User Centered Web Design

Teilleistung Andreas Strauß

Nutzung plattformübergreifender HTML5 Webframeworks auf mobile Devices

Andreas Strauß

Wintersemester 2012

Umfang: 17995 Zeichen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

App.....mobile Anwendung für Smartphones und Tablets
ASP.....Active Server Pages
CSS.....Cascading Style Sheets
GPL.....General Public License
GPS.....Global Positioning System
HTTP.....Hypertext Transfer Protocol
HTML5.....Hypertext Markup Language Version 5
IDE.....Integrated Development Environment
IEEE .....Institute of Electrical and Electronics Engineers
iOS.....Standard-Betriebssystem der Apple-Produkte iPhone, iPod und iPad.
JS.....JavaScript
JQM.....jQuery mobile
MVC.....Model View Controller
PG.....PhoneGap
PHP.....Hypertext Preprozessor
RIA.....Rich Internet Applications
ST.....Sencha Touch
UI.....User Interface
WWW.....World Wide Web

1 Motivation

Das Angebot und die Nachfrage nach Smartphones und Tablets ist in den letzten Jahren immens angestiegen. Durch die starke Nachfrage aus dem Consumer Bereich halten auch immer mehr Geräte in den Unternehmen den Einzug und der Bedarf an mobilen Enterpriselösungen nimmt stark zu. Dabei haben die meisten Firmen mit einer heterogenen Gerätelandschaft und immer kürzeren Produktlebenszyklen der Hersteller im Hinblick auf Device-Management, System- und Anwendungsentwicklungen zu kämpfen. Die Herausforderung, plattformunabhängige Anwendungen zu entwickeln, besteht somit mehr denn je zuvor.

Softwarehersteller und Unternehmen stehen im Bezug plattformunabhängiger Anwendungsentwicklung vor gewaltigen Problemen. Für Unternehmen besteht neben der Nachhaltigkeit, also der weiteren Nutzung auf anderen Plattformen, sowie der Änderungsfreundlichkeit für bestehende Funktionen und Neuentwicklung neuer Anforderungen zu entwickeln. Das Hauptziel gerade in diesem Bereich der Anwendungsentwicklung liegt darin, mit möglichst wenig Quellcode (eine Quellbasis), möglichst viele Plattformen zu erreichen, um Änderungen, Weiterentwicklungen und Fehlerbereinigung drastisch zu reduzieren, sowie wie den Produktlebenszyklus der Apps möglichst nachhaltig zu gestalten[1].

Diese Ausarbeitung soll als Leitfaden dienen und einen Überblick geben, welche Möglichkeiten mit den bestehenden Entwicklungsframeworks zur Verfügung stehen und inwieweit Vorteile und Risiken gegenüber der nativen App Entwicklung vorhanden sind.

[1] Vgl. iX Developer-App Entwicklung, 03/2012

2 Problemstellung bei der Entwicklung mobiler Anwendungen

Die Anwendungsentwicklung bei mobilen Geräten ist gegenüber der Desktopentwicklung bei herkömmlichen PC's sehr viel komplexer und heterogener. Durch die immer stärkere Nachfrage nach mobilen Endgeräten verlagert sich der Focus der Anwendungsentwicklung sowohl im Consumer als auch Businessbereich zusehends in Richtung der mobilen Anwendungen. Das generelle Problem dabei besteht darin, dass der Aufwand bei nativer Anwendungsentwicklung für die gängigen Betriebssysteme wie Android, IOS, BlackBerry und Windows Phone erheblich größer ist als bei der Desktopentwicklung, bei der Windows den größten Marktanteil mit über 90% hat[2]. Im mobilen Bereich sieht es da schon anders aus, hier verteilen sich die Marktanteile etwas diffiziler, die Platzhirsche Android mit fast 60% und IOS 25% beherrschen fast den ganzen Markt[3]. Bei der mobilen Entwicklung müssen für jedes OS sämtliche Anwendungen hauptsächlich mit eigener Entwicklungsumgebung (IDE), sowie expliziter Programmiersprache, wie z.B. Java (Android), Objective C (IOS), C++ (BlackBerry) oder C# (Windows Phone), geschrieben werden. Bei Neuentwicklungen, Änderungen oder bugfixing müssen dann, sofern man für alle gängigen Plattformen entwickelt hat, sämtliche Anwendungen berücksichtigt, geprüft und erneut freigegeben werden. Desweiteren stellen die Schnittstellen zu den Geräten ein größeres Problem dar, um an hardwarenahe Funktionen wie z.B. GPS, WLAN, NFC oder Kamera heranzukommen, muss meist mit nativer Entwicklung ans Werk gegangen Werden. Diese Arbeit soll prüfen, ob sich dieser Aufwand durch Nutzung plattformunabhängiger Frameworks deutlich reduzieren lässt.

