ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Cross-Plattform Development einer Mobile Business Application am Beispiel einer Tourismus Anwendung

Masterarbeit an der Universität Ulm

Vorgelegt von:
Daniel Glunz
Matrikelnummer: 702033
Wirtschaftswissenschaften
daniel.glunz@uni-ulm.de

Gutachter:
1. Prof. Dr. Manfred Reichert
2. Prof. Dr. Franz Schweiggert

Betreuer:
M. Sc. Johannes Schobel
Dipl.-Inf. Marc Schickler

2014

“Cross-Plattform Development einer Mobile Business Application am Beispiel einer Tourismus Anwendung”
Fassung vom 13.10.2014

Satz: PDF-LaTeX 2_ε

Zusammenfassung

Möchte man eine mobile Applikation für ein mobiles Betriebssystem entwickeln, so programmiert und kompiliert man diese mit dem jeweiligen SDK der Plattform und hinterlegt sie dann in dem jeweiligen App Store. Um sich den Aufwand der mehrfachen Entwicklung für die unterschiedlichen Plattformen zu sparen, entwickelt man eine Cross-Plattform-Anwendung, also eine Anwendung, die unabhängig vom Betriebssystem des mobilen Endgerätes ist. Dafür gibt es inzwischen verschiedene Frameworks, die diese Entwicklungsmöglichkeit unterstützen. Einige davon werden in dieser Arbeit vorgestellt.

Diese Arbeit behandelt die Entwicklung einer plattformunabhängigen Applikation für mobile Endgeräte anhand eines solchen Frameworks. Als Anwendungszenario wurde eine Tourismus-Applikation gewählt, welche als Client Informationen wie Nachrichten, Veranstaltungen, Bürgermeldungen und Sehenswürdigkeiten abruft und auf dem mobilen Endgerät des Nutzers darstellt. Serverseitig wird hierzu eine REST-API benötigt, die den Zugriff auf die bereits bestehende Datenbank regelt, in welcher die Daten für die einzelnen Kommunen vorliegen. Die Entwicklung der REST-Schnittstelle ist ebenfalls Gegenstand dieser Arbeit. Anschließend wird die Cross-Plattform-Entwicklung vorgestellt.

Im Fazit der Arbeit lässt sich feststellen, dass eine solche plattformunabhängige Entwicklung viele Vorteile mit sich bringt und diese auch in der nächsten Zeit noch zunehmen werden, andererseits wird jedoch vieles dennoch weiterhin mit nativen Applikationen programmiert werden, da solche Cross-Plattform-Entwicklungsmöglichkeiten noch lange nicht an die Performance und die Möglichkeiten der nativen Entwicklung herankommen werden.

Abstract

For the development of a mobile application for a specific device you have to program and compile the application with the SDK of the mobile OS and deploy it on the respective OS Store. This means a lot of programming effort if the mobile application is to be used on every OS. A more effective way is to write a source code once and deploy it on every OS by using a cross-platform framework. This allows you to create a mobile application platform-indepently. A lot of those cross-platform technologies have been developed by now. In the first part, this thesis gives a brief overview of the current frameworks used for cross-platform development.

This thesis describes the development of such a mobile cross-platform application. The mobile app that serves as example is a tourismus application which provides the user with data like news, events, live-reports and points of interest. On the server side a REST-API was programmed in association with the database which provides all the data of the communities for this mobile application. The thesis concludes with the presentation of the cross-platform development process.

It can be observed that using such a framework has many advantages and carries great future potential, but many mobile applications will be developed conventionally because these frameworks still lack some native performance and features.

Danksagung

An dieser Stelle möchte ich mich herzlich bei allen bedanken, die mich bei der Anfertigung dieser Arbeit unterstützt haben.

Zuerst gebührt mein Dank meinen beiden Betreuer, Johannes Schobel und Marc Schickler. Sie haben mir immer sehr weitergeholfen und mich nicht als „Fachfremden“ angesehen, sondern waren immer bemüht meine Anliegen zu klären. Auch wahren Sie jederzeit erreichbar und haben stets auf meine Fragen geantwortet. Insbesondere erwähnen möchte ich hier Johannes Schobel. Die Unterstützung, die ich von Johannes erhalten habe, reicht von konstruktiver Kritik und Unterstützung für die Implementierung bis hin zum Korrekturlesen der Arbeit. Er hat sich sehr viel Zeit für mich genommen.
Ich kann jedem, der sich mit mobilen Applikationen befasst und überlegt darin eine Abschlussarbeit zu schreiben, die beiden uneingeschränkt weiterempfehlen.

Des Weiteren möchte ich auch Herrn Prof. Dr. Reichert danken, der diese fachübergreifende Abschlussarbeit ermöglicht hat.

Nicht zuletzt gilt Herrn Prof. Dr. Schweiggert ein herzliches Dankeschön, der mein Interessengebiet in meinem Studium geweckt und gefördert hat.

Mein ganz besonderer Dank gilt abschließend meinen Eltern, die mich während des Studiums immer unterstützt haben.

Inhaltsverzeichnis

1

Einleitung

Möchte man für eine der bekannten Plattformen für mobile Endgeräte, wie Android, Windows Phone oder iOS eine mobile Anwendung entwickeln, so steht man vor dem Problem, dass man verschiedene Programmiersprachen beherrschen und die jeweils plattformabhängigen SDKs benutzen muss, um eine solche App zu entwickeln. Alternativ implementiert man eine Web-App. Eine solche Web-App ist eine mobile Webseite, die auf die Eigenschaften eines mobilen Endgeräts hin optimiert ist. Im Vergleich zu einer gewöhnlichen Webseite unterscheidet sie sich also in der Art der Bedienelemente und des Layouts. Eine klare Trennung zwischen Web-App und mobiler Webseite gibt es jedoch nicht (siehe auch Kapitel 2.4). Die Vorteile einer mobilen Web Applikation scheinen offensichtlich zu sein:

- Man kann damit webbasierte Applikationen auf allen Plattformen ausführen.
- Man muss sie nicht auf dem Endgerät installieren, sondern sie können über den Webbrowser des Geräts aufgerufen werden.
- Man muss sich nicht um Updates kümmern, da bei jedem Aufruf automatisch die aktuelle Version geladen wird.

Nachteilig an solchen mobilen Web-Applikationen ist aber, dass man nicht auf die Hardwarekomponenten und Schnittstellen des mobilen Endgeräts zugreifen kann. Auch ist eine Web-App nicht so performant wie eine native App, denn sie wird im Browser ausgeführt. [FI13]

Um diese Probleme zu beheben gibt es immer mehr Frameworks für die Entwicklung von plattformunabhängigen Applikationen für mobile Endgeräte. Man versucht, eine Art hybride Applikation zu entwickeln, die die Vorteile einer Web-App mit denen einer nativen App kombiniert. Dabei kann man eine sogenannte *Cross-Plattform*-Applikation auf verschiedene Weisen implementieren. Diese Varianten werden im weiteren Verlauf dieser Arbeit behandelt.

Darüber hinaus diskutiert diese Arbeit, ob eine solche Cross-Plattform die native Entwicklung vollständig ersetzen kann und welche Vor- und Nachteile die Entwicklung einer Cross-Plattform-Anwendung bietet. Auch wird der zukünftige Entwicklungstrend für mobile Applikationen erörtert.

