**Zürcher Hochschule für angewandte Wissenschaften**

**Diplomstudium Informatik**

****

**Seminararbeit 5. Semester**

**Entwickeln von Anwendungen für Handheld B**

**Evaluation von 10 Frameworks für Touch Devices**

|  |  |
| --- | --- |
| Autoren | Oliver Aeschbacher  René Kamer |
| Dozent | Christian Vils |
| Projektstart | 4. Oktober 2012 |
| Projektpräsentation | 16. Januar 2013 |

Inhaltsverzeichnis

[1. Aufgabenstellung 5](#_Toc343016040)

[1.1. Ausgangslage 5](#_Toc343016041)

[1.2. Ziele der Arbeit 5](#_Toc343016042)

[1.3. Erwartetes Resultat 5](#_Toc343016043)

[1.4. Motivation 6](#_Toc343016044)

[2. Planung und Termine 7](#_Toc343016045)

[2.1. Projektplan 7](#_Toc343016046)

[2.2. Termine 7](#_Toc343016047)

[2.3. Soll-/Ist-Vergleich 8](#_Toc343016048)

[3. Einleitung 8](#_Toc343016049)

[3.1. Vorgehen 9](#_Toc343016050)

[3.1.1. Vorgehen: Evaluation der Frameworks 9](#_Toc343016051)

[3.1.2. Vorgehen: Schreiben der Lupen-App 11](#_Toc343016052)

[3.2. Aufteilen der Aufgaben 11](#_Toc343016053)

[3.2.1. Aufteilung der Evaluation 11](#_Toc343016054)

[3.2.2. Aufteilung der Programmierung 12](#_Toc343016055)

[4. Evaluation der Frameworks 12](#_Toc343016056)

[4.1. Appcelerator Titanium 12](#_Toc343016057)

[4.1.1. Einleitung 12](#_Toc343016058)

[4.1.2. Wie funktioniert das Framework 12](#_Toc343016059)

[4.1.3. Evaluation 12](#_Toc343016060)

[4.1.4. Finale Bewertung 15](#_Toc343016061)

[4.2. Appcelerator Titanium 15](#_Toc343016067)

[4.2.1. Einleitung 15](#_Toc343016068)

[4.2.2. Wie funktioniert das Framework 16](#_Toc343016069)

[4.2.3. Evaluation 16](#_Toc343016070)

[4.2.4. Finale Bewertung 19](#_Toc343016071)

[4.3. PhoneGap 19](#_Toc343016072)

[4.3.1. Einleitung 19](#_Toc343016073)

[4.3.2. Wie funktioniert das Framework 19](#_Toc343016074)

[4.3.3. Evaluation 20](#_Toc343016075)

[4.3.4. Finale Bewertung 22](#_Toc343016076)

[4.4. Sencha Touch 22](#_Toc343016077)

[4.4.1. Einleitung 22](#_Toc343016078)

[4.4.2. Wie funktioniert das Framework 23](#_Toc343016079)

[4.4.3. Evaluation 23](#_Toc343016080)

[4.5. Gegenüberstellung der Frameworks 25](#_Toc343016081)

[5. Projekt Lupen-App 25](#_Toc343016082)

[5.1. Umgebung des Projekts 25](#_Toc343016083)

[5.1.1. Umgebung der Evaluation 25](#_Toc343016084)

[5.1.2. Umgebung der Programmierung / Framework 25](#_Toc343016085)

[5.1.2.1. Wahl des Frameworks 25](#_Toc343016086)

[5.1.2.2. Tools 25](#_Toc343016087)

[6. Quellen 25](#_Toc343016088)

[7. Abbildungsverzeichnis 26](#_Toc343016089)

# Aufgabenstellung

# Ausgangslage

Bereits die ersten Mobiltelefone enthielten kleine Anwendungen, wie Kalender,

Taschenrechner oder Spiele. Diese waren jedoch vom Hersteller exakt für die jeweiligen

Geräte konzipiert und fix installiert.  
Das erscheinen moderner Devices wie iPhone oder Android-Geräten erlaubt nun, dass man den Funktionsumfang der Geräte beinahe beliebig erweitern kann mit selbstgeschriebenen und/oder fremden Apps.  
Aus der heutigen Zeit sind Touch-Devices nicht mehr wegzudenken. Handys wie das iPhone oder das Samsung Galaxy beziehungsweise Tablets wie das iPad oder das Asus Eee Pad bieten eine schier unendliche Menge an Applikationen. Um diese zu entwickeln bedarf es der entsprechenden Frameworks.

# Ziele der Arbeit

Das Ziel dieses Projekts ist, auf der Basis von <http://woorkup.com/2010/08/25/10-useful-frameworks-to-develop-html-based-webapps-for-touch-devices/> eine Evaluation der 10 dort vorgestellten Frameworks zu vollziehen auf theoretischer Basis:

* Einfachheit
* Entwicklungsumgebung
* Support
* Tutorials
* Building
* Testing
* Verbreitung

und diese Erkenntnisse zur Verfügung zu stellen.

Auf der Basis dieser Evaluation werden 1 - 2 dieser Frameworks herangezogen, um eine kleine Lupen-Anwendung zu schreiben für Android-Telefone und darin die Erkenntnisse der Evaluation einfliessen zu lassen.   
Die Lupen-Anwendung soll eine Hilfe sein für Leute, welche nicht mehr so gut sehen können und mittels der Lupen-App kleine Schriften oder ähnliches wieder lesen können, welche zuvor von der Device-Kamera abfotografiert wurden (im Idealfall sogar eine Lupe in Realtime).

# Erwartetes Resultat

Dokument, welches die 10 Frameworks gegenüberstellt in den unter Aufgabenstellung genannten Punkten.   
Des Weiteren eine App, welche unter den 1 - 2 verheissungsvollsten Frameworks geschrieben wurden. Dazu wird ein Teaser erstellt und der ganzen Klasse zur Verfügung gestellt. Ebenfalls wird eine Präsentation von ca 40 - 60 Minuten gehalten vor der Klasse (Die Zeitdauer bestimmt sich dadurch, dass an diesem Projekt 2 Personen Arbeiten)

# Motivation

Schon länger beschäftigt uns das Thema Handhelds und die Programmierung dazu. Dieses Seminar gibt uns nun die Möglichkeit, dieses Wissen aufzubauen und zu festigen.  
Im Speziellen interessieren uns Frameworks (nicht nur in Bezug auf Handhelds) und deren Möglichkeiten, unsere Arbeit zu vereinfachen und zu vereinheitlichen. Nicht zuletzt ist auch die Möglichkeit spannend, die unter den Frameworks geschriebenen Programme auf verschiedenen Plattformen laufen zu lassen, quasi 7 Fliegen mit einer Klatsche (iOS, Android, Blackberry, Windows Phone, Palm, WebOS, Bada, Symbian).  
Daher haben wir uns entschieden, 10 dieser Frameworks einem Evaluationsprozess zu unterziehen und unter ausgewählten Frameworks eine kleine App zu realisieren.  
Da leider unsere Kriegskasse ein wenig eingeschränkt ist, wird diese Realisierung ausschliesslich auf Android stattfinden.

# Planung und Termine

# Projektplan



Der Projektplan wurde mit GanttProject entwickelt. Leider hat die aktuelle Version einen Bug und somit können die Vorgänger sowie Nachfolger eines Arbeitszeitraumes nicht visuell dargestellt werden.  
Der Projektplan in dieser Form zeigt an, in welchem Zeitraum der jeweilige Task zu erledigen ist, für die exakte Stunden-Aufteilung wird auf den Soll-/Ist-Vergleich verwiesen.  
Da der Plan aufgrund seiner Grösse in diesem Dokument etwas schlecht lesbar ist, ist dieser noch als JPG im „Documents“ Ordner des Projekts abgelegt sowie auch das Gantt-File, welches mit GanttProject geöffnet werden kann.