[2] Vgl.: http://www.zdnet.de/41559156/betriebssystemmarkt-windows-xp-verliert-erneut-anteile-an-windows-7/

[3] Vgl.: http://www.heise.de/resale/meldung/Android-und-iOS-beherrschen-die-Smartphone-Welt-1584381.html

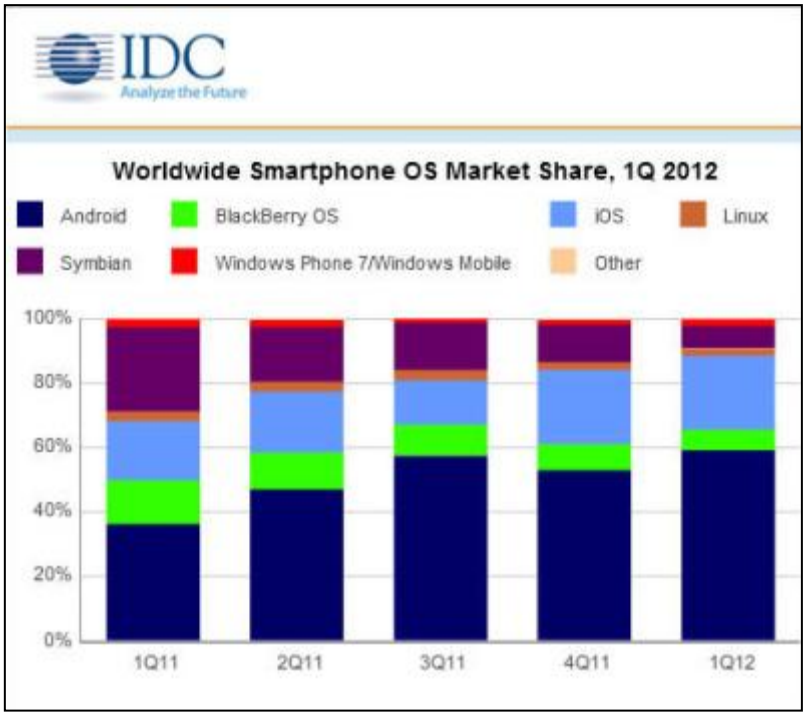

Abbildung 1: Verteilung mobile OS[4]

[4] Entnommen, http://www.heise.de/resale/meldung/Android-und-iOS-beherrschen-die-Smartphone-Welt-1584381.html

2.1 Überblick unterschiedliche Programmiersprachen

Hier soll veranschaulicht werden, wie unterschiedlich die Sprachsyntax der einzelnen mobilen Betriebssysteme anhand an einer hello-World App sind, wobei die Unterschiede zwischen Java, C# und C++ eher gering ausfallen.