1.1 Zielsetzung

Diese Arbeit evaluiert die Entwicklung einer plattformunabhängige Applikation mit Hilfe einer Cross-Plattform.

Die mobile Applikation ist dabei eine Endanwendung, die auf einer bereits bestehenden Idee gründet. Kommunen pflegen nämlich unterschiedliche Daten (z. B. Nachrichten, Veranstaltungen, Maps (POI), Ausschreibungen, Übernachtungsmöglichkeiten, Gastronomie, ÖPNV und Nahverkehr, Unternehmen und Politik) über eine gemeinsame Schnittstelle in eine große Datenbank ein. Diese Daten werden diesen Kommunen visualisiert und den Bürgern präsentiert. Dieses Konzept möchte man nun auch für mobile Applikationen umsetzen.

Die App soll nun, je nach Gemeinde, die jeweiligen dazugehörigen Informationen anzeigen. Sie soll also erst einmal eine Schnittstelle bzw. einen REST-Service aufrufen, der die Daten zurückliefert und anschließend die erhaltenen Daten für die jeweilige Gemeinde darstellen.

Dabei muss zuerst serverseitig die REST-API definiert und implementiert werden, welche dann die ganzen Daten aus der vorhandenen Datenbank zur Verfügung stellt.

Diese Arbeit behandelt nur die ersten 4 Module dieser Tourismus-Applikation: Nachrichten, Veranstaltungen, Live-Reports und Karte.

Sehr gerne wird für diese Art der Entwicklung des mobilen Clients das Cross-Plattform-Framework *PhoneGap/Cordova* benutzt, denn es können die bekannten Webtechnologien (CSS3, HTML5, Javascript) für die Programmierung genutzt werden. Ziel dieser Arbeit ist es auch, hierzu einen alternativen Ansatz zu finden.

1.2 Gliederung der Arbeit

Der weitere Aufbau der Arbeit befasst sich zuerst mit den grundlegenden Entwicklungsmöglichkeiten einer nativen App in Android, Windows Phone und iOS, sowie der Entwicklung einer Web-App in Kapitel 2.

Dann werden die verschieden Arten von Cross-Plattformen in Kapitel 3 vorgestellt und dann daraufhin untersucht, welche am besten für die Verwendung der Cross-Plattform-App geeignet ist, die im weiteren Verlauf vorgestellt wird.

Anschließend wird, nachdem die Anforderungen für die mobile App in Kapitel 4 ermittelt wurden, der Entwicklungsprozess der App mit der Cross-Plattform in Kapitel 5 vorgestellt. Hierbei wird zuerst auf die serverseitige Entwicklung der REST-API eingegangen, bevor dann die clientseitige mobile Anwendung entwickelt wird.

Es folgt in Kapitel 6 ein Anforderungsabgleich mit den Anforderungen an die mobile Applikation.

Das abschließende Kapitel 7 zieht ein Fazit zur Cross-Plattform-Entwicklung von mobilen Applikationen und gibt einen Ausblick für die zukünftige Entwicklung mobiler Applikation. Ebenso wird die mögliche Weiterentwicklung der implementierten mobilen Applikation diskutiert.

2

Grundlagen zur Entwicklung mobiler Applikationen

Die drei meistgenutzten Betriebssysteme für mobile Endgeräte sind heutzutage Android, iOS und Windows Phone, glaubt man der Statistik von IDC. Rund 80 % deckt dabei Android, 13 % iOS und ca. 4 % Windows Phone ab [IDC13].

Bevor die plattformunabhängige Programmierung einer mobilen Applikation in Kapitel 5 vorgestellt wird, wird die native Entwicklung von Applikationen der drei bekanntesten Betriebssysteme für mobile Geräte betrachtet. Im zweiten Teil dieses Kapitels wird dann auf die Entwicklung von Web-Apps eingegangen.

Eine native Applikation zeichnet sich dadurch aus, dass sie speziell für eine bestimmte Zielplattform geschrieben und kompiliert wurde, also direkt auf ein Endgerät zugeschnitten wird. Gewöhnlich werden solche Apps über einen App Store oder Marketplace veröffentlicht und verkauft [Vis14].

2.1 Android OS

Android ist ein Open-Source-Betriebssystem, das von dem Unternehmen Android Inc. entwickelt wurde, das dann später von Google aufgekauft wurde. Die Open Handset

Alliance[1], welche einen offenen Standard für die Entwicklung von Mobiltelefonen vorsieht, erkennt Android als solchen offenen Standard an [Kün12].

Das OS Android basiert auf einem Linux-Kernel (aktuell in Version 3,x) [Lic13], der aus Hardware-, IPC- und Gerätetreibern besteht und für die Prozess- und Speicherverwaltung zuständig ist (siehe Abbildung 2.1) [BP09].

Das Wesentliche des Android OS ist die Laufzeitumgebung *Android Runtime*. Sie besteht zum einem aus Java-Klassenbibliotheken, wie sie bei der Virtuellen Maschine (VM) von Sun üblicherweise vorzufinden sind[2], und zum anderen aus der *Dalvik Virtual Machine* (DVM); diese wandelt die Java .class-Dateien mit Hilfe des *dx*-Tools in einen *.dex-Code um (siehe auch Anhang A.2, Abbildung A.1), nimmt also die Umwandlung von Stapelmaschinencode in Registermaschinencode vor. Wird eine App auf dem mobilen Endgerät gestartet, so bekommt sie eine eigene DVM zugeteilt. Dies hat den Vorteil, dass die App nur innerhalb dieser DVM lebt und bei Termination der Anwendung auch nur diese beendet wird [BP09].

Des Weiteren gibt es Standardbibliotheken (*libraries*), die in C oder C++ geschrieben sind und allen Anwendungen zur Verfügung stehen. Sie enthalten Schnittstellen zu Datenbanken, Grafikbibliotheken, Webzugriffen, Multimedia-Verwaltung, etc. Der Entwickler greift darauf indirekt über das Android Application Framework zu.

Weitere Bestandteile der Architektur sind die inbegriffenen Programmierschnittstellen, das sogenannte Android Application Framework, das jene Programmierschnittstellen bereitstellt, die die Kommunikation zwischen Endanwender und Anwendung ermöglichen. Außerdem bietet das Framework Zugriff auf die Hardware des Smartphones (Kamera, Netzwerk, Sensoren) oder ermöglicht das Lesen und Schreiben von Kontaktdaten oder Terminen. Auch kann eine Applikation ein andere Applikation aufrufen bzw. deren Funktionen anfordern. Zu den wichtigsten Bestandteilen einer Applikation gehören die Views, die die Benutzeroberflächen bilden. Darüber hinaus gibt es Content Provider, welche Applikationen Zugriff auf andere Applikationen ermöglichen. Ein weiterer Bestandteil ist der Resource Manager, der auf lokalisierte Zeichenketten, Grafiken und Layoutdaten Zugriff bietet; der Notification Manager hingegen gewährt Zugriff auf Status-Meldungen. Außerdem gibt es noch den Activity Manager, der den Lebenszyklus einer Anwendung steuert [Kün12].

