Signet farbig_ Format

**Entwurf und Umsetzung einer**

**Beschreibungssprache für mobile Anwendungen**

Bachelorarbeit von Edmund Senkleiter

Bachelorarbeit

Fakultät für Informatik   
Institut für Softwaretechnologie  
Professur für Programmierung kooperativer Systeme  
Prof. Dr. Michael Koch



betreut von:  
Peter Lachenmaier, M. Sc.

Abgabetermin der Arbeit: 14.01.2015

Hiermit versichere ich an Eides statt, dass die vorliegende Arbeit von mir selbständig verfasst wurde und ich alle verwendeten Quellen, auch Internetquellen, ordnungsgemäß angegeben habe.

Datum, Unterschrift (Name)

Kurzzusammenfassung

Hier steht später eine kurze und prägnante Zusammenfassung der Arbeit in ca. fünf bis maximal zehn Sätzen

Inhaltsverzeichnis

Kurzzusammenfassung i

Inhaltsverzeichnis ii

Abbildungsverzeichnis iv

Abkürzungsverzeichnis v

1 Einleitung 1

1.1 Motivation 1

1.2 Problemstellung 2

1.3 Zielsetzung 2

1.4 Aufbau der Arbeit 4

2 Entwicklung plattformübergreifender mobiler Anwendungen 5

2.1 Bewertungskriterien plattformübergreifender Anwendungsentwicklungsframeworks 6

2.2 Bewertung aktueller plattformübergreifender Anwenungsentwicklungsframeworks 8

2.3 Zusammenfassung existierender Sprachen 8

3 Beschreibungssprache 9

3.1 Anforderungen 9

3.2 Entwurf 9

4 iOS Interpreter 10

4.1 Architektur 10

4.2 Entwurf 10

5 Beispiel iOSTemplateLanguage Applikation 11

5.1 Anforderungen 11

5.2 CommunityMashup 11

5.3 Umsetzung/Funktionsweise 11

5.4 Fazit 11

5.4.1 Erreichter Funktionsumfang 11

5.4.2 Verbesserte Performanz 11

6 Zusammenfassung und Ausblick 12

Literaturverzeichnis 13

Abbildungsverzeichnis

[Abbildung 1: Übersicht Risikobewertung und Bewertungsmodell **Fehler! Textmarke nicht definiert.**](#_Toc211259057)

[Abbildung 2: Zuordnung der IT-Risikofaktoren zu den Managementtätigkeiten **Fehler! Textmarke nicht definiert.**](#_Toc211259058)

