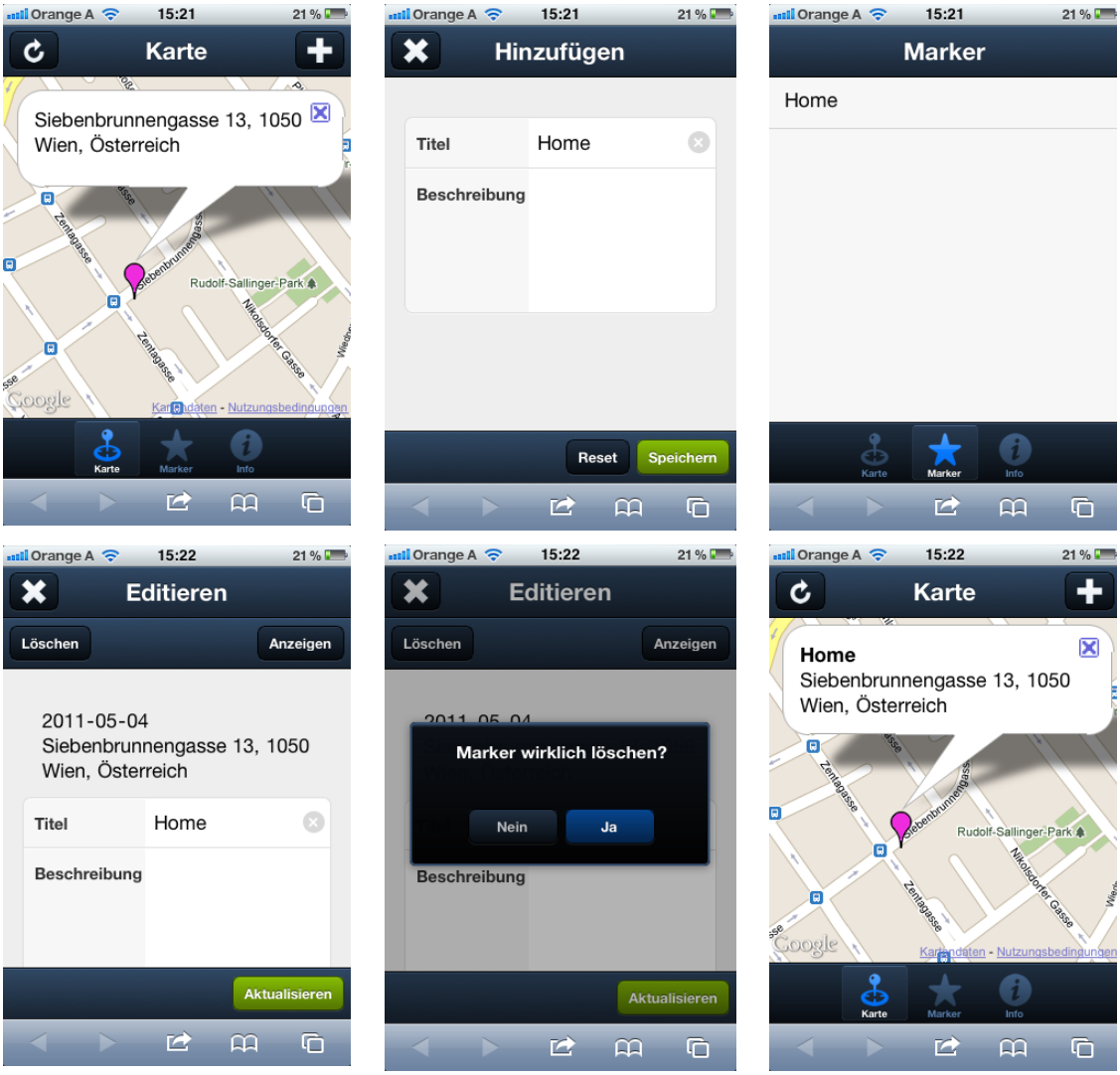


Abbildung 66: Sencha Touch UI iOS

Die Kriterien Geolocation (Abbildung 67/1) und Karte erhalten die Benotung *Sehr Gut*, da das von Sencha Touch zur Verfügung gestellte Karten Modul mit integrierter Geolocation mit voller Funktionsfähigkeit implementiert werden konnte und auch die Implementierung der beiden Kriterien ohne Sencha Touch Modul sehr einfach möglich ist. Die Kriterien offline Applikationen (Abbildung 67/2) und lokale Datenbank

(Abbildung 67/3) in Kombination mit Sencha Touch funktionieren einwandfrei und erhalten somit die Benotung *Sehr Gut*.

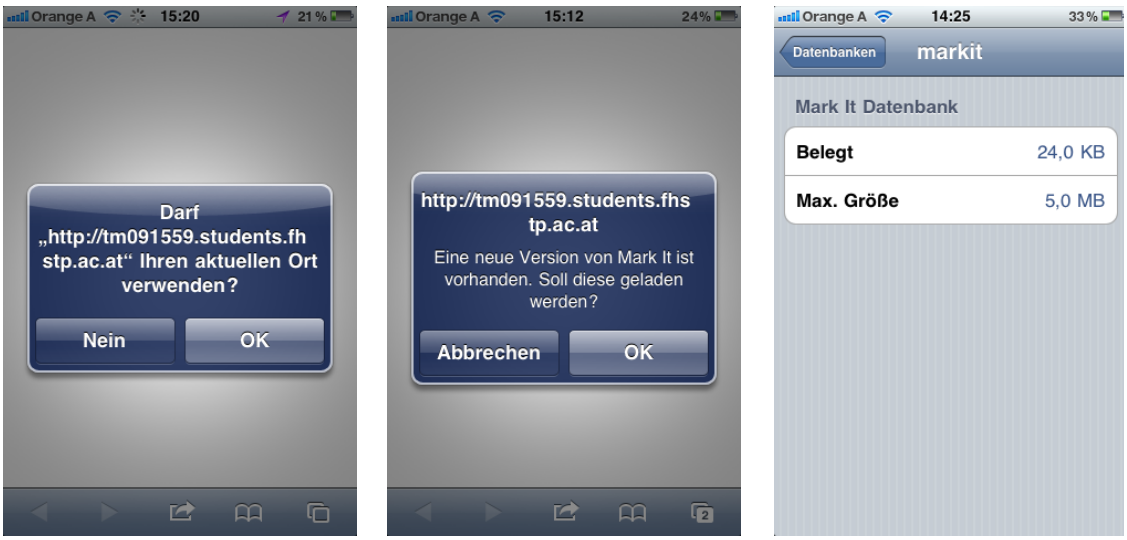


Abbildung 67: Sencha Touch Kriterien iOS

Der Zugriff auf die Kamera wird von iOS mobilen Webbrowser derzeit noch nicht unterstützt, deshalb die Bewertung *Nicht Genügend*. Da beim Test der mobilen Web Applikation im Desktop Browser und im Simulator gegenüber dem Test auf einem echten Smartphone nichts gefehlt oder anders war, haben wir das Kriterium mit *Sehr Gut* benotet. Da die Vorteile und Nachteile des Kriteriums Distribution recht ausgeglichen sind, die iOS Plattform jedoch einige Konfigurationen für eine nativ wirkende Applikation zur Verfügung stellt, wird dieses mit *Gut* bewertet. Die Bewertung aller Kriterien des Sencha Touch Frameworks für die iOS Plattform ist in Tabelle 17 ersichtlich.

Kriterium	iOS
Installation und Entwicklungsumgebung	1
UI	1
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1
Kamera	5
Test	1
Distribution	2
Durchschnitt	**1,5**

Tabelle 17: Bewertung Sencha Touch für iOS

10.2.5 Bewertung Sencha Touch für alle Plattformen

Tabelle 18 zeigt die Bewertung der Kriterien des Sencha Touch Frameworks für alle Plattformen.

Kriterium	Symbian	Android	BlackBerry	iOS	Durchschnitt je Kriterium
Installation und Entwicklungsumgebung	5	1	1	1	**2**
UI	5	1	1	1	**2**
Geolocation	5	1	2	1	**2,25**
Karte	5	1	2	1	**2,25**
Offline Applikationen	5	1	5	1	**3**
Lokale Datenbank	5	1	1	1	**2**
Kamera	5	5	5	5	**5**
Test	5	1	3	1	**2,5**
Distribution	5	3	3	2	**3,25**
Durchschnitt je Plattform	**5**	**1,6**	**2,5**	**1,5**	**2,7**

Tabelle 18: Bewertung Sencha Touch für alle Plattformen

10.3 PhoneGap

Wir setzten die Beispielapplikation mit der PhoneGap Version 0.9.5 um (vgl. PhoneGap, 2011b). Das Kriterium Installation und Entwicklungsumgebung hat sich beim PhoneGap Framework als sehr aufwendig herausgestellt, da nicht nur die Einarbeitung in das Framework, sondern auch in die damit verbundenen Anforderungen an die verschiedenen Plattformen notwendig ist. Damit das PhoneGap Framework eingesetzt werden kann, muss das entsprechende SDK jeder Plattform (außer Symbian) installiert werden und PhoneGap sieht für jede Plattform eine Projektvorlage mit eigener PhoneGap JavaScript Datei vor. Zusätzlich sind zwei verschiedene Desktop Betriebssystem notwendig, da das iOS SDK nur auf Mac OS X und das BlackBerry WebWorks SDK nur auf Windows läuft.

PhoneGap verfügt über kein eigenes UI, es kann jedoch ohne Probleme das jQuery Mobile oder Sencha Touch Framework als UI eingesetzt werden. Da das UI von Sencha Touch bei der Entwicklung der mobilen Web Applikation viel besser abgeschnitten hat, haben wir dieses in der PhoneGap Beispielapplikation herangezogen. Der große Vorteil an PhoneGap ist, dass bestehende mobile Web Applikationen für die Weiterentwicklung mit PhoneGap verwendet werden können. Da

das PhoneGap Framework auch auf W3C Spezifikationen aufbaut, konnte die bereits mit Sencha Touch realisierte mobile Web Applikation mit all ihren Kriterien für die Weiterentwicklung herangezogen werden. Somit funktionierten die Kriterien Geolocation, Karte und lokale Datenbank auf Anhieb und mussten nicht angepasst werden. Die in der mobilen Web Applikation vorgenommenen Konfigurationen für offline Applikationen sind mit dem PhoneGap Framework nicht notwendig, da alle Dateien automatisch nach der Kompilierung und Installation auf den Plattform offline zur Verfügung stehen. Ein weiterer sehr großer Vorteil von PhoneGap ist, dass auf verschiedene Software- und Hardwarefunktion, die derzeit in den mobilen Webbrowsern noch nicht implementiert sind, zugegriffen werden kann. Der Zugriff auf die Kamera konnte sehr rasch und ohne Problem in die Beispielapplikation integriert werden. Der funktionierende Quellcode einer Plattform kann ohne Probleme für die anderen Plattformen herangezogen werden. Die Wartung ist jedoch sehr aufwendig, da alle Quellcodeänderungen für die einzelnen Plattformen nachgezogen werden müssen. Eine zentrale Quellcode-Basis, wie bei mobilen Web Applikationen gibt es nicht.

Sehr aufwendig und zeitintensiv ist auch der Test der Beispielapplikation auf den verschiedenen Plattformen. Die Applikation muss in den jeweiligen Entwicklungsumgebungen (Xcode für iOS, Eclipse für Android, Kommandozeilentool Ant für BlackBerry) kompiliert und auf Simulatoren/Emulatoren oder echten Smartphones installiert werden, um getestet werden zu können. Jede kleinste Änderung im Quellcode erfordert die Neukompilierung der Applikation. Nachteilig ist auch, dass die BlackBerry und iOS Plattform zum Testen auf echten Smartphones eine Signierung vorsieht, dies ist mit viel Zeit, Aufwand und bei iOS auch mit Kosten verbunden. Die mit PhoneGap realisierte hybride Applikation kann über Application Stores vertrieben und verkauft werden. Dies stellt einen großen Vorteil dar, da viele Benutzerinnen/Benutzer Applikationen über App Stores suchen. Nachteilig ist, dass die Applikation nach Fertigstellung nicht sofort im Application Store zur Verfügung steht, da diese zunächst einen Genehmigungsprozess durchlaufen muss, der einige Wochen in Anspruch nehmen kann und auch nach jeder Aktualisierung der Applikation durchgeführt werden muss.

10.3.1 Symbian

Für die Verwendung des PhoneGap Frameworks ist kein SDK notwendig, weshalb wir das Kriterium Installation und Entwicklungsumgebung mit *Sehr Gut* bewertet haben. Das UI des Sencha Touch Frameworks funktioniert auf Symbian nicht, das jQuery Mobile Framework jedoch schon, aus diesem Grund die Bewertung *Befriedigend*. Die

Kriterien Geolocation und Karte (Abbildung 68/1) werden von Symbian unterstützt, die Karte allerdings nur in statischer Form, deshalb die Benotung *Befriedigend*. Wenn die Applikation im offline Modus aufgerufen wird, wird das UI nur in Textform (Abbildung 68/2) dargestellt und kann somit nicht verwendet werden, deshalb die Note *Nicht Genügend*. Die Kriterien lokale Datenbank und Kamera (Abbildung 68/3) erhalten die Benotung *Nicht Genügend*, da diese auf der Symbian Plattform nicht funktionieren.