[1] Die Open Handset Alliance ist ein Zusammenschluss von mehreren Softwarefirmen, Chip-, Prozess-, Smartphoneherstellern und Netzbetreibern, siehe auch http://www.openhandsetalliance.com/

[2] Dazu gehören Pakete wie *java.lang*, *java.io*, *java.math*, *java.net*, *java.util*, etc.

Abschließend müssen die eigentlichen Applikationen erwähnt werden. Dabei greift eine App auf die oben genannten Programmierschnittstellen (APIs) zu (siehe Abbildung 2.1) [BP09, Kün12].

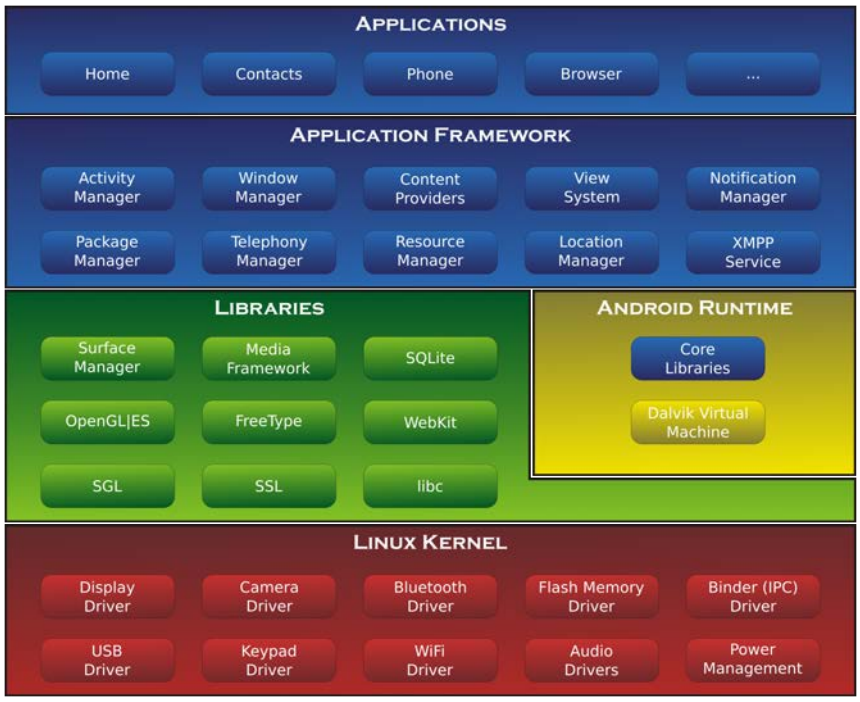


Abbildung 2.1: Android Architektur (nach [Goo13a])

2.1.1 Anwendungsentwicklung

Das von Google zur Verfügung gestellte SDK enthält einige Tools für die Anwendungsentwicklung von Android Apps, welche in Java geschrieben werden. Es beinhaltet das Plugin für die Open-Source-basierte Entwicklungsumgebung Eclipse, das sogenannte ADT, das die Entwicklung vereinfacht. Indem es zahlreiche Hilfsmittel wie Wizards, Editoren und Views enthält, unterstützt es bei der Erstellung von Applikationen [Kün12]. Zusätzlich existiert mit dem Native Development Kit (NDK) die Möglichkeit Teile der Apps in C/C++ zu entwickeln, was in einer höheren Performance resultiert [Goo14a].

Ein wichtiges Tool zur Unterstützung der Entwicklung ist der Emulator. Er bietet die Möglichkeit, mobile Geräte inklusive ihrer Hard- und Software zu simulieren, um die entwickelten Applikationen zu testen. Die Einstellungen dafür können im AVD-Manager vorgenommen werden. Ebenso enthalten sind die unterschiedlichen Android-Plattformen (Cupcake 1.5, Froyo 2.2.,Gingerbread 2.3, Honeycomb 3.x, Ice Cream Sandwich 4.x) sowie Dokumentation und Programmier-Beispiele. Auch sind weitere Tools für das

Layouting und die grafischen Oberflächen integriert. Für eine vollständige Auflistung aller Tools siehe auch [Goo14c].

Die Android App kann über den Google Play Store zur Verfügung gestellt werden. Dabei fällt lediglich eine einmalige Registrierungsgebühr (aktuell 25 $) für den Entwickler an. Kostenlose Apps können einfach bereitgestellt werden. Will man allerdings eine kostenpflichtige App über den Play Store verkaufen, so beansprucht Google 30 % der Einnahmen für sich [Goo13b, Goo14b].

2.2 iOS

Apple brachte 2007 das erste iPhone auf den Markt [App14f]. Allerdings konnte der Endanwender nur die von Apple bereitgestellten Apps nutzen, da das iPhone ein gesperrtes Smartphone war. So kam schnell der Wunsch nach einem SDK auf, damit auch andere Entwickler Applikationen für das iPhone programmieren konnten. So hat Apple 2008 das eigene SDK öffentlich zugänglich gemacht [Kim08].

Das iOS fungiert als System-Schnittstelle zwischen den Apps und der Hardware des mobilen Endgeräts. Das iOS Device Framework besteht dabei im Wesentlichen aus vier Schichten (siehe Abbildung 2.2). Die unterste Schicht, *Core OS*, bietet quasi die Systemschnittstelle zur Hardware. Es werden Hardware Accelerator, Bluetooth Unterstützung, externe Zugriffe, Security und das System (Kernel, Treiber, Prozessverwaltung, Filesystem, etc.) zugänglich gemacht. Die Frameworks der oberen Schichten beinhalten alle diese Technologien, möchte man aber explizit darauf zurückgreifen, so benutzt man diese Schicht [App13c].

Die *Core Services* Schicht stellt mit dem Core Foundation Framework und Foundation Framework die Grundlagen für die Entwicklung einer App zur Verfügung. Sie beinhaltet wesentliche Funktionen, auf denen höhere Schichten aufbauen. Hierzu gehören Datenbankfunktionen, Datenverwaltung, Zugriff auf den App Store und Dienste wie iCloud, Social Media und die Lokalisierung [App13d].

Die dritte Schicht, *Media*, ermöglicht die Integration von Grafik- (OpenGL ES Framework), Audio- (AV Foundation Framework) und Videoelementen (Media Player Framework). Ebenso sind das Zeichnen zweidimensionaler Elemente (Core Graphics Framework) und Animationen möglich [App13f].

Die oberste Schicht, *Cocoa Touch*, enthält Frameworks, die das Aussehen der App gestalten, also die grafische Oberfläche für den Entwickler bereitstellen. Ebenso enthal-

ten diese Frameworks die grundlegende Programmlogik und weitere Technologien wie Multitasking, die Push-Erinnerungs-Funktion, touch-basierter Input und einige Services [App13b].

Allgemein empfiehlt Apple, dass man ein Framework einer höheren Schicht einem Niedrigeren vorziehen soll, falls dieses die gleichen Funktionen bietet [App13e].