2.1.1 Hello World mit Objective C für Apple IOS

```
UIAlertView *alert =
[[UIAlertView alloc] initWithTitle:@”Hello World”
message:@”iPhone!”
delegate:self
```

Codebeispiel 5.1.1:Objective C

2.1.2 Hello World mit Java für Android

```
TextView text = new TextView(this);
        text.setText("Hello World, Android");
        setContentView(text);
```

Codebeispiel 5.2.1:Java

2.1.3 Hello World mit C++ für BlackBerry

```
Page* root = new Page;
Label* label = Label::create()
.text("Hello world, BlackBerry");
root->setContent(label);
app->setScene(root);
```

Codebeispiel 5.3.1:C++

2.1.4 Hello World mit C# für Windows Phone

```
Label label1 = new Label();
label1.Text = „Hello World, Windows Phone“;
```

Codebeispiel 5.4.1:C#

3 Überblick mobiler Plattformen und deren Möglichkeiten zur Applikationsentwicklung

Bei der Realisierung plattformübergreifender mobile Apps kann man grundlegend zwei Ansätze unterscheiden. Auf der einen Seite das sogenannte Cross Compiling: Hier wird aus einer Quellbasis (beispielsweise C#) nativer Code für die jeweilige Zielplattform erzeugt. Beispiele sind die Frameworks Titanium und MonoTouch. Beide erlauben es, Apps aus einer Codebasis gleichzeitig für die Android- und die IOS-Plattform zu generieren[5].

Auf der anderen Seite steht der Ansatz, Plattformunabhängigkeit durch Nutzung standardisierter Webtechnologien zu erhalten, worauf der Schwerpunkt dieser Ausarbeitung liegt. Diese Richtung verfolgt die plattformunabhängige Entwicklung mit HTML5. HTML5 als Oberbegriff[6] des Funktionspaketes bezeichnet dabei die Erstellung des Designs in Anwendung von CSS3, die Struktur mit HTML und für die Funktionalität JavaScript. Eine nach dieser Sammeldefinition mit HTML5 entwickelte App läuft im Kontext des Browsers, der die Laufzeitumgebung zur Verfügung stellt. Da die größten mobilen Plattformen wie Android, IOS, BlackBerry und Windows Phone HTML5 unterstützen, ist eine Plattformunabhängigkeit dadurch grundsätzlich gegeben.

Diese beiden Ansätze lassen sich zudem nach der Art ihrer Ausgabe unterscheiden. Beide haben sie Apps zum Ziel. Man unterscheidet dabei drei Arten: Native-Apps, Hybrid-Apps und Web-Apps. Der Cross-Compiling Ansatz erzeugt Native-Apps, die auf den jeweiligen Plattform-SDK's basieren und direkt auf dem mobilen Betriebssystem laufen. Der HTML5-Ansatz mündet zunächst einmal in eine Web-App. Eine Web-App ist eine Anwendung, die mit Webtechnologien wie JavaScript, HTML und CSS entwickelt wurde und in einer Webumgebung, wie sie ein Browser zur Verfügung stellt, abläuft. Web-Apps sind im Laufzeitverhalten von Servertechnologien wie PHP, ASP oder Java unabhängig und laufen vollständig auf der Client-Seite, können aber natürlich zur Datenabfrage auf Server-Schnittstellen zugreifen. Hinsichtlich der Funktionen können Web-Apps bisher noch nicht alle Funktionen der Plattformen nutzen. So ist beispielsweise eine Kameraschnittstelle

[5] [iX Developer App-Entwicklung], Seite 127 ff.
[6] [Lubbers], Seite 7 ff.

im HTML5-Standard vorgesehen, jedoch noch in keinem der standardmäßig installierten Browser auf Android, IOS, BlackBerry oder Windows Phone implementiert. Diesen Nachteil gleichen Hybrid-Apps aus. Hybrid-Apps sind als Native-Apps verpackte Web-Apps. Hierzu wird die Web-App in einem nativen Container in einer speziellen „Web-View" angezeigt. Die native App besteht aus einer einzigen Komponente, in der wiederum die Web-App abläuft. Zwischen der WebView und der nativen Laufzeitumgebung lassen sich Daten austauschen. So ist die Nutzung nativer Funktionen gewährleistet, was den Nachteil der Web-App damit aufhebt.

