Vergleichsstudie zur Entwicklung von mobilen Applikationen auf der Basis von Android, iOS und HTML5

Bader & Jene
Software-Ingenieurbüro

Impressum

Herausgeber:
Bader&Jene Software-Ingenieurbüro GmbH
Schauenburgerstrasse 116
D-24118 Kiel
Tel: 0431 . 908 900-0
Fax: 0431 . 908 900-88

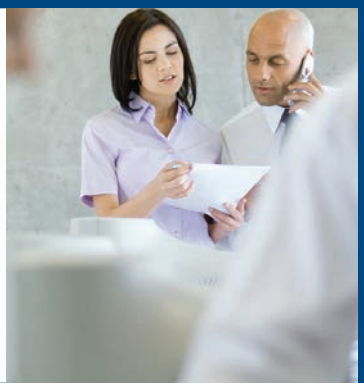
Papenreye 53
22453 Hamburg
Tel: 040 . 3866 132-0
Fax: 0431 . 908 900-88

E-Mail: info@bader-jene.de
Web: www.bader-jene.de

Gestaltung/Layout: Carola Stich (Bader&Jene)

Foto Titel links: © Robert Kneschke - Fotolia.com

Inhalt



Firma

Die Bader&Jene Software-Ingenieurbüro GmbH wurde 2003 in Kiel von Thomas Bader und Andreas Jene gegründet. Das Unternehmen basiert auf den Säulen: individuelle Softwareentwicklung, Standardsoftware und Consulting. Besonders die Spezialisierung auf Java und die Methodik SCRUM zeichnet das Unternehmen für ihre individuellen IT-Lösungen aus. Seit mehreren Jahren entwickelt Bader& Jene mobile Anwendungen sowie moderne App-Lösungen für Smartphones bzw. Tablets und berät Unternehmen bei der Auswahl geeigneter Plattformen.

Zielsetzung

Bei dieser Untersuchung geht es um einen Vergleich zwischen den Alternativen bei der App-Programmierung. Dabei werden insbesondere die Möglichkeiten bzw. Einschränkungen eines Frameworks gegenüber nativer Entwicklung beleuchtet.

Allgemeine Marktpräsenz

Der mobile Markt für Smartphones wächst rasant. Android® besitzt dabei die größte Verbreitung der mobilen Betriebssysteme. Laut einer Gartner® Studie besaß Android® im 3. Quartal 2011 einen Marktanteil von 51%, Symbian® 17% und iOS® 15%. Trotz des schnellen Wachstums von Android®, ist auch in den nächsten Jahren nicht mit einer Konsolidierung des Marktes zu rechnen. Daher wird es auch in Zukunft wichtig sein, verschiedene Plattformen zu unterstützten, um eine möglichst große Kundenzahl zu erreichen.

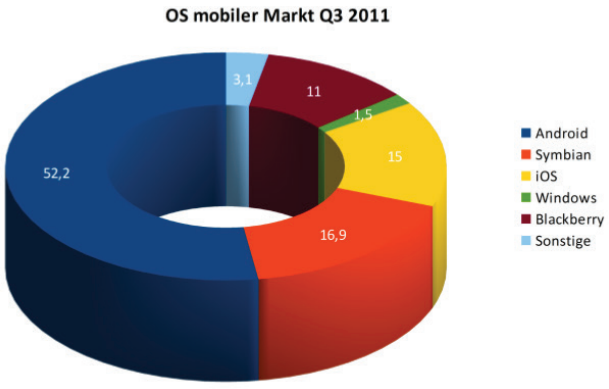


Abbildung 1: Marktverteilung in Prozent (Quelle: November 15, 2011 Gartner®, http://www.gartner.com/it/page.jsp?id=1848514, Datum des Zugriffs: 16.05.2012)

Technologien

Bei der Realisierung einer mobilen Applikation gibt es verschiedene Technologien am Markt, die genutzt werden können.