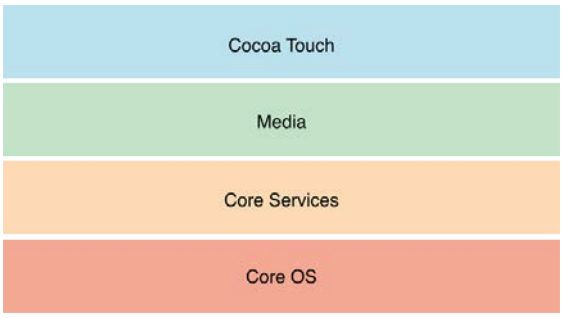


Abbildung 2.2: Layer des iOS (nach [App13e])

2.2.1 Anwendungsentwicklung

Die Apps werden in Objective-C programmiert, einer Programmiersprache, die auf C basiert und objektorientierte Konzepte nutzt. Auch Apple bietet ein eigenes SDK für die Entwicklung von iOS Anwendungen an, das die benötigten Klassen und einige Tools zum Debuggen und Testen der App zur Verfügung stellt. Die Entwicklung selbst erfolgt in der bekannten IDE Xcode von Apple. Dies setzt allerdings einen Macintosh OS X-fähigen Rechner voraus [App13e].

Xcode enthält einen iOS-Simulator, der es auch ermöglicht, die i-Geräte zu simulieren und die entwickelte Applikation darauf zu testen. Es kann kostenlos über den App Store heruntergeladen werden. [App14h]

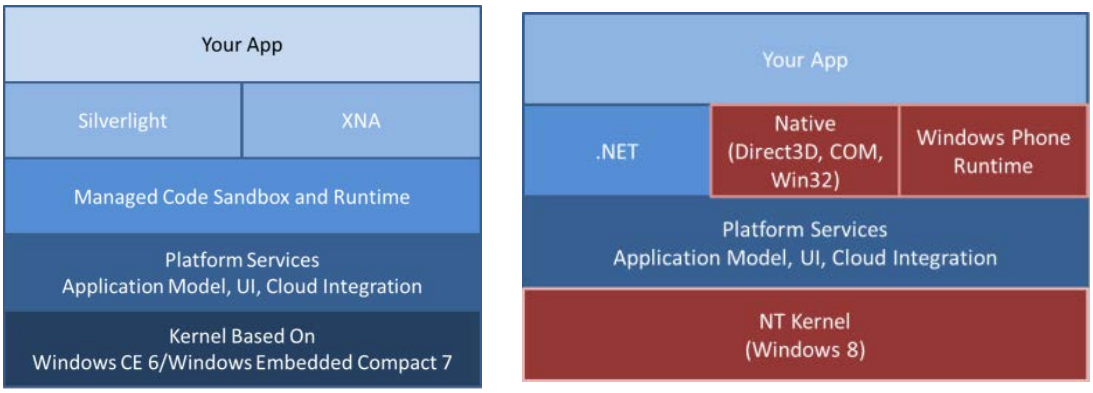
Möchte man seine entwickelte Apps im App Store veröffentlichen, so muss man kostenpflichtig (aktuell: 99 $ pro Jahr) dem iOS Developer Program beitreten. Ebenso wie Google behält Apple 30 % der Einnahmen für sich [App14g].

2.3 Windows Phone

Den drittgrößten Marktanteil der Betriebssysteme für mobile Endgeräte hat das Windows Phone. Es kommt im 2. Quartal des Jahres 2013 auf einen Marktanteil von rund 4 % [IDC13]. Es ist aber erst seit 2010 auf den Markt [Hol10].

Windows Phone ist das Nachfolger-Betriebssystem von Windows Mobile. Bisher gibt es nur zwei Versionen davon, die erste ist die Version 7, welche noch auf dem alten

Windows CE basiert, die aktuelle ist Phone 8, welche nun den gleichen Kernel wie das Windows 8 Betriebssystem besitzt [Rub12].



(a) Architektur des Windows Phone 7

(b) Architektur des Windows Phone 8

Abbildung 2.3: vereinfachte Darstellung der Windows Phone Architekturen (nach [Kuh12])

Die Architektur von diesen beiden mobilen Betriebssystemen kann ebenfalls als Schichtenmodell betrachtet werden. Jedoch gibt es leichte Unterschiede, wie man in Abbildung 2.3 sehen kann.

Windows Phone 7 besitzt noch den alten Kernel von Windows CE 6, wohingegen Windows Phone 8 auf dem NT-Kernel basiert. Beide Versionen bieten die Platform Services an, welche Dienste wie das UI, das Application Model (welches die App erstellt, zentrale Systemfunktionen und Laufzeitinformationen bietet und den Lebenszyklus der App steuert) und die Cloud Integration bereitstellen [Mic14f].

Windows Phone 7 hat noch eine eigene Sandbox- und Runtime-Umgebung, in der die Applikationen ausgeführt werden. Ebenso stellt es noch Technologien wie XNA und Silverlight zur Verfügung. Diese werden von Windows Phone 8 nicht mehr unterstützt.

Bei Windows Phone 8 ist die Laufzeitumgebung eine abgeschlackte Form des Windows 8 Runtime Environment. Der Kern von Windows Phone 8 ist ja identisch mit dem normalen Windows 8 Betriebssystem. Die hierzu zugehörige .NET-API besitzt Klassenbibliotheken, Systemtypen und die Microsoft.Phone-Namensräume. Zusätzlich zu diesen APIs hat man in Phone 8 noch die Möglichkeit mit den APIs Win32, COM und Direct3D weitere low-level Eigenschaften, wie z. B. Kamera, Rendering, Audioaufnahmen oder Winsock für Networking, also alles was mit nativer Implementierung zu tun hat, zu nutzen [Mic14d].

2.3.1 Anwendungsentwicklung

Windows Phone Apps werden in C# oder in Visual Basic.NET geschrieben. Die Programmierung erfolgt dabei in Visual Studio, der IDE von Microsoft. Auch das Windows Phone SDK stellt selbstverständlich einen Emulator zur Verfügung.

Auch wenn Windows Phone 8 kein Silverlight und XNA mehr unterstützt, laufen dennoch die für Windows Phone 7 programmierten Apps auf Windows Phone 8, jedoch nicht umgekehrt [Mic14e]. Möglich ist dies über den sogenannten Quirks Modus. Entweder wird der Code einer mobilen Applikation direkt ausgeführt oder über den Quirks Modus in anderen Code transformiert, welche das alte Verhalten von Windows Phone 7 nachahmt, wie es in Abbildung 2.4 dargestellt ist [Kuh12].

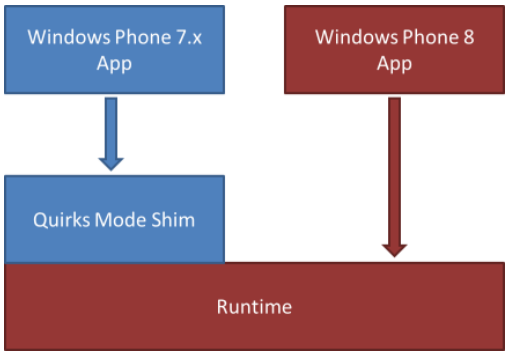


Abbildung 2.4: Ausführung von Windows Phone 7 Code auf Windows Phone 8 (nach [Kuh12])

Des Weiteren bietet das SDK eine Möglichkeit, nativen Code in C++ zu integrieren [Mic14b].