3.1 Vergleich aller zur Verfügung stehenden Technologien

Typ	Programmiersprache	Vorteile	Nachteile
Native App	Plattformspezifisch: - Android: Java - IOS:Objective C - Black Berry: Java oder C++ oder Java Script - Windows Phone: C#	- Performancestark, da die App native auf der Plattform läuft und sie speziell für eine Plattform optimiert werden kann - Voller Funktionsumfang der Plattform nutzbar	- Läuft auf nur einer Plattform (Android /BlackBerry Kooperation ausgenommen) - Kann zum Teil nur über App-Stores veröffentlicht werden - Benötigt Entwickler-Zertifikate, die oftmals kostenpflichtig sind.
Hybride App	Plattformspezifisch und HTML5 (JavaScript+HTML+CSS)	- Kann auf den gesamten Funktionsumfang der Plattform zugreifen - Kann in den App-Stores veröffentlich werden - Bietet Plattformunabhängigkeit	- Verliert die Flexibilität der Web-App hinsichtlich des Vertriebs. - Kann zum Teil nur App-Stores vertrieben werden. - Benötigt Entwickler-Zertifikate, die oftmals kostenpflichtig sind.
Web-App	HTML5 = JavaScript + HTML + CSS	- Standardisierte Technologien - Bietet Plattformunabhängigkeit - Lauffähig auf Desktop, Mobile- und anderen Geräten	- Langsamer, da der Browser eine weitere Abstraktionsschicht darstellt und Optimierung nicht im Umfang einer nativen Entwicklung gemacht werden kann. - Kann nicht auf den gesamten Plattform-Funktionsumfang zugreifen. Hybride Apps haben diesen Nachteil auch.

Tabelle 1: Vergleich der App Modelle

Plattformübergreifende HTML5 web-frameworks

Name	Lizenz	Primäre Programmiersprache	Beschreibung
M-Project	MIT oder GPL	JavaScript	MVC-Framework mit einer Reihe vorgefertigter UI-Komponenten, Data Providern und umfassenden Build-Werkzeug (Espresso).
Sencha Touch	GPLv3 oder Sencha Touch Commercial License	JavaScript	MVC-Framework, dass die gleichen Bereiche wie M-Project abdeckt, unter anderem UI-Komponenten und Datenschnittstellen.
jQuery Mobile	MIT oder GPL	HTML/JavaScript	Reines UI-Framework, das eine Reihe vorgefertigter UI-Komponenten, Events und Funktionen zur Verfügung stellt. Es beruht auf jQuery und ist dank Progressive Enhancement auch auf älteren Geräten einsetzbar.
PhoneGap	Apache License	JavaScript	Stellt eher eine Bibliothek als ein Framework dar. PhoneGap stellt Schnittstellen zu einer Vielzahl nativer Gerätefunktionen zur Verfügung und bietet für verschiedene Plattformen bereits vorgefertigte Container, um aus einer Web-App eine hybride App zu machen.

Tabelle 2: Übersicht HTML5 web-frameworks[7]