# Termine

Kick-Off: 19.09.2012

Projektfreigabe: 04.10.2012

Zwischenbesprechung: 12.12.2012

Präsentation: 16.01.2012

# Soll-/Ist-Vergleich

Generell werden die Projekt-Mitarbeiter mit Kürzeln versehen, René Kamer mit RKA, Oliver Aeschbacher mit OLA.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Zuständig | Soll | Ist |
| **Projektplanung** | - | **11** | **10** |
| Erstes Informieren über die Frameworks | RKA + OLA | 2 | 2 |
| Vorgehen besprechen | RKA + OLA | 1 | 2 |
| Aufteilung der Aufgaben | RKA + OLA | 2 | 1 |
| Erst-Erstellung der Dokumentation | RKA | 3 | 2 |
| Projektplan erstellen | RKA | 3 | 3 |
| **Erstellung des Teasers** | OLA | **3** |  |
| **Analyse der Frameworks** | - | **70** |  |
| Analyse von Appcelerator Titanium |  | 7 |  |
| Analyse von Sencha Touch |  | 7 |  |
| Analyse von Sproutcore Touch |  | 7 |  |
| Analyse von PhoneGap |  | 7 |  |
| Analyse von Rhodes |  | 7 |  |
| Analyse von iUi |  | 7 |  |
| Analyse von iWebKit |  | 7 |  |
| Analyse von XUI |  | 7 |  |
| Analyse von jQPad |  | 7 |  |
| Analyse von jQuery Mobile |  | 7 |  |
| **Vergleich der Frameworks** | - | **6** |  |
| Zusammenstellung der Resultate |  | 2 |  |
| Diskussion der Frameworks (inkl Doku) |  | 4 |  |
| **Projektumgebung erstellen** | - | **8** |  |
| Projektumgebung Framework 1 |  | 4 |  |
| Projektumgebung Framework 2 |  | 4 |  |
| **Lupen APP** | - | **30** |  |
| Erstellen mit Framework 1 |  | 8 |  |
| Building mit Framework 1 |  | 3 |  |
| Testing mit Framework 1 |  | 4 |  |
| Erstellen mit Framework 2 |  | 8 |  |
| Building mit Framework 2 |  | 3 |  |
| Testing mit Framework 2 |  | 4 |  |
| **Dokumentation abschliessen** | RKA + OLA | **4** |  |
| **Präsentation + Handzettel erstellen** | RKA + OLA | **10** |  |
| **Reserve** |  | **8** |  |
| **Total** |  | **150** |  |

Alle Einheiten sind in Stunden angegeben.

# Einleitung

Dieser Teil beschreibt, wie wir vorgegangen sind in der Projekt-Anfangsphase, um die detaillierte Vorgehensweise der Evaluation der Frameworks zu beschreiben sowie die detaillierte Vorgehensweise der Entwicklung der Lupen App voranzutreiben. Hierbei beschreiben wir die Kriterien und wie sie auf die Evaluation angewendet werden sowie die Gewichtung derselbigen. Des Weiteren wird der Entwicklungszyklus der Lupen-App beschrieben sowie die Anforderungen, welche an die App gestellt werden.  
Daraus resultierend werden die Aufgaben auf die jeweiligen Projektmitarbeiter aufgeteilt, was im zweiten Abschnitt dieses Kapitels dargestellt ist, sowohl für die Evaluation der Frameworks als auch für das Schreiben der Lupen-App.

# Vorgehen

# Vorgehen: Evaluation der Frameworks

Die Frameworks werden jeweils durch dieselben 8 Kriterien bewertet: Einfachheit, Entwicklungsumgebung, Support, Dokumentation, Tutorials, Building, Testing und Verbreitung. Zum Schluss werden alle Frameworks einander gegenübergestellt in einer Entscheidungsmatrix, in welcher die jeweiligen Kriterien mit Zahlen von 1 - 10 bewertet werden und danach die Summe gebildet, um die 1 - 2 geeignetsten Frameworks zu ermitteln. Die Frameworks mit der höchsten Gesamtpunktzahl werden für die Lupen-App berücksichtigt.

Die Kriterien werden nachfolgend einzeln beschrieben betreffend der Bewertung:

* Einfachheit
  + Subjektives Empfinden: Die Einfachheit des eingesetzten Frameworks ist schwerlich objektiv zu beschreiben, da unterschiedliches Vorwissen oder generelle Präferenzen stark in die Bewertung mit einfliessen können. Deshalb wird hier ausnahmsweise auch das subjektive Empfinden als Kriterium miteinbezogen. Dies ist aber das schwächere Kriterium als die Anzahl benötigter Technologien, um die nicht-technische Beschreibung des subjektiven Empfindens ein wenig abzuschwächen.
  + Anzahl benötigter Technologien: Man kann verschiedene Technologien einsetzen. Alle Frameworks basieren auf HTML, CSS und JavaScript, jedoch in unterschiedlicher Ausprägung. Weiters gibt es noch verschiedene zusätzliche Möglichkeiten wie z.B. XML im W3C Widget Specification Format oder ähnliches. Als ideal werden 2-3 Technologien angesehen, welche zum Schreiben einer App ausreichen. Zu wenige Möglichkeiten in der Basis sind schlecht sowie auch zu viele Technologien, welche in einer Basisanwendung eingesetzt werden müssen. Dem wird in diesem Kriterium Rechnung getragen.
  + Intuitive Benutzbarkeit: Werden die eingesetzten Technologien so eingesetzt, wie man sie aus dem Alltag kennt oder werden diese anders benutzt, umgeordnet, eventuell sogar eingeschränkt in der Benutzung? Ein hohes Mass an Normalgebrauch gibt eine schnelle Einarbeitungszeit, weshalb dies hoch zu gewichten ist.
* Entwicklungsumgebung
  + Aufzählung: Eine Aufzählung der Entwicklungsumgebungen. Mehr Möglichkeiten bieten Entwicklergruppen grössere Flexibilität: Der Web-Programmierer schreibt lieber mit einem Web-Programmier Tool, der alteingesessene Programmierer eher in einem Plaintext-Editor.
  + Alternativen/Erweiterungen: Gibt es Alternativen (evtl. auch etwas exotische) welche unter dem Punkt „Aufzählung“ beschriebenen Tools ergänzen/verbessern?
  + Hilfestellungen durch Framework: Wie weit geben die Frameworks Hilfestellung, bieten z.B. Text-Completion, auf das Framework zugeschnittene Fertigbausteine oder Hinweise an? Je mehr Hilfestellungen existieren, desto besser wird dieser Punkt bewertet.
* Support
  + Supporthotline: Existiert eine Supporthotline? Wie schätzt man deren Qualität ein, ist diese in Landessprache verfügbar?
  + Online-Möglichkeiten: Gibt es weitere Möglichkeiten den Support zu erreichen? Z.B. Mail, Chat, Live-Support oder ähnliches? Gibt es ein Entwicklerforum oder eine Online-Gruppe, in welchen man sich austauschen kann? Je mehr Möglichkeiten existieren, umso besser wird hier bewertet.
  + Preis: Was sind die Kosten für den Support? Wie wird dies auf die Projektgrösse berechnet, auf die Anzahl Entwickler? Was ist das Preismodell? Das Preis/Leistungsverhältnis wird bewertet.
  + Anleitungen im Netz: Gibt es anständige Anleitungen, welche im Netz verfügbar sind? Sind sie vom Anbieter erstellt oder muss auf Dritt-Anleitungen zurückgegriffen werden? Je mehr Anleitungen von guter Qualität vorhanden sind, umso besser wird dieses Kriterium bewertet.
* Dokumentation
  + Verfügbarkeit: Wie ist die Verfügbarkeit der Dokumentation? Gibt es sie nur als Buch oder sind sie online verfügbar? Je weniger Aufwand betrieben werden muss, umso besser wird dieses Kriterium bewertet.
  + Struktur: Hat die Dokumentation eine Struktur oder ist es ein simples Textfile ohne Möglichkeiten zur Suche? Ist es einfach, etwas zu finden in der Dokumentation? Je einfacher etwas zu finden ist, umso besser wird hier bewertet.
  + Ausführlichkeit: Wie ausführlich ist die Dokumentation? Sobald man eine Ausführlichkeit erreicht, welche das Verständnis erleichtert, wird hier gut bewertet. Eine zu hohe Ausführlichkeit kann jedoch kontraproduktiv sein, was ebenfalls in diesen Punkt einfliesst.
* Tutorials
  + Google Tutorial Suche: Hier wird eine simple Google-Suche gemacht der Form: „<Framework-Name>“ + „Tutorial“. Je mehr Treffer Google zurückgibt, umso mehr Punkte werden vergeben. 10 Punkte für die meisten Treffer, 1 Punkt für die wenigsten Treffer.
  + Qualität der ersten 5 Google-Hits: Es wird die Ausführlichkeit und Verständlichkeit der ersten 5 Google Treffer bewertet. Tutorials sollten einfach aufgebaut und simpel sein, jedoch trotzdem ein gewisses Mass an Komplexität aufweisen. Das richtige Mittelmass sollte angestrebt werden und wird entsprechend bewertet.
* Building
  + Wie einfach /aufwändig ist der Prozess: Ist das Building einfach zu machen? Muss man hohe Aufwände betreiben, um den Build zu erstellen? Je einfacher der Prozess umso besser wird bewertet.
  + Verschiedene Möglichkeiten: Existieren verschiedene Möglichkeiten, den Build zu erstellen? Z.B. Lokal, Online, per Post?
  + Build-Tool: Stellt der Framework-Anbieter Build-Tools zur Verfügung oder muss auf ein IDE oder ähnliches zurückgegriffen werden (Cross-Platform). Gibt es ein Build-Tool wird dies positiv bewertet.
* Testing
  + Wie kann getestet werden: Sind Testing-Tools oder Frameworks vorhanden? Kann anständig Debugging betrieben werden? Muss man auf Methoden wie Print-Outs zurückgreifen? Je mehr Möglichkeiten das Framework von sich aus bietet, desto besser wird hier bewertet.
  + Methodik: Sind die einsetzbaren Test-Möglichkeiten methodisch oder strukturiert? Konformität mit der gängigen Lehrmeinung wird hier hoch bewertet.
* Verbreitung
  + Google Suche: Hier wird eine simple Google-Suche gemacht mit dem Namen des Frameworks. Je mehr Treffer Google zurückgibt, umso mehr Punkte werden vergeben. 10 Punkte für die meisten Treffer, 1 Punkt für die wenigsten Treffer.
  + Anzahl Apps: Wie viele mit diesem Framework geschriebene Apps sind verfügbar. Je mehr Apps damit erstellt wurden, umso besser fällt die Punktzahl aus.
  + Persönliche Erfahrung: Hat man schon von diesem Framework gehört? Gibt es gute oder schlechte Kritiken, welche man schon gelesen hat? Eine Zusammenfassung der Eindrücke spiegelt sich in der vergebenen Punktzahl wieder.
* Spezielles: Gibt es noch irgendwelche speziell nennenswerte Features oder Vorteile, welche dieses Framework auszeichnen? Hiermit können maximal 2 Zusatzpunkte vergeben werden.