Abkürzungsverzeichnis

App Applikation

Aufl. Auflage

Bd. Band

bzw. beziehungsweise

ca. circa

d. h. das heißt

et al. et altera

etc. et cetera

f. folgende

ggf. gegebenenfalls

Hrsg. Herausgeber

i. d. R. in der Regel

S. Seite

s. siehe

sog. sogenannt

u. a. unter anderem

usw. und so weiter

v. a. vor allem

vgl. vergleiche

z. B. zum Beispiel

z. T. zum Teil

# Einleitung

## Motivation

Mobile Endgeräte wie Smartphones oder Tablets sind aus unserem Alltag nicht mehr wegzudenken. Die Verkaufsrekorde werden jährlich gebrochen und die klassischen Desktop PCs sowie Laptops langsam verdrängt(Bitkom 2014). Mobile Anwendungen gewinnen dabei natürlich auch immer mehr an Bedeutung und damit auch die Anzahl an Softwareentwicklern in diesem Bereich. Genauso wie für die „klassischen“ Geräte, existieren auch mehrere unterschiedliche Betriebssysteme für mobile Geräte (Apple iOS, Google Android, Microsoft Windows Phone usw.). Um nun eine möglichst große Nutzerzahl abzudecken, ist es erforderlich die Anwendung für mehrere Betriebssystem / Plattformen zu entwickeln, was aber mit erheblichem Mehraufwand verbunden sein kann, gerade auch wenn man an Erweiterungen oder Korrekturen denkt. Die Apps können nativ für jede Plattform extra entwickelt werden, wodurch sie zwar sehr effizient ausgeführt werden und Zugriff auf alle Geräte- /Betriebssystemfunktionen bieten. Dies jedoch eben nur auf der dafür entwickelten Plattform. Im Gegensatz dazu werden Web Apps im Browser ausgeführt was derzeit mit fast jedem mobilen Endgerät möglich ist, sodass hier nicht für jede Plattform extra Code geschrieben werden muss. Die Benutzerschnittstelle entspricht jedoch eher einer gewöhnlichen Webseite als den nativen Elementen. Einen Kompromiss zwischen beiden Varianten bieten hybride Apps, die zwar auch durch Web Technologien wie bspw. HTML5 realisiert werden, zur Laufzeit jedoch nativ in einem WebView Container laufen und somit auch Zugriff auf viele Geräte / Betriebssystemfunktionen bieten. Welche Variante die passendere ist lässt sich an Merkmalen wie Performanz, Nutzung nativer Funktionalitäten, Installation, Erreichbarkeit, offline Nutzbarkeit usw. festmachen(Davit Svand 2014). Ständige Leistungssteigerungen erweitern die Entwicklungs- und somit Einsatzmöglichkeiten, jedoch werden mobile Geräte hauptsächlich zur schnellen Informationsversorgung genutzt(Sonja Knaub 2014), was größtenteils nur einfaches Abrufen und Darstellen bestimmter Informationen aus einer (halb-) öffentlichen Quelle ist.

Im Rahmen dieser Arbeit wird speziell auf mobile Anwendungen eingegangen, die personenzentrierte Daten aus einem sog. Mashup laden und dem Nutzer zur Verfügung stellen. Für die Mensch und Computer Tagung 2014 wurde bereits eine solche App für iOS entwickelt und veröffentlicht. Diese unterstützt die Teilnehmer der Tagung, indem Informationen zu Personen (Teilnehmern), Organisationen und Inhalten (bspw. Vorträgen) anzeigt werden. Informationen sind bspw:

* Kontaktdaten zu Teilnehmern, Referenten, Organisationen
* Überblick sowie detailliertere Angaben zu einem Vortrag
* Auflistung aller Vorträge zu einer bestimmten Person
* Agenda / Programmplan der Konferenz

Den Nutzern wird außerdem noch die Möglichkeit gegeben selbst Informationen, in Form von Kommentaren und Bewertungen, den Inhalten hinzuzufügen. Das eben genannte Mashup, genauer gesagt Community Mashup, stellt eine Schnittstelle zwischen externen sozialen Dienstleistungen, wie z.B Facebook, Twitter, Flickr, und Endbenutzer Anwendungen, wie der eben genannten App Muc2014 dar. Im Mashup werden personenzentrierte Daten gesammelt, gefiltert und konfigurierbar vereinheitlicht (Lachenmaier 2013).

## Problemstellung

Die Mensch und Computer 2014 Tagungsapp ist derzeit nur nativ für Apples Betriebssystem iOS entwickelt worden, sodass eine Portierung auf andere Plattformen wie Android oder Windows Phone mit viel Entwicklungsaufwand verbunden ist. Dasselbe Problem hat man auch bei Änderungen, die mehrere oder sogar alle angebotenen Plattformen betreffen. Außerdem müssen alle Änderungen / Updates bei Apple erneut den sog. „App Review Process“ durchlaufen der mehrere Tage dauern kann, das verhindert kurzfristige Änderungen.

Ein weiteres Problem ist der langwierige erste Start der App, bei dem alle Informationen zunächst abgerufen, lokal gespeichert und // in Beziehung zu einander gesetzt werden müssen //. Das kann bei großen Datenmengen und schlechter Konnektivität mehrere Minuten dauern, währenddessen die App nicht nutzbar ist.