Abbildung 68: PhoneGap Symbian

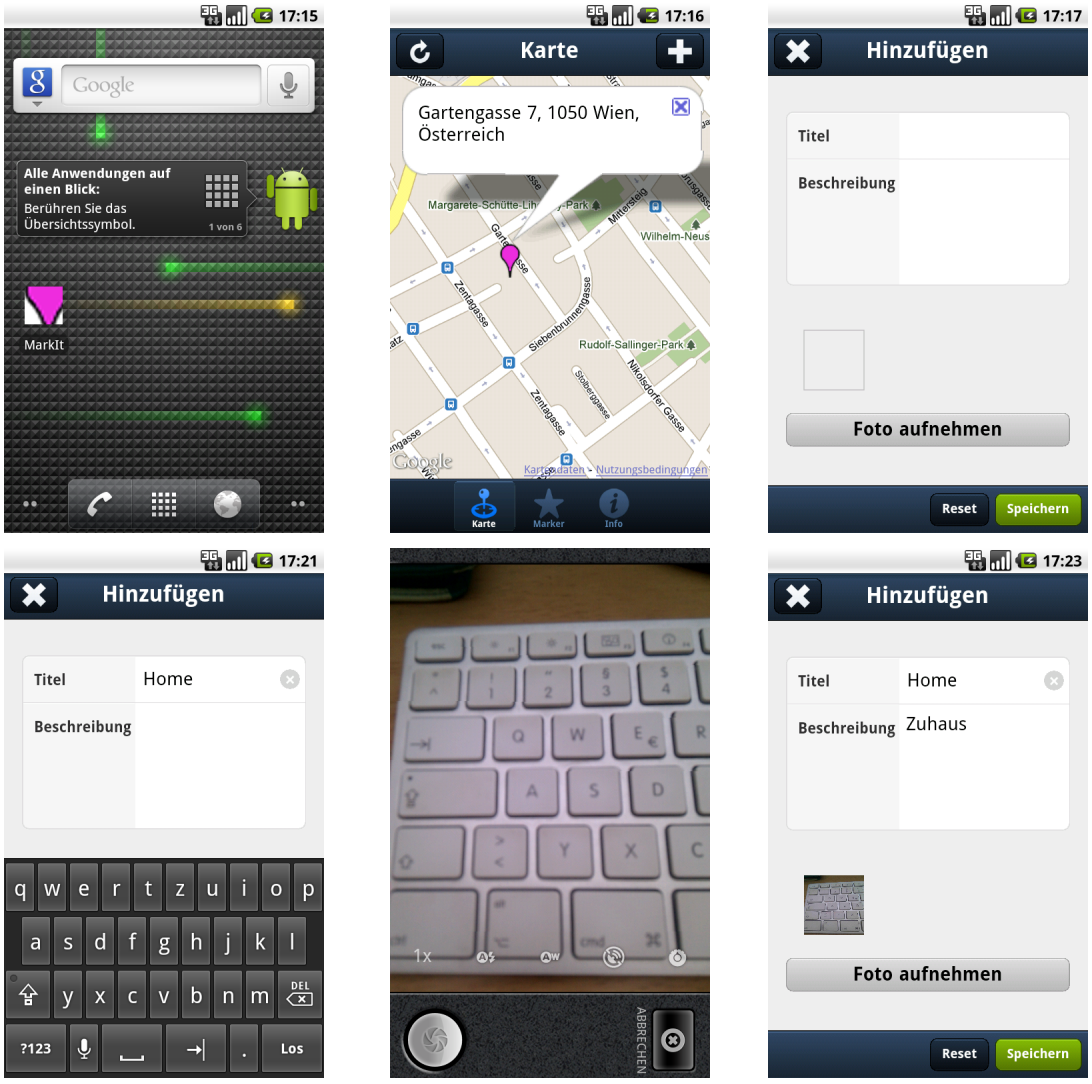
Da sich der Test auf einem Nokia Smartphone gegenüber den anderen Plattformen einfacher gestaltet, haben wir die Note *Befriedigend* vergeben. Die Vorteile und Nachteile des Kriteriums Distribution sind recht ausgeglichen. Da die für Symbian realisierte Beispielapplikation jedoch letztendlich aufgrund der fehlenden Unterstützung der lokalen Datenbank nicht verwendet und somit nicht vertrieben werden kann, haben wir die Note *Nicht Genügend* vergeben. Die Bewertung aller Kriterien des PhoneGap Frameworks für die Symbian Plattform ist in Tabelle 19 ersichtlich.

Kriterium	Symbian
Installation und Entwicklungsumgebung	1
UI	3
Geolocation	1
Karte	3
Offline Applikationen	5
Lokale Datenbank	5
Kamera	5
Test	3
Distribution	5
Durchschnitt	**3,2**

Tabelle 19: Bewertung PhoneGap für Symbian

10.3.2 Android

Das Kriterium Installation und Entwicklungsumgebung erhält die Bewertung *Genügend*, da diese für die Android Plattform sehr aufwendig ist. Das Kriterium UI funktioniert, wie in der mobilen Web Applikation einwandfrei und wird mit *Sehr Gut* bewertet. Die Kriterien Geolocation, Karte (Abbildung 69/2), offline Applikationen, lokale Datenbank und Kamera (Abbildung 69/5) werden mit *Sehr Gut* bewertet, da diese ohne Probleme und funktionsfähig umgesetzt werden konnten. Da der Test der Applikation sehr aufwendig und zeitintensiv ist, wird die Note *Genügend* vergeben. Da die Vorteile und Nachteile des Kriteriums Distribution recht ausgeglichen sind und die Applikation in Vollbild-Modus auf dem Home-Bildschirm zur Verfügung steht, wird dieses mit *Gut* bewertet. Die Bewertung aller Kriterien des PhoneGap Frameworks für die Android Plattform ist in Tabelle 20 ersichtlich.



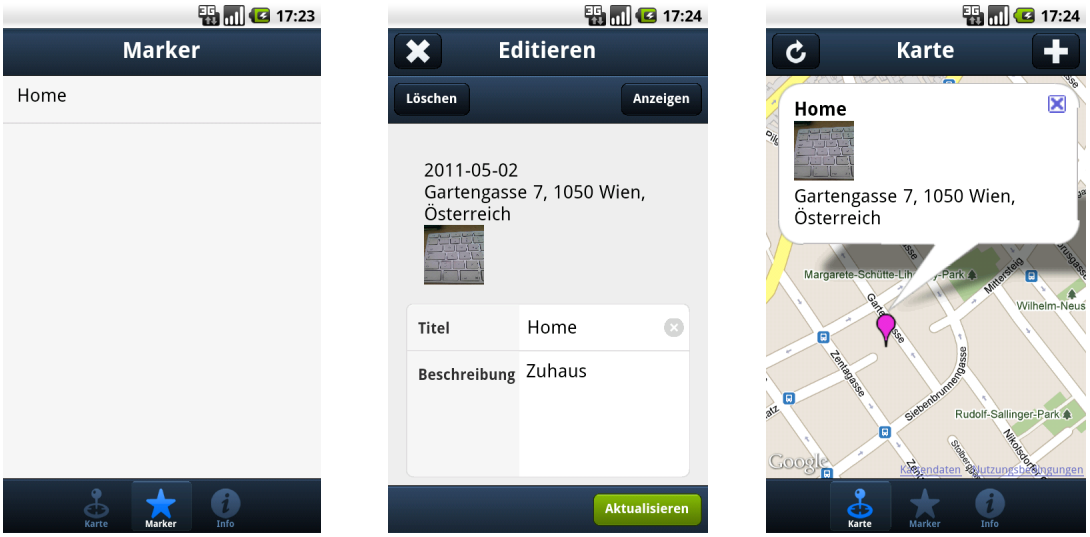


Abbildung 69: PhoneGap Android

Kriterium	Android
Installation und Entwicklungsumgebung	4
UI	1
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1
Kamera	1
Test	4
Distribution	2
Durchschnitt	**1,7**

Tabelle 20: Bewertung PhoneGap für Android

10.3.3 BlackBerry

Die Benotung der BlackBerry Plattform sieht aus denselben Gründen genau gleich wie bei der Android Plattform (Kapitel 10.3.2) aus. Die Bewertung aller Kriterien des PhoneGap Frameworks für die BlackBerry Plattform ist in Tabelle 21 ersichtlich.

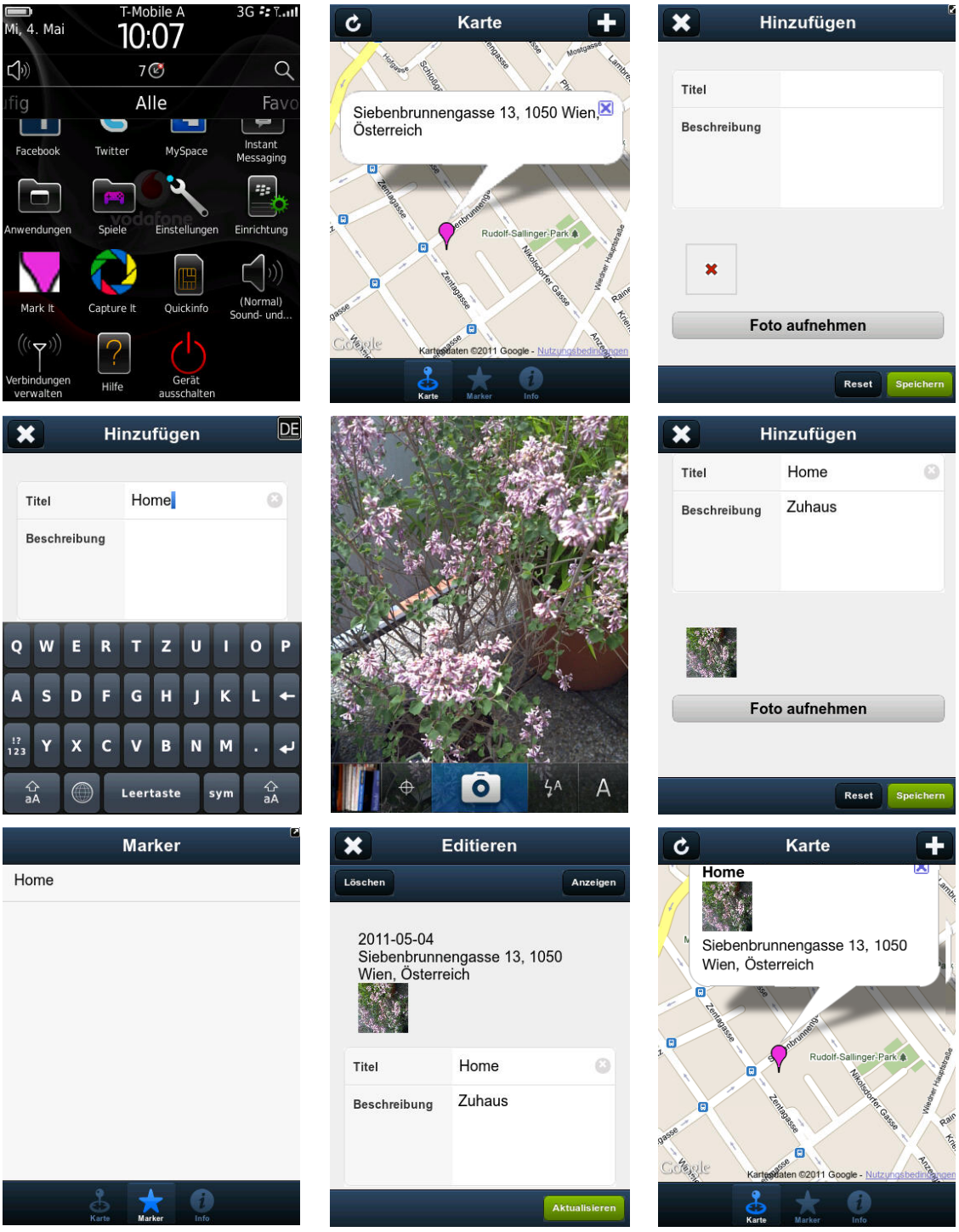


Abbildung 70: PhoneGap BlackBerry

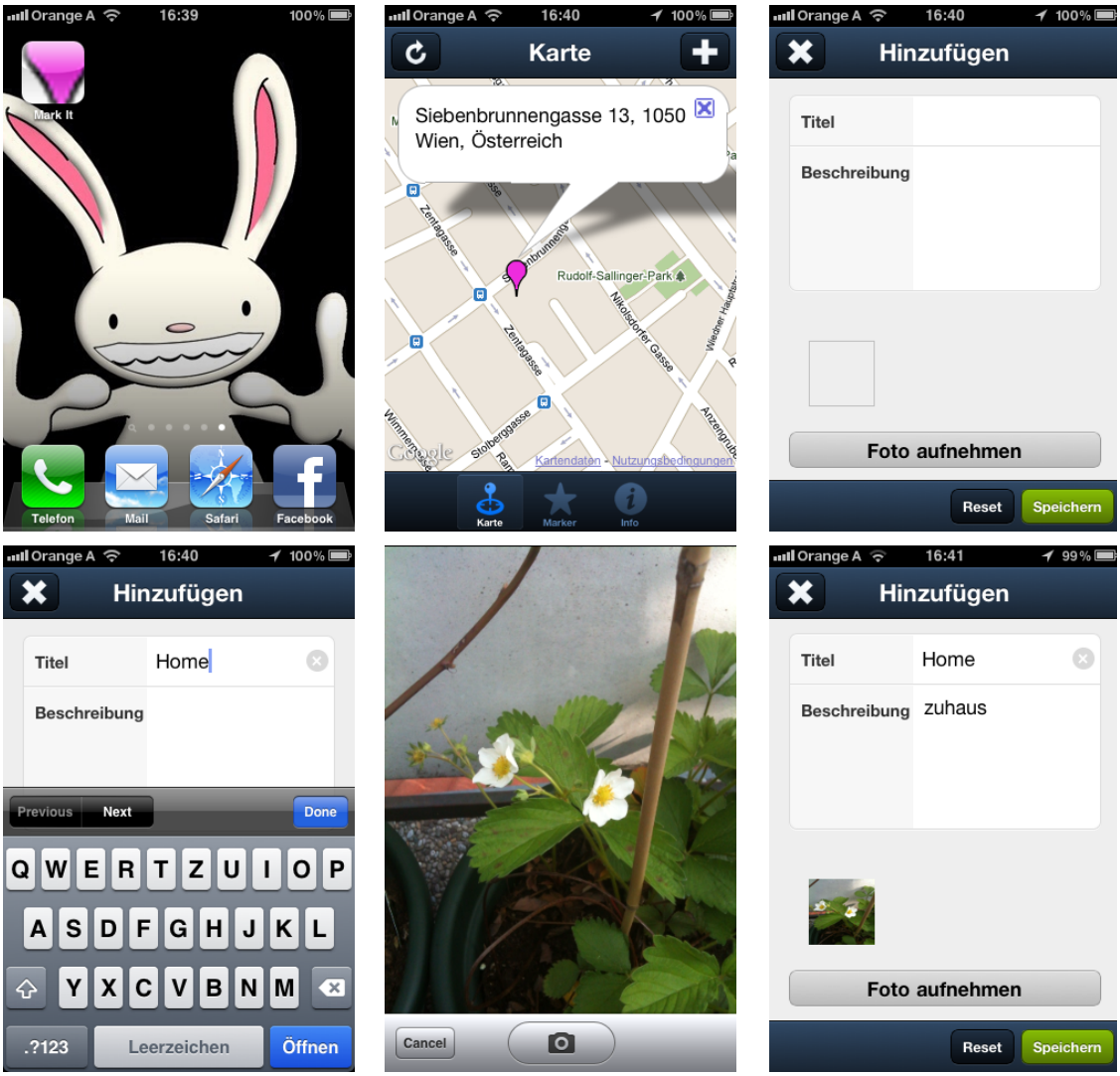
Kriterium	BlackBerry
Installation und Entwicklungsumgebung	4
UI	1
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1

Kamera	1
Test	4
Distribution	2
Durchschnitt	**1,7**

Tabelle 21: Bewertung PhoneGap für BlackBerry

10.3.4 iOS

Die Benotung der iOS Plattform sieht aus denselben Gründen genau gleich wie bei der Android Plattform (Kapitel 10.3.2) aus. Die Bewertung aller Kriterien des PhoneGap Frameworks für die iOS Plattform ist in Tabelle 22 ersichtlich.



