**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 4](#_Toc338095972)

[1.1. Ausgangslage 4](#_Toc338095973)

[1.2. Ziele der Arbeit 4](#_Toc338095974)

[1.3. Erwartetes Resultat 4](#_Toc338095975)

[1.4. Motivation 5](#_Toc338095976)

[2. Planung und Termine 6](#_Toc338095977)

[2.1. Projektplan 6](#_Toc338095978)

[2.2. Termine 6](#_Toc338095979)

[2.3. Soll-/Ist-Vergleich 7](#_Toc338095980)

[3. Einleitung 7](#_Toc338095981)

[3.1. Vorgehen 8](#_Toc338095982)

[3.1.1. Vorgehen: Evaluation der Frameworks 8](#_Toc338095983)

[3.1.2. Vorgehen: Schreiben der Lupen-App 10](#_Toc338095984)

[3.2. Aufteilen der Aufgaben 10](#_Toc338095985)

[3.2.1. Aufteilung der Evaluation 10](#_Toc338095986)

[3.2.2. Aufteilung der Programmierung 11](#_Toc338095987)

[3.3. Umgebung des Projekts 11](#_Toc338095988)

[3.3.1. Umgebung der Evaluation 11](#_Toc338095989)

[3.3.2. Umgebung der Programmierung / Framework 11](#_Toc338095990)

[3.3.2.1. Wahl des Frameworks 11](#_Toc338095991)

[3.3.2.2. Tools 11](#_Toc338095992)

[4. Quellen 12](#_Toc338095993)

[5. Abbildungsverzeichnis 12](#_Toc338095994)

# 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

# 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. Daher kann man sogar noch von einer höheren Anzahl ausgehen. 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.

Aktuell liegt die Version 2 vor, allerdings wird die Version 1.x immernoch angeboten und supportet. Die Version 2 bietet allerdings zusätzliche Features an wie Native-APIs, mit welchem Hardware-Komponenten wie beispielsweise die Kamera angesprochen werden kann.

# 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. Der HTML-Inhalt 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, JavaScript ergänzend zu HTML zu schreiben, 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. Allerdings werden wir einen kleinen Abzug für die eher ungewohnte Notation 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, ... von der Community 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:** Es stehen grundsätzlich drei Support-Lizenzen zur Verfügung, welche in eine Standard und zwei Premium Lizenzen unterteilt werden. Die Lizenzen sind jeweils Entwicklerbasiert und der Preis bewegt sich zwischen 299$ und 4995$. Dafür erhält man Updates, Upgrades und Zugang zu Standard und Premium Foren. Bei einer Premium-Lizenz stehen zusätzlich Telefonsupport und Emergency-Bugfixes zur Verfügung.

Kostenlos kann natürlich über das Sencha-Forum der Community ebenfalls um Ratschlag gebeten werden.

**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 Framework, Version, Kundenmodell, etc. präsentiert. 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:** Das Building erfolgt lokal über die Sencha SDK Tools. Dabei wird die gebuildete Sencha-Library optimiert, d.h. es werden nur die benötigten Komponenten in den Build kopiert. Diese Einsparung macht sich dann beim Download und der Laufzeit bemerkbar.

Um ein natives Package zu erstellen können ebenfalls die Sencha SDK Tools eingesetzt werden, allerdings muss die App mit zusätzlichen Herstellerspezifischen Tools signiert werden. Wenn die App beispielsweise über den Android Market angeboten werden soll, muss die App vor dem Build über das Android SDK zertifiziert werden. Dies gilt natürlich auch für den Apple Store, hier ist der Aufwand jedoch noch etwas höher, da Apple ja bekanntermassen hohe Qualitätsanforderungen und –prüfungen durchführt.

Das bedeutet letztendlich, dass ein Native-Build geräteabhängig durchgeführt werden muss. Allerdings finden sich viele Verweise auf den PhoneGap-Online-Build. Dies gelingt allerdings nicht mit der reinen Sencha-App.

Positiv: Der Build lässt sich mit dem Tool Ant automatisieren. Dies dürfte vorallem Java-Entwicklern bestens bekannt sein und die Einarbeitung dürfte sich entsprechend mühelos gestalten.

Alles in allem etwas enttäuschend, dass die App für jede Art Device eigens gebuildet werden muss, allerdings erfreulich, dass ein externes Build-Tool eingesetzt werden kann.

**Testing:** Das erste Testwerkzeug ist laut diversen Foreneinträgen und Blogs der Simulator, also manuelles Testing. Dies ist im Grunde ein allgemeines Problem von RIAs (Rich Internet Applications), da sich die Struktur des Dokuments zur Laufzeit ändert.

Entsprechend findet sich leider (noch) kein Tutorial In der Sencha-Dokumentation in Punkto „automatisierte Tests“. Allerdings gibt es einen sehr schönen und ausführlichen Blog-Eintrag unter <http://www.sencha.com/blog/automating-unit-tests>. Dort wird auf Syntax Checks und Unit-Tests eingegangen. Für Web-Entwickler dürfte der Einstieg relativ einfach sein, da insbesondere für Unit-Tests das Jasmine-Framework eingesetzt wird, mit welchem JavaScript getestet werden kann.

**Verbreitung:** Die Suche mit „Sencha Touch“ ergibt ein Ergebnis von 1,49 Mio Einträgen. Lustigerweise ergibt die Suche mit „Sencha Touch 2“ ein Ergebnis von 4,26 Mio Einträgen. Dieses Ergebnis ist aus unserer Sicht relevanter, da die neue Version mehr Features bietet und auch in der Dokumentation mehr zu finden ist. Das Ergebnis ist schon recht beeindruckend und deutet auf eine grosse Verbreitung hin.

Auf der Sencha-Homepage finden sich unter der Rubrik „Who’s using it“ interessante Informationen. Seit 2010 haben mehr als 500‘000 Entwickler das Framework runtergeladen und damit Zehntausende von Apps geschrieben. Einige davon werden in der App Gallery vorgestellt, welche wöchentlich aktualisiert wird und über 200 Apps beinhaltet.

**Spezielles:** Speziell zu erwähnen sind die Live-Demos in den Dokumentationen. Mit einem WebKit-Fähigen Browser wie Chrome oder Safari können die Demos direkt im Browser ausprobiert werden.

# Finale Bewertung

|  |  |
| --- | --- |
| Kriterium | Punkte |
| Einfachheit | 6 |
| Entwicklungsumgebung | 8 |
| Support | 7 |
| Dokumentation | 8 |
| Tutorials | 8 |
| Building | 6 |
| Testing | 6 |
| Verbreitung | ?? |
| Spezielles | 1 |

# 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/>

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

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

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

<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/>

Sencha:

<http://www.sencha.com/blog/event-recorder-for-android-web-applications/>

<http://miamicoder.com/>

# Abbildungsverzeichnis