Native Entwicklung

Unterst. Plattformen	1
Marktdurchdringung	1,5 – 52 %
Vermarktung	App Market
Hardwarefunktion	Alle
Natives Aussehen	Ja
Debugging	Ja

Native Entwicklung bedeutet, dass direkt für ein bestimmten Gerätetyp (Android®, iOS®) programmiert wird. Dabei können alle Möglichkeiten des Gerätes genutzt werden, das Programm ist allerdings nur auf diesem Gerätetyp lauffähig. Bei der nativen Entwicklung einer App ist je nach Anzahl der Zielplattformen ein erhöhter Zeitaufwand gegenüber einer HTML5 basierten Lösung vorhanden. Dabei spielt nicht nur die absolute Zahl der zu entwickelden App-Versionen eine Rolle. Da jedes System auch eine unterschiedliche Architektur und verschiedene Programmiersprachen verwendet (Java® bei Android®, Objective-C® bei iOS®, C/C++ bei Bada®, C#® bei Windows® Mobile) sollte ein breites Wissen in diesen Bereichen vorhanden sein. Ansonsten muss eine Einarbeitung in diese Sprachen und Architekturen bei der Zeitschätzung beachtet werden. Daneben bietet native Programmierung aber auch zahlreiche Vorteile bei der App-Entwicklung. So können sämtliche Hardwarefunktionen eines Smartphones wie Kamera, GPS, Mikrofon usw. genutzt werden.

Auch GPU unterstütztes Rendering von Grafiken ist (bislang) nur mit nativer Programmierung möglich. Die Performance der Apps – insbesondere bei der Reaktionszeit auf Touch-Events der Benutzer – ist unter nativen Applikationen immer noch spürbar besser. Des Weiteren integriert sich eine solche App in das bestehende Look and Feel des Systems, ohne dass weitere Anpassungen von Nöten sind. Zusätzlich können für die Verteilung und Bezahlung der Apps die App Stores der entsprechenden Plattformen genutzt werden, was bei einer mobilen Webseite nicht möglich ist.

Mobile Webseite

Unterst. Plattformen	Alle
Marktdurchdringung	100%
Vermarktung	Eigenes System
Hardwarefunktion	Keine
Debugging	Ja

Inhalte können auf ein Smartphone auch in Form einer Webseite gebracht werden, da alle Smartphones einen Browser zum Betrachten von Webseiten mitbringen. Eine solche Seite sollte an die Gegebenheiten der mobilen Nutzung angepasst sein. Dabei sind nicht nur die Ausmaße des Bildschirms eines Handys zu beachten, sondern auch die verringerte CPU Leistung gegenüber eines Desktop PCs (rechenlastige Elemente wie aufwendige Java® Applets sollten vermieden werden). Des Weiteren müssen die Einschränkungen der mobilen Browser in Bezug auf Flash oder andere Technologien berücksichtigt werden. Zuletzt ist die mobile Internetanbindung nicht mit einem stationären Breitbandanschluss gleichzusetzen, so dass das Datenvolu-

men der Webseite möglichst niedrig gehalten werden muss. Neben diesen Anpassungen müssen auch die Nachteile einer klassischen Webseite im Vergleich zu einer App bedacht werden: So ist eine durchgängige Internetverbindung zum Server erforderlich. Gerade in Gebäuden oder eher ländlichen Gebieten ist eine solche Verbindung oft nicht gegeben. Des Weiteren können bei einer solchen Lösung keine Hardwarefunktionen des Smartphones wie Kamera, Mikrofon oder GPS genutzt werden. Allerdings ist eine mobile Webseite – besonders für einen Webentwickler – schnell und kostengünstig zu erstellen und unterstützt alle mobilen Plattformen, da sie mit jedem Browser – unabhängig von der Architektur – genutzt werden kann. Anpassungen können dabei, insbesondere beim CSS, jedoch von Nöten sein. Beim Verkauf eines solchen „Programms“ können die App Stores der entsprechenden Plattformen nicht genutzt werden, so dass man sich ein eigenes Abrechnungssystem aufbauen muss.

HTML5 Container Framework

Unterst. Plattformen	7 (iOS®, Android®, HP webOS®, Windows Mobile, Symbian®, Blackberry® und Bada®)
Marktdurchdringung	Nahe 100%
Vermarktung	App Market
Hardwarefunktion	Alle
Natives Aussehen	Nein
Debugging	Nur mit externen Tools

