Plattformunabhängige App-Entwicklung für mobile Geräte
–
Grenzen und Möglichkeiten

Daniel Morgenstern

22. April 2014

Inhaltsverzeichnis

1 Einleitung

1.1 Ziel der Arbeit

1.2 Motivation

1.3 Aufbau der Arbeit

2 Theoretische Grundlagen

2.1 Ausgangsproblematik

Zumeist nutzen die verschiedenen Plattformen nicht nur unterschiedliche Dateiformate für ihre Anwendungen sondern auch unterschiedliche Programmiersprachen und Toolkits für die Programmierung. So verwendet beispielsweise Android die Programmiersprache Java in Kombination mit einem eigenen Android SDK, während Apps für Apples iOS mit ObjectiveC in der Firmeneigenen Entwicklungsumgebung Xcode (die auch nur auf dem firmeneigenen Betriebssystem Mac OSX läuft) geschrieben werden. Somit ist die Plattformunabhängigkeit mit den herkömmlichen Entwicklungsmethoden kaum zu erreichen, da die verschiedenen Produkte und Plattformen zueinander nicht kompatibel sind.

bessere Einleitung finden!

welche technischen grundlagen kann ich denn voraussetzen?

Muss man das überhaupt erwähnen? schließlich wird das ja vermutlich auch Inhalt von Motivation und so.

Im Planungsprozess einer Softwarelösung ist in bestimmten Fällen eine denkbare Alternative zur nativen App eine Webanwendung, die mithilfe des Browsers abrufbar ist, also konkret dynamische Website, wie man sie auch von der herkömmlichen Internet-Nutzung eines Desktop-Systems kennt.

Diese Variante hat zwar den Vorteil, dass die Anwendung plattformunabhängig ist, da jedes moderne (mobile) Betriebssystem einen Browser besitzt, der Entwicklungsaufwand also unabhängig von der Anzahl der Zielplattformen der gleiche bleibt, allerdings stößt der Entwickler schnell an seine Grenzen, wenn Hardware- oder betriebssystemnahe Anforderungen gestellt sind. So ist es beispielsweise mit Webtechnologien wie HTML oder Javascript nicht ohne weiteres möglich, auf die Kamera oder das GPS-Modul eines Gerätes oder das Adressbuch eines Betriebssystems zuzugreifen.

nachweis!

Ebenso kommt die Webanwendung im Normalfall nicht für Offline-Anwendungen in Frage, da die meisten mobilen Betriebssysteme nicht vorsehen, dass der Nutzer aus dem Dateisystem eine HTML-Seite öffnet.Dahingehend liegt ein Ansatz zur plattformunabhängigen App-Entwicklung auf der Hand: Die Hybrid-App, also die Verwendung von Webtechnologien mithilfe einer Schnittstelle zwischen der Webanwendung und der nativen Ebene des Betriebssystems.

nachweis!

2.2 Technologien

2.2.1 Grundlegende Webtechnologien

Konkret heißt das, die Anwendung wird als Webanwendung mit den zugehörigen Technologien (HTML, Javascript, CSS) entwickelt und für die jeweiligen benötigten Plattformen

in eine native App eingebettet, welche hier allerdings hauptsächlich aus einer Web-View, also einer abgespeckten Variante eines Web-Browsers zur Anzeige der Webanwendung, besteht. Da dieser Ansatz der am häufigsten von Cross-Platform-Frameworks genutzte ist, stellt dieser auch den Schwerpunkt der in dieser Arbeit explorierten Technologien dar und soll hier näher erläutert werden.

nachweis!

2.2.2 Phonegap

Die entscheidende Lücke bei der auf dem Weg zur Hybrid-App schließt *PhoneGap* von Adobe. Bei der Webanwendung taten sich in der Herleitung zwei grundlegende Problemfelder gegenüber der nativen App-Entwicklung auf. Zum Einen besteht eine Webanwendung aus mehreren Dokumenten, die auf einem Server liegen und nur über den Browser des Betriebssystems abgerufen werden, fühlt sich also für den Nutzer durch die ausbleibende Installation, die benötigte Internetverbindung und die optische Präsenz des Browser-Oberfläche nicht wie eine Anwendung an. Zum Anderen können die Anforderungen an die Anwendung einen Zugriff auf native Features des Geräts oder seines Betriebssystems erfordern, der mittels herkömmlicher Webtechnologien nicht oder nur sehr eingeschränkt möglich sind.