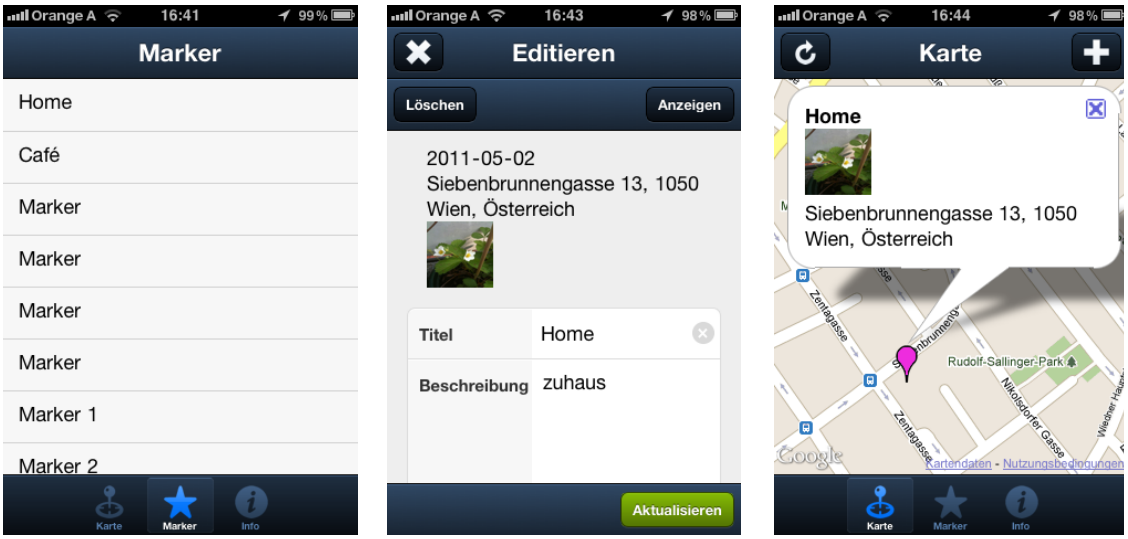


Abbildung 71: PhoneGap iOS

Kriterium	iOS
Installation und Entwicklungsumgebung	4
UI	1
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1
Kamera	1
Test	4
Distribution	2
Durchschnitt	**1,7**

Tabelle 22: Bewertung PhoneGap für iOS

10.3.5 Bewertung PhoneGap für alle Plattformen

Tabelle 23 zeigt die Bewertung der Kriterien des PhoneGap Frameworks für alle Plattformen.

Kriterium	Symbian	Android	BlackBerry	iOS	Durchschnitt je Kriterium
Installation und Entwicklungsumgebung	1	4	4	4	**3,25**
UI	3	1	1	1	**1,5**
Geolocation	1	1	1	1	**1**
Karte	3	1	1	1	**1,5**
Offline Applikationen	5	1	1	1	**2**
Lokale Datenbank	5	1	1	1	**2**

Kamera	5	1	1	1	**2**
Test	3	4	4	4	**3,75**
Distribution	5	2	2	2	**2,75**
Durchschnitt je Plattform	**3,2**	**1,7**	**1,7**	**1,7**	**2,1**

Tabelle 23: Bewertung PhoneGap für alle Plattformen

10.4 Titanium Mobile

Wir setzte die Beispielapplikation mit der Titanium Mobile SDK Version 1.6.2 um. Voraussetzung für die Verwendung von Titanium Mobile ist die Installation der Titanium Developer IDE sowie die SDKs für Android und iOS. Gegenüber PhoneGap konnte das Kriterium Installation und Entwicklungsumgebung einfach und rasch erledigt werden, da die IDE die zentrale Verwaltung übernimmt, die SDKs automatisch erkannt werden und an keine bestimmte Entwicklungsumgebung gebunden ist.

Die Entwicklung mit dem Titanium Mobile Framework bedurfte aufgrund der eigenen JavaScript Syntax eine längere Einarbeitungszeit, konnte jedoch durch ausführliche Dokumentationen und Beispiele einfach erlernt werden. Das UI von Titanium Mobile baut auf native iOS und Android Komponenten auf und konnte ohne Probleme entwickelt werden. Die Kriterien Geolocation, Karte, lokale Datenbank und Kamera konnten mit wenig Aufwand in die Beispielapplikation implementiert werden. Die in der mobilen Web Applikation vorgenommenen Konfigurationen für offline Applikationen sind mit dem Titanium Mobile Framework nicht notwendig, da alle Dateien automatisch nach der Kompilierung und Installation auf den Plattformen offline zur Verfügung stehen. Die mit Titanium Mobile realisierte Beispielapplikation besteht aus einer Quellcode-Basis, die plattformunabhängig eingesetzt werden kann. Ein großer Vorteil für Entwicklerinnen/Entwickler ist, dass Änderungen in nur einer Quellcode-Basis erfolgen müssen.

Der Test der Beispielapplikation ist zeitintensiv, da diese bei jeder kleinsten Änderung im Quellcode für Android und iOS neu kompiliert werden muss. Ein großer Vorteil gegenüber PhoneGap ist jedoch, dass die Kompilierung der Beispielapplikation zentral über die Titanium Developer IDE für alle Plattformen vorgenommen werden kann und somit nicht verschiedene Entwicklungsumgebungen benötigt werden. Die Beispielapplikation kann sowohl im Emulator von Android/Simulator von iOS oder direkt auf den Smartphones getestet werden. Nachteilig ist, dass die iOS Plattform zum Testen auf echten Smartphones eine Signierung vorsieht und dies ist mit Kosten

verbunden. Die mit Titanium Mobile realisierte hybride Applikation kann über Application Stores vertrieben und verkauft werden. Dies stellt einen großen Vorteil dar, da viele Benutzerinnen/Benutzer Applikationen über App Stores suchen. Nachteilig ist, dass die Applikation nach Fertigstellung nicht sofort im Application Store zur Verfügung steht, da diese zunächst einen Genehmigungsprozess durchlaufen muss, der einige Wochen in Anspruch nehmen kann und auch nach jeder Aktualisierung der Applikation durchgeführt werden muss. Vorteilhaft ist, dass die finale vertriebsbereite Paketierung für die iOS und Android Plattform über die Titanium Developer IDE vorgenommen werden kann.

10.4.1 Symbian

Die Symbian Plattform wird von Titanium Mobile nicht unterstützt, weshalb wir jedes Kriterium mit *Nicht Genügend* bewertet haben. Die Bewertung aller Kriterien des Titanium Mobile Frameworks für die Symbian Plattform ist in Tabelle 24 ersichtlich.

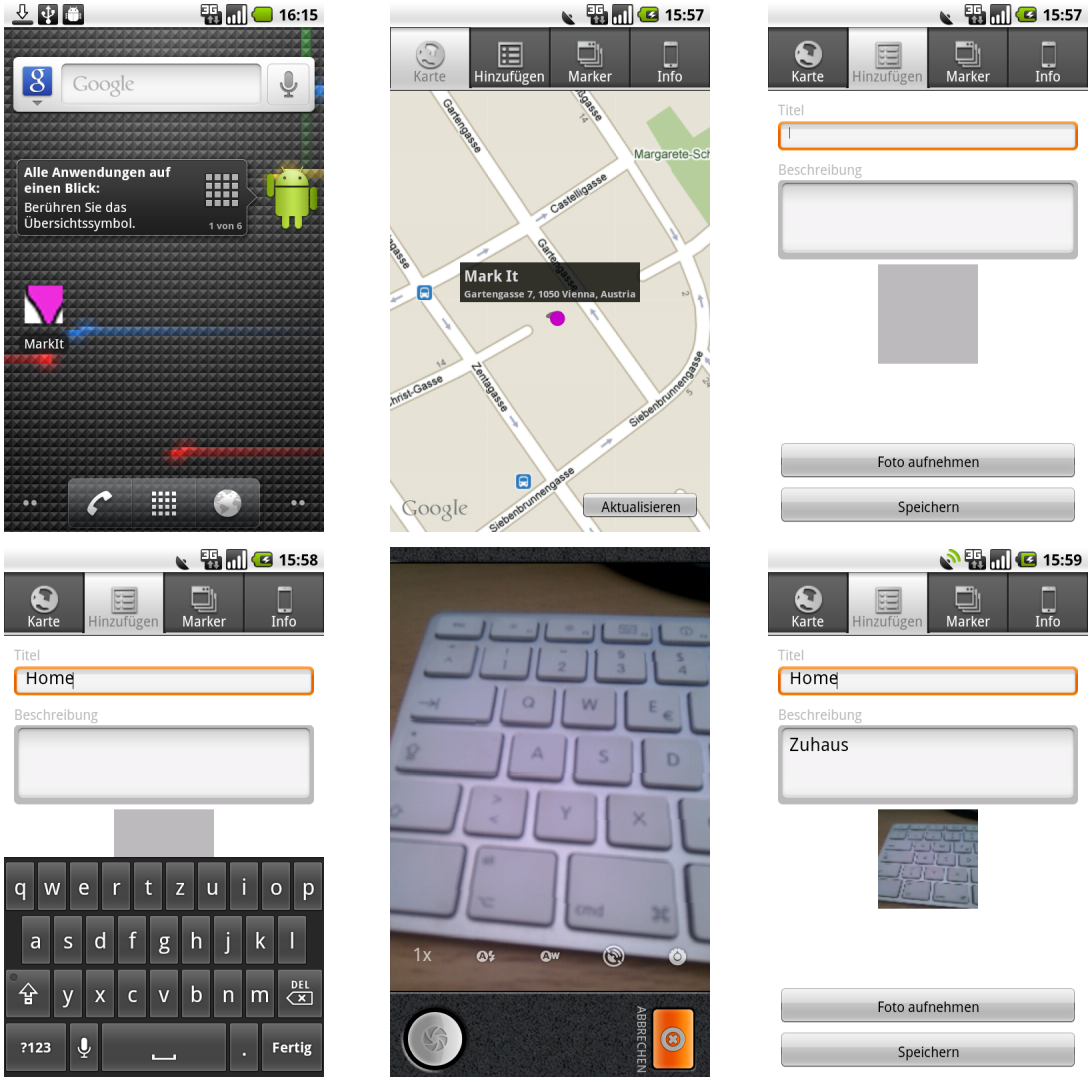
Kriterium	Symbian
Installation und Entwicklungsumgebung	5
UI	5
Geolocation	5
Karte	5
Offline Applikationen	5
Lokale Datenbank	5
Kamera	5
Test	5
Distribution	5
Durchschnitt	**5**

Tabelle 24: Bewertung Titanium Mobile für Symbian

10.4.2 Android

Das Kriterium Installation und Entwicklungsumgebung erhält die Bewertung *Gut*, da die Installation des Frameworks sehr rasch und einfach vorgenommen werden kann, allerdings im Gegensatz zur mobilen Web Applikation zusätzlich die Titanium Developer IDE und das SDK von Android benötigt wird. Das Kriterium UI wird mit *Sehr Gut* bewertet, die dieses ohne Problem umgesetzt werden konnten und den nativen Komponenten der Android Plattform entsprechen (Abbildung 72/3, 4, 6, 7). Die Kriterien Geolocation, Karte (Abbildung 72/2), offline Applikationen, lokale Datenbank und Kamera (Abbildung 72/5) werden mit *Sehr Gut* bewertet, da diese ohne Probleme

und voll funktionsfähig umgesetzt werden konnten. Der Test kann über die Titanium Developer IDE bequem im Simulator oder direkt auf dem Android Smartphone durchgeführt werden. Allerdings nimmt der Test, im Gegensatz zur mobilen Web Applikation, mehr Zeit in Anspruch, da bei jeder Änderung die Applikation neue kompiliert und installiert werden muss, aus diesem Grund die Bewertung *Befriedigend*. Da die Vorteile und Nachteile des Kriteriums Distribution recht ausgeglichen sind und die Applikation in Vollbild-Modus auf dem Home-Bildschirm zur Verfügung steht, wird dieses mit *Gut* bewertet. Die Bewertung aller Kriterien des Titanium Mobile Frameworks für die Android Plattform ist in Tabelle 25 ersichtlich.



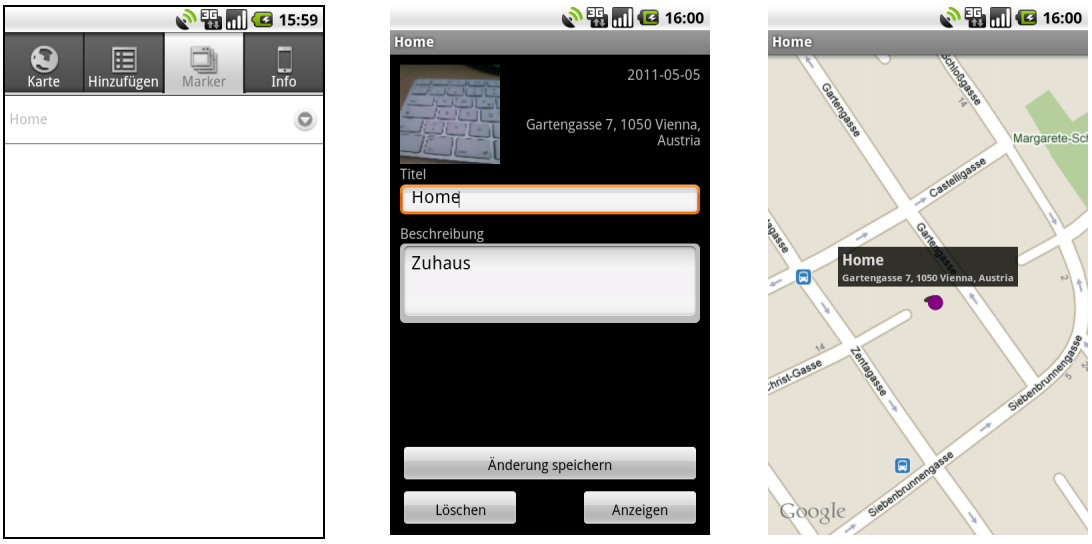


Abbildung 72: Titanium Mobile Android

Kriterium	Android
Installation und Entwicklungsumgebung	2
UI	1
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1
Kamera	1
Test	3
Distribution	2
Durchschnitt	**1,4**

Tabelle 25: Bewertung Titanium Mobile für Android

10.4.3 BlackBerry

Die BlackBerry Plattform wird derzeit noch nicht von Titanium Mobile unterstützt, weshalb wir jedes Kriterium mit *Nicht Genügend* bewertet haben. Die Bewertung aller Kriterien des Titanium Mobile Frameworks für die BlackBerry Plattform ist in Tabelle 26 ersichtlich.