[7] Entnommen aus iX Developer App-Entwicklung, Seite 128

4 Übersicht mobile HTML5 Frameworks

4.1 Sencha Touch

Sencha Touch entstand aus dem für Internetbrowser Applikationen bewährten JavaScript RIA-Framework Sencha (Ext-JS). ST unterstützt den HTML5 basierten cross-plattform Ansatz und gehört zur Kategorie der browserbasierten Web-Apps. Auch hier benötigen Entwickler keine Hochsprachkenntnisse, es reichen ebenfalls HTML, CSS und JavaScript Kenntnisse völlig aus. Durch die Anpassung und Öffnung der mobilen Betriebssysteme sind nun teilweise auch schon systemnahe Funktionen wie Kamera und Sensorik über JS ansprechbar. Desweiteren kann ST auch mit PhoneGap gekoppelt werden, so dass auch hier Hybrid-Apps erstellt werden können. ST kann vom UI look her so gestaltet werden, dass auf den unterschiedlichen OS Versionen die Apps alle gleich aussehen oder wenn gewollt mit den systemspezifischen Controls[8]. Mit ST erstellte Apps laufen in den Internetbrowsern der mobilen Geräte und können problemlos über das Filesystem implementiert werden[9]. Für die Entwicklung benötigt man keine spezielle IDE, es genügt ein Text Editor oder wer es komfortabler möchte, den von Sencha lizenzpflichtig erwerbbaren SenchaArchitect.

[8] Vgl.: [Garcia], Seite 9
[9] Vgl.: [Garcia], Seite 305

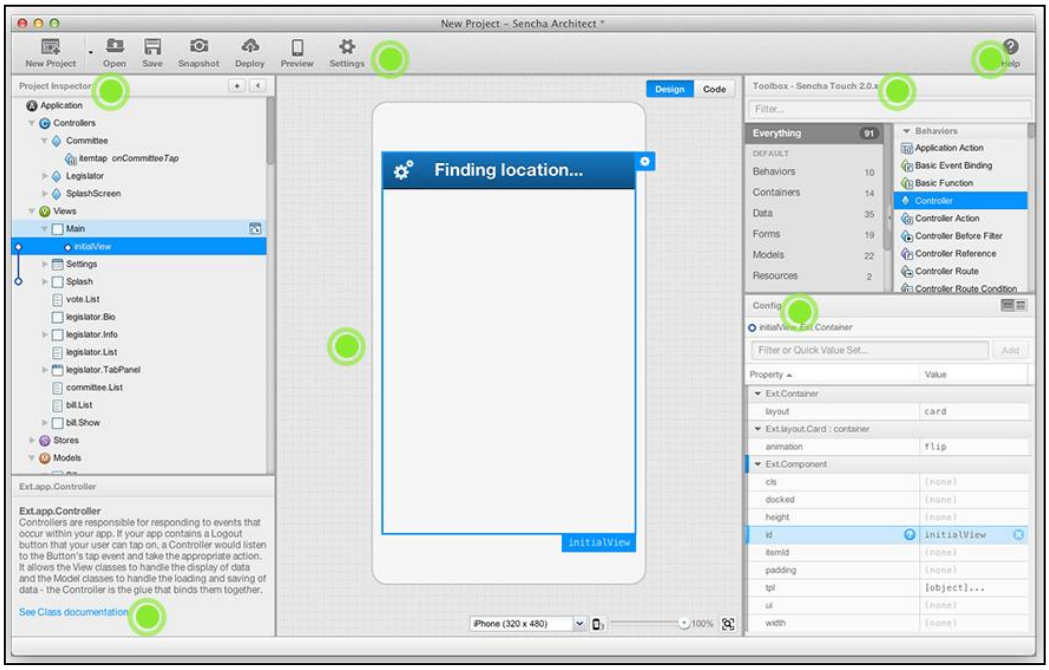

Abbildung 2: Sencha Architect

Abbildung 3: SenchaTouch UI[10]

[10] Entnommen aus [Garcia], Seite 202

```
<script type="text/javascript">
    Ext.setup({ #1
        onReady : function() { #2
        Ext.Msg.alert(
        'Hello!',
        'Hello there from Sencha Touch!'
        );
      }
  });
</script>
```

Codebeispiel 5.1.1:ST MessageBox mit HTML5 und JavaScript

4.2 jQuery mobile

jQuery mobile ist ein auf jQuery basierendes Framework zur Erstellung von Webseiten für vor allem mobile und per Touch gesteuerte. Es unterstützt die Entwicklung von Webseiten mit vordefiniertem Verhalten, das auf die Fähigkeiten und Einschränkungen mobiler Geräte angepasst ist. Möchte man die eigene Anwendung in Form einer Web-App entwickeln, so gibt es zwei grundsätzliche Vorgehensweisen:

Bei der ersten Version wird die Web-App mit nativen HTML5 und eigenen CSS-Klassen entwickelt. Der Feinschliff wird über JavaScript durchgeführt und so angepasst, dass die App auf jedem mobilen Gerät genutzt werden kann[11].