## Zielsetzung

Ziel dieser Arbeit ist der Entwurf sowie die Umsetzung einer Beschreibungssprache für mobile Anwendungen um die zuvor genannten Probleme zu lösen. Mit der auf JSON basierenden Beschreibungssprache werden Aussehen, Inhalte sowie mögliche Funktionalitäten der App plattformunabhängig definiert. Damit liegt für alle Plattformen eine gemeinsame Beschreibung vor, die jedoch plattformspezifisch interpretiert wird, sodass nur der Interpreter plattformabhängig entwickelt werden muss. Im Implementierungs- bzw. Umsetzungsteil dieser Arbeit wird nur der Interpreter für iOS als Beispiel und zur Evaluation entwickelt. Ein natives Framework an abstrakteren Grundfunktionalitäten ermöglicht das Zuweisen von Aktionen zu Benutzerschnittstellenelementen und definiert damit die erwähnten möglichen Funktionalitäten der App. Unter abstrakten Grundfunktionalitäten kann man sich Einträge ins Telefonbuch, das Öffnen eines Links, den Wechsel zu einer anderen Ansicht vorstellen. Darauf wird später noch genauer eingegangen. Die Anwendungsbeschreibung soll von einer externen Quelle wie dem Community Mashup heruntergeladen und zur Laufzeit interpretiert werden, was bedeutet, dass das Mashup keine reine Informationsquelle ist, sondern auch das Aussehen sowie die möglichen Funtionalitäten der App mithilfe von Templates über eine REST Schnittstelle definiert. Auf diese Art und Weise können jegliche Korrekturen oder Verbesserungen, die nicht den Sprachumfang bzw. dessen Interpretation betreffen, schnell und mit geringem Aufwand für alle Plattformen gleichzeitig umgesetzt werden. Ein weiteres Ziel ist die Verkürzung der App Initialisierung durch sog. „Caching“. Das wird erreicht indem nur dann Beschreibungsinformationen heruntergeladen werden, wenn sie das erste Mal gebraucht werden oder eine Aktualisierung gewünscht bzw. notwendig ist. Bereits geladene Informationen sind lokal gespeichert und können auch offline abgerufen werden. Damit reduzieren sich die benötigten Informationen bei Anwendungsstart auf das Notwendigste, nämlich der ersten Ansicht. Ein weiteres Ziel ist die entwickelte Sprache sowie den dazugehörigen Interpreter für iOS zu nutzen, indem eine App nach Vorbild der MuC2014 umgesetzt wird. Dabei liegt der Schwerpunkt nicht darauf, die App in der Form einzusetzen bzw. zu Veröffentlichen sondern zu Zeigen, dass mit der Sprache alle grundlegenden Funktionen umgesetzt und die Benutzbarkeit, insbesondere die Ladezeiten sich verbessert haben.

Entwurf sowie Implementierung der UI Beschreibungssprache.

* Beschreibungssprache
  + Aussehen
  + Daten
* Übertragung Server-Client
  + Zur Laufzeit änderbar
* Interpretation auf mobile Clientseite
  + Framework
  + Verschiedene Plattformen
* Fähigkeiten
  + Caching
  + Aktualisierung
  + Listen von den oben motivierten Objekten
  + Details dazu
  + Miteinander verlinkt
  + => Browsing App
* Implementierung/Test für iOS bzw. iPhone

## Aufbau der Arbeit

Am Anfang werden aktuelle Beschreibungssprachen anhand einiger Bewertungskriterien näher untersucht:

* Bewertungskriterium a
* B

Abschließend wird begründet warum eine eigene Beschreibungssprache entwickelt werden soll.

Die Entwicklung der Beschreibungssprache wird in Kapitel 3 vorgestellt. Dazu werden zunächst alle Anforderungen beschrieben, die sich aus Kapitel 2 und den Zielen der Arbeit ergeben. Im folgenden Entwurf werden diese dann umgesetzt.

Die Evaluation der entworfenen Sprache beginnt bereits in Kapitel 4 mit der Entwicklung eines Interpreters für iOS. $Architektur , $Entwurf

Nachdem der Interpreter nun entworfen ist, folgt die Umsetzung einer Beispielanwendung nach Vorbild der MuC2014 App. Hierbei werden wieder Anforderungen und die Verbindung zum Community Mashup erläutert. Anschließend folgen Details zum Aufbau und der Funktionsweise der App. Zum Schluss gibt es noch ein Fazit, dass insbesondere auf den erreichter Funktionsumfang und die verbesserte Performanz eingeht.