Für einen Developer Account bei Microsoft und die damit verbundene Veröffentlichung im Windows Store zahlt man jährlich als Entwickler 14 € oder 75 € als Unternehmen [Mic14a].

Beim Verkauf neuer Apps muss man 30% der Erlöse Microsoft als Store-Gebühren überlassen [Mic14c].

2.4 Webbasierte Entwicklung

Im Gegensatz zu einer gewöhnlichen Webseite zeichnet sich eine mobile Web-Applikation dadurch aus, dass sie dem typischen Softwareschema Eingabe, Verarbeitung und Ausgabe im Webbrowser folgt. Typisches Beispiel für eine solche Applikation ist die Seite

Google Mail https://mail.google.com[3]. Man loggt sich ein, empfängt die Mails und kann sich diese anzeigen lassen. Durch die Optik entsteht hierbei eher der Eindruck, sich in einem Programm, anstatt auf einer Webseite zu befinden. Eine klare Trennung dieser beiden Begriffe *Web-App* und *mobile Website* gibt es jedoch nicht [FI13].

Ebenso gibt es keine allgemeingültige Definition einer mobilen Web- Applikation. Vielmehr wird versucht, gewisse Eigenschaften mit einer mobilen Web-Applikation zu assoziieren.

Diese benötigt dabei immer einen Browser als Client, in dem sie ausgeführt wird [Nat14].

Während eine native App immer plattformabhängig ist, also mit einer Programmiersprache geschrieben, die dann in plattformspezifischen Maschinencode umgewandelt wird, so ist die Web-App nur abhängig von dem Browser, in dem sie ausgeführt wird. Ausgeführt wird sie dabei regelmäßig in einer WebKit Browser-Engine, welche die mittlerweile gängigen Features wie HTML5 und CSS3 unterstützt. Die WebKit-Engine bildet darüber hinaus die Grundlage für die Engines der Browser Google Chrome[4] und Opera[5]. Der Internet Explorer[6] und Firefox[7] nutzen eigene Engines.

Der Aufruf in einem Browser erfordert jedoch immer eine Internetverbindung.

Eingesetzte Programmier- und Skriptsprachen für die Web-App sind HTML5, sowie JavaScript für die Logik und CSS3 für das Layout.

2.4.1 HTML5, JavaScript und CSS3 in der mobilen Webentwicklung

Als neueste Version von HTML bietet HTML5 weitere Features, die speziell für mobile Endgeräte entwickelt wurden. Im Gegensatz zur ausschließlich hierarchischen Anordnung, wie man sie bei klassischen Websites kennt (Überschriften, ...), werden mit HTML5 semantische Elemente unterstützt. Folgende Tags stehen dabei unter anderem zur Auswahl: *section, nav, article, aside, hgroup, header, footer*.

Ebenso gibt es auch einige weitere Tags für Multimedia, wie *canvas, audio, video*, oder neue *input* types für Formularelemente, sowie *time* und *mark* für Texte und noch weitere Tags, die aber für diese Arbeit nicht weiter von Relevanz sind [FI13].

[3] aufgerufen am:06.10.2014
[4] http://www.chromium.org/blink, aufgerufen am: 24.09.2014
[5] http://thenextweb.com/insider/2013/04/04/opera-confirms-it-will-follow-google-and-ditch-webkit-for-blink-as-part-of-its-commitment-to-chromium/, aufgerufen am: 24.09.2014
[6] http://msdn.microsoft.com/de-de/library/aa741317.aspx, aufgerufen: 24.09.2014
[7] https://developer.mozilla.org/de/docs/Gecko, aufgerufen am: 24.09.2014

Allerdings ist diese Markupsprache immer noch in der Entwurfsphase und noch nicht offiziell von der W3C Kommission beschlossen, ist aber bereits weit verbreitet. Da es noch keinen einheitlichen Parser gibt, kann jeder Browser selbst entscheiden, wie er die HTML5 Tags parsen möchte. Das kann mitunter zur unterschiedlichen Integration und Darstellung von Elementen führen. Einen ersten Arbeitsentwurf (Working Draft) [8] für solch einen einheitlichen Parser gibt es bereits. Bis Ende 2014 soll eine einheitliche Empfehlung (Candidate Recommendation) von der W3C Kommission beschlossen werden [W3C13b].

JavaScript ist die Programmiersprache für Webseiten, mit der das Verhalten der Webseite gesteuert wird. JavaScript wird dabei von einem Interpreter, der in dem Webbrowser enthalten ist, über die DOM-Schnittstelle ausgeführt. JavaScript unterscheidet sich von der Programmiersprache Java, orientiert sich aber bei der Syntax an Java und implementiert objektorientierte und funktionale Konzepte [Fla11].

Cascading Style Sheets (CSS), eine textbasierte Gestaltungssprache, beschreibt das Aussehen der Seite mit Hilfe eines Stylesheets, in welchem Layout, Farben und Schrift beschrieben werden. Es kann somit das Aussehen an unterschiedliche Endgeräte angepasst werden. Ein Vorteil der Nutzung von CSS ist die Trennung von Inhalt und Aussehen. Damit kann ein Stylesheet leicht für andere Seiten wiederverwendet werden [W3C13a].

Die Version CSS3 bietet für die Erstellung des Layouts neue Möglichkeiten, speziell für mobile Endgeräte, wie Media Queries. Diese Neuerungen sollen eine Webseite möglichst wie eine native App aussehen lassen. Für CSS3 gibt es bereits eine erste Candidate Recommendation von der W3C Kommission [W3C14].

2.4.2 Eine webbasierte App offline verfügbar

Da die Web-App eine ständige Internetverbindung benötigt, um die HTML-, CSS- und JavaScript-Dateien vom Server zu laden, ist die App nicht offline ausführbar. Deswegen muss man sich überlegen, wie man die Daten dennoch offline auf dem mobilen Gerät speichern kann. Grundsätzlich gibt es hierzu zwei Lösungen.

Zum einen besteht die Möglichkeit, Caching-Mechanismen zu benutzen. Jeder Browser verfügt über solche Caches, allerdings funktionieren diese nicht immer für mobile Geräte.

[8] Working Draft ist der erste Entwurf , Candidate Recommendation die 2. Stufe, die letzte die Proposed Recommendation der W3C Kommission, siehe auch http://www.w3.org/2005/10/Process-20051014/tr.html#rec-advance

Deswegen wurde mit HTML5 eine Application-Cache-Schnittstelle eingeführt, nämlich eine Cache-Manifest-Datei, die Daten offline speichert. Dabei gibt es drei wichtige Keys für dieses Manifest, siehe Listing 2.1. Unter *CACHE* gibt man die Dateien an, welche offline gespeichert werden sollen, unter *NETWORK* definiert man, welche Dateien unbedingt vom Server geladen werden und schließlich gibt es noch das *FALLBACK*, bei dem beschrieben wird, was passiert, wenn eine Datei nicht online geladen werden kann (z. B. zeigt man stattdessen eine andere Datei an).