# Vorgehen: Schreiben der Lupen-App

# Aufteilen der Aufgaben

# Aufteilung der Evaluation

Die Aufteilung erfolgt der Logik nach mit 50/50 aufgeteilt auf die Projektmitarbeiter. Dabei wurde der Grösse der jeweiligen Frameworks einigermassen Rechnung getragen, dass die Arbeitslast in etwa gleich verteilt ist, wie in der Planung bereits berücksichtigt.

|  |  |
| --- | --- |
| Analyse von Appcelerator Titanium | RKA |
| Analyse von Sproutcore Touch | RKA |
| Analyse von PhoneGap | RKA |
| Analyse von Rhodes | RKA |
| Analyse von iUi | RKA |
| Analyse von iWebKit | OLA |
| Sencha Touch | OLA |
| Analyse von XUI | OLA |
| Analyse von jQPad | OLA |
| Analyse von jQuery Mobile | OLA |

# Aufteilung der Programmierung

# Evaluation der Frameworks

# Appcelerator Titanium

# Einleitung

**F:\Schule\HSZ-T\3. Jahr\Handheld\Semesterarbeit\Handheld_Repo\Media\Appcelerator Titanium\Appcelerator_Logo.JPG**Appcelerator Titanium ist eine Plattform, um mobile Apps, Tablet-Apps sowie Desktop Apps zu entwickeln. Entwickelt wurde es von Appcelerator und der Öffentlichkeit wurde es im Jahre 2008 präsentiert. Es unterstützt verschiedene Systeme: iPhone, iPad, Android Devices und neu werden dank der strategischen Partnerschaft mit RIM auch Blackberry Devices unterstützt.   
Es werden im Wesentlichen zwei Technologien eingesetzt: JavaScript sowie Titanium API. Oftmals wird Appcelerator Titanium als Cross-Compiler bezeichnet, dies stimmt jedoch nur begrenzt. Vielmehr wird aus dem JavaScript mit der Titanium API auf dem Gerät der gesamte Code interpretiert, daher wäre eine Interpretiersprache wohl die bessere Bezeichnung.  
Mittlerweile haben sich sehr viele Entwickler mit Appcelerator Titanium beschäftigt, soeben hat das Unternehmen die Grenze von 300‘000 registrierten Entwicklern durchbrochen.

# Wie funktioniert das Framework

Die Entwicklung innerhalb dieses Frameworks basiert hauptsächlich auf JavaScript, hinzu kommt noch die Titanium-Schnittstelle, über welche man Zugriff auf über 5‘000 Schnittstellen der jeweiligen Hersteller haben soll. Es wird NICHT nativ entwickelt mit z.B. Objective C oder ähnlichem, sondern ausschliesslich in der genannten Umgebung JavaScript und Titanium API.   
Eine Kompilierung der jeweiligen Codeteile ist nicht notwendig. Die Files können, so wie sie sind, direkt auf das Device geladen werden und werden erst auf dem Gerät interpretiert. Dies gibt eine kleine Verzögerung beim Starten der Apps, welche sich jedoch Testberichten zufolge nicht all zu stark auswirkt.



# Evaluation

**Einfachheit:** Hier liegt wohl der grösste Knackpunkt von Appcelerator Titanium: Es ist ziemlich undurchsichtig auf den ersten sowie auf den zweiten Blick. Der Einsteiger findet zwar viele Code-Snippets, doch die eingesetzten Beispiele setzen sofort auf einem sehr hohen Level an und zeigen eher Features auf als dass sie beim Einstieg helfen. Zwar beschränken die zwei eingesetzten Technologien JavaScript und Titanium API den Lernaufwand, doch die sehr komplexen Möglichkeiten überfordern ziemlich schnell, eine gründliche Einarbeitung ins Framework ist zwingend notwendig, um diesen Berg an Informationen einigermassen überblicken zu können.  
Die gänzliche Absenz von strukturierten Elementen wie z.B. bei HTML verwirren zu Anfang sehr stark, erinnern jedoch in ihrer Art der GUI-Programmierung mit Java. Erfahrene JavaScript Programmierer haben hier bestimmt einen klaren Vorteil, doch auch sie müssen sich noch mit der Titanium API beschäftigen. Jedoch ist positiv zu bewerten, dass mit relativ wenigen Technologien gearbeitet wird, man also nicht allzu stark links und rechts schauen muss, um eine Implementation zu erreichen.  
Weiters ist die Homepage sehr business-lastig aufgebaut, auf den ersten Blick ist nicht ersichtlich, wie man als Entwickler an die benötigte Information kommt. Es wird eher der Fokus auf Verkauf, Marketing, Möglichkeiten und Zahlen gelegt, hier Punkten andere Frameworks mit einer wesentlich entwickler-orientierteren Präsentation der Fakten.  
Final lässt sich sagen, dass eine steile Lernkurve vonnöten ist, um sich schnell im Framework zurechtzufinden, dies schlägt sich auch in der Punktebewertung nieder, die eher mässig ist.