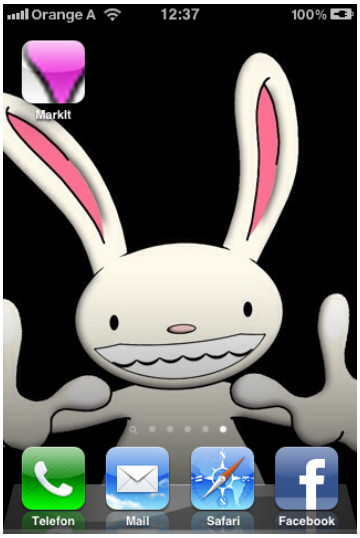
Kriterium	BlackBerry
Installation und Entwicklungsumgebung	5
UI	5
Geolocation	5
Karte	5
Offline Applikationen	5

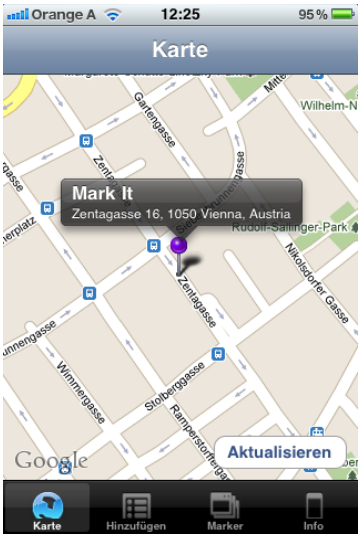
Lokale Datenbank	5
Kamera	5
Test	5
Distribution	5
Durchschnitt	**5**

Tabelle 26: Bewertung Titanium Mobile für BlackBerry

10.4.4 iOS

Das Kriterium Installation und Entwicklungsumgebung erhält die Bewertung *Gut*, da die Installation des Frameworks sehr rasch und einfach vorgenommen werden kann, allerdings im Gegensatz zur mobilen Web Applikation zusätzlich die Titanium Developer IDE und das SDK von iOS benötigt wird. Das Kriterium UI wird mit *Sehr Gut* bewertet, die dieses ohne Problem umgesetzt werden konnten und den nativen Komponenten der iOS Plattform entsprechen (Abbildung 73/3, 4, 6, 7). Die Kriterien Geolocation, Karte (Abbildung 73/2), offline Applikationen, lokale Datenbank und Kamera (Abbildung 73/5) werden mit *Sehr Gut* bewertet, da diese ohne Probleme und voll funktionsfähig umgesetzt werden konnten. Der Test kann über die Titanium Developer IDE bequem im Simulator oder direkt auf dem iPhone durchgeführt werden. Allerdings nimmt der Test, im Gegensatz zur mobilen Web Applikation, mehr Zeit in Anspruch, da bei jeder Änderung die Applikation neue kompiliert und installiert werden muss, aus diesem Grund die Bewertung *Befriedigend*. Da die Vorteile und Nachteile des Kriteriums Distribution recht ausgeglichen sind und die Applikation in Vollbild-Modus auf dem Home-Bildschirm zur Verfügung steht, wird dieses mit *Gut* bewertet. Die Bewertung aller Kriterien des Titanium Mobile Frameworks für die iOS Plattform ist in Tabelle 27 ersichtlich.





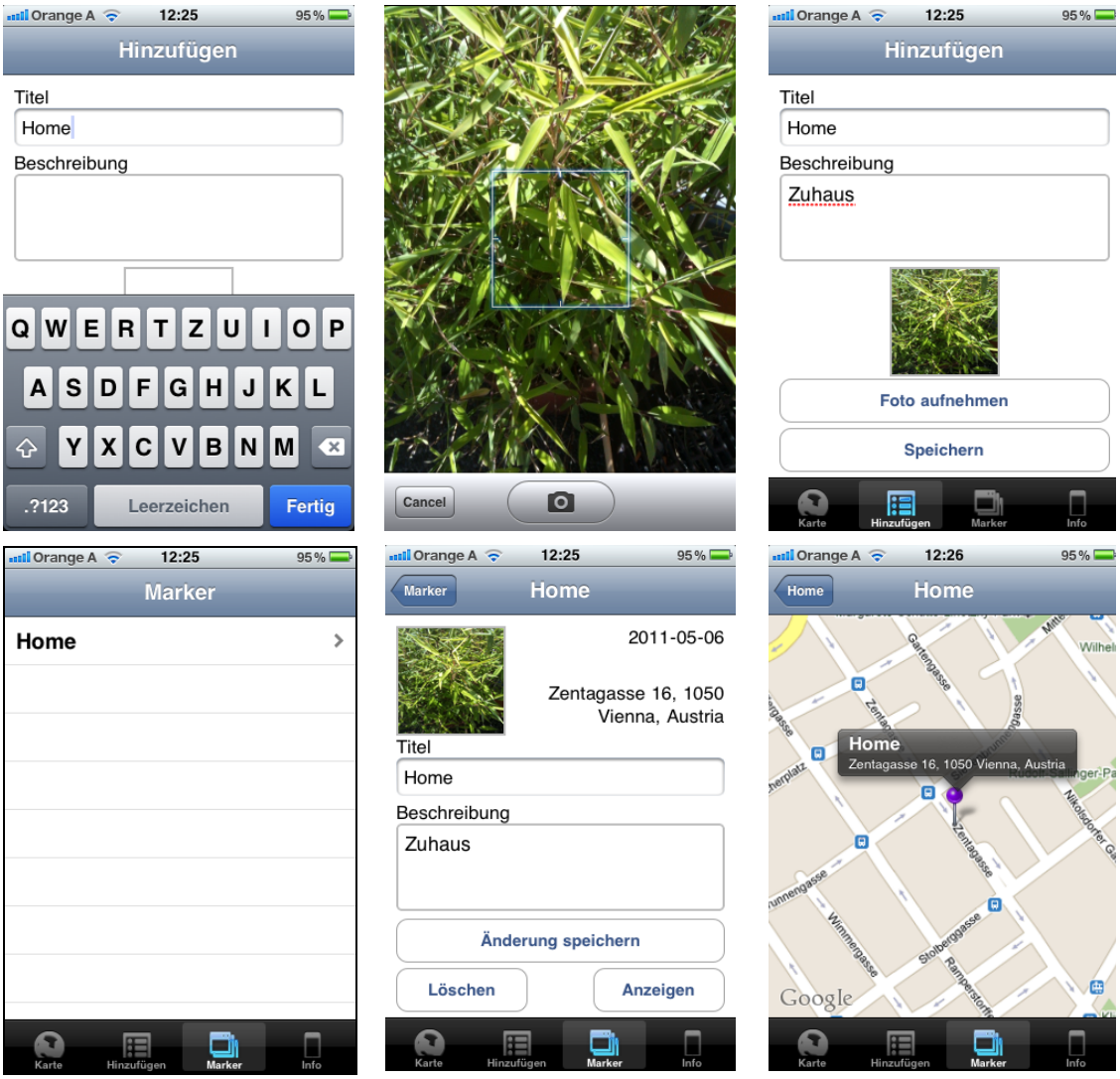


Abbildung 73: Titanium Mobile iOS

Kriterium	iOS
Installation und Entwicklungsumgebung	2
UI	1
Geolocation	1
Karte	1
Offline Applikationen	1
Lokale Datenbank	1
Kamera	1
Test	3
Distribution	2
Durchschnitt	**1,4**

Tabelle 27: Bewertung Titanium Mobile für iOS

10.4.5 Bewertung Titanium Mobile für alle Plattformen

Tabelle 28 zeigt die Bewertung der Kriterien des Titanium Mobile Frameworks für alle Plattformen.

Kriterium	Symbian	Android	BlackBerry	iOS	Durchschnitt je Kriterium
Installation und Entwicklungsumgebung	5	2	5	2	**3,5**
UI	5	1	5	1	**3**
Geolocation	5	1	5	1	**3**
Karte	5	1	5	1	**3**
Offline Applikationen	5	1	5	1	**3**
Lokale Datenbank	5	1	5	1	**3**
Kamera	5	1	5	1	**3**
Test	5	3	5	3	**4**
Distribution	5	2	5	2	**3,5**
Durchschnitt je Plattform	**5**	**1,4**	**5**	**1,4**	**3,2**

Tabelle 28: Bewertung Titanium Mobile für alle Plattformen

11 Empfehlung

Die Stärke-Schwäche Profile aus dem vorherigen Kapitel führen noch zu keiner endgültigen Empfehlung, da diese auch von unterschiedlichen Anforderungen an eine mobile Applikation seitens der Entwicklerinnen/Entwickler abhängig ist. Diese Anforderungen betreffen das UI, den Zugriff auf Hardware- und Softwarefunktionen sowie den Vertrieb über das Internet oder den jeweiligen Markt einer Plattform. Aus diesen Gründen ergibt sich die Empfehlung eines Frameworks mit jeweiliger Begründung anhand von drei verschiedenen Szenarien unter Berücksichtigung der Ergebnisse aus der Umsetzung der Beispielapplikation aus Kapitel 10. Die Szenarien zeigen auch die damit verbundenen Einschränkungen und Nachteile der Frameworks auf.

11.1 Erstes Szenario

Die Entwicklerin/der Entwickler benötigt ein Framework, mit welchem eine mobile Applikation für möglichst viele Plattformen umgesetzt werden kann. Das UI soll auf allen Plattformen einheitlich aussehen und es sind keine speziellen Smartphone-spezifischen Funktionen, die im mobilen Webbrowser nicht zur Verfügung stehen, notwendig. Die mobile Applikation soll so schnell wie möglich im Internet zur Verfügung stehen und Änderungen/Aktualisierungen sollen bei der Benutzerin/beim Benutzer sofort ersichtlich sein.

Für dieses Szenario empfiehlt sich die Realisierung einer mobilen Web Applikation mit dem Sencha Touch Framework aus folgenden Gründen:

- Es werden die Plattformen Android, BlackBerry und iOS unterstützt.
- Das UI funktioniert auf allen drei Plattformen einwandfrei, sieht einheitlich aus, lässt natives Feeling aufkommen, hat eine gute Performance und ist anpassbar.
- Von allen mobilen Webbrowsern der drei Plattformen werden typische Smartphone-Funktionalitäten, wie Geolocation, Karte und lokale Datenbank unterstützt.
- Die mobile Web Applikation benötigt keine Signierung und muss keinen Genehmigungsprozess durchlaufen, sondern kann nach Fertigstellung sofort im Internet abgerufen werden. Aktualisierungen sind jederzeit problemlos möglich.

Weitere Vorteile von Sencha Touch sind:

- Das Sencha Touch Framework ist Open Source.
- Für die Installation von Sencha Touch sind keine zusätzlichen SDKs und IDEs notwendig und für die Entwicklung wird nur ein Desktop Betriebssystem benötigt.
- Sencha Touch verfügt über eine gute Dokumentation und viele Beispiele.
- Die Entwicklerin/der Entwickler arbeitet an einer Quellcode-Basis, die plattformübergreifend eingesetzt werden kann. Dadurch ergibt sich eine Zeitersparnis und schnellere Entwicklung.
- Änderungen im Quellcode sind sofort ersichtlich, da Test der mobilen Web Applikation in Desktop Webbrowsern erfolgen kann.
- Die mobile Web Applikation kann mit Hilfe des PhoneGap Frameworks in eine hybride Applikation umgewandelt und über Application Stores vertrieben werden.

Die Nachteile von Sencha Touch sind:

- Die Einarbeitungszeit bedarf aufgrund der eigenen JavaScript Syntax länger.
- Es kann nicht auf alles Software- und Hardwarefunktionen eines Smartphones über den mobilen Webbrowser zugegriffen werden. Die Entwicklerin/der Entwickler sind von den derzeit angebotenen Funktionen der mobilen Webbrowser abhängig.
- Für manche Funktionen muss ein plattformspezifischer Workaround gefunden werden. Beispielsweise wird die offline Speicherung vom mobilen Webbrowser der BlackBerry Plattform nicht einwandfrei unterstützt.
- Konfigurationen der mobilen Web Applikation für eine nativ wirkende Applikation stehen derzeit nur von der iOS Plattform zur Verfügung.
- Derzeit gibt es noch wenig bekannte Stores für den Vertrieb von mobilen Web Applikationen.

Warum wird nicht das jQuery Mobile Framework empfohlen:

- Dass sich jQuery Mobile im Alpha Stadium befindet, kann am teilweise noch sehr unausgereiften UI erkannt werden. Das UI verursacht Darstellungsprobleme auf den drei Plattformen Android, BlackBerry und iOS und die Performance ist gegenüber Sencha Touch schlecht.
- Für den produktiven Einsatz ist jQuery Mobile noch nicht geeignet.

Die Vorteile von jQuery Mobile gegenüber Sencha Touch sind:

- Die Einarbeitungszeit für die Umsetzung des UI ist aufgrund der einfachen und klaren HTML Struktur gering.
- Gutes Zusammenspiel mit jQuery core, da jQuery Mobile darauf aufbaut.

11.2 Zweites Szenario

Die Entwicklerin/der Entwickler benötigt ein Framework, mit welchem eine mobile Applikation für möglichst viele Plattformen umgesetzt werden kann. Das UI soll auf allen Plattformen einheitlich aussehen und es soll der Zugriff auf Hardware- und Softwarefunktionen eines Smartphones möglich sein. Die mobile Applikation soll über Application Stores vertrieben werden können.

Für dieses Szenario empfiehlt sich die Realisierung einer hybriden Applikation mit dem PhoneGap Framework aus folgenden Gründen:

- Es werden die Plattformen Android, BlackBerry und iOS unterstützt.
- Ein UI Framework, wie Sencha Touch kann sehr leicht in PhoneGap eingebunden werden.
- Das PhoneGap Framework kann auf eine bestehende mobile Web Applikation aufbauen.
- PhoneGap ermöglicht den Zugriff auf verschiedene Software- und Hardwarefunktionen eines Smartphones, die im mobilen Webbrowser nicht zur Verfügung stehen.
- Mit PhoneGap realisierte hybride Applikationen können über die jeweiligen Application Stores vertrieben und verkauft werden.

Weitere Vorteile von PhoneGap sind:

- Das PhoneGap Framework ist Open Source.
- Die Einarbeitungszeit in die PhoneGap API ist sehr gering.
- PhoneGap verfügt über eine gute Dokumentation und Beispiele.
- Die hybride Applikation ist für Benutzerinnen/Benutzer auf dem Home-Bildschirm immer sichtbar.