Am Ende der Arbeit werden die zu Beginn genannten Ziele in Zusammenhang mit den erreichten Ergebnissen gebracht und ein Ausblick auf weitere Entwicklungen sowie Verbesserungen gegeben

# Entwicklung plattformübergreifender mobiler Anwendungen

Für die Entwicklung plattformübergreifender mobiler Anwendungen gibt es 3 verschiedene Ansätze (Heitk, Hanschke 2013):

1. Nutzung einer Beschreibungssprache die zur Laufzeit interpretiert wird
2. Nutzung einer gemeinsamen Codebasis, die plattformspezifisch übersetzt wird
3. Modellgetriebener Ansatz

Der erste Ansatz entspricht den sog. Web und hybriden Apps. Die App wird durch eine gemeinsame Beschreibungssprache für alle Plattformen entwickelt und erst zur Laufzeit interpretiert. Dazu muss jedoch für jede Plattform der Interpreter nativ entwickelt worden sein. Web Apps nutzen diesen Ansatz, wobei als Beschreibungssprachen HTML5, CSS und JavaScript verwendet werden und der Interpreter der native Webbrowser ist. Um deutlich mehr Geräte- / Betriebssystemfunktionen nutzen zu können, kann hier ein eigener Interpreter entwickelt werden, worauf in Kapitel 4 näher eingegangen wird.

Der zweite Ansatz ähnelt dem ersten, jedoch wird die Beschreibung zunächst plattformspezifisch übersetzt und damit gleichzeitig für alle Plattformen eine native Anwendung erstellt. Das hat natürlich den Vorteil, dass nichts zur Laufzeit interpretiert werden muss und die Anwendung somit deutlich effizienter ausgeführt wird.

Der dritte Ansatz versucht, wie der Name schon sagt, über eine Modelldefinition plattformübergreifende Apps zu generieren, wird aber aufgrund fehlender marktreifer Lösungen nicht weiter betrachtet.

Im Folgenden werden zunächst Bewertungskriterien für Entwicklungsframeworks die dem ersten Ansatz entsprechen vorgestellt und anschließend aktuelle Lösungen mithilfe dieser bewertet.

Evtl DSLs für mobile Apps

## Bewertungskriterien für Entwicklungsframeworks

Die folgenden 14 Kriterien wurden aus verschiedenen Quellen zusammengestellt. Die Quellen sind kleine bis mittel große Softwarefirmen, Literatur, Zusammenstellung von Problemen aus Online Communities sowie Erfahrungen aus der Entwicklung eigener Prototyp Apps(Heitk, Hanschke 2013). Die Kriterien werden in 2 Kategorien unterteilt: Infrastruktur und Entwicklung. Mit Infrastruktur sind Kriterien gemeint die zur Laufzeit von Bedeutung sind wie z.B Benutzbarkeit, Aussehen und Funktionsumfang. In der Kategorie Entwicklung befinden sich Kriterien zu den angebotenen / verwendeten Tools, Wartbarbeit usw. Die folgende Tabelle listet die ersten 7 Kriterien der Kategorie Infrastruktur auf:

|  |  |
| --- | --- |
| Lizenz und Kosten | Welche Lizenz wird verwendet?  Freie Software / evtl. sogar Open Source  Nutzung für kommerzielle / private Zwecke  Supportkosten |
| Unterstütze Plattformen | Welche Plattformen werden unterstützt?  Wie gut werden diese unterstützt? |
| Zugang zu plattformspezifischen Funktionen | Gibt es Zugang zu Hardwarekomponenten wie GPS, Kamera usw. ?  Vergleich zu nativer und Web App |
| Zukunftsfähigkeit | Kurze sowie reguläre Updatezyklen  Unterstützung aktueller mobiler Betriebssysteme  Aktive Community sowie viele Entwickler  Kommerzielle Unterstützer |
| Erscheinungsbild und Benutzbarkeit | Unterstützung nativer / Web Benutzerschnittstellen  Anwendungen eigenen sich zur mobilen Nutzung |
| Anwendungsgeschwindigkeit | Benötigte Startzeit  Geschwindigkeit zur Laufzeit |
| Distribution | Veröffentlichungsprozess  Nutzung der plattformspezifischen App Stores  Updatemöglichkeiten |

Abbildung 1:Bewertungskriterien zur Infrastruktur mobiler Anwendungsentwicklungsframeworks und Umgebungen