**Entwicklungsumgebung:** Die Entwicklungsumgebung ist fest vorgegeben, sie nennt sich Titanium Studio. Obwohl noch ein SDK herunterladbar ist, wird von diversen Seiten abgeraten mit normalen Editoren zu entwickeln, weil damit viele Möglichkeiten verloren gehen, unter anderem das direkte Ausführen in einem Emulator, welcher im Titanium Studio eingebettet ist.  
Obwohl es aussieht wie ein Eclipse, auch in seiner Funktionalität stark daran angelehnt ist, kommt das IDE nicht von der Eclipse Foundation. Entwickelt wurde es aus dem Aptana Studio, welches von Appcelerator im Januar 2011 aufgekauft worden ist.   
Titanium Studio bietet viele Annehmlichkeiten wie Code-Completion, Dive-In Code, Compiler Correction marks, automatische Prüfung von Plausibilität. Manchmal sogar etwas zu viel, denn viele Funktionen werden mittels des Titanium API ein wenig versteckt, man möchte öfters viel granularer auf die jeweiligen Funktionen zugreifen.  
Viele Entwickler raten von Titanium Studio ab, es wird oft verschrien als Prototyping-Sprache, die meisten der Apps würden nach dem Prototyping-Stadium nochmals in Native-Programmiersprachen geschrieben. Dies konnte aus Zeitmangel in der Evaluationsphase nicht ausprobiert und beurteilt werden.  
Im Grossen und Ganzen überzeugen die Möglichkeiten der Entwicklungsumgebung, jedoch müssen kleine Abstriche gemacht werden wegen des Verdeckens der Möglichkeiten, der begrenzten Auswahl von Alternativen sowie dem Hinweis auf eine Prototyping Sprache.

**Support:** Wie viele andere hat auch Appcelerator einen Slogan für seinen Support: „Get Access to mobile Experts“. Es werden einige Kontaktmöglichkeiten vorgestellt: Chat, E-Mail und Telefon. Der Service umfasst auch mehrere gut klingende Möglichkeiten wie: Hilfe um Titanium schnell zum Laufen zu bringen, Fixes für die Applikationen zu entwickeln oder beim Launch der App zu helfen. Ziemlich schnell wird jedoch klar, dass der Support mehr Business-Orientiert angelegt ist als bei anderen Anbietern.   
Es wird kein einziger Preis genannt bei den Supportangeboten, alles läuft über Offerten, somit kann man sich als Kunde nur ein Bild verschaffen, wenn man solche Offerten einholt, was ein klarer Minuspunkt ist.  
Eine positiv zu bewertende Funktion ist die FAQ-Sektion von Appcelerator, welche als offene Plattform angelegt ist und innert kurzer Zeit Antworten liefert. Der Rest des Supports kann nur schwer bewertet werden.

**Dokumentation:** Hier liegt die Stärke von Appcelerator Titanium, die Dokumentation ist sehr reichhaltig und äusserst Umfangreich. Auf der Homepage von Appcelerator finden sich sehr viele Entwickler-Dokumentationen, an gewissen Stellen sogar übertrieben viele, dies ist aber der Komplexität des Frameworks zuzuschreiben. Ebenfalls ist die Dokumentation sehr gut in die Entwicklungsumgebung eingebettet und man kann sich auch einige Bücher zum Framework bestellen. Ebenfalls ist ein Wiki verfügbar, welches sich aber noch im Aufbau befindet.  
Die Struktur der Dokumentation ist dabei sehr einfach, aber äusserst effizient gehalten. Man findet schnell, was man benötigt, hat aber an einigen Stellen einen gewissen Overhead, da zu viele Informationen in einen Menüpunkt integriert sind.  
Abschliessend lässt sich sagen, dass die Dokumentation sehr gut gehalten ist, für Einsteiger jedoch zu massig und überladen, daher wird es hier leichte Abstriche in der Bewertung geben.

**Tutorials:** Die Google Suche fördert 78‘400 Ergebnisse zu Tage:  
Eine erschreckend niedrige Zahl für ein so weit verbreitetes Framework. Auch die Qualität der gefundenen Tutorials ist eher mager, die meisten beschränken sich darauf, Features von Appcelerator Titanium zu zeigen, indem man vorgefertigte Code-Stücke herunterlädt und ausprobiert.   
Unter den ersten 5 Links, welche aus der Suche resultiert sind, findet sich als erstes das Appcelerator-Wiki, welches in etwa dasselbe abdeckt wie die Appcelerator Homepage selbst, das Wiki scheint sich noch im Aufbau zu befinden. Die restlichen Links sind eher Feature-Präsentationen von Appcelerator als Tutorials. Es werden oft Codestücke zum Download angeboten, welche man ausprobieren kann. Der Lerneffekt ist eher gering, wenn man sich jedoch schon ein wenig auskennt kann das durchaus nützlich sein.  
Die Absenz von guten Tutorials und deren geringe Zahl schlagen sich entsprechend in der Bewertung nieder.

**Building:** Building gibt es bei Appcelerator eigentlich nicht. Die Codestücke, welche mit Titanium Studio programmiert wurden, können direkt auf das Gerät geladen werden und werden dort interpretiert. Somit entfällt das Build / Compile des Codes, der in Titanium Studio eingebaute Live-Emulator gibt schon im Vorfeld einen guten Einblick, wie das Programm aussehen wird.  
Dieses Vorgehen erzeugt aber auch relativ starke Probleme. Diverse Berichte zeigen, dass auf dem eingebetteten Emulator alles Problemlos läuft, sich aber auf dem Zielgerät Fehler zeigen, welche bei einem vorrangigen Build ausgeräumt würden.  
Eine Bewertung ist hier schwierig, da es ja eigentlich kein Building gibt. Jedoch wird die Möglichkeit, die Scripts direkt auf das Mobile Gerät zu spielen, als positiv bewertet. Jedoch sind die Probleme, die dabei entstehen wiederum als negativ zu bewerten. Ein guter Mittelweg ist in der Bewertung das fairste.

**Testing:** Hier liegt wieder eine der Stärken von Appcelerator Titanium. Da eine Registrierung als Technology Partner möglich ist, haben diverse Firmen diese Möglichkeit wahrgenommen. Technology Partners können über den Appcelerator Marketplace Erweiterungen für Appcelerator Titanium programmieren. Dies schlägt sich in diversen Erweiterungen nieder, welche explizit für das Testing geschrieben wurden, es werden diverse Möglichkeiten abgedeckt: Debugging-Tools, Trace-Tools, Beta Testing Apps und vieles mehr.   
Wie in diversen Marktplätzen tummeln sich hier gute wie weniger gute Tools, nach ein wenig Suchen hat man jedoch gute Tools zur Hand, welche Methodisch gut aufgebaut sind und Testing nach Schulbuch oder Industriestandard ermöglichen. Die Preise sind teilweise jedoch eher hoch.  
Alles in allem zeichnet sich jedoch für das Testing ein sehr gutes Bild.

**Verbreitung:** Die Google Suche nach „Appcelerator Titanium“ zeigt eine Anzahl von 621‘000 Ergebnissen an, was doch ziemlich erstaunt. Denn die Hersteller-Homepage gibt eine Anzahl von 35‘000 geschriebenen Apps an. Diese Zahl lässt sich jedoch schwer überprüfen. Wenn man jedoch einige Apps anschaut wird schnell klar, dass es einige Global Players hat, welche ihre Apps mit Appcelerator Titanium geschrieben haben, so zum Beispiel: Mitsubishi, ebay, Merck oder PayPal. Daher kann man von einer grossen Verbreitung ausgehen.  
Auf Nachfrage hat der eine oder andere Kollege schon vom Framework gehört, jedoch nach kurzer Zeit wieder die Finger davon gelassen. Dies ist wohl hauptsächlich auf die unter „Einfachheit“ erwähnten Punkte sowie durch das unglückliche Design der Homepage zu erklären.

**Spezielles:** Einige Spezialfunktionen bzw. Services sind löblich zu erwähnen. So bietet Appcelerator zum Beispiel eigene Clouds in verschiedenen Ausbaustufen an. Diese werden z.B. benötigt für Push-Nachrichten oder Mobile Backend as a Service.  
Der eigene Marktplatz für Entwickler ist ebenfalls löblich, vor allem, weil dort viele Tools gefunden werden können, welche das Entwickeln vereinfachen, z.B. Testing Tools, Analyse Tools (Verbreitung, Benutzung) und vieles mehr.