Eine andere Möglichkeit Daten im Browser offline zu speichern besteht darin, ein sogenanntes *localStorage*-Objekt, also eine Offline-Datenbank, zu benutzen, in der die Daten gespeichert werden. Angesprochen wird dieses Objekt über JavaScript. Vorteil dieser Variante ist, dass die Daten auch nach dem Beenden der Web-App oder nach dem Ausschalten des Smartphones erhalten bleiben [FI13].

Listing 2.1: Beispiel Cache-Manifest nach [Bid13]

```
1  CACHE MANIFEST
2
3  # Explicitly cached 'master entries'.
4  CACHE:
5  /favicon.ico
6  index.html
7  stylesheet.css
8  images/logo.png
9  scripts/main.js
10
11 # Resources that require the user to be online.
12 NETWORK:
13 login.php
14 /myapi
15 http://api.twitter.com
16
17 # static.html will be served if main.py is inaccessible
18 # offline.jpg will be served in place of all images in images/large/
19 # offline.html will be served in place of all other .html files
20 FALLBACK:
21 /main.py /static.html
22 images/large/ images/offline.jpg
23 *.html /offline.html
```

Es ist einfacher, eine Web-App als eine native Applikation zu entwickeln. Denn das Know-How der einzelnen Plattformen ist hierfür nicht erforderlich. Lediglich die Webtechnologien müssen für die Programmierung verwendet werden. Jedoch hat man mit den Webtechnologien keine Möglichkeit auf die Hardware eines mobilen Endgerätes zuzugreifen. Die Cross-Plattform-Frameworks, die die Vorteile beider Entwicklungsmöglichkeiten kombinieren, werden im folgenden Kapitel 3 vorgestellt.

3

Cross-Plattformen

Eine Cross-Plattform-Entwicklung bedeutet in der Regel, dass eine Software auf mehreren Plattformen laufen soll. Hier gibt es jedoch zwei prinzipielle Definitionen.

Man denke nur an die Programmiersprache Java. Java wird als plattformunabhängig angesehen, da es auf allen Betriebssystemen läuft, auf denen ein Java Runtime Environment verfügbar ist, in dem der Java Bytecode ausgeführt werden kann. Ebenso gilt auch ein interpretierbarer Code weitestgehend als plattformunabhängig, da nur das Skript-Environment gegeben sein muss (z. B. Javascript). Auch eine Webanwendung, die nur von einem Browser abhängig ist, gilt als plattformunabhängig, da ein Browser auf den meisten Geräten und Plattformen vorhanden ist.

Alternativ kann eine Cross-Plattform-Software auch plattform-spezifisch sein, wenn einfach für dieselbe Software individueller Code oder mehrere Kompilierungen für die jeweiligen Plattformen vorhanden sind [TCLC14, Tec14].

In dieser Arbeit wird die erste Definition von Plattform-Unabhängigkeit verwendet, d. h. dass eine solche Anwendung für verschiedene Zielplattformen einen einzigen Quellcode verwendet.

Ziel einer solchen Entwicklung ist dabei die Nutzung eines gemeinsamen Quellcodes, auch die *Shared Code Base* genannt. Anstatt mehrerer Quellcodes für die entsprechenden Plattformen existiert nur ein gemeinsamer Quellcode, der die mobile Applikation für die Betriebssysteme der Plattformen erzeugt. Die Vorteile eines solchen Code scheinen offensichtlich.

Man muss nur einen einzigen Code für alle Plattformen entwickeln, was in geringen Entwicklungskosten und einer schnelleren Veröffentlichung der Anwendung resultiert. Ebenso ist der Wartungsaufwand geringer, da nur ein Quellcode angepasst werden muss. Ein weiterer Vorteil ist, dass durch die Distribuierung auf verschiedene Plattformen mehr Kunden erreicht werden können [AT11].

Es gibt dabei verschiedene Arten von Frameworks, die eine solche Cross-Plattform-Entwicklung unterstützen. Zum einen gibt es native Frameworks, welche auch Schnittstellen, also native Funktionen, zum jeweiligen Endgerät bereitstellen. Das kann einerseits über einen nativen Container ermöglicht werden, andererseits wird der Quellcode zusammen mit einem nativen kompilierten Code von einem Interpreter ausgeführt. Alternativ wird der Quellcode in einen nativen Code umgewandelt oder es wird direkt in einer plattformübergreifenden nativen Programmiersprache entwickelt. Die fünfte Möglichkeit besteht aus der Wiedergabe mit Hilfe von Adobe Flash.

Darüber hinaus gibt es mobile Web Frameworks, die mit Hilfe von Webtechnologien versuchen, das Aussehen einer nativen Applikation der jeweiligen Plattform zu imitieren. Dabei wird der Code in einer *embedded* Umgebung, also beispielsweise dem Browser, ausgeführt.

Im Folgenden werden die verschiedenen Frameworks vorgestellt und mit den Anforderungen an ein solches Framework abgeglichen. Auf Grundlage dieser Untersuchung wird eine Wahl für das zu benutzende Framework für die Implementierung der zu entwickelnden Applikation getroffen.

Es werden hierbei nur die relevantesten Plattformen der unterschiedlichen Varianten zur Entwicklung vorgestellt. Nicht vorgestellte Frameworks finden sich im Anhang unter A.3. Da ständig neue Plattformen vorgestellt und alte Entwicklungen nicht mehr aktualisiert werden, bietet auch die Literatur in dieser Hinsicht keine aktuelle Übersicht und keine klare Trennung oder Abgrenzung dieser Frameworks. Einen Vorschlag für eine Klassifikation dieser Cross-Plattformen bietet die Diplomarbeit von Toca, [AT11], mit den Kategorien *Web, Boxed web* (Plattformen mit nativer Hülle), *hybride* (wie Boxed web + native UI), *interpretierbare* (Plattformen mit Interpreter) und *native Plattformen*, wobei auch hier einige der gelisteten Cross-Plattformen inzwischen veraltet sind. In der hier vorliegenden Arbeit wird eine leicht vereinfachte Abwandlung dieser Aufteilung verwendet. Die prinzipielle Trennung von *nativen und mobilen Web Frameworks* wird übernommen, wobei innerhalb der nativen Frameworks nochmals eine Unterscheidung

zwischen *Frameworks mit nativem Container, Frameworks mit interpretierten Code, Frameworks mit Umwandlung in Binärcode, Frameworks, die ganz mit nativem Quellcode geschrieben werden* und *Frameworks, die auf dem Adobe Flash Player basieren,* stattfindet (siehe auch Kapitel 3.3).