Die zweite Variante bündelt die Vorteile und Funktionen von Web-Apps und nativen Apps in einer Applikation. In diesem Fall spricht man von Hybrid-Apps. Diese werden mit dem nativen PhoneGap Plugin an die Bedienungsmöglichkeiten und Schnittstellen einzelner Geräte angepasst.

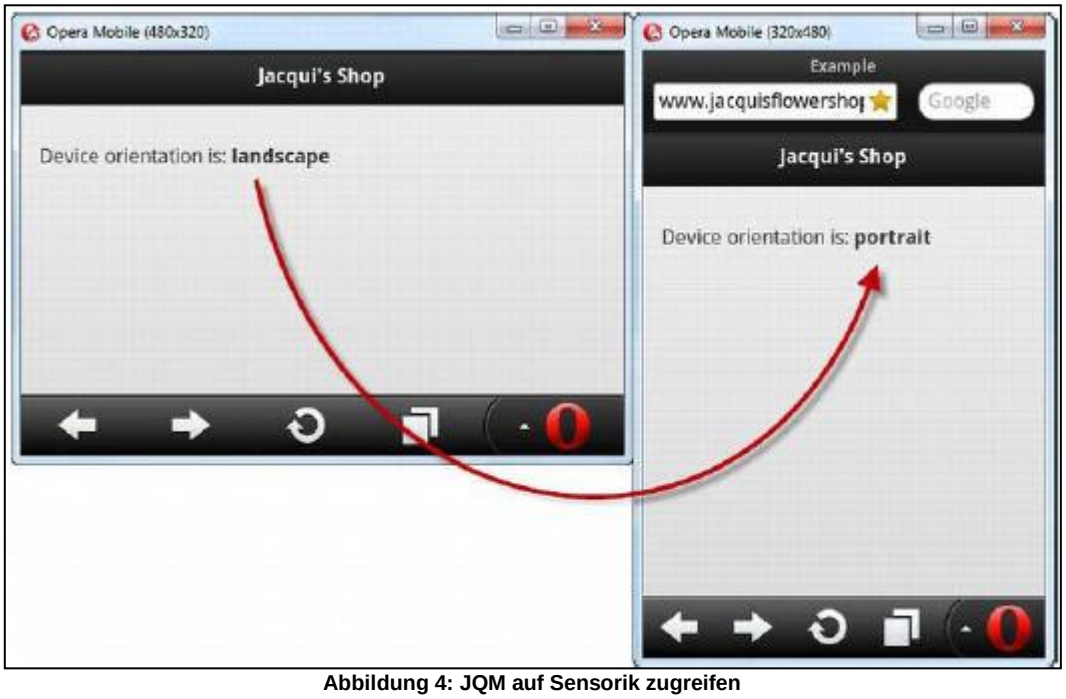

Abbildung 4: JQM auf Sensorik zugreifen

[11] Vgl.: [Freeman], Seite 762

```
<!DOCTYPE html>
<html>
<head>
    <title>Example</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="jquery.mobile-1.0.css" type="text/css" />
    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
        $(document).bind("pageinit", function() {
            $(window).bind("orientationchange", function(e) {
                $('#status').text(e.orientation)
            })
            $('#status').text(jQuery.event.special.orientationchange.orientation())
        });
    </script>
    <script type="text/javascript" src="jquery.mobile-1.0.js"></script>
</head>
<body>
    <div data-role="page">
        <div data-role="header">
                <h1>Jacqui's Shop</h1>
        </div>
        <div data-role="content">
            <p>Device orientation is: <b><span id=status></span></b></p>
        </div>
   </div>
</body>
</html>
```