# Finale Bewertung

|  |  |
| --- | --- |
| Kriterium | Punkte |
| Einfachheit | 4 |
| Entwicklungsumgebung | 6 |
| Support | 4 |
| Dokumentation | 7 |
| Tutorials | ?? |
| Building | 5 |
| Testing | 9 |
| Verbreitung | ?? |
| Spezielles | 2 |



# Sproutcore Touch

# F:\Schule\HSZ-T\3. Jahr\Handheld\Semesterarbeit\Handheld_Repo\Media\Sproutcore Touch\light2.pngEinleitung

Sproutcore Touch ist ein Open-Source Java-Script Framework dessen Ziel es ist, Web-Applikationen zu entwickeln mit erweiterten Möglichkeiten der Hardwarebeschleunigung von Touch Devices. Wenn mit Sproutcore entwickelt wird, ist fast jeglicher Code in JavaScript gehalten. Die Unterstützung von Mobile Devices beschränkt sich hierbei grösstenteils auf Apple-Produkte. Dies auch darum, weil der Gründer von Sproutcore noch bis 2010 bei Apple gearbeitet hat (Gründung von Sproutcore im Jahr 2007) und diverse Apple-Entwickler im Rahmen der Web 2.0 Initiative stark bei Sproutcore mitgearbeitet haben. Apples Service MobileMe wurde sogar komplett mit Sproutcore entwickelt.  
So erklärt sich auch das Look-and-Feel des Frameworks, welches sowohl auf Desktop-Browsern wie auch auf Mobile-Browsern stark an Apple-Produkte erinnert.

# Wie funktioniert das Framework

F:\Schule\HSZ-T\3. Jahr\Handheld\Semesterarbeit\Handheld_Repo\Media\Sproutcore Touch\MVC.JPGDie Entwicklung in Sproutcore ist primär in JavaScript gehalten, es lassen sich aber prinzipiell sehr viele verschiedene Technologien mit einbinden. Die Entwicklung von Sproutcore kommt historisch von Ruby, dies merkt man auch beim Entwickeln mit Sproutcore stark.   
Zu Beginn wird mittels wenigen Befehlen die Struktur und erste Dateien erzeugt, auf welchen man aufbaut. Diese werden dann immer stärker verknüpft und erweitert. Die Verknüpfung basiert dabei auf dem Model-View-Controller (MVC) Schema, was für ein Touch-Framework eher erstaunlich ist. Zum guten Schluss kann mittels einem lokalen Build eine schlanke, schnelle Applikation erstellt werden.



# Evaluation

**Einfachheit:** Das Framework ist im Aufbau ziemlich simpel, der Schwierigkeitsgrad steigt jedoch ziemlich schnell. Es lässt sich aber, wenn man sich ans MVC-Paradigma gewohnt ist, relativ schnell erlernen. Die ersten einfachen Programme lassen sich einfach verstehen (z.B. mit den Beispielen auf der SproutCore Homepage) und kommen auch schnell zum Laufen. Dies auch dank der einfachen Schritt-für-Schritt Anweisungen auf der Sproutcore Homepage.  
Die eingesetzten Technologien beschränken sich, zumindest in den ersten Beispielen, auf HTML, CSS und Javascript. Damit lassen sich schon ziemlich gute Resultate erzielen. Will man jedoch für andere Produkte entwickeln als iPad und iPhone im Mobilbereich, dann muss man auch andere Technologien einsetzen (z.B. Objective-J, JSS), es bestehen ebenfalls Schnittstellen zu den bedeutendsten anderen Frameworks (Sencha Touch, Phonegap, etc.).  
Da die Entwicklung mittels MVC-Paradigma passiert, könnte man sagen, die Bedienung ist nicht intuitiv. Da sich aber dieses Paradigma immer grösserer Beliebtheit erfreut, wird es viele Leute geben, welche es durchaus als intuitiv bezeichnen würden. Jedoch ist die Entwicklung von HTML auf Basis von JavaScript doch schon etwas speziell, daher bezeichnen wir es als leicht unintuitiv.

**Entwicklungsumgebung:** Ein fest vorgegebenes IDE gibt es in diesem Sinne nicht bei Sproutcore, auch keine offizielle Empfehlung. Es gibt mittlerweile Greenhouse, ein IDE nur für Sproutcore, jedoch befindet sich dies noch im Alpha-Stadium, weshalb noch nicht sehr viele Entwickler damit arbeiten, obwohl es öffentlich ist.  
Prinzipiell lassen sich alle Texteditoren verwenden, idealerweise natürlich ein Texteditor, welcher JavaScript unterstützt. Da es auch einige Rails-Komponenten in Sproutcore hat, kann es auch von Vorteil sein, ein IDE zu wählen, welches Rails unterstützt.  
Populär sind NetBeans (es kann hier praktisch alles eingebunden werden), Rubymine (Eigentlich eher Rails/Ruby zentriert, versteht mittlerweile jedoch auch JavaScript) sowie WebStorm (JavaScript-Zentriert).  
Da nichts vorgegeben ist, hat man hier ein freies Gefäss und kann sich sein IDE so zusammenstellen, wie man es möchte. Bei der Bewertung wird hier daher ein Mittelweg gewählt, jedoch ein wenig höher gewichtet wegen der guten Möglichkeiten der Auswahl.

**Support:** Bei Sproutcore findet man keinen klassischen Support. Weder per Telefon, noch per Mail. Es gibt jedoch Möglichkeiten, mit dem Sproutcore Team in Verbindung zu treten, dies sind: Eine Mailing List, ein Blog (mit Möglichkeit RSS), Twitter und Facebook.  
Sproutcore betreibt auch User-Groups. Dies sind Gruppierungen an verschiedenen Orten, welche sich mehr oder weniger regelmässig treffen, um sich auszutauschen und ihre Erfahrungen abzugleichen.  
Sproutcore veranstaltet auch in unregelmässigen Abständen Events, bei welchen sie informieren, aber auch meist für Diskussionen bereit stehen. Speziell erwähnt sei das zweijährlich stattfindende „SproutCore WWDC Bash“, eine Veranstaltung, die einen ernsten Kern aber auch viele Spass-Events hat.  
Alles in allem ist der Support wohl einer der Schwachpunkte von Sproutcore, was durch seine breite Userbasis mit all ihren Homepages ein wenig ausgeglichen wird. Dennoch können hier nur wenige Punkte vergeben werden.

**Dokumentation:** Die Dokumentation gehört zu den Stärken von Sproutcore. An mehreren Stellen online verfügbar bietet diese einen sehr sauber ausgeführten Service, um schnell an Informationen zu kommen. Dies zeigt sich in den Guides (Step-By-Step für Anfänger, weiterführende Tutorials für Leute die schon weiter sind und grob verschachtelte Beispiele für Profis), bei der eigentlichen Funktionsdokumentation sowie bei der Hilfe, die beim Erzeugen des Framework-Stammes verfügbar ist. Der Aufbau ist immer sehr fein gegliedert und bietet für jedermann das richtige Niveau an, ohne einem jemals zu viel zuzumuten.  
Durch die Summe der Möglichkeiten erreicht hier Sproutcore eine sehr gute Bewertung, ein wenig Abzug muss gemacht werden, weil man den Desktopteil von Sproutcore immer mitschleppt. Eine separate Dokumentation für die Touch-Funktionen wäre wünschenswert.