Die Nachteile von PhoneGap sind:

- Für die Installation von PhoneGap werden die jeweiligen SDKs und IDEs der Plattformen benötigt. Dies erfordert einen hohen Zeitaufwand und Einarbeitungszeit.
- Die Entwicklung von hybriden Applikationen für die iOS Plattform setzt ein Mac OS X Betriebssystem voraus, für die BlackBerry Plattform ein Windows Betriebssystem.
- Jede Plattform bekommt von PhoneGap eine separate Projektvorlage mit eigener PhoneGap JavaScript Datei zur Verfügung gestellt. Änderungen im Quellcode müssen deshalb für jede Plattform nachgezogen werden.
- PhoneGap hat kein eigenes UI und PhoneGaps API verfügt über keinen Zugriff auf native UI Komponenten.
- Nicht alle Gerätefunktionen der PhoneGap API stehen für alle Plattformen einheitlich zur Verfügung. Deshalb können plattformspezifische Entwicklungen notwendig sein.
- Die Entwicklerin/der Entwickler ist von den vom PhoneGap Framework angebotenen Gerätefunktionen abhängig. Neue Hardware- und Softwarefunktionen werden nicht immer sofort in das Framework aufgenommen.
- Jede kleinste Änderung im Quellcode erfordert die Neukompilierung der hybriden Applikation.
- Der Test der hybriden Applikation ist nicht in Desktop Webbrowsern, sondern nur in Simulatoren/Emulatoren oder auf echten Smartphones möglich.
- Die Paketierung der hybriden Applikation für die Veröffentlichung in Application Stores muss mit den jeweiligen SDKs und IDEs durchgeführt werden. Mit PhoneGap Build könnte sich dies zukünftig ändern.
- Eine mit PhoneGap realisierte hybride Applikation steht nicht sofort für Benutzerinnen/Benutzer zur Verfügung. Der Vertrieb über die Application Stores ist langwierig, da dieser mit Signaturen und Genehmigungsprozessen verbunden ist und bis zur Veröffentlichung der Applikation ein paar Wochen vergehen können.
- Änderungen/Aktualisierung an der hybriden Applikation können nicht sofort bereit gestellt werden.

11.3 Drittes Szenario

Die Entwicklerin/der Entwickler benötigt ein Framework, mit welchem eine mobile Applikation mit nativem UI umgesetzt werden kann. Diese soll über verschiedene Application Stores vertrieben werden und auf verschiedene Hardware- und Softwarefunktionen zugreifen können.

Für dieses Szenario empfiehlt sich die Realisierung einer hybriden Applikation mit dem Titanium Mobile Framework aus folgenden Gründen:

- Titanium Mobile ermöglicht den Zugriff auf native UI Komponenten der Android und iOS Plattform. Dadurch ergibt sich eine gewohnte Usability für die Benutzerinnen/Benutzer.
- Titanium Mobile ermöglicht den Zugriff auf verschiedene Software- und Hardwarefunktionen eines Smartphones, die im mobilen Webbrowser nicht zur Verfügung stehen.
- Mit Titanium Mobile realisierte hybride Applikationen können über die jeweiligen Application Stores vertrieben und verkauft werden.

Weitere Vorteile von Titanium Mobile sind:

- Das Titanium Mobile Framework ist Open Source.
- Die Einarbeitungszeit in die Titanium Mobile API ist sehr gering.
- Titanium Mobile verfügt über eine gute Dokumentation und Beispiele.
- Die Entwicklerin/der Entwickler arbeitet an einer Quellcode-Basis, die plattformübergreifend eingesetzt werden kann. Dadurch ergibt sich eine Zeitersparnis und schnellere Entwicklung.
- Die Titanium Developer IDE übernimmt die zentrale Verwaltung der Applikationen. In dieser kann die hybride Applikation auf Simulatoren/Emulatoren und echten Smartphones getestet sowie die Paketierung für den Vertrieb vorgenommen werden.
- Die hybride Applikation ist für Benutzerinnen/Benutzer auf dem Home-Bildschirm immer sichtbar.

Die Nachteile von Titanium Mobile sind:

- Derzeit wird nur die Android und iOS Plattform unterstützt. BlackBerry Unterstützung wird es in absehbarer Zeit geben.

- Für die Installation von Titanium Mobile werden die jeweiligen SDKs der Plattformen benötigt.
- Die Entwicklung von hybriden Applikationen für die iOS Plattform setzt ein Mac OS X Betriebssystem voraus.
- Nicht alle nativen UI Komponenten werden von beiden Plattformen unterstützt.
- Titanium Mobile verfügt über kein einheitliches UI.
- Titanium Mobile kann nicht so einfach auf bestehende mobile Web Applikationen aufbauen. Die in einem WebView integrierte mobile Web Applikation funktioniert nur auf der iOS Plattform einwandfrei.
- Nicht alle Gerätefunktionen der Titanium Mobile API stehen für alle Plattformen einheitlich zur Verfügung. Deshalb können plattformspezifische Entwicklungen notwendig sein.
- Die Entwicklerin/der Entwickler ist von den vom Titanium Mobile Framework angebotenen Gerätefunktionen abhängig. Neue Hardware- und Softwarefunktionen werden nicht immer sofort in das Framework aufgenommen.
- Jede kleinste Änderung im Quellcode erfordert die Neukompilierung der hybriden Applikation.
- Der Test der hybriden Applikation ist nicht in Desktop Webbrowsern, sondern nur in Simulatoren/Emulatoren oder auf echten Smartphones möglich.
- Eine mit Titanium Mobile realisierte hybride Applikation steht nicht sofort für Benutzerinnen/Benutzer zur Verfügung. Der Vertrieb über die Application Stores ist langwierig, da dieser mit Signaturen und Genehmigungsprozessen verbunden ist und bis zur Veröffentlichung der Applikation ein paar Wochen vergehen können.
- Änderungen/Aktualisierung an der hybriden Applikation können nicht sofort bereitgestellt werden.

12 Zusammenfassung

Der systematische Vergleich von Frameworks für die plattformübergreifende Entwicklung mobiler Applikationen mit Web Technologien hat ergeben, dass Sencha Touch, PhoneGap und Titanium Mobile die Entwicklerinnen/Entwickler je nach Anforderungen an die mobile Applikation bei der Umsetzung unterstützen können, jedoch auch Nachteile mit sich bringen.

Das jQuery Mobile Framework wird aufgrund des derzeit noch sehr fehlerhaften UI für den produktiven Einsatz nicht empfohlen. Das Sencha Touch Framework zeichnet sich vorallem durch ein ausgereiftes, nativ wirkendes UI aus, das auf den Plattformen Android, BlackBerry und iOS erfolgreich eingesetzt werden kann. Sencha Touch eignet sich einerseits besonders für die rasche Umsetzung von mobilen Web Applikationen mit von mobilen Webbrowsern bereits integrierten Gerätefunktionen und andererseits als UI und Grundlage für die Weiterentwicklung mit dem PhoneGap Framework. PhoneGap wiederum zeichnet sich durch die Zugriffsmöglichkeit auf Smartphone-spezifische Funktionen und die Vertriebsmöglichkeit der mobilen Applikation über die verschiedenen Märkte der Plattformen aus. Die Realisierung der Beispielapplikation mit PhoneGap hat gezeigt, dass im Gegensatz zu Sencha Touch alle gerätespezifischen Kriterien für die Plattformen Android, BlackBerry und iOS umgesetzt werden konnten. Die Installation des Frameworks, der Test der mobilen Applikation, die Wartung des Quellcodes sowie die Paketierung der mobilen Applikation für den Vertrieb sind jedoch gegenüber Sencha Touch sehr aufwendig. Titanium Mobile ist besonders gut geeignet, wenn die mobile Applikation mit nativen UI Komponenten ausgestattet sein soll. Der große Nachteil ist, dass derzeit nur die Plattformen Android und iOS unterstützt werden.

Die Realisierung der Beispielapplikation mit der Unterstützung von Frameworks hat gezeigt, dass eine plattformübergreifende Umsetzung mit Web Technologien durchaus möglich ist und auch zukünftig der Fokus auf die Entwicklung von mobilen Applikationen basierend auf HTML5, CSS3 und W3C Spezifikationen liegen könnte.

Literaturverzeichnis

A1 Telekom Austria Presseabteilung (25.09.2002). mobilkom austria: Österreich startet mit dem ersten UMTS Netz Europas. Wien: A1 Telekom Austria AG. http://newsroom.a1telekom.at/2002/09/20020925_mobilkom-austria-oesterreich-startet-mit-dem-ersten-umts-netz-europas/ (Abruf: 01.12.2010)

A1 Telekom Austria Presseabteilung (15.04.2003). mobilkom austria Kunden nutzen als erste die dritte Mobilfunk-Generation in Österreich. Wien: A1 Telekom Austria AG. http://newsroom.a1telekom.at/2003/04/20030415_mobilkom-austria-kunden-nutzen-als-erste-die-dritte-mobilfunk-generation-in-oesterreich/ (Abruf: 01.12.2010)

Accenture (2010). Mobile Web Watch – Studie 2010. Accenture. http://www.accenture.com/de-de/Pages/insight-mobile-web-2010-summary.aspx (Abruf: 18.10.2010)

Access (2011). Access. Access Co., Ltd. http://www.access-company.com/ (Abruf: 01.03.2011)

Acid3 (2011). Acid3 Browser Test. The Web Standards Project. http://www.webstandards.org/action/acid3/ (Abruf: 05.03.2011)

AdMob (30.06.2010). AdMob Mobile Metrics Highlights May 2010. AdMob. http://metrics.admob.com/ (Abruf: 12.10.2010)

Alby, Tom (2008). Das mobile Web. München: Carl Hanser Verlag.

Allen, Sarah & Graupera, Vidal & Lundrigan, Lee (2010). Pro Smartphone Cross-Platform Development. New York, USA: Apress.

Android Developers (22.10.2008). Android Market: Now available for users. California, USA: Google Inc. http://android-developers.blogspot.com/2008/10/android-market-now-available-for-users.html (Abruf: 03.03.2011)

Android Developers (06.12.2010). Android 2.3 Platform and Updated SDK Tools. California, USA: Google Inc. http://android-developers.blogspot.com/2010/12/android-23-platform-and-updated-sdk.html (Abruf: 02.03.2011)

Android Developers (09.02.2011, 2011a). Android 2.3.3 Platform, New NFC Capabilities. California, USA: Google Inc. http://android-developers.blogspot.com/search/label/Android%202.3.3 (Abruf: 02.03.2011)

Android Developers (2011b). Installaing the SDK. California, USA: Google Inc. http://developer.android.com/sdk/installing.html (Abruf: 02.03.2011)

Android Developers (2011c). Download the Android SDK. California, USA: Google Inc. http://developer.android.com/sdk/index.html (Abruf: 02.03.2011)

Android Developers (2011d). Installing the ADT Plugin. California, USA: Google Inc. http://developer.android.com/sdk/eclipse-adt.html#installing (Abruf: 02.03.2011)

Android Developers (2011e). Building Web Apps in WebView. California, USA: Google Inc. http://developer.android.com/guide/webapps/webview.html (Abruf: 02.03.2011)

Android Developers (2011f). Android 2.3 Platform Highlights. California, USA: Google Inc. http://developer.android.com/sdk/android-2.3-highlights.html (Abruf: 03.03.2011)

Android Market (2011). Android Market. California, USA: Google Inc. https://market.android.com/ (Abruf: 03.03.2011)

AndroLib (2011). Statistics: Accumulated number of Application and Games in the Android Market. AndroLib.com. http://www.androlib.com/appstats.aspx (Abruf: 03.03.2011)

Anygraaf (2011). Anygraaf. Anygraaf USA Inc. http://www.anygraaf.com/anyusa/index.htm (Abruf: 01.03.2011)

Apache Ant (2011). Apache Ant. The Apache Software Foundation. http://ant.apache.org/bindownload.cgi (Abruf: 17.03.2011)

Appcelerator (2011). Download Titanium. Mountain View, CA, USA: Appcelerator Inc. http://www.appcelerator.com/products/download/ (23.03.2011)

Appcelerator Mobile API (2011a). Titanium.Media.showCamera. Mountain View, CA, USA: Appcelerator Inc. http://developer.appcelerator.com/apidoc/mobile/latest/Titanium.Media.showCamera-method.html (Abruf: 06.05.2011)

Appcelerator Mobile API (2011b). Titanium.UI.AlertDialog. Mountain View, CA, USA: Appcelerator Inc. http://developer.appcelerator.com/apidoc/mobile/latest/Titanium.UI.AlertDialog-object (Abruf: 06.05.2011)

Appcelerator Mobile API (2011c). Titanium.UI.OptionDialog. Mountain View, CA, USA: Appcelerator Inc. http://developer.appcelerator.com/apidoc/mobile/latest/Titanium.UI.OptionDialog-object (Abruf: 06.05.2011)

Appcelerator Mobile API (2011d). Titanium.UI. Mountain View, CA, USA: Appcelerator Inc. http://developer.appcelerator.com/apidoc/mobile/latest/Titanium.UI-module (Abruf: 06.05.2011)

Appcelerator Mobile API (2011e). Titanium.UI.Button. Mountain View, CA, USA: Appcelerator Inc. http://developer.appcelerator.com/apidoc/mobile/latest/Titanium.UI.Button-object (Abruf: 06.05.2011)

Appcelerator Mobile API (2011f). Titanium.UI.Window. Mountain View, CA, USA: Appcelerator Inc. http://developer.appcelerator.com/apidoc/mobile/latest/Titanium.UI.Window-object (Abruf: 06.05.2011)

Appcelerator Wiki (2011a). Getting Started with Titanium. Mountain View, CA, USA: Appcelerator Inc. http://wiki.appcelerator.org/display/guides/Getting+Started+with+Titanium (Abruf: 11.03.2011)

Appcelerator Wiki (2011b). The Application Project Structure. Mountain View, CA, USA: Appcelerator Inc. http://wiki.appcelerator.org/display/guides/The+Application+Project+Structure (Abruf: 11.03.2011)

Appcelerator Wiki (2011c). The Titanium Architecture. Mountain View, CA, USA: Appcelerator Inc. http://wiki.appcelerator.org/display/guides/The+Titanium+Architecture (Abruf: 11.03.2011)

Appcelerator Wiki (2011d). Designing the User Interface. Mountain View, CA, USA: Appcelerator Inc. http://wiki.appcelerator.org/display/guides/Designing+the+User+Interface (Abruf: 11.03.2011)

Appcelerator Wiki (2011e). Using Location services. Mountain View, CA, USA: Appcelerator Inc. http://wiki.appcelerator.org/display/guides/Using+Location+services (Abruf: 11.03.2011)

Apple (09.06.2008). Apple Introduces the New iPhone 3G. Cupertino, California, USA: Apple Inc. http://www.apple.com/pr/library/2008/06/09iphone.html (Abruf: 04.03.2011)

Apple (2011a). iPhone 3GS: Technische Daten. Cupertino, California, USA: Apple Inc. http://www.apple.com/de/iphone/iphone-3gs/specs.html (Abruf: 04.03.2011)

Apple (2011b). iOS 4. Cupertino, California, USA: Apple Inc. http://www.apple.com/de/iphone/ios4/ (Abruf: 04.03.2011)

Apple (2011c). iPhone: Multitasking. Cupertino, California, USA: Apple Inc. http://www.apple.com/de/iphone/features/multitasking.html (Abruf: 04.03.2011)

Apple (2011d). iPhone: Ordner. Cupertino, California, USA: Apple Inc. http://www.apple.com/de/iphone/features/folders.html (Abruf: 04.03.2011)

Apple (2011e). iOS 4.3 Software Update. Cupertino, California, USA: Apple Inc. http://www.apple.com/de/ios/ (Abruf: 04.03.2011)

Apple (2011f). Web Apps. Cupertino, California, USA: Apple Inc. http://www.apple.com/webapps/ (Abruf: 04.03.2011)

Apple (22.01.2011, 2011g). Apple's App Store Downloads Top 10 Billion. Cupertino, California, USA: Apple Inc. http://www.apple.com/pr/library/2011/01/22appstore.html (Abruf: 04.03.2011)

Apple Developer (2011a). iOS Dev Center. Cupertino, California, USA: Apple Inc. http://developer.apple.com/devcenter/ios/index.action (Abruf: 23.02.2011)

Apple Developer (2011b). iOS Developer Program. Cupertino, California, USA: Apple Inc. http://developer.apple.com/programs/ios/ (Abruf: 23.02.2011)

Apple Developer (2011c). iOS Developer Enterprise Program. Cupertino, California, USA: Apple Inc. http://developer.apple.com/programs/ios/enterprise/ (Abruf: 23.02.2011)

Apple Developer (2011d). UIWebView Class Reference. Cupertino, California, USA: Apple Inc.

http://developer.apple.com/library/ios/#documentation/uikit/reference/UIWebView_Class/Reference/Reference.html (Abruf: 04.03.2011)

Apple Developer (2011e). Configuring Web Applications. Cupertino, California, USA: Apple Inc. http://developer.apple.com/library/safari/#documentation/appleapplications/reference/safariwebcontent/ConfiguringWebApplications/ConfiguringWebApplications.html (Abruf: 17.03.2011)

AppRejections (2011). App Rejections: Send app-rejection news to @redglassesapps on twitter. http://apprejections.com/ (Abruf: 04.03.2011)

AppStoreHQ (2010). AppStoreHQ. MobilMeme Inc. http://web.appstorehq.com/search/results?crumb[platform]=Web&crumb_order=platform (Abruf: 17.03.2011)

Aptana Studio (2011). Download Aptana Studio 2.0.5. Aptana, Inc. http://www.aptana.org/products/studio2/download (Abruf: 17.03.2011)

Aurelio, David (2010). Plattformübergreifende Apps mit Web-Technologien entwickeln. t3n Magazin Nr. 22. Hannover: yeebase media GbR.