Für letzteres bietet PhoneGap eine Javascript-Bibliothek, die auf das Cross-Platform-Framework *Cordova* aufbaut und den Zugriff auf native Features des jeweiligen Betriebssystems ermöglicht.
todowie genau, weiß ich noch nicht, noch rausfinden! Bei der konkreten Verwendung von nativen Features muss jedoch teilweise wieder auf die Unterstützung durch die jeweiligen Plattformen geachtet werden.

checken: oder zumindest deklariert werden, was verwendet werden soll.

Darüber hinaus übernimmt das Online-Portal PhoneGap Build den Bauprozess der App, also das Überführen in ein installierfähiges App-Format. Während der Entwickler ohne diesen Build-Service die verschiedenen Toolkits aller Zielplattformen lokal verwalten müsste, um Cordova zu verwenden,[1] reicht es hier aus, die Web-Anwendung (bspw. per öffentlichem Git-Repository) auf das Portal zu laden, den Build-Prozess im Browser abzuwarten und die fertigen Apps auf dem gewünschten Zielgerät zu installieren.

Für Auftraggeber oder Entwickler, die ihren Code als sehr sensibel und vertraulich ansehen, könnte diese Variante allerdings heikel sein, da der Quellcode dann stets auch bei Adobe liegt.

gehört vllt. eher woanders hin.

2.2.3 JQueryMobile

Die Javascript-Bibliothek JQueryMobile baut auf JQuery auf und bietet ein Oberflächen-Toolkit für mobile Webseiten. Sie besteht zum Einen aus einer Javascript-Datei, die in die HTML-Seite eingebunden wird und zum Anderen aus einem CSS, das für das an mobile Touch-Geräte angepasste aussehen verantwortlich ist. So können einerseits UI-Elementen explizit Style-Klassen aus dem JQueryMobile-CSS zugewiesen werden, andererseits sorgt das JQuery-Javascript ohnehin bei allen verwendeten HTML-Elementen dafür, dass die

[1] link

Style-Klassen dynamisch vergeben werden und die UI-Komponenten somit ihr App-typisches Aussehen erhalten. Dieser Mechanismus erleichtert es dem Frontend-Entwickler erheblich, eine auf Touchscreens ausgerichtete Oberfläche zu entwerfen, da er im Grunde sämtliche herkömmliche HTML-Elemente verwenden kann.

Lediglich mit dem zusätzlichen Attribut "data-role"werden den UI-Elementen Eigenschaftentypen zugewiesen, über die die JQueryMobile-Bibliothek erkennt, wie dieses dargestellt werden soll.

Konkreter, Mit Beispiel?

2.2.4 Knockout.JS

Auch bei Knockout.js handelt es sich um eine Javascript-Bibliothek, die per <script>-Tag der HTML-Seite hinzugefügt wird. Knockout.js übernimmt mit einem MVVM (Model-View-View-model) das Data-Binding, also die Verknüpfung zwischen Daten-Feldern im Programm und UI-Elementen. So lässt sich die UI (View) sehr klar von der Programmlogik (Model) trennen, was der Lesbarkeit, Erweiterbarkeit und Wartbarkeit zugute kommt.

Statt also Javascript mit HTML-String-Schnipseln zu vermischen indem per Javascript das DOM direkt manipuliert und UI-Elemente zur Laufzeitprogrammatisch erweitert werden, definiert der Entwickler in einem separaten Javascript ein View-Model und bindet mit dem HTMl-Attribut "data-bindëine bestimmte View-Eigenschaft an ein Datanfeld aus dem View-Model.

gibt es bei javascript überhaupt eine laufzeit?

Konkreter, Beispiel, Grafik?

3 Praktische Umsetzung