Codebeispiel 6.2: JQM Sensorik Syntax mit HTML5[12]

[12] Entnommen: [Freeman], Seite 748

4.3 PhoneGap

PhoneGap nutzt HTML5, JavaScript und CSS3 für die Entwicklung mobiler Anwendungen. Diese sind heutzutage Standard in der Webentwicklung. Entwickler benötigen keine tiefgründigen Programmierkenntnisse in der jeweils gegebenen nativen Hochsprache, ausschließlich grundlegende HTML, CSS und JavaScript Kenntnisse sind notwendig. PG benutzt entsprechend des mobilen OS ein Plugin, welches Zugriff auf die Hardware- und Systemnahen Funktionen ermöglicht, quasi ein Interface zwischen nativem Basiscode und HTML5 basiertem UI-Layer. Werden neue Features in folgenden Betriebssystemen geboten, so muss natürlich auch das Plugin erneuert werden, was zusätzlichen Aufwand hervorruft[13]. Das Framework gehört zur Gattung der Hybriden-Apps, durch die Nutzung des systemnahen Plugins sind sie weder voll Web basiert noch vollkommen native, sondern eben eine Mischung aus beidem. PG Apps müssen ggf. über die jeweiligen App-Stores verteilt werden, da aus ihnen mit der Verwendung eine compilierte Anwendung erstellt wurde.

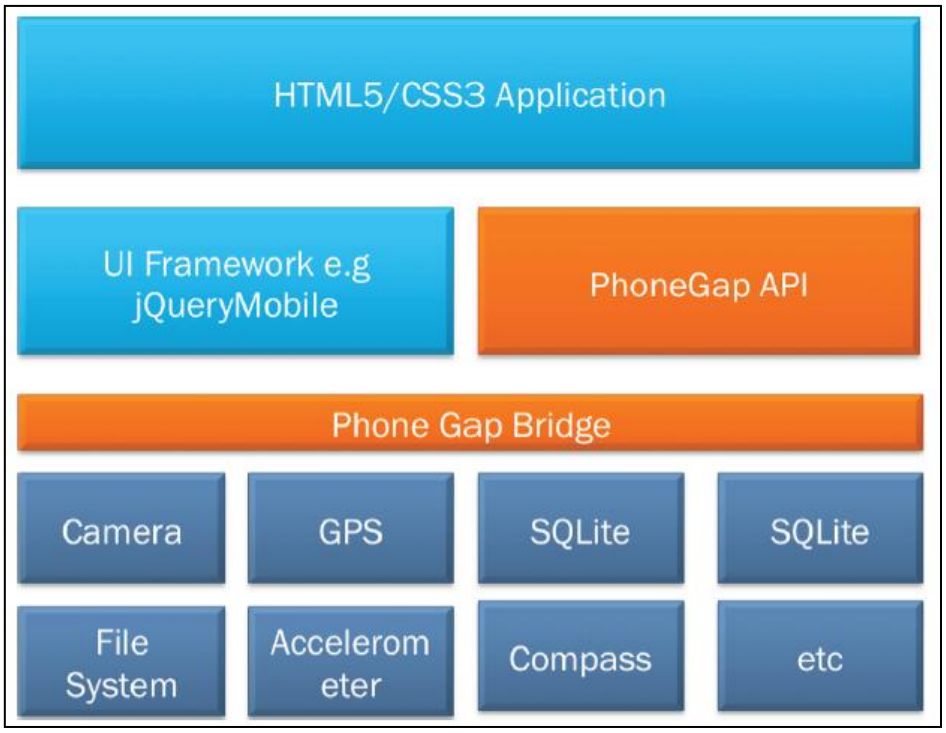

Abbildung 5: PhoneGap Architektur

[13] Vgl.: [Ghatol, Patel], Seite 17

PG bringt keine eigene IDE zur Entwicklung von Apps mit sich. Das Plugin setzt auf den jeweiligen Entwicklungsumgebungen der einzelnen mobilen OS auf und vereinfacht durch das Einbinden des Plugins den Zugriff auf die systemnahen Funktionen mit HTML5.