**Tutorials:** Die Google Suche fördert 257‘000 Ergebnisse zu Tage:  
Kein schlechtes Ergebnis, denn wenn man bedenkt, was Sproutcore selbst schon auf der Homepage anbietet. Somit ist es realistisch, nicht mehr allzu viele Tutorials ausserhalb zu platzieren, denn man kann seine Tutorials auch direkt an Sproutcore schicken, sie begrüssen dies sogar explizit und rufen dazu auf. Diese Tutorials sind auf SproutCore selbst mit einer WordCloud sehr schnell auffindbar, auch ohne Suche (welche es selbstredend auch gibt), daher ist es nur logisch, dass Sproutcore selbst in den Suchergebnissen als erstes auftaucht.  
Der zweite Link verweist auf das Wiki von Sproutcore, welches den gewohnten Standard an Dokumentation und Tutorials bietet, jedoch in einer Wiki-Struktur eingebettet. Somit kann entschieden werden, welches einem besser liegt: Wiki oder normale Web-Struktur.  
Der dritte Link behandelt ein vieldiskutiertes Thema in einem Tutorial, den Umgang mit SC.TableView. Dieses Tutorial ist äusserst ausführlich gehalten, was ein grosses Plus ist. Ebenfalls sind alle Code-Snippets einzeln herunterladbar. Äusserst gelungen.  
Der vierte Link gibt in mehreren Video-Tutorials verschiedene Funktionalitäten zu sehen, dies ebenfalls im gewohnt aufgeräumten Look und Code.  
Der fünfte Link stammt von BroadcastingAdam, wer sich ein wenig in der Web-Entwickler-Szene auskennt, kennt auch diese Page und weiss um ihre grossen Bemühungen, gute Tutorials zu bieten.  
Alles in allem sind alle betrachteten Tutorials-Pages von exzellenter Qualität, nicht zu viel, nicht zu wenig, (meist) mit Abstufungen des Schwierigkeitsgrades und guten Beispielen.

**Building:** Da es sich bei Sproutcore nicht um ein Building für eine App handelt, sondern ein Building für eine Web-Page, muss hier ein Abzug in der Bewertung gemacht werden. Nichtsdestotrotz ist der Building-Prozess sehr eindrücklich: Es wird vom vorgefertigten Code ein Build erzeugt mit einem einfachen Befehl. Dieser löst eine ganze Reihe von Aktionen aus: Der gesamte JavaScript-Code wird in ein einziges File geschrieben (dies erhöht die Geschwindigkeit der Page enorm), ebenso werden andere Sources sauber zusammengefasst. Der Clou ist jedoch, dass Bilder automatisch auf die richtige Grösse zugeschnitten werden und auch für die Verwendung im Web optimiert werden (Auflösung, etc.). Der Build, der dabei erzeugt wird, unterstützt praktisch alle nativen Funktionen eines iPads oder iPhones.  
Im Grossen eine tolle Methode, einen Build zu erzeugen, jedoch schade, dass dies nicht in eine fertige App passiert, welche auf verschiedenen Systemen läuft und über einen Store bezogen werden kann. Dies schlägt sich entsprechend in der Bewertung nieder.

**Testing:** Sproutcore ist eines der wenigen Frameworks, welches starken Wert auf Testing legt. Es steht ein komplettes Unit-Testing Framework zur Verfügung, um den Code auf Herz und Nieren zu überprüfen. Die gelungenen Anleitungen dazu sind auch für Anfänger sehr gut zu verstehen und könnten, ohne Veränderungen, im Schulunterricht eingesetzt werden. Dies liegt einerseits an ihrer Verständlichkeit und auch an deren Praxisorientierung, jeder kann sofort umsetzen, was in den Anleitungen steht. Ebenso legen sie starken Wert auf Test-Driven-Development, dieser Methode wird ebenfalls eine komplette Anleitung gewidmet.  
Allgemein ist das Testframework sehr aufgeräumt und stellt die Ergebnisse der Tests sauber dar in verschiedenen Tiefen und Ausprägungen. Die Methodik verläuft dabei fast schulbuchmässig und die Struktur ist jederzeit gut eingehalten und gut verständlich.  
Dieser Punkt wird sehr gut bewertet, da die Ausführlichkeit, Handhabung und Dokumentation exzellent gehalten sind.

**Verbreitung:** Die Google Suche nach „Sproutcore“ zeigt eine Anzahl von 481‘000 Ergebnissen an, dies wohl ebenfalls, da Sproutcore schon sehr vieles unter ihrem eigenen Dach vereint.   
Eine Anzahl der Apps, die damit geschrieben wurden liess sich leider nicht in Erfahrung bringen, denn diese sind im Web nicht unbedingt als solche zu erkennen und auch Sproutcore selbst stellt hier keine Zahlen zur Verfügung  
Jedoch stolpert man immer wieder über dieses Framework, sei es, wenn man im Internet zu Web-Entwicklung googelt, sich mit Freunden oder Arbeitskollegen unterhält oder in einschlägigen Publikationen. Dies kann auch damit zu tun haben, dass dieses Framework auch schon eher zu den älteren Hasen in diesem Business zählen.

**Spezielles:** Ein spezieller Build für schlanke Page-Strukturen (näher beschrieben unter „Building“) wird angeboten. Da dies jedoch nicht in ein App gewandelt werden kann, werden hier keine Punkte vergeben.

# Finale Bewertung

|  |  |
| --- | --- |
| Kriterium | Punkte |
| Einfachheit | 7 |
| Entwicklungsumgebung | 6 |
| Support | 4 |
| Dokumentation | 9 |
| Tutorials | ?? |
| Building | 5 |
| Testing | 9 |
| Verbreitung | ?? |
| Spezielles | 0 |

# PhoneGap

# Einleitung

****Phonegap ist ein Open Source Framework, um in kurzer Zeit Cross-Plattform Mobile Apps zu erstellen mit HTML5, Javascript und CSS. Es können damit Apps für iPhone, Android, Windows Mobile, Blackberry, WebOS, Symbian, Tizen und Bada erstellt werden.   
Es werden allgemein Standard-Web-Technologien eingesetzt um die Grenze zwischen Web-Basierter Technologie und Mobile Devices immer stärker zu verwischen.   
Das Framework wurde schon über eine Million Mal heruntergeladen und zählt mittlerweile eine Community von 400‘000 Entwicklern. Das Framework benutzt die Technologie Apache Cordova, welche unter der Ägide der Apache Software Foundation mitentwickelt wurde. Dadurch kann gesagt werden, dass PhoneGap auch in Zukunft Open Source bleiben wird. Mittlerweile wurde PhoneGap von Adobe aufgekauft und weiterentwickelt

# Wie funktioniert das Framework

Die Entwicklung von PhoneGap Apps folgt dem klassischen Web-Entwicklungs-Ansatz. Es wird ein HTML erstellt, welches mittels CSS formatiert wird, danach werden spezifische JavaScript Tags benutzt um die Gerätefunktionen anzusteuern. Es wird NICHT nativ entwickelt mit z.B. Objective C für iOS, es können auch keine Zusatzfunktionen damit erreicht werden.  
Diese Skripts werden danach kompiliert, hier gibt es zwei Möglichkeiten: Ein lokales kompilieren auf dem eigenen Rechner (jeweils für 1 Framework) oder das Kompilieren auf build.phonegap.com (Kompilieren auf mehrere Frameworks gleichzeitig). Was herauskommt sind binary Files, welche mit wenigen Abwandlungen direkt in den jeweiligen Store (iTunes Store, etc.) geladen werden können.



# Evaluation

**Einfachheit:** Die sehr guten und ausführlichen Beispiele geben schnell einen Eindruck, wie das Framework funktioniert, sehen einfach aus in der Anwendung und es ist schnell ersichtlich, dass mit wenigen Zeilen JavaScript eine hohe Funktionsdichte geboten wird.  
Die Technologie wird eingesetzt wie man es aus HTML, CSS und Javascript kennt, jeder Programmierer, welcher jemals eine Homepage selbst geschrieben hat, wird sich innert kürzester Zeit zurechtfinden. Die Schichtung der drei Haupttechnologien ist genauso, wie man sie tagtäglich einsetzt beim Programmieren einer Homepage, jedoch ist noch ein zusätzliches JavaScript einzubinden, um die Funktionalität der Cordova Schnittstelle vollständig ausschöpfen zu können.  
Die Basis der Entwicklung sind HTML, CSS und JavaScript. Mehr braucht es nicht, um eine App erstellen zu können. Es bieten sich für Fortgeschrittene jedoch noch mehr Möglichkeiten wie z.B. das Anlegen eines config.xml zur Angabe von Meta-Daten, einbinden eigener JavaScripts sowie WebView Embedding für ausgewählte Touch-Devices. In der Summe ist hier von einem einfachen und gut verständlichen Framework zu sprechen.