Beccue, Mark (06.05.2010). Smartphone Downloads from Mobile App Stores to Peak in 2013. New York: Allied Business Intelligence, Inc. http://www.abiresearch.com/press/3417-Smartphone+Downloads+from+Mobile+App+Stores+to+Peak+in+2013 (Abruf: 07.03.2011)

BlackBerry App World (2011). BlackBerry App World. Waterloo, Ontario, Canada: Research In Motion Limited. https://appworld.blackberry.com/webstore/ (Abruf: 05.04.2011)

BlackBerry Developers (2011a). BlackBerry Java Development Environment. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/developers/javaappdev/javadevenv.jsp (Abruf: 05.04.2011)

BlackBerry Developers (2011b). BlackBerry Java Plug-in for Eclipse. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/developers/javaappdev/javaplugin.jsp (Abruf: 05.04.2011)

BlackBerry Developers (2011c). Class BrowserField. Waterloo, Ontario, Canada: Research In Motion Limited. http://www.blackberry.com/developers/docs/6.0.0api/net/rim/device/api/browser/field2/BrowserField.html (Abruf: 05.04.2011)

BlackBerry Developers (2011d). BlackBerry WebWorks Development. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/developers/browserdev/ (Abruf: 05.04.2011)

BlackBerry Developers (2011e). BlackBerry WebWorks Advanced Features. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/developers/browserdev/advfeatures.jsp (Abruf: 05.04.2011)

BlackBerry Developers (2011f). Code Signing Keys. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/developers/javaappdev/codekeys.jsp (Abruf: 05.04.2011)

BlackBerry Developers (2011g). BlackBerry App World Distribution. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/developers/appworld/distribution.jsp (Abruf: 05.04.2011)

BlackBerry Developers (2011h). Tools & Downloads. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/developers/browserdev/devtoolsdownloads.jsp (Abruf: 17.03.2011)

BlackBerry Developers (2011i). BlackBerry WebWorks SDK. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/developers/browserdev/widgetsdk.jsp (Abruf: 17.03.2011)

BlackBerry Documentation (2010a). Turn on geolocation in the browser. Waterloo, Ontario, Canada: Research In Motion Limited. http://docs.blackberry.com/en/smartphone_users/deliverables/18577/Turn_on_geolocation_in_the_browser_60_1072867_11.jsp (Abruf: 13.02.2011)

BlackBerry Documentation (2010b). Supported web standards and technologies. Waterloo, Ontario, Canada: Research In Motion Limited. http://docs.blackberry.com/en/developers/deliverables/18169/Standards_support_in_60_browser_1120158_11.jsp (Abruf: 17.02.2011)

BlackBerry Documentation (2010c). BlackBerry Java Development Environment. Waterloo, Ontario, Canada: Research In Motion Limited. http://docs.blackberry.com/de-de/developers/deliverables/9976/BlackBerry_JDE_446979_11.jsp (Abruf: 05.03.2011)

BlackBerry Documentation (2010d). Feature and Technical Overview - BlackBerry Browser - 6.0. Waterloo, Ontario, Canada: Research In Motion Limited. http://docs.blackberry.com/de-de/developers/deliverables/18169/Overview_BB_Browser_with_WebKit_1146916_11.jsp (Abruf: 05.03.2011)

BlackBerry Software (2011a). BlackBerry Enterprise Server. Waterloo, Ontario, Canada: Research In Motion Limited. http://us.blackberry.com/apps-software/business/server/full/ (Abruf: 05.04.2011)

BlackBerry Software (2011b). Download BlackBerry App World. Waterloo, Ontario, Canada: Research In Motion Limited. http://de.blackberry.com/services/appworld/download.jsp (Abruf: 05.03.2011)

Brauner, Roman (17.08.2010). Web App oder besser HTML5 App?. http://romanbrauner.wordpress.com/2010/08/17/web-app-oder-besser-html5%C2%A0app/ (Abruf: 17.03.2011)

Çelik, Tantek & Etemad, Elika J. & Glazman, Daniel & Hickson, Ian & Linss, Peter & Williams, John (15.12.2009). CSS3: Selectors Level 3. W3C. http://www.w3.org/TR/2009/PR-css3-selectors-20091215/ (Abruf: 05.03.2011)
Constantinou, Andreas (23.11.2010). Apps is the new Web: sowing the seeds for Web 3.0. London, UK: VisionMobile Limited. http://www.visionmobile.com/blog/2010/11/apps-is-the-new-web-sowing-the-seeds-for-web-3-0/ (Abruf: 17.03.2011)

CSS3 Selectors Test (2011). CSS3 Selectors Test. WEBFLUX. http://www.css3.info/selectors-test/ (Abruf: 05.03.2011)

Delbrouck, Dirk (21.01.2003). ZDNet.de: Reqwireless: HTML-Browser für Handys. CBS Interactive GmbH. http://www.zdnet.de/news/wirtschaft_telekommunikation_reqwireless_html_browser_fuer_handys_story-39001023-2129046-1.htm (Abruf: 01.03.2011)

Department of Defense (DoD) World Geodetic System 1984 (WGS 84) (03.01.2000). Its Definition and Relationships with Local Geodetic Systems. National Imagery and Mapping Agency (NIMA) Technical Report 8350.2, Third Edition. St. Louis, MO: NIMA. http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf (Abruf: 14.02.2011)

Dern, Daniel (2010). Cross-Platform Smartphone Apps Still Difficult. New York, USA: IEEE Spectrum. http://spectrum.ieee.org/geek-life/tools-toys/crossplatform-smartphone-apps-still-difficult/0 (Abruf: 17.03.2011)

Dziemba, Oliver & Horx, Matthias & Kirig, Anja & Mijnals, Patrick (2009). Trend-Report 2010. Soziokulturelle Schlüsseltrends für die Märkte von morgen. Kelkheim, Deutschland: Zikunftsinstitut GmbH.

Duwe, Colin (20.11.2002). ZDNet.de: RIM BlackBerry 5810 Wireless Handheld im Test. CBS Interactive GmbH. http://www.zdnet.de/mobiles_arbeiten_mit_handheld_pda_handy_smartphone_rim_blackberry_5810_wireless_handheld_reviewstory-20000152-20001224-1.htm (Abruf: 05.04.2011)

Eclipse (2011). Eclipse Downloads. The Eclipse Foundation. http://www.eclipse.org/downloads/ (Abru: 12.03.2011)

Eddy, Nathan (16.02.2009). eWeek. MWC: Nokia Announces 'Smart Store' Ovi. New York, USA: Ziff Davis Enterprise Holdings Inc. http://www.eweek.com/c/a/Mobile-and-Wireless/MWC-Nokia-Announces-Smart-Store-Ovi/ (Abruf: 03.03.2011)

eMarketer (27.10.2010). Mobile Users Prefer Browsers over Apps. eMarketer Inc. http://www.emarketer.com/Article.aspx?R=1008010&AspxAutoDetectCookieSupport=1 (Abruf: 08.03.2011)

Fakhre, Alam (14.12.2010). Nokia planned New UI for Symbian, Dual-Core phones, Upgradeable HTML5 Browser in 2011. **Symbian Tweet.** http://www.symbiantweet.com/nokia-planned-new-ui-for-symbian-dual-core-phones-upgradeable-html5-browser-in-2011 (Abruf: 01.03.2011)

Firtman, Maximiliano (2010). Programming the Mobile Web. Sebastopol, California: O'Reilly Media, Inc.

Fittkau & Maaß Consulting (26.07.2010). Das mobile Internet. 1. April bis 9. Mai 2010, 30. Erhebungswelle der W3B-Studie. Hamburg: Fittkau & Maaß Consulting GmbH. http://www.w3b.org/technik/das-mobile-internet-fuer-die-klasse-nicht-fuer-die-masse.html (Abruf: 11.10.2010)
Fitzek, Frank H.P. & Mikkonen Tommi & Torp Tony (2010). Qt for Symbian. Chichester, UK: John Wiley & Sons, Ltd.

Fletcher, Nik (17.10.2007). Apple: "we plan to have an iPhone SDK in developers' hands in February". AOL Inc. http://www.tuaw.com/2007/10/17/apple-we-plan-to-have-an-iphone-sdk-in-developers-hands-in-fe/ (Abruf: 04.03.2011)

Forum Nokia (2011a). Qt. Espoo, Finnland: Nokia. http://www.forum.nokia.com/Develop/Qt/ (Abruf: 01.03.2011)

Forum Nokia (2011b). Nokia N8-00. Espoo, Finnland: Nokia. http://www.forum.nokia.com/Devices/Device_specifications/N8-00/ (Abruf: 01.03.2011)

Forum Nokia (12.04.2011, 2011c). Ovi Store statistics. Espoo, Finnland: Nokia. http://www.forum.nokia.com/Distribute/Ovi_Store_statistics.xhtml (Abruf: 12.04.2011)

Forum Nokia (2011d). Remote Device Access. Espoo, Finnland: Nokia. http://www.forum.nokia.com/Devices/Remote_device_access/ (Abruf: 17.03.2011)

Gabriel, Hermann (19.10.2010). A1 Telekom Austria startet LTE in der Bundeshauptstadt Wien: A1 Telekom Austria Generaldirektor Hannes Ametsreiter übergibt Bürgermeister Michael Häupl den ersten LTE Datenstick. Wien: A1 Telekom Austria AG. http://newsroom.a1telekom.at/2010/10/a1-telekom-austria-startet-lte-in-der-bundeshauptstadt-wien-a1-telekom-austria-generaldirektor-hannes-ametsreiter-ubergibt-burgermeister-michael-haupl-den-ersten-lte-datenstick/ (Abruf: 01.12.2010)

Gartner (13.01.2010, 2010a). Gartner Highlights Key Predictions for IT Organizations and Users in 2010 and Beyond. Stamford, Connecticut: Gartner, Inc. http://www.gartner.com/it/page.jsp?id=1278413 (Abruf: 11.10.2010)

Gartner (12.08.2010, 2010b). Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down. Egham, UK: Gartner, Inc. http://www.gartner.com/it/page.jsp?id=1421013 (Abruf: 11.10.2010)

Gartner (09.02.2011, 2011a). Gartner Says Worldwide Mobile Application Store Revenue Forecast to Surpass $15 Billion in 2011. Egham, UK: Gartner, Inc. http://www.gartner.com/it/page.jsp?id=1529214 (Abruf: 01.03.2011)

Gartner (26.01.2011, 2011b). Gartner Says Worldwide Mobile Device Sales to End Users Reached 1.6 Billion Units in 2010; Smartphone Sales Grew 72 Percent in 2010. Egham, UK: Gartner, Inc. http://www.gartner.com/it/page.jsp?id=1543014 (Abruf: 01.03.2011)

Global Intelligence Alliance (2010). Native or Web Application? How Best to Deliver Content and Services to Your Audiences over the Mobile Phone. Helsinki, Finnland: Global Intelligence Alliance. http://www.globalintelligence.com/insights-analysis/white-papers/native-or-web-application-how-best-to-deliver-cont (Abruf: 17.03.2011)

Glyphish (2010). Glyphish. Create icons for mobile Apps. http://glyphish.com/ (Abruf: 30.01.2011)

Gonsalves, Antone (11.10.2007). InformationWeek: Apple Launches iPhone Web Apps Directory. New York, USA: United Business Media LLC. http://www.informationweek.com/news/hardware/mac/showArticle.jhtml?articleID=202401732 (Abruf: 04.03.2011)

Google (2011). Mobile Bookmark Bubble. Google Inc. http://code.google.com/p/mobile-bookmark-bubble/ (Abruf: 17.03.2011)

Google Maps (2011a). Google Maps JavaScript API V3: Willkommen. Google. http://code.google.com/intl/de-DE/apis/maps/documentation/javascript/basics.html#Welcome (Abruf: 14.02.2011)

Google Maps (2011b). Google Maps JavaScript API V3: Angeben des Sensorparameters. Google. http://code.google.com/intl/de-DE/apis/maps/documentation/javascript/basics.html#SpecifyingSensor (Abruf: 14.02.2011)

Google Maps (2011c). Google Maps JavaScript API V3 – Tutorial. Google. http://code.google.com/intl/de-DE/apis/maps/documentation/javascript/tutorial.html (Abruf: 15.02.2011)

Google Maps (2011d). Google Maps JavaScript API V3 – Dienste: Geocoding. Google. http://code.google.com/intl/de-DE/apis/maps/documentation/javascript/services.html#Geocoding (Abruf: 15.02.2011)

Hausmann, Roger (2010). Die Welt der Apps ist grösser als iPhone & Co. E-Shopping Ausgabe 2010. Zürich: Anthrazit AG. http://www.anthrazit.org/index.php?apsid=25f67554&apid=1077587179&apparentid=872197826 (Abruf: 16.10.2010)