Abbildung 6: PhoneGap BlackBerry / Android / IOS Demo

5 Vor- und Nachteile frameworkbasierte vs. native Entwicklung

Die größte Konkurrenz für Web-Apps stellen die nativen Apps der verschiedenen Geräte dar, die zunächst mehrere Vorteile bieten: So lassen sich native Apps einfach, meist nur mit ein paar Klicks auf dem Smartphone installieren. Auch die Performance spielt eine Rolle. Native Apps werden unter IOS mit Objective-C und auf Android mit Java entwickelt. Mithilfe dieser Sprachen spricht man die Schnittstellen des Betriebssystems direkt an, was eine höhere Geschwindigkeit bedeutet, und direkten Zugriff auf die Hardware des Gerätes erlaubt. Darüber hinaus benötigen native Apps nicht zwingend eine bestehende Internetverbindung, da die für die App notwendigen Dateien auf dem Gerät gespeichert sind.

Es sind allerdings nicht nur technische Aspekte, die native Apps den Web-Apps gegenüber besser stellen. Sie punkten auch mit den Marketingmöglichkeiten über die App-Stores, denn die stellen eine stabile und von Kunden akzeptierte Infrastruktur für Verkäufe zur Verfügung, über die Entwickler eine große Zahl potenzieller Kunden erreichen können – und damit letztlich Geld verdienen.

Auf der anderen Seite haben aber auch Web-Apps ihre Vorteile gegenüber nativen Apps. Hier sind in erster Linie die Kosten der Entwicklung zu erwähnen. Native Apps werden meist in höheren Programmiersprachen wie Objective-C, Java oder C#/C++ entwickelt. Entwicklungskosten liegen hier um ein Vielfaches höher, als bei Webapplikationen auf Basis von HTML5, CSS3 und JavaScript[14].

[14] [iX Developer App-Entwicklung], Seite 127 ff.

6 Fazit und Beurteilung

Entwickelt man nativ für ein spezielles Gerät, verbaut man sich dadurch meist den Weg auf andere Systeme umzustellen und man muss für die gleiche Anwendung auf einem weiteren Gerät in einer anderen Sprache erneut entwickeln. Mit Web-Apps stellt sich das Problem nicht, denn HTML wird auf Desktoprechnern ebenso wie auf Smartphones ausgeführt. Ist eine Web-App auf die Besonderheiten der jeweiligen Plattform angepasst, ist lediglich ein Teil des Programmcodes und der Anzeigelogik auszutauschen, und man erhält eine für verschiedene Plattformen optimierte Applikation.

Und noch ein Punkt spricht für Web-Apps: Für ihre Verbreitung benötigt man keinen App-Store, eine einfache Webseite reicht aus. So entstehen keine Kosten bei der Verbreitung. Auch Updates und Veröffentlichungen sind einfacher, da kein Genehmigungsprozess oder Review notwendig ist. Updates lassen sich tagesaktuell verbreiten, erreichen alle Clients problemlos und stehen zuverlässig zur Verfügung[15].

[15] [iX Developer App-Entwicklung], Seite 128

Literaturverzeichnis

Ghatol, Patel (2012): Beginning PhoneGap, Apress.

Lubbers, Albers, Salim (2010): Pro HTML5 Programming, Apress.

Freeman (2012): pro jQuery, Apress.

Garcia, De Moss, Simoens (2012): Sencha Touch in Action, Manning.

iX Sonderheft Developer (03/2012) App Entwicklung, heise Verlag.

iX Sonderheft (02/2012), Webdesign, heise Verlag.

„Ich versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen."

Waiblingen, 27.12.2012 

_____

Ort, Datum, Unterschrift