Die folgende Tabelle listet die 7 Bewertungskriterien der Kategorie Entwicklung auf:

|  |  |
| --- | --- |
| Entwicklungsumgebung | Funktionalitäten (Autovervollständigung)  Vorhandene / unterstütze Tools (IDE , Debugger, Emulator)  Einfachheit der Umgebungseinrichtung |
| GUI Design | Möglichkeiten Benutzerschnittstellen zu erstellen (Tools zur Unterstützung und Test)  GUI Vorschau vorhanden? |
| Einfachheit der Entwicklung | Qualität der API und Dokumentation  Erlernbarkeit (Tutorials, Beispiele)  Nutzung bekannter Softwareentwicklungsparadigmen |
| Wartbarbeit | Zur Evaluation wird die Anzahl der benötigten Codezeilen verwendet. |
| Skalierbarkeit | Unterstützung größerer Entwicklungsteams und Projekte  Modularität des Frameworks und der Anwendung |
| Möglichkeiten für zukünftige Entwicklungen | Wiederverwendung von Code |
| Entwicklungsdauer und Kosten | Dauer des Entwicklungsprozesses sowie Faktoren die ihn behindern  Kosten werden nicht explizit angegeben, stehen in Zusammenhang mit der Entwicklungsdauer |

App

UI

Einsatzzeit im Entwicklungsprozess

Generierung?

Interpretation zur Laufzeit?

## Bewertung aktueller Entwicklungsframeworks

Die Studie „Evaluation of mobile Web applications“ (Heitk 2013) bewertet 4 aktuelle Lösungen / Frameworks im Hinblick auf die zuvor genannten Kriterien. Als vierte Lösung werden die nativ entwickelten Anwendungen betrachtet, auf die hier jedoch nicht mitaufgeführt werden, da diese nicht dem ersten Ansatz entsprechen, da sie keine Beschreibungssprache zur Laufzeit interpretieren. Die erste Lösung sind die gewöhnlichen Web Apps. Die zweite ist ein bekanntes Open Source Framework für die Entwicklung von hybriden Apps: PhoneGap (PhoneGap 2014). Die Anwendungen werden mit HTML, CSS und JavaScript entwickelt, jedoch wird der Webinterpreter um eine JavaScript API in Kombination mit einer Brücke zur Hardware erweitert. Ein Beispiel für die Nutzung eines eigenen Interpreters liefert das Framework Titanium Mobile (Titanium Mobile 2014). Die Benutzerschnittstelle, Logik und Daten werden in JavaScript programmiert und zur Laufzeit interpretiert.

// TEXT LÖSUNG TABELLE //Die erste Lösung sind die gewöhnlichen Web Apps. Die meisten Tools und Frameworks sind OpenSource und es fallen nur für einige Ausnahmen Gebühren bzw. Kosten an. Support erhält man von großen aktiven Communities, jedoch ist es üblich für OpenSource Projekte mit Support Geld zu verdienen. Weitere Kosten können für den Betrieb der Webseite anfallen. Es werden alle Plattformen unterstützt die einen Webbrowser besitzen. Der Zugang zu plattformspezifischen Funktionen hat sich durch HTML5 verbessert, da nun das lokale speichern von Daten möglich ist. Abspielen von Video und Audio Dateien sowie die Nutzung von Multi-Touch Gesten ist ebenfalls möglich. Aufgrund der Nutzung von Sprachen wie HTML, CSS und JavaScript ist die Lösung sehr wohl zukunftssicher, jedoch ist dies je nach Nutzung externer Frameworks (für JavaScript bspw.) nicht mehr sichergestellt. Das Erscheinungsbild wird über CSS definiert und entspricht nicht dem der nativen Apps. Zur Benutzbarkeit trägt HTML5 bei, da aufgrund der lokalen Speichermöglichkeiten es nun möglich ist den alten Zustand der Web App beim erneuten Öffnen wiederherzustellen. Die Anwendungsgeschwindigkeit ist natürlich abhängig von der Konnektivität, jedoch ist die Ausführung zur Laufzeit durch die optimierten Browser sehr performant. Die Distribution ist sehr einfach, da die Nutzer nur die URL eingeben müssen und damit auch die aktuellste Version der Web App erhalten. // ENDE TEXTLÖSUNG TABELLE//