3.1 Bewertungskriterien einer Cross-Plattform-Technologie

Es gibt verschiedene Versuche, bestimmte Kriterien für die Auswahl eines Cross-Plattform-Frameworks für die Entwicklung einer mobilen Applikation zu finden. Als erfolgreich haben sich bestimmte Kriterien aus der Praxis und von Seiten der Experten erwiesen. Man kann diese im Wesentlichen auf zwei verschiedene Sichtweisen betrachten. Die *Applikationsperspektive* beschäftigt sich mit den Funktionen und dem Lebenszyklus einer App, wohingegen sich die *Entwicklungsperspektive* mit der Entwicklungsmöglichkeiten der Cross-Plattform-Technologie selbst beschäftigt (siehe folgende Kriterien aus [HHM13, AT11]):

Applikation

Kriterium	**Fragestellung**
Lizenzierung & Kosten	Ist das Framework Open-Source-basiert und unter welcher Lizenz ist das Framework veröffentlicht?
Unterstützte Plattformen	Welche mobilen Betriebssystem zur Entwicklung werden von dem Framework unterstützt?
Hardwareunterstützung	Kann ich mit dem Framework auf native Elemente, also Hardware, zugreifen?
Langzeitunterstützung	Wird das Framework auch weiterentwickelt, gibt es eine aktive Community?
Look & Feel	Unterstützt die Cross-Plattform native Bedienelemente?
Anwendungsperformance	Wie fühlt sich die subjektive Geschwindigkeit zur Start- und Laufzeit der Applikation an?
Distribution	Wie einfach ist es, mobile Applikation auf verschiedene Distributionsmöglichkeiten zu vertreiben? Welche Möglichkeiten gibt es zusätzlich zu den plattformspezifischen App Stores?

Entwicklung

Kriterium	Fragestellung
Entwicklungsumgebung	Wie gut unterstützt die Entwicklungsumgebung Auto-Completion und automatisches Testen?
GUI Design	Ist es möglich mit einem grafischen Editor UI-Elemente zu erstellen?
Einfache Entwicklung	Gibt es Dokumentation, Codebeispiele und Links zu ähnlichen Problemen?
Wartung	Wie viel Lines of Code werden für die Implementierung benötigt? Auch wenn LOC keine optimale Metrik für die Komplexität ist, so ist sie dennoch einfach ermittelbar und liefert einen ersten Anhaltspunkt. Andere Metriken wären hierfür sinnvoller.
Skalierbarkeit	Wie gut ist der Code modularisiert und wie gut können große Teams mit dem Framework arbeiten?
Weiterentwicklung	Wie gut kann ich den Code für andere Cross-Plattform-Ansätze wiederverwenden?
Zeit und Kosten der Entwicklung	Wie hoch ist der Zeitaufwand und der damit verbundene Kostenaufwand für die Entwicklung?

Oft wird für die Auswahl auch eine gewichtete Betrachtung der einzelnen Kriterien vorgenommen.

3.2 Anforderungen

Um eine geeignete Auswahl der Cross-Plattform für die zu entwickelnde App treffen zu können, muss zunächst eine Anforderungsanalyse an eine solche Cross-Plattform durchgeführt werden. Die Auswahl wird in Kapitel 5 vorgestellt.

Als zu unterstützende Betriebssysteme werden Android und iOS, optional auch noch Windows Phone, definiert, da diese aktuell die wichtigsten und am weitesten verbreiteten Betriebssysteme für mobile Endgeräte sind. Die Distribution soll über die App Stores der jeweiligen Plattform möglich sein.

Natürlich sollte es auch möglich sein, einen REST/HTTP-Request wie GET, PUT, POST und DELETE zu machen.

Auch sollte die Cross-Plattform aktiv weiterentwickelt und gepflegt werden, damit Sicherheitslücken und Fehler behoben werden. Der Entwickler soll sich schnell in die Entwicklung mit dem bereitgestellten Framework einarbeiten können. Dies setzt auch eine gute Entwicklungsumgebung voraus. Weiter sollte diese Entwicklungsumgebung, sowie das Framework möglichst Open-Source oder sonst frei verfügbar sein.

Native Bedienelemente wie auch hardwarenahe Funktionen sollten durch das Framework bereitgestellt werden.

Weitere Aspekte, die in Kapitel 3.1 aufgelistet wurden, werden als weiche Kriterien für die Auswahl eines passenden Frameworks angesehen.

Für einen Vergleich von verschiedenen Frameworks bietet Markus Falk eine Homepage an, auf der Anforderungen definiert werden können und Vorschläge zur Auswahl eines passenden Frameworks präsentiert werden (http://mobile-frameworks-comparison-chart.com/[9]). Auch hier sind nicht alle aktuellen Frameworks vorhanden, die Seite wird jedoch ständig erweitert.

Frameworks, die den genannten Anforderungen nicht entsprechen, sind im Anhang aufgezählt (siehe Kapitel A.3).

3.3 Native Frameworks für mobile Endgeräte

Im Folgenden werden einige Frameworks vorgestellt und bezüglich der Unterstützung eines REST-Aufrufs zur Kommunikation mit einem Server untersucht. Hierbei wird nochmals zwischen Frameworks, die den Quellcode in nativen Binärcode für die entsprechende Plattform umwandeln oder Quellcode in einer nativen Umgebung ausführen, unterschieden. Ebenso gibt es Frameworks, die nur einen nativen Container auf der jeweiligen Plattform bereitstellen und dann über Webtechnologien die Applikation in diesem Container darstellen. Die vierte Möglichkeit besteht in der Verwendung von Programmiersprachen, die es erlauben, auf alle mobilen Endgeräten nativ zu programmieren. Die letzte Möglichkeit der nativen Frameworks ist schließlich die Ausführung der App mit Adobe Flash.

[9] aufgerufen am: 07.10.2014

3.3.1 Native Frameworks mit nativem Container

Die App wird hierbei über den nativen Container ausgeführt, welcher die App bildschirmfüllend darstellen soll und Zugriffe auf die Hardware-Schnittstellen erlaubt.

3.3.1.1 PhoneGap/Cordova

Eines der bekanntesten Frameworks aus dieser Kategorie ist PhoneGap. Es wurde unter der Firma Notobi gegründet, welche 2011 von Adobe aufgekauft wurde. Das Open-Source Framework ermöglicht die Programmierung über die bekannten Webtechnologien JavaScript, HTML5 und CSS3 [AS14a, OR11].

Die PhoneGap Codebasis wurde 2011 von Adobe der Apache Software Foundation unter dem Namen Apache Cordova, welche nun unter der Apache Lizenz 2.0 veröffentlicht ist, übergeben. Seither ist Adobe PhoneGap eine Distribution von Apache Cordova [Fou13].

PhoneGap ermöglicht folgendermaßen die native Entwicklung: Der erzeugte Code für die App wird nicht in dem Browser, sondern in einer nativen Hülle ausgeführt, welche die Aufgabe hat, die Web-App bildschirmfüllend anzuzeigen. PhoneGap generiert den nativen Code für die jeweilige Plattform größtenteils nur für die Komponente, die diese Applikation anzeigt. Die Applikation selbst ist jedoch noch immer eine Web-App, die ausgeführt wird. Jedoch kann sie über die bereitgestellte Hülle wie eine native App auf gewisse Hardwarekomponenten wie Kamera, Dateisystem und Telefonbuch zugreifen. Möchte man die Applikation für eine Plattform deployen, so wird ein Projekt erstellt, das mit dem plattformspezifischen SDK kompiliert wird und den nativen Container bereitstellt. Dieser nativen Container wird je nach Plattform unterschiedlich implementiert. Über JavaScript kann die mobile Applikation dann mit dem nativen Container asynchron kommunizieren und so auf die nativen Elemente zugreifen. Dabei wird ein IFrame-Objekt erzeugt, das eine URI (gap://) lädt, mit der die native Applikation umgehen kann [Whi12, SHS13].