**Entwicklungsumgebung:** Als Entwicklungsumgebung wird von PhoneGap Eclipse angegeben, welches mit dem jeweiligen SDK (für die jeweiligen Betriebssystemumgebungen wie Android oder iOS) und Cordova erweitert wird. Es sind aber auch schon Plug-Ins für Dreamweaver entwickelt worden. Prinzipiell reicht ein Text-Editor, um die Anwendungen zu entwickeln, da ein Online-Build-Tool verwendet werden kann für die Kompilierung. Es wird den Entwicklern also viel Freiheit gelassen, wie eine App entwickelt werden kann.  
Das empfohlene IDE, Eclipse, kennt fast jeder Entwickler und kennt seine Stärken wie: Text-Completion, Korrekturvorschläge, Fertigbausteine, Code-Highlighting, Containering, Variable-Lookup, Variablen-Unterstützung und vieles mehr. Die extrem einfache Erweiterbarkeit tut ihren Rest und daher eignet sich dieses Tool hervorragend, um zu entwickeln.  
Erweiterungen sind, wie oben Erwähnt, für Dreamweaver vorhanden. Mit dieser Erweiterung lassen sich Cordova-Funktionen auch in Dreamweaver ansteuern und verwenden. Eine weitere Erweiterung bzw. abgeleitetes Framework stellt appMobi PhoneGap XDK dar, welches auf PhoneGap aufbaut, aber ein noch weiter gefasstes Framework mit diversen Erweiterungen und eigenem IDE darstellt.  
Die verschiedenen Möglichkeiten der Entwicklungsumgebung tragen zu einer guten Bewertung bei, einziger Wehrmutstropfen ist, dass es etwas kompliziert ist, bis man alles bereit hat bei der Installation der Umgebung.

**Support:** Unter dem Slogan „Get all the tools, help and training you need to build great PhoneGap apps” stehen diverse Supportmöglichkeiten zur Verfügung, wie es unter dem Dach von Adobe zu vermuten war. Es stehen diverse Support-Packets zur Verfügung, vom einfachen Ein-Personen Entwickler (1 Person, Best Effort, 24.95$) bis zum Enterprise-Kunden (X Personen, 24x7, 4h Reaktionszeit, ab 2000$) können diese Support-Packets diverse Bedürfnisse abdecken.  
Es stehen dabei diverse Möglichkeiten wie Bug-Fix-Patching, Knowledge-Base, Chats, privates Forum und noch vieles mehr bereit. Dies zeigt, dass für wenige Dollars ein immenses Wissen zur Verfügung gestellt wird, daher wird das Preis Leistungsverhältnis mit sehr gut bewertet.  
Es gibt aber auch genügend Möglichkeiten, kostenlos mit den Entwicklern in Verbindung zu treten, so z.B. Google Groups, ein PhoneGap Forum und diverse andere Foren, worin sich die Entwickler von PhoneGap bewegen. Ein kurzer Test zeigt, dass man z.B. in den Google Groups meist innert 2 Stunden eine Antwort erhält. Auch dies zeigt eine vorbildliche Haltung gegenüber den Usern.  
Es werden eine Vielzahl von sauber ausgearbeiteten Anleitungen angeboten und auch gepflegt. Die Übersicht ist dabei stets gewährleistet und bietet grossen Komfort.

**Dokumentation:** Wie bereits unter dem Punkt „Support“ erwähnt steht eine perfekt gepflegte Dokumentation im Netz bereit, welche einfach navigierbar und gut durchdacht ist. Die Struktur ist nicht eindimensional gehalten, über viele Stellen gelangt man an denselben Ort. Die Dokumentationen sind allesamt Online oder Offline verfügbar und kosten nichts, auch hier wird der Open-Source Charakter des gesamten Frameworks hervorgehoben.  
Die Dokumentationen enthalten viele Informationen, doch allesamt erscheinen nützlich und helfen weiter. Falls man in die Tiefe gehen will, ist mit wenig ausprobieren und ein wenig Suche im Netz schnell das richtige gefunden.

**Tutorials:** Die Google Suche fördert 1.09 Mio. Ergebnisse zu Tage:  
Dies spricht für eine grosse Unterstützung durch Tutorials. Als erster Link taucht gleich das Developer-Portal von PhoneGap auf, welches wie oben erwähnt einige gute Tutorials bietet.   
Auch die restlichen vier Links, welche ausprobiert wurden, bieten sehr gut ausgearbeitete Tutorials, der 5. Link ist sogar ein 10 Minütiges Video-Tutorial für Android-Entwicklung und wie man die Entwicklungsumgebung aufsetzt. Alles in allem ist die Qualität absolut erstaunlich. Die meisten führen von einfachen Aktionen auf kompliziertere Zusammenhänge ohne jemals den Schwierigkeitsgrad zu stark zu erhöhen.

**Building:** Die Builds sind mehr oder weniger einfach zu erstellen. Wählt man den lokalen Weg, kommt man nicht umhin, für jedes unterstützte Touch-Betriebssystem den Build im entsprechenden IDE zu machen. Wählt man jedoch den Weg über das Building Tool build.phonegap.com, ist dies sehr einfach und die Build-Engine erstellt für jedes Touch-Betriebssystem in einer guten Geschwindigkeit die entsprechenden Binary Files.   
Vor allem die Möglichkeit, über das Online-Build Tool zu arbeiten, bietet extrem komfortable Möglichkeiten, es können nicht nur ZIP Files mit den entsprechenden HTML-Dateien hochgeladen werden, es bietet auch die Möglichkeit, ein Git-Repo direkt anzugeben. Das direkte Ausbringen der entsprechenden Files auf ein Testgerät rundet das üppige Angebot ab.

**Testing:** Auf den ersten Blick ist Testing eher schwierig. Auf den zweiten Blick jedoch sieht die Situation anders aus. PhoneGap stellt zwar von Hause aus keine Testing-Tools zur Verfügung, jedoch ist die Community hier schon wesentlich weiter. Diverse Entwickler haben Test-Tools entwickelt, welche sich einerseits direkt während der Entwicklung benutzen lassen (im IDE) oder auf dem Touch-Device selbst.   
Das beste gefundene Tool, welches ein aussenstehender Entwickler erstellt hat, ist nun sogar von PhoneGap in ihr Portfolio übernommen worden. Erreichbar über <http://debug.phonegap.com/> bietet dieses eine gute Möglichkeit, über eingebettete Skriptteile die komplette App zu debuggen.

**Verbreitung:** Die Google Suche nach „PhoneGap“ zeigt eine Anzahl von 4.4 Mio Ergebnissen an. Die Anzahl der damit geschriebenen Apps vergrössert sich natürlich täglich, der aktuelle Stand weist eine Zahl von 1181 Apps auf der Feature-List von PhoneGap an, inoffiziell ist von über 30‘000 Apps die Rede. Dabei sind auch sehr respektable Apps zu finden, wie z.B. der Logitech Squeezebox™ Controller, welcher eine komplette Steuerung einer Multi-Room-Media Lösung ermöglicht.  
Auch bei Nachfragen im Kollegenkreis kennt man PhoneGap, es wurde auch schon öfters bei Mobile-Konferenzen erwähnt, und dies nur lobend.

**Spezielles:** In diesem Framework ist so vieles gut gemacht, man sehe sich nur mal die Building-Homepage an, die Unterstützung durch die Entwicklercommunity (welche selbst noch Tools beisteuert), die Umsetzung des Konzepts, usw.

# Finale Bewertung

|  |  |
| --- | --- |
| Kriterium | Punkte |
| Einfachheit | 8 |
| Entwicklungsumgebung | 7 |
| Support | 9 |
| Dokumentation | 8 |
| Tutorials | ?? |
| Building | 10 |
| Testing | 6 |
| Verbreitung | ?? |
| Spezielles | 2 |

# Sencha Touch

# Einleitung

E:\Media\Sencha Touch\sencha_touch_logo.JPGSencha Touch ist ein HTML5-Framework, welches unter verschiedenen Lizenzen erworben werden kann. Zum Einen stehen Gratisversionen unter einer kommerziellen und einer OpenSource-Lizenz zur Verfügung und zum Anderen eine kostenpflichtige OEM-Version. Unterstützt werden die Systeme iOS, Android, BlackBerry, Kindle Fire und weitere.