|  |  |  |
| --- | --- | --- |
| Web Apps | Phone Gap | Titanium Mobile |
| Lizenz und Kosten | | |
| Meisten Tools und Frameworks Open Source  Support durch große Communities  Kosten für den Betrieb der Webseite | Open Source Software unter Apache Lizenz 2.0 / GPL/MIT Lizenz  Anbieter verkauft Support zwischen 25$ und 2000$ pro Monat | Community Edition ist kostenlos, der Funktionsumfang jedoch limitiert  Proprietäre Edition enthält alle Funktionen sowie Support  Eher geschlossenes System |
| Unterstütze Plattformen | | |
| Alle Plattformen mit Webbrowser | iOS, Andorid, BlackBerry OS, Windows Phone 8, Ubuntu, Firefox OS (Phone Gap 2014) | iOS, Android, BlackBerry (Titanium Mobile 2014) |
| Zugang zu plattformspezifischen Funktionen | | |
| Verbessert durch HTML5: lokaler Speicherplatz begrenzt nutzbar  Video, Audio Dateien abspielbar, Nutzung Multi Touch Gesten | Zugang zu fast allen Hardwarefunktionen[[1]](#footnote-1) | Ähnlich zu Phone Gap |

|  |  |  |
| --- | --- | --- |
| Zukunftsfähigkeit | | |
| Ja - Nutzung fest etablierter Standards (HTML, CSS, Java Script) | Aktive Community, Unterstützer: Adobe und IBM, als Apache Projekt geführt, regelmäßige Updates | Große Community und viele Entwickler, folgt aktuellen Trends, nutzt neue Funktionalitäten der Betriebssysteme, regelmäßige Updates  Abhängig von einer einzigen Firma |
| Erscheinungsbild und Benutzbarkeit | | |
| CSS definiert Erscheinungsbild, native Benutzerschnittstellenelemente nicht möglich  HTML5 verbessert Benutzbarkeit durch lokale Speichernutzung (bspw. Wiederherstellung Appzustand nach Pause) | Nutzt keine nativen Benutzerschnittstellenelemente, CSS definiert plattformspezifisches Aussehen(hoher Aufwand)  Nutzt Events um Laufzeitveränderungen zu behandeln (z.B bei Anruf, Pause) | Über Java Script werden native Benutzerschnittstellenelemente genutzt – erfordert viel Kenntnisse der API  Gute Benutzbarkeit implementierbar |
| Anwendungsgeschwindigkeit | | |
| Abhängig von der Netzwerkverbindung, optimierte Browser ermöglichen performante Ausführung | Vergleichbar mit nativer App | Startzeit ähnlich zu anderen Frameworks, zur Laufzeit jedoch Probleme bei zu vielen Objekten |
| Distribution | | |
| Sehr einfach: den Nutzern die URL mitteilen | Normalerweise im jeweiligen App Store (Apple verbietet reine Web Apps) | Im jeweiligen App Store |
|  | | |

|  |  |  |
| --- | --- | --- |
| Entwicklungsumgebung | | |
| Nutzung vorhandener Umgebungen aus dem Bereich Webentwicklung | Ähnlich zu Web Apps, nicht jede IDE unterstützt die PhoneGap API | Eclipse basierte DIE Titanium Studio, Autovervollständigung für die Titanium API, unterstützt Ausführung sowie Veröffentlichung im App Store, einfache Einrichtung |
| GUI Design | | |
| Identisch zur Webentwicklung | Identisch zur Webentwicklung (z.B Adobe Dreamweaver) | Mit JavaScript programmiert – mühsam und zeitaufwendig, kein Editor vorhanden |
| Einfachheit der Entwicklung | | |
| Viele Dokumentationen und Beispiele / Tutorials in HTML, CSS und Java Script , Beachtung Limits mobiler Geräte (CPU, GPU usw.) | Gute Dokumentation, viele Beispiele, wenig Framework spezifisches Wissen | Gute Dokumentation, wenig Codebeispiele, viel Framework spezifisches Wissen erforderlich |
| Wartbarkeit | | |
| Wenig Codezeilen nötig | Ähnlich Web Apps, für Hardwarezugang wird extra Code benötigt, dennoch insgesamt sehr kurz und klar strukturiert | Viele Codezeilen, jedoch sehr modular aufgebaut |
| Skalierbarkeit | | |
| Modularisierbar, Aufteilung in mehrere Dateien | Siehe Web Apps | sehr einfache Modularisierung |
| Möglichkeiten für zukünftige Entwicklungen | | |
| Gut, einfach portierbar zur PhoneGap App, kann auch im WebView von Titanium Mobile angezeigt werden | Ohne Nutzung von plattformspezifischer Funktionen ähnlich Web Apps | Schlecht da viel Titanium spezifischer Code benutzt wird |
| Entwicklungsdauer und Kosten | | |
| Web App Prototype Entwicklung war am kürzesten, mächtige Entwicklerwerkzeuge beschleunigen und begünstigen den Prozess | Ähnlich zu Web Apps mit zusätzlichem Entwicklungsaufwand für plattformspezifische Funktionalitäten | Viel spezielle Erfahrung nötig, verlangsamte Entwicklung der UI aufgrund der notwenigen Ausführung im Emulator oder auf dem Gerät |