Bei der Benutzung eines HTML5 Container Framework wie PhoneGap® wird die GUI in HTML und CSS erstellt. Die Programmlogik wird in JavaScript® geschrieben. Die resultierende Webseite wird in einem WebView in einer App dargestellt, so dass der Benutzer im Idealfall gar nicht bemerkt, dass er keine native Anwendung benutzt. So muss die App für mehrere Zielplattformen (theoretisch) nur einmal programmiert werden. Es muss allerdings für jede Plattform eine eigene Toolchain zum Deployment vorhanden sein. Unterstütze Plattformen sind: iOS®, Android®, HP webOS®, Windows® Mobile, Symbian®, Blackberry® und Bada®. Auch sind Anpassungen an die einzelnen Plattformen oftmals nötig, insbesondere bei Features wie Push Notifications, die nicht im JavaScript® des „Web-Kerns“ der App realisiert werden können. Daneben muss ein Android® Widget zusätzlich nativ programmiert werden. Bei der Erstellung der internen Webseite der App können Frameworks wie jQuery© mobile, Sencha Touch 2® oder Application Craft® unterstützend verwendet werden. Diese erleichtern dem Programmierer die Arbeit und bieten Möglichkeiten, die

Bedienung der App näher an native Anwendungen zu rücken, beispielsweise mit der Unterstützung von Swipe Gesten. Das größte Problem bei der Entwicklung stellte das Debugging dar. Da die eigentliche Programmlogik nicht im nativen Code vorliegt, sondern in einer WebView geladen wird, gibt es direkt in der IDE keine Möglichkeit den eigenen Code zu debuggen. Es können weder Breakpoints gesetzt, noch Funktionen durchgesteppt werden. Debugging ist nur mithilfe von externen Tools möglich. Für die GUI bietet sich hier weinre an, damit lässt sich die UI direkt auf dem Simulator oder auf dem Gerät testen. Vom Aufbau her ähnelt weinre Firebug® oder Web Inspector / Google® Chrome® Developer Tools. Ein Firebug ähnliches Tool ist auch nötig, um die JavaScript® Funktionen zu debuggen.

JavaScript® Interpretierungsframeworks

Unterst. Plattformen	2 nativ (iOS® und Android®) plus Web Variante
Marktdurchdringung	70% nativ, Rest per Web
Vermarktung	App Market für Android® und iOS®
Hardwarefunktion	Alle für Android® und iOS®, keine für den Rest
Natives Aussehen	Ja bei Android® und iOS®
Debugging	ja

Eine andere Möglichkeit der Cross-Plattform Entwicklung sind Frameworks die JavaScript® Code interpretieren und daraus native Elemente generieren, wie beispielsweise Appcelerator®. Zur Laufzeit besteht die App aus drei Hauptteilen: zum ersten der JavaScript® Code (als codierter String in einer Java® oder einer Objective-C® Datei), zum anderen die Titanium® API in der nativen Programmiersprache der Plattform und zuletzt dem verwendeten JavaScript® Interpreter. So lassen sich sogenannte „Proxy-Objekte“ mit JavaScript® und den Titanium® API Funktionen erstellen. Über diese Objekte lassen sich native Funktionen aufrufen. Es handelt sich damit nicht, wie oftmals angenommen um ein Cross-Compiling von JavaScript® nach Objective-C® oder Java®. UI und Programmlogik werden dabei in JavaScript® geschrieben. Dabei können wie bei PhoneGap® Hardwarefunktionen des Smartphones verwendet werden. Aufgrund der internen Vorgehensweise können mit Appcelerator® nur native Android® und iOS® Apps sowie Web Anwendungen erstellt werden. Im Gegensatz zurPhoneGap® Lösung bestehen diese Apps jedoch aus nativen UI Elementen und reihen sich daher nahtlos in andere native Apps auf der Zielplattform ein. Beide Arten von Frameworks ermöglichen auch die Nutzung der App Stores des Zielsystems für die Verbreitung und Abrechnung der Applikation.

Code Generierung

Abseits dieser Möglichkeiten gibt es auch Cross-Compiler Toolchains wie beispielsweise XMLVM, das die Möglichkeit bietet, Programme in Java®, .NET CL oder Ruby zu schreiben und diese in Java®, .NET CL, JavaScript®, Python®, Objective-C® oder C++ zu übersetzen. Dabei wird die ursprüngliche Sprache erst in einen XML Modell überführt, welches danach in die Zielsprache übersetzt wird.

Diese Möglichkeit erweitert die benutzbaren Programmiersprachen für die App Entwicklung, weist aber einige entscheidende Nachteile auf: Wie bei Code-Generierungswerkzeugen oft der Fall, ist der erzeugte Code schon bei simplen Grundfunktionen aufgebläht und unleserlich. Fehler im generierten Code sind somit nur schwer zu finden. Besonderheiten der Zielplattformen könnten es zudem nötig machen, den generierten Code direkt zu manipulieren, um so Fehler zu beheben. Diese Technik ist zum heutigen Zeitpunkt mehr eine Machbarkeitsstudie denn eine Möglichkeit für den produktiven Einsatz.