# Wie funktioniert das Framework

Das Framework basiert auf HTML5 und bietet mittels JavaScript ein Menge weiterer Funktionen, unter anderem verschiedene Komponenten, Layouts, Zeichnungs- und Animations-Tools, Diagramme sowie Ansteuerung von Geräteschnittstellen. Speziell ist, dass das gesamte UI in JavaScript geschrieben wird. Dies wird mit einer JSON-ähnlichen Notation erreicht.

# Evaluation

**Einfachheit:** Die Dokumentation und die How-Tos auf der Sencha-Homepage geben einen guten Einblick in die Funktionsweise des Frameworks. Auffällig ist, dass nur JavaScript-Code für die Programmierung verwendet wird. Das HTML-Layout wird dann mittels HTML-Attributen in JSON angegeben. Für Kenner von anderen Sencha-Frameworks wie beispielsweise Ext.Js wird dies kein Problem sein. Wenn man sich jedoch gewohnt ist, den JavaScript-Code in HTML einzubetten, erscheint die Sencha-Art eher fremd. Dies gilt allerdings nur für die Verwendung der von Sencha Touch bereitgestellten Komponenten. Wenn die Komponenten selbst entwickelt werden kann dies auch gewohnt mit einem HTML-Layout und eingebettetem JavaScript entwickelt werden. Deshalb werden wir einen kleinen Abzug notieren müssen.

**Entwicklungsumgebung:** Die Dokumentation lässt keine Schlüsse offen für eine spezifische IDE. Im Prinzip kann ein normaler Texteditor verwendet werden. Die Internet-Recherche ergab, dass Entwicklungsumgebungen wie Netbeans, Eclipse / Aptana, WebStorm, ... verwendet werden.

Nach der Installation des Sencha SDKs und den dazugehörigen SDK Tools kann es los gehen. Mit dem durch die SDK Tools installierten „Sencha“-Tools kann eine komplette Applikationsstruktur generiert werden. Dieser Ansatz ist bereits durch andere Frameworks (wie beispielsweise Rails) bekannt. Die Applikation kann anschliessend bereits über einen Browser aufgerufen werden und eine schöne Begrüssung wird angezeigt. Darauf ist auch auf das Main.js verwiesen, welches für das Rendering der Startseite verantwortlich ist. Dies bietet einen sehr komfortablen und schnellen Einstieg und wird entsprechend gut bewertet.

**Support:**

**Dokumentation:** Sencha bietet auf ihrer Homepage eine sehr ausführliche, teilweise extrem technische Dokumentation. Trotzdem ist diese immer zweckgemäss gehalten und bietet viele In-Browser-Beispiele, welche die fertige App demonstrieren. Die Gliederung ähnelt einer API, bei welcher die JavaScript-Prototypen und –Methoden aufgelistet werden. Dies macht den Einstieg für einen Anfänger eher schwierig, allerdings ist dies bei allen Sencha-Frameworks so gehalten, sodass sich Sencha-Fans sicherlich darin wohlfühlen.

**Tutorials:** Diese sind in den meisten Fällen direkt in die Dokumentation eingebettet. So wird die technische Dokumentation gelungen abgerundet. Besonders die In-Browser-Beispiele geben eine sehr gute Vorstellung, wie sich das App-Beispiel präsentiert und auch anfühlt.

Direkt auf der Website ist auch ein Forum verfügbar, welches eine gute Gliederung nach beispielsweise nach Framework, Version, Kundenmodell, etc. aufgeteilt ist. Unter anderem bietet das Forum sogenannte Showcases, welche Beispielimplementierungen, Tutorials, Repositories zu eigens geschriebenen Komponenten, etc. beinhalten. Diese Einträge machen einen sehr guten Eindruck, da bei den meisten Posts ebenfalls Feedback und Anregungen vorhanden sind.

Die Suche nach “Sencha Touch Tutorial” ergab 387’000 Treffer. Der erste Treffer verweist auf die Sencha-Homepage. Dort befindet sich eine nette Sammlung an Tutorials verschiedener Themenbereiche. Sehr angenehm wirkt auf den ersten Blick auch die Angabe des Schwierigkeitsgrads, welcher über „Easy“, „Medium“ und „Hard“ angegeben wird. Die Anleitungen sind unterschiedlich aufgebaut: Gewisse werden über ein Video abgehandelt und andere werden herkömmlich über Text und Quellcode dargestellt. Positiv bei den Video-Tutorials ist, dass bei den meisten ein Source-Repository angegeben ist, in welchem sich dann der verwendete Code befindet.

Der zweite Link verweist ebenfalls auf die Sencha-Homepage und beschreibt ausführlich ein „Hello World“-Beispiel. Dies ist sehr einfach gehalten und abgesehen von den HTML-Tags wird jede Zeile einzeln erklärt.

Der dritte Link verweist auf den Sencha-Blog, welcher ebenfalls wertvolle Beiträge beinhaltet wie zum Beispiel „Wie schreibe ich eigene Komponenten“. Hier fällt allerdings auf, dass nicht nur Sencha Touch sondern ebenfalls andere Sencha-Frameworks beschrieben sind. Also ist bei den Artikeln darauf zu achten, dass über das richtige Framework gesprochen wird.

Der dritte und vierte Link verweisen auf die Seite miamicoder.com von Jorge Ramon. Dort werden Schritt-Für-Schritt-Anleitungen zur Entwicklung von eigenen Sencha-Touch-Apps angeboten. Diese basieren auf dem vom Website-Inhaber geschriebenen Buch „Building a Sencha Touch Application“. Die Tutorials sind sehr einfach aufgebaut und werden detailliert Step-by-Step beschrieben.

Gesamthaft bewerten wir die Tutorials sehr gut, allerdings werden wir auf Grund des Blogs – welcher verschiedene Frameworks beinhaltet – einen Abzug machen.

**Building:**

**Testing:**

**Verbreitung:**

**Spezielles:**

# Gegenüberstellung der Frameworks

# Projekt Lupen-App

# Umgebung des Projekts

# Umgebung der Evaluation

# Umgebung der Programmierung / Framework

# Wahl des Frameworks

# Tools

# Quellen

<http://www.appcelerator.com/>

<https://github.com/appcelerator/KitchenSink/tree/master/Resources>

<http://www.d-mueller.de/blog/appentwicklung-webtechniken-mit-appcelerator-titanium/>

<http://www.universalmind.com/mindshare/entry/mobile-html5-phonegap-vs-appcelerator-titanium>

<http://usingimho.wordpress.com/2011/06/14/why-you-should-stay-away-from-appcelerators-titanium/>

<https://github.com/appcelerator/KitchenSink/tree/master/Resources>

<http://www.sencha.com/products/touch/>

<http://www.sproutcore.com/>

<http://opensource.org/licenses/MIT>

<http://ajaxian.com/archives/sproutcore-2010>

<http://broadcastingadam.com/2011/04/sproutcore_login_tutorial/>

<http://www.veebsbraindump.com/2010/10/sproutcore-crud-tutorial-using-sc-tableview/>

[https://groups.google.com/forum/?fromgroups=#!topic/sproutcore/EFoUcd5fC2w](https://groups.google.com/forum/?fromgroups=%23!topic/sproutcore/EFoUcd5fC2w)

<http://www.phonegap.com/>

<http://dev.appmobi.com/?q=node/153>

<http://www.adobe.com>

<http://www.google.ch>

<http://build.phonegap.com/>

<http://debug.phonegap.com/>

<http://mobile.tutsplus.com/tutorials/phonegap/creating-an-android-hello-world-application-with-phonegap/>

[https://groups.google.com/forum/?fromgroups#!forum/phonegap](https://groups.google.com/forum/?fromgroups%23!forum/phonegap)

<http://rhomobile.com/products/rhodes/>

<http://code.google.com/p/iui/>

<http://www.iui-js.org/>

<http://iwebkit.net/>

<http://xuijs.com/>

<http://code.google.com/p/jqpad/>

<http://code.google.com/p/jqpad/>

# Abbildungsverzeichnis