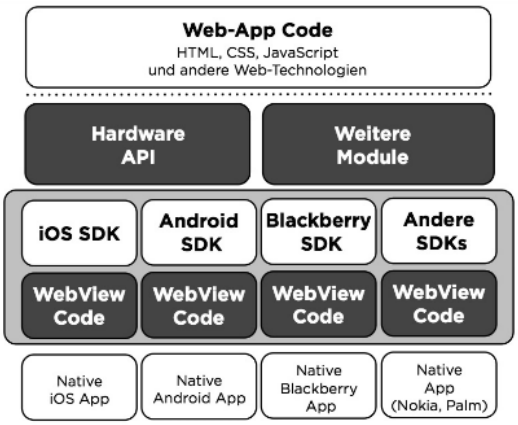


Abbildung 3.1: PhoneGap Funktionsweise (nach [SHS13])

Für jede Plattform erstellt PhoneGap mit dem erstellten Web-App Code, dem WebView Code und dem plattformspezifischen SDK eine native App (siehe auch Abbildung 3.1) [SHS13].

Ebenso unterstützt PhoneGap native Plugins, wie beispielsweise jQuery Mobile (siehe auch Kapitel 3.4.1), womit der Funktionsumfang der implementierten Anwendung erweitert werden kann [pho14].

Des Weiteren existiert mit PhoneGap Build ein cloud-basierter Service, der es ermöglicht, die Apps mit PhoneGap ohne Verwendung der plattformspezifischen SDKs über die Cloud für die jeweilige Plattform zu erstellen und zu übersetzen. Man benötigt also nicht mehr einen Mac OS-fähigen Rechner, um eine Applikation für iOS zu kompilieren. Gleiches gilt auch für Windows. Open-Source Applikationen können darüber kostenlos erstellt werden. Private Apps kosten $9,99 im Monat für 25 Apps oder man besitzt eine Adobe Creative Cloud Membership[10] [AS14a].

Dabei unterstützt PhoneGap inzwischen eine große Anzahl von nativen Features auf den verschiedenen Plattformen, wie in der Abbildung 3.2 zu sehen ist.

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 6.0+	Blackberry 10	WebOS	Windows Phone 7 + 8	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	x	✓	✓	x	✓	✓	✓	x	✓
Contacts	✓	✓	✓	✓	✓	x	✓	✓	✓
File	✓	✓	✓	✓	✓	x	✓	x	x
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	x	✓	x	✓	x	x
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	x	x

✓ - supported feature
X - unsupported feature due to hardware or software restrictions

Abbildung 3.2: unterstützte Features von PhoneGap (nach [AS14c])

PhoneGap unterstützt dabei alle gängigen mobilen Betriebssysteme, wie Android, Windows Phone, iOS, Blackberry und weitere. Für den REST- bzw. HTTP-Request kann der *XMLHttpRequest*-Aufruf von JavaScript benutzt werden.

Die Entwicklung selbst erfolgt dabei in einem cordova-spezifischen *command line interface* (CLI), einem Kommandozeilentool. Auch können die Apps darüber mit Emulatoren der verschiedenen Plattformen getestet werden. Dabei müssen die jeweiligen SDKs

[10] https://build.phonegap.com/, aufgerufen: 30.09.2014

installiert sein, oder aber man greift auf das Cloud-basierte Build zurück. Bei Eclipse, Dreamweaver CS oder Xcode ist es auch möglich, PhoneGap direkt einzubinden und diese für die Entwicklung zu nutzen.

Die erstellten Apps können dann über die gewöhnlichen Stores der jeweiligen Plattform veröffentlicht werden.

Eine typische „Hello World“-Anwendung besteht dabei nur aus HTML-Code, welcher mit den eingebundenen Bibliotheken in einer Webview dargestellt wird (siehe Listing 3.1).

Listing 3.1: PhoneGap Hello World (nach [AS14b])

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <title>Cordova</title>
5 <script type="text/javascript" charset="utf-8" src="cordova-1.9.0.js"/>
6 </head>
7 <body>
8     <h1>Hello World</h1>
9 </body>
10 </html>
```

3.3.1.2 Rhomobile

Ein weiteres Framework für diese Kategorie ist das Open-Soure Framework Rhomobile, welches früher unter Rhodes bekannt war. Es gehört heute Motoral Solutions.

Gängige mobile Betriebssysteme wie Android, iOS, Blackberry und Windows Phone 8 werden unterstützt [MS14a, MS14c].

Die Applikationen werden dabei mit dem klassischen MVC-Pattern programmiert. Im Wesentlichen findet die Entwicklung mit Ruby statt. Dabei kann auf die Webtechnologien und die nativen Erweiterungen zugegriffen werden. Ähnlich wie PhoneGap erstellt auch Rhomobile einen nativen Container beim Kompilieren der Anwendung. Im Gegensatz zu PhoneGap stellt aber Rhomobile auch teilweise native UI-Elemente in diesem gerenderten Container als native Kontrollelemente zur Verfügung. Wobei die meisten UI-Elemente wie Buttons, Text, Grafiken, Listen, etc. immer noch webbasiert umgesetzt werden. Die Controller-Komponenten der App werden in Ruby-Bytecode kompiliert [MS14b, MS12].

3.3.2 Native Frameworks mit interpretiertem Code

Bei diesem Ansatz wird der generierte Quellcode, zusammen mit einem plattformspezifischen nativen Code, für die nativen Elemente in einem Interpreter zur Laufzeit ausgeführt.

3.3.2.1 Appcelerator Titanium

Eines der bekannteren Frameworks ist das von Appcelerator bereitgestellte Titanium-SDK, welches unter der Apache-Lizenz veröffentlicht ist. Es ist ein Open-Source-Framework, das mit einem einzigen JavaScript-Code eine App für die Betriebssysteme Android, iOS, Blackberry OS erzeugt [App14d]. Auch wird gerade an einer Unterstützung für Windows Phone 8 gearbeitet, welche zum Zeitpunkt dieser Arbeit allerdings noch nicht fertig entwickelt ist [Fel14].

Titanium unterstützt alle bekannten Webtechnologien, wie HTML5, CSS3 und JavaScript.

Während Titanium das kostenlose Open-Source Framework ist, bietet Appcelerator auch noch ein kostenpflichtiges Framework für Unternehmen an. Es ist eine offene, cloudbasierte Plattform, die den erstellten Quellcode optimiert, Möglichkeiten zum Testen bietet und Analysemethoden (Performance, Abstürze) bereitstellt. Natürlich kann die Anwendung für die jeweilige Plattform auch in der Cloud generiert werden, ohne die SDKs der jeweiligen Plattformen installieren zu müssen [App14a].

Zudem gibt es noch ein *Mobile Backend as a Service in the Cloud*, bei dem Unternehmen wie SAP, Oracle, Microsoft und Salesforce APIs zur Verfügung stellen, um über Appcelerator auf heterogene Datenquellen zugreifen zu können. Auch Zugriffe auf Dropbox, LinkedIn, Facebook, Twitter oder PayPal werden direkt unterstützt. Preise für die Nutzung des Dienstes werden dabei individuell festgelegt [App14b].

Im Vergleich zu PhoneGap (siehe Kapitel 3.3.