Marktanalyse

Auch wenn es einige Beispiele von gelungenem Design von mobilen Webseiten gibt (Facebook®, Amazon®, flickr®, NikeLap® …) geht der Trend einer mobilen Lösung zur App. Dabei zeichnet sich keine eindeutige Entwicklung ab, ob die native Programmierung oder eine Lösung mit einem Framework wie PhoneGap® oder Appcelerator® sich durchsetzen wird. Dabei gibt es von allen genannten Technologien zahlreiche populäre Apps.

PhoneGap®: Wikipedia®, IGN® Dominate, …

Appcelerator®: NBC®, LEGOLAND®, …

Von der Verbreitung her lässt sich kein Argument für oder gegen eine der Lösungen finden.

Praxistest

Für das geplante Projekt waren Hardwarefunktion wie das Nutzen der Kamera gefordert, so dass eine reine Webseite ausschied. Um möglichst viele Plattformen unterstützen zu können, wurde das Framework PhoneGap® eingesetzt. Zur Unterstützung ist die JavaScript® Bibliothek jQuery© Mobile verwendet worden. Für einen direkten Vergleich wurde die App jeweils für Android® und iOS® nativ programmiert.

PhoneGap® punktet mit einer geringen Einarbeitungsdauer für den Programmierer und mit der Unterstützung von mehreren Plattformen bei einer Codebasis. Sie bietet daher besonders für kleinere App Projekte eine gewaltige Kostenersparnis gegenüber einer nativen Entwicklung. Allerdings ist die Performance schlechter als bei nativen Apps, was sich in einer trägeren Bedienung negativ bemerkbar macht. Da bei PhoneGap® keine nativen UI Elemente zum Einsatz kommen, passt die App ohne weitere Anpassung nicht in das „Look and Feel“ der Zielplattform. Dabei sind hingegen die weitreichenden Anpassungsmöglichkeiten der GUI mittels CSS positiv anzumerken. Neben diesem grafischen Aspekt tritt das größte Problem des Frameworks bei der Programmierung, genauer gesagt beim Debugging auf, welches nur eingeschränkt möglich ist.

Native Programmierung hingegen erfordert wesentlich mehr Zeit für Einarbeitung und Programmierung, bietet allerdings eine nahtlose Integration in die Zielplattform, bessere Performance und ausgereifteres Tooling für Programmierung und Debugging.

Fazit

Je nach Einsatzgebiet kann eine HTML5 Lösung sinnvoll für ein App Projekt sein. Unterstützung mehrerer Plattformen sowie die Verwendung bekannter Technologien wie HTML, CSS und JavaScript® sind starke Argumente für die Verwendung einer solchen Lösung. Mit einem entsprechenden Framework können auch Hardwarefunktionen von Smartphones verwendet werden. Beim Tooling hat diese Herangehensweise allerdings noch großen Nachholbedarf.

Bei erhöhter Komplexität des Projektes, insbesondere wenn es sich um performance-kritische Anwendungen handelt, fällt die Wahl allerdings (noch) auf native Entwicklung. Die (subjektiven) Erfahrungen mit den einzelnen Technologien im Test waren wie folgt:

	Nativ	PhoneGap®
Einarbeitung	+	+
UI Design	+	+
Look and Feel	+	-
Customizing	o	+
IDE Unterst.	+	o
Programmierung	+	o
Debugging	+	-
Zeitaufwand	-	+

Zukünftig ist die Weiterentwicklung von HTML von zentralem Interesse, könnten doch hier weitere Standards für Hardwarefunktionen eine Mittelschicht wie PhoneGap® überflüssig machen. 3D Animationen mittels WebGL sind ein erster Schritt in diese Richtung. Bis eine Unterstützung verabschiedet und letzten Endes in den mobilen Browsern angekommen ist, bieten hybrid Lösungen aus eingebettetem HTML5 eine interessante Lösung kostengünstig Apps zu entwickeln.

Bader&Jene Software-Ingenieurbüro GmbH

Schauenburgerstrasse 116
D-24118 Kiel
Tel: 0431 . 908 900-0
Fax: 0431 . 908 900-88

Papenreye 53
D-22453 Hamburg
Tel: 040 . 3866 132-0
Fax: 0431 . 908 900-88

E-Mail: info@bader-jene.de
Web: www.bader-jene.de

Bader & Jene
Software-Ingenieurbüro