Abbildung 2: Bewertungsergebnisse

## Zusammenfassung und Fazit

Die Ergebnisse zeigen die Stärken und Schwächen der unterschiedlichen Frameworks und des rein webbasierten Ansatzes. Es gilt nun zu prüfen inwiefern die aktuellen Lösungen für die eigene verwendetet werden können. Die Web App kann zwar lokalen Speicherplatz nutzen, momentan ist dieser aber auf 5MB[[2]](#footnote-2) beschränkt, sodass das Anfangs erwähnte „caching“ nicht umgesetzt werden kann. Der Zugriff auf Funktionen wie das Telefonbuch sind hier auch nicht möglich womit klar ist, dass eine rein webbasierte App zu wenig Funktionalität besitzt. PhoneGap und Titanium Mobile hingegen bieten den gewünschten Zugriff auf die Geräte / Betriebssystem Funktionalitäten. PhoneGap überzeugt gerade im Punkt Einfachheit der Entwicklung bietet jedoch keine nativen Benutzerschnittstellenelemente an, sodass die App eher wie eine Web App wirkt. Darüber hinaus //schlechte Speicher anbindung

Schluss ziehen warum eigene Sprache

Gründe für den Entwurf einer neuen Sprache

# Beschreibungssprache

## Anforderungen

Siehe Kapitel 2

Erweiterbarkeit

Generisch für mobile Plattformen

## Entwurf

# iOS Interpreter

Sprachmerkmal -> UITableViewController

## Architektur

## Entwurf

Technologieauswahl

Restkit

# Beispiel iOSTemplateLanguage Applikation

Um die entwickelte Beschreibungssprache einer ersten Evaluation zu unterziehen, bedarf es einer Beispiel App, deren Entwurf und Umsetzung im Folgenden nun beschreiben wird. Als Vorlage dient die App Muc2014[src], welche bereits im Rahmen einer Masterarbeit entwickelt wurde. Ziel der Beispiel App ist es, möglichst viele unterschiedliche Funktionalitäten der Vorlage mithilfe der Beschreibungssprache zu realisieren.

## Anforderungen

Das sind Anforderungen an konkrete App

* Datenquelle: CommunityMashup MUC
* JSON Parsing und Mapping
* Persistenz der Daten
* Geschwindigkeit, Verwendung mit geringer Bandbreite
* UI Design stark an Vorlagen App orientiert

## CommunityMashup

Architektur

Funktionsweise

Schnittstellen

Templates

## Umsetzung/Funktionsweise

Hier Schreenshots der App mit Beschreibung

## Fazit

### Erreichter Funktionsumfang

### Verbesserte Performanz

# Zusammenfassung und Ausblick

Literaturverzeichnis

Phanouriou, C. (2000). UIML: A Device-Independent User Interface Markup. Retrieved from http://vtechworks.lib.vt.edu/handle/10919/28625

1. http://phonegap.com/about/feature/ [↑](#footnote-ref-1)
2. http://www.w3schools.com/html/html5\_webstorage.asp [↑](#footnote-ref-2)