Hazaël-Massieux, Dominique (13.08.2010). What is a Web-based mobile application or Web app? Here's expert opinion from the W3C. dotMobi. http://mobithinking.com/blog/what-is-a-Web-app (Abruf: 08.03.2011)

Hence, Stefan (06.05.2010). In der App-Entwicklung ist weniger mehr. Ausgabe 19/10. Computerwoche. http://www.computerwoche.de/netzwerke/mobile-wireless/1931422/ (Abruf: 27.10.2010)

Hickson, Ian (18.11.2010, 2010a). Web SQL Database. W3C. http://www.w3.org/TR/2010/NOTE-webdatabase-20101118/ (Abruf: 17.02.2011)

Hickson, Ian (18.11.2010, 2010b). Web SQL Database: Errors and exceptions. W3C. http://www.w3.org/TR/2010/NOTE-webdatabase-20101118/#sqlerror (Abruf: 17.02.2011)

Hickson, Ian (18.11.2010, 2010c). Web SQL Database: Asynchronous database API. W3C. http://www.w3.org/TR/2010/NOTE-webdatabase-20101118/#asynchronous-database-api (Abruf: 17.02.2011)

Hickson, Ian (2011a). HTML5: Embedding custom non-visible data with the data-* attributes. W3C. http://dev.w3.org/html5/spec/Overview.html#embedding-custom-non-visible-data-with-the-data-attributes (Abruf: 18.01.2011)

Hickson, Ian (2011b). HTML5: Offline Web applications: Introduction. W3C. http://dev.w3.org/html5/spec/offline.html#introduction-3 (Abruf: 15.02.2011)

Hickson, Ian (2011c). HTML5: Offline Web applications: Application Cache API. W3C. http://dev.w3.org/html5/spec/offline.html#application-cache-api (Abruf: 16.02.2011)

Hickson, Ian (2011d). HTML5: Offline Web applications: Event summary. W3C. http://dev.w3.org/html5/spec/offline.html#appcacheevents (Abruf: 16.02.2011)

Honan, Mathew (09.01.2007). Apple unveils iPhone. San Francisco, USA: Mac Publishing LLC. http://www.macworld.com/article/54769/2007/01/iphone.html (Abruf: 04.03.2011)

HTC (2011a). T-Mobile G1: Overview. HTC. http://www.htc.com/www/product/g1/overview.html (Abruf: 02.03.2011)

HTC (2011b). Nexus One: Photo Gallery. HTC. http://www.htc.com/www/product/nexusone/gallery.html (Abruf: 03.03.2011)

Ihlenfeld, Jens (15.06.2010). Ext JS + jQTouch + Raphaël = Sencha. Berlin: Golem.de, Klaß & Ihlenfeld Verlag GmbH. http://www.golem.de/1006/75792.html (Abruf: 01.04.2011)

Jacobs, Mike (17.03.2011). Living on the Edge of Mobile Development. Ulitzer, Inc. http://java.sys-con.com/node/1719019 (Abruf: 06.05.2011)

Jobs, Steve (14.06.2007). PC World: Safari: iPhone Development Platform. San Bruno, California, USA: YouTube LLC. http://www.youtube.com/watch?v=8Vq993Td6ys (Abruf: 04.03.2011)

Johnson, Dave (19.08.2010). PhoneGap Native Bridge. Null is not an Object. http://nullisnotanobject.com/2010/08/phonegap-native-bridge/ (Abruf: 17.03.2011)

jQuery (2010). jQuery. jQuery Project. http://jquery.com/ (Abruf: 17.03.2011)

jQuery Mobile (2010a). jQuery Mobile Framework. jQuery Project. http://jquerymobile.com/ (Abruf: 21.04.2011)

jQuery Mobile (2010b). Anatomy of a Page. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/pages/docs-pages.html (Abruf: 21.04.2011)

jQuery Mobile (2010c). Header bars. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/toolbars/docs-headers.html (Abruf: 21.04.2011)

jQuery Mobile (2010d). Footer configuration. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/toolbars/docs-footers.html (Abruf: 21.04.2011)

jQuery Mobile (2010e). Fixed toolbars. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/toolbars/bars-fixed.html (Abruf: 21.04.2011)

jQuery Mobile (2010f). Fullscreen fixed header. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/toolbars/bars-fullscreen.html (Abruf: 21.04.2011)

jQuery Mobile (2010g). Theming pages. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/pages/pages-themes.html (Abruf: 21.04.2011**)**

jQuery Mobile (2010h). Configuring Defaults. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/api/globalconfig.html (Abruf: 21.04.2011**)**

jQuery Mobile (2010i). Button icons. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/toolbars/../buttons/buttons-icons.html (Abruf: 21.04.2011)

jQuery Mobile (2010j). Navbar. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/toolbars/docs-navbar.html (Abruf: 21.04.2011)

jQuery Mobile (2010k). Transitions. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/pages/docs-transitions.html (Abruf: 21.04.2011)

jQuery Mobile (2010l). Dialogs. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/pages/docs-dialogs.html (Abruf: 21.04.2011)

jQuery Mobile (2010m). Link formats. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/pages/link-formats.html (Abruf: 21.04.2011)

jQuery Mobile (2010n). Button markup options. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/buttons/buttons-types.html (Abruf: 21.04.2011)

jQuery Mobile (2010o). Inline buttons. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/buttons/buttons-inline.html (Abruf: 21.04.2011)

jQuery Mobile (2010p). Grouped buttons. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/buttons/buttons-grouped.html (Abruf: 21.04.2011)

jQuery Mobile (2010q). HTML Formatting. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/content/content-html.html (Abruf: 21.04.2011)

jQuery Mobile (2010r). Collapsible content. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/content/content-collapsible.html (Abruf: 21.04.2011)

jQuery Mobile (2010s). Layout grids. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/content/content-grids.html (Abruf: 21.04.2011)

jQuery Mobile (2010t). Forms. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/forms/docs-forms.html (Abruf: 21.04.2011)

jQuery Mobile (2010u). Forms. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/forms/forms-all.html (Abruf: 21.04.2011)

jQuery Mobile (2010v). Text inputs. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/forms/forms-text.html (Abruf: 21.04.2011)

jQuery Mobile (2010w). Lists. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/lists/docs-lists.html (Abruf: 21.04.2011)

jQuery Mobile (2010x). Events. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/api/events.html (Abruf: 21.04.2011)

jQuery Mobile (2010y). Download. jQuery Project. http://jquerymobile.com/download/ (Abruf: 21.04.2011)

jQuery Mobile (2010z). Supported platforms. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/about/platforms.html (Abruf: 21.04.2011)

jQuery Mobile (2010aa). Accessibility. jQuery Project. http://jquerymobile.com/demos/1.0a4.1/#docs/about/accessibility.html (Abruf: 21.04.2011)

Kaikkonen, Anne (2008). Full or Tailored Mobile Web- Where and How do People Browse on Their Mobiles? Mobility '08: Proceedings of the International Conference on Mobile Technology, Applications, and Systems. New York, USA: ACM. http://portal.acm.org/citation.cfm?doid=1506270.1506307 (Abruf: 21.11.2010)

Kincaid, Jason (07.07.2010). YouTube Mobile Goes HTML5, Video Quality Beats Native Apps Hands Down. TechCrunch. http://techcrunch.com/2010/07/07/youtube-iphone-mobile-html5/ (Abruf: 08.03.2011)

Koch, Peter-Paul (20.04.2009). Introduction to W3C Widgets. Amsterdam, Niederlande: quirksmode. http://www.quirksmode.org/blog/archives/2009/04/introduction_to.html (Abruf: 17.03.2011)

Kröner, Peter (2010). HTML5: Webseiten innovativ und zukunftssicher. München: Open Source Press.

Leenheer, Niels (2010). HTML5 TEST. http://html5test.com/ (Abruf: 17.02.2011)

Marketwire (14.02.2011). BlackBerry App World Now Available In Over 100 Markets. Marketwire, Inc. http://www.marketwire.com/press-release/BlackBerry-App-World-Now-Available-In-Over-100-Markets-NASDAQ-RIMM-1394866.htm (Abruf: 7.03.2011)

MeeGo (2011). About. http://meego.com/about (Abruf: 01.03.2011)

Mehta, Nikunj & Sicking Jonas & Graf, Eliot & Popescu, Andrei & Orlow, Jeremy (2011). Indexed Database API. W3C. http://dvcs.w3.org/hg/IndexedDB/raw-file/tip/Overview.html (Abruf: 17.02.2011)

Mittwich, Michael (01.07.2009). Blogger-Aktion: Zeigt her eure Handys! Early Adopter. http://www.early-adopter.info/2009/07/blogger-aktion-zeigt-her-eure-handys/ (Abruf: 04.03.2011)

mobiThinking (13.08.2010). What is a Web-based mobile application or Web app? Here's expert opinion from the W3C. dotMobi. http://mobithinking.com/blog/what-is-a-Web-app (Abruf: 08.03.2011)

Morris, Ben (2007). The Symbian OS Architecture Sourcebook: Design and Evolution of a Mobile Phone OS. Chichester, UK: John Wiley & Sons, Ltd.

Morrow, Fergus (2010). PhoneGap – The Internals: Part 1. "phonegap.js" laid bare. http://www.scribd.com/doc/34401287/PhoneGap-Internals-Part-1 (Abruf: 25.02.2010)

Motl, Matthias (13.10.2010). Re: Studie "Das mobile Internet". [Persönliche E-mail].

Neil, Drew (2010a). Sencha Toch – Intro to Panels. New York, USA: Vimeo, LLC. http://vimeo.com/15879797 (Abruf: 01.04.2011)

Neil, Drew (2010b). Sencha Touch – Into to Layouts. New York, USA: Vimeo, LLC. http://vimeo.com/15888504 (Abruf: 01.04.2011)

Neil, Drew (2010c). Sencha Touch – Into to Listeners. New York, USA: Vimeo, LLC. http://vimeo.com/17414405 (Abruf: 01.04.2011)

Neil, Drew (2011a). Sencha Touch – Tabs and Toolbars. New York, USA: Vimeo, LLC. http://vimeo.com/22251762 (Abruf: 01.04.2011)

Neil, Drew (2011b). Sencha Touch – Intro to List Component. New York, USA: Vimeo, LLC. http://vimeo.com/19245335 (Abruf: 01.04.2011)

Nielsen (13.09.2010). The State Of Mobile Apps. The Nielsen Company. http://blog.nielsen.com/nielsenwire/media_entertainment/insights-on-the-emerging-mobile-app-economy/ (11.10.2010)

Nokia (24.06.2008). Mobile leaders to unify the Symbian software platform and set the future of mobile free. Espoo, Finnland: Nokia. http://press.nokia.com/2008/06/24/mobile-leaders-to-unify-the-symbian-software-platform-and-set-the-future-of-mobile-free/ (Abruf: 10.12.2010)

Nokia (08.11.2010, 2010a). Nokia reaffirms commitment to Symbian platform. Espoo, Finnland: Nokia. http://press.nokia.com/2010/11/08/nokia-reaffirms-commitment-to-symbian-platform-2/ (Abruf: 08.12.2010)

Nokia (21.10.2010, 2010b). Nokia further refines development strategy to unify environments for Symbian and MeeGo. Espoo, Finnland: Nokia. http://press.nokia.com/2010/10/21/nokia-further-refines-development-strategy-to-unify-environments-for-symbian-and-meego/ (Abruf: 28.02.2011)

Oksanen, Ilkka & Hazaël-Massieux (28.09.2010). HTML Media Capture. W3C. http://www.w3.org/TR/2010/WD-html-media-capture-20100928/ (Abruf: 17.03.2011)

Open Handset Alliance (2011). Members. California, USA: Google Inc. http://www.openhandsetalliance.com/oha_members.html (Abruf: 02.03.2011)

OpenAppMkt (2010). OpenAppMkt. San Francisco, CA, USA: OpenAppMkt. http://openappmkt.com/ (Abruf: 17.03.2011)

Opera (2011). Opera. Opera Software ASA. http://www.opera.com/ (Abruf: 01.03.2011)

Oracle (2011). Java SE Downloads. Redwood Shores, CA, USA: Oracle Corporation. http://www.oracle.com/technetwork/java/javase/downloads/index.html#jdk (Abruf: 17.03.2011)

Ovi Store (2011). Nokia Ovi Store. Espoo, Finnland: Nokia. http://store.ovi.com/ (Abruf: 03.03.2011)

Pappas, Lisa & Schwerdtfeger, Rich & Seeman, Lisa (18.01.2011). Accessible Rich Internet Applications (WAI-ARIA) 1.0. W3C. http://www.w3.org/TR/2011/CR-wai-aria-20110118/ (Abruf: 17.03.2011)

Pearce, James (24.01.2011). Tutorial:A Sencha Touch MVC application with PhoneGap. Redwood City, CA, USA: Sencha. http://www.sencha.com/learn/Tutorial:A_Sencha_Touch_MVC_application_with_PhoneGap (Abruf: 25.02.2011)

Pederick, Chris (2011). User Agent Switcher. http://chrispederick.com/work/user-agent-switcher/ (Abruf: 17.03.2011)

PhoneGap (2011a). About. Vancouver BC, Canada: Nitobi Software. http://www.phonegap.com/about/ (Abruf: 23.02.2011)

PhoneGap (2011b). PhoneGap. Vancouver BC, Canada: Nitobi Software. http://www.phonegap.com/ (Abruf: 23.02.2011)

PhoneGap (2011c). Get Started: iOS. Vancouver BC, Canada: Nitobi Software. http://www.phonegap.com/start#ios (Abruf: 23.02.2011)

PhoneGap (2011d). Supported Features. Vancouver BC, Canada: Nitobi. http://www.phonegap.com/features (Abruf: 26.02.2011)

PhoneGap (2011e). Get Started: Android. Vancouver BC, Canada: Nitobi Software. http://www.phonegap.com/start#android (Abruf: 13.03.2011)

PhoneGap (2011f). Get Started: Symbian WRT. Vancouver BC, Canada: Nitobi Software. http://www.phonegap.com/start#symbian (Abruf: 17.03.2011)

PhoneGap (2011g). Get Started: BlackBerry. Vancouver BC, Canada: Nitobi Software. http://www.phonegap.com/start#bb (Abruf: 17.03.2011)

PhoneGap Build (2011). PhoneGap Build. https://build.phonegap.com/

PhoneGap Documentation (2011a). Geolocation. Vancouver BC, Canada: Nitobi Software. http://docs.phonegap.com/phonegap_geolocation_geolocation.md.html (Abruf: 26.02.2011)

PhoneGap Documentation (2011b). Storage. Vancouver BC, Canada: Nitobi Software. http://docs.phonegap.com/phonegap_storage_storage.md.html (Abruf: 26.02.2011)

PhoneGap Documentation (2011c). Camera. Vancouver BC, Canada: Nitobi Software. http://docs.phonegap.com/phonegap_camera_camera.md.html (Abruf: 26.02.2011)

PhoneGap Documentation (2011d). deviceready. Vancouver BC, Canada: Nitobi Software. http://docs.phonegap.com/phonegap_events_events.md.html (Abruf: 26.02.2011)

Pilgrim, Mark (2010). HTML5: Up and Running. Sebastopol, California: O'Reilly Media, Inc.

Popescu, Andrei (07.09.2010, 2010a). Geolocation API Specification: Security and privacy considerations. W3C. http://www.w3.org/TR/2010/CR-geolocation-API-20100907/#security (Abruf: 12.02.2011)

Popescu, Andrei (07.09.2010, 2010b). Geolocation API Specification: 5.1 Geolocation interface. W3C. http://www.w3.org/TR/2010/CR-geolocation-API-20100907/#geolocation_interface (Abruf: 12.02.2011)

Popescu, Andrei (07.09.2010, 2010c). Geolocation API Specification: 5.2 PositionOptions interface. W3C. http://www.w3.org/TR/2010/CR-geolocation-API-20100907/#position_options_interface (Abruf: 14.02.2011)

Popescu, Andrei (07.09.2010, 2010d). Geolocation API Specification: 5.4 Coordinates interface. W3C. http://www.w3.org/TR/2010/CR-geolocation-API-20100907/#coordinates_interface (Abruf: 14.02.2011)

Popescu, Andrei (07.09.2010, 2010e). Geolocation API Specification: 5.5 PositionError interface. W3C. http://www.w3.org/TR/2010/CR-geolocation-API-20100907/#position_error_interface (Abruf: 14.02.2011)

Qt Reference Documentation (2011). WebKit in Qt. Espoo, Finnland: Nokia. http://doc.qt.nokia.com/4.7-snapshot/qtwebkit.html (Abruf: 01.03.2011)

Rogers, Rick (2010). Developing portable mobile web applications. Linux Journal. Volume 2010 Issue 197, September 2010. Seattle, WA, USA: Specialized Systems Consultants, Inc. http://delivery.acm.org/10.1145/1890000/1883522/10789.html?key1=1883522&key2=0652697921&coll=DL&dl=ACM&ip=195.202.144.9&CFID=9337726&CFTOKEN=11853995 (Abruf: 17.02.2011)

Savov, Vlad (02.02.2011). Android Market gets a web store with OTA installations, in-app purchases coming soon. AOL Inc. http://www.engadget.com/2011/02/02/android-market-gets-a-web-store/ (Abruf: 03.03.2011)

Schmiedl, Grischa & Baumert, Joachim & Seidl, Markus & Temper, Klaus & Dobiasch, Christine & Religa, Robert & ... Gunacker, Manuel (2009a). Verwendbarkeit und Verwendung des mobilen Webs. St. Pölten: Institut für Medieninformatik der Fachhochschule St. Pölten. http://medieninformatik.fh-stpoelten.ac.at/index.php?option=com_content&task=view&id=185&Itemid=10 (Abruf: 14.10.2010)

Schmiedl, Grischa & Seidl, Markus & Temper, Klaus (2009b). Mobile Phone Web Browsing – A Study on Usage and Usability Of The Mobile Web. Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. New York, USA: ACM. http://portal.acm.org/citation.cfm?doid=1613858.1613942 (Abruf: 14.10.2010)

Schonfeld, Erick (19.02.2009). Pinch Media Data Shows The Average Shelf Life Of An iPhone App Is Less Than 30 Days. TechCrunch. http://techcrunch.com/2009/02/19/pinch-media-data-shows-the-average-shelf-life-of-an-iphone-app-is-less-than-30-days/ (Abruf: 07.03.2011)

Sencha (2011a). Sencha Touch Mobile JavaScript Framework. Redwood City, CA, USA: Sencha Inc. http://www.sencha.com/products/touch/ (Abruf: 01.04.2011)

Sencha (2011b). Sencha Touch Licensing Options. Redwood City, CA, USA: Sencha Inc. http://www.sencha.com/products/touch/license/ (Abruf: 01.04.2011)

Sencha (2011c). Download Sencha Touch. Redwood City, CA, USA: Sencha Inc. http://www.sencha.com/products/touch/download/ (Abruf: 01.04.2011)

Sencha (2011d). Kitchen Sink. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/examples/kitchensink/ (Abruf: 01.04.2011)

Sencha (2011e). Getting Started with Sencha Touch. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/getting-started.html (Abruf: 01.04.2011)

Sencha (2011f). Icons. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/examples/icons/ (Abruf: 01.04.2011)

Sencha (2011g). Forms. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/examples/forms/ (Abruf: 01.04.2011)

Sencha (2011h). Ext.ux.touch.ListPullRefresh. Redwood City, CA, USA: Sencha Inc. http://www.sencha.com/forum/showthread.php?115326-Ext.ux.touch.ListPullRefresh (Abruf: 01.04.2011)

Sencha (2011i). Pull Refresh. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/examples/pullrefresh/ (Abruf: 01.04.2011)

Sencha Touch API (2011a). Ext.setup Method. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/source/Ext1.html (Abruf: 01.04.2011)

Sencha Touch API (2011b). Class Ext.Panel. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.Panel (Abruf: 01.04.2011)

Sencha Touch API (2011c). Class Ext.Container. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.Container (Abruf: 01.04.2011)

Sencha Touch API (2011d). Class Ext.MessageBox. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.MessageBox (Abruf: 01.04.2011)

Sencha Touch API (2011e). Class Ext.Carousel. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.Carousel (Abruf: 01.04.2011)

Sencha Touch API (2011f). Class Ext.Video. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.Video (Abruf: 01.04.2011)

Sencha Touch API (2011g). Class Ext.Audio. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.Audio (Abruf: 01.04.2011)

Sencha Touch API (2011h). Class Ext.util.Draggable. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.util.Draggable (Abruf: 01.04.2011)

Sencha Touch API (2011i). Class Ext.util.Sortable. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.util.Sortable (Abruf: 01.04.2011)

Sencha Touch API (2011j). Class Ext.form.FormPanel. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.form.FormPanel (Abruf: 01.04.2011)

Sencha Touch API (2011k). Class Ext.DatePicker. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.DatePicker (Abruf: 01.04.2011)

Sencha Touch API (2011l). Class Ext.List. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.List (Abruf: 01.04.2011)

Sencha Touch API (2011m). Class Ext.data.Model. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.data.Model (Abruf: 01.04.2011)

Sencha Touch API (2011n). Class Ext.NestedList. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.NestedList (Abruf: 01.04.2011)

Sencha Touch API (2011o). Class Ext.util.GeoLocation. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.util.GeoLocation (Abruf: 01.04.2011)

Sencha Touch API (2011p). Class Ext.Map. Redwood City, CA, USA: Sencha Inc. http://dev.sencha.com/deploy/touch/docs/?class=Ext.Map (Abruf: 01.04.2011)

Shum, Alex & Cole, Alan (22.09.2010). Building mobile applications for WebSphere Commerce using the hybrid application programming model. IBM. http://www.ibm.com/developerworks/websphere/library/techarticles/1009_shum/1009_shum.html (Abruf: 11.03.2011)

Spiering, Markus & Haiges, Sven (2010). HTML5-Apps für iPhone und Android. Poing: Franzis Verlag GmbH.

SQLite (2011). SQLite. Charlotte, North Carolina, USA: Hwaci. http://www.sqlite.org/ (Abruf: 22.02.2011)

Stever, Dan (08.12.2010).Titanium Mobile: Database Driven Tables with SQLite. Melbourne, Australien: Mobiletuts+, Envato. http://mobile.tutsplus.com/tutorials/appcelerator/titanium-mobile-database-driven-tables-with-sqlite/ (Abruf: 11.03.2011)

Symbian (2010a). The History of Symbian. Wales, England: Symbian Foundation Limited. http://www.symbian.org/about-us/history-symbian (Abruf: 08.12.2010)

Symbian (04.02.2010, 2010b). Symbian Completes Biggest Open Source Migration Project Ever. Wales, England: Symbian Foundation Limited. http://www.symbian.org/news-and-media/2010/02/04/symbian-completes-biggest-open-source-migration-project-ever (Abruf: 08:12.2010)

Symbian (08.11.2010, 2010c). Symbian Foundation to Transition to a Licensing Operation. Wales, England: Symbian Foundation Limited. http://www.symbian.org/news-and-media/2010/11/08/symbian-foundation-transition-licensing-operation (Abruf: 08.12.2010)

Symbian Developer (2010a). Symbian^1. Wales, England: Symbian Foundation Limited. http://developer.symbian.org/wiki/Symbian%5E1 (Abruf: 08:12.2010)

Symbian Developer (2010b). Symbian^2. Wales, England: Symbian Foundation Limited. http://developer.symbian.org/wiki/Symbian%5E2 (Abruf: 08:12.2010)

Symbian Developer (2010c). Symbian^3. Wales, England: Symbian Foundation Limited. http://developer.symbian.org/wiki/Symbian%5E3 (Abruf: 08:12.2010)

Symbian Developer (2010d). Symbian Foundation web sites to shut down. Wales, England: Symbian Foundation Limited.

http://developer.symbian.org/wiki/Symbian_Foundation_web_sites_to_shut_down (Abruf: 08.12.2010)

Tamas, Dan (08.08.2010, 2010a). Seven days with Titanium – day 1 – windows, views, navigation and tabs. cssgallery.info. http://cssgallery.info/seven-days-with-titanium-day-1-windows-views-navigation-and-tabs/ (Abruf: 11.03.2011)

Tamas, Dan (17.08.2010, 2010b). Seven days with Titanium – day 2 – tables and pickers. cssgallery.info. http://cssgallery.info/seven-days-with-titanium-day-2-tables-and-pickers/ (Abruf: 11.03.2011)

Tamas, Dan (06.11.2010, 2010c). Seven days with Titanium – day 4 – Media – images, movies and sounds. cssgallery.info. http://cssgallery.info/seven-days-with-titanium-day-4-media-images-movies-and-sounds/ (Abruf: 11.03.2011)

Tanner, Rudolf & Hofstetter, Rolf (2008). LTE, der nächste Mobilfunkstandard. Ausgabe 15/2008. Fehraltorf, Schweiz: Bulletin SEV/VSE. http://www.electrosuisse.ch/g3.cms/s_page/71290/displayType/artikelDetail/artikelLanguage/DE/artikelId/563 (Abruf: 20.11.2010)

Thornton, Brent & Schmit Ed (12.08.2010). Introducing the BlackBerry® Torch™ 9800 and the BlackBerry 6 Operating System. San Francisco, California, USA: Scribd Inc. http://www.scribd.com/doc/36065870/BlackBerry-6-Torch-9800-AT-T-Developer-Webcast-Slides (Abruf: 05.03.2011)

Tißler, Jan (12.07.2010). t3n: Die Königsfrage – Native App oder Mobile Website? http://t3n.de/news/mobile-konigsfrage-native-app-mobile-website-275476/ Hannover: yeebase media GbR. (Abruf: 19.10.2010)

Tran, D Dzung & Oksanen, Ilkka & Kliche, Ingmar (28.09.2010). The Media Capture API. http://www.w3.org/TR/2010/WD-media-capture-api-20100928/ (Abruf: 17.03.2011)

Tyberg, Justin (2011). PhoneGap Wiki: Getting Started with PhoneGap BlackBerry WebWorks. Vancouver BC, Canada: Nitobi Software. http://wiki.phonegap.com/w/page/31930982/Getting-Started-with-PhoneGap-BlackBerry-WebWorks (Abruf: 26.02.2011)

User Agent String (2011). List of User Agent Strings. User Agent String. http://www.useragentstring.com/pages/useragentstring.php (Abruf: 17.03.2011)

Van Beelen, Arend (2011). PhoneGap Symbian (Qt). Vancouver BC, Canada: Nitobi Software. http://wiki.phonegap.com/w/page/16494811/PhoneGap-Symbian-%28Qt%29 (Abruf: 17.03.2011)

Van Kesteren, Anne & Hickson, Ian (30.05.2008). Offline Web Applications: SQL. W3C. http://www.w3.org/TR/2008/NOTE-offline-webapps-20080530/#sql (Abruf: 17.02.2011)

Virkus, Rober & Gülle, Roland & Rouffineau Thibaut & Brady, Chris & Kriesing, Wolfram & Müffke, Friedger & ... Readfern-Gray, Gary (2011). Mobile Developer's Guide To The Galaxy 7th Edition. Bremen: Enough Software GmbH + Co. KG. http://www.enough.de/products/mobile-developers-guide/ (Abruf: 01.03.2011)

Virkus, Rober & Gülle, Roland & Rouffineau Thibaut & Brady, Chris & Kriesing, Wolfram & Müffke, Friedger & ... Schmidt, André (2010). Mobile Developer's Guide

To The Galaxy 6th Edition. Bremen: Enough Software GmbH + Co. KG. http://www.enough.de/products/mobile-developers-guide/ (Abruf: 06.12.2010)

W3C (2010). Mobile Web Application Best Practices Cards. W3C. http://www.w3.org/2010/09/MWABP/Overview.html.en (Abruf: 08.03.2011)

W3C (2011a). W3C Mobile Web Initiative. W3C. http://www.w3.org/Mobile/ (Abruf: 08.03.2011)

W3C (2011b). JavaScript Interfaces Current Status. W3C. http://www.w3.org/standards/techs/js#w3c_all (Abruf: 08.03.2011)

W3C (2011c). Standards for Web Applications on Mobile: February 2011 current state and roadmap. W3C. http://www.w3.org/2011/02/mobile-web-app-state.html (Abruf: 08.03.2011)

Wei-Meng, Lee (2009). mobiForge: Deploying iPhone Apps to Real Devices. Dublin, Ireland: dotMobi. http://mobiforge.com/developing/story/deploying-iphone-apps-real-devices (Abruf: 23.02.2011)

Whinnery, Kevin (01.07.2010). How-To: Perform CRUD operations on a local database. Mountain View, CA, USA: Appcelerator Inc. http://developer.appcelerator.com/blog/2010/07/how-to-perform-crud-operations-on-a-local-database.html (Abruf: 11.03.2011)

Wikipedia (10.08.2005). Nokia 9210 Communicator. Wikimedia Foundation, Inc. http://en.wikipedia.org/wiki/Nokia_9210 (Abruf: 08.12.2010)

WIP Connector (2010). App Stores tagged with Android. Wireless Industry Partnership Connector Inc. http://www.wipconnector.com/index.php/appstores/tag/android (Abruf: 03.03.2011)

Zokem (04.10.2010, 2010a). Mobile Insights - Are Smartphones Pervasive Devices? Espoo, Finnland: Zokem Ltd. http://www.zokem.com/2010/10/are-smartphones-pervasive-devices/ (Abruf: 18.10.2010)

Zokem (10.09.2010, 2010b). Applications Capture Already Half of Mobile Internet Traffic. Espoo, Finnland: Zokem Ltd. http://www.zokem.com/2010/09/applications-capture-already-half-of-mobile-internet-traffic/ (Abruf: 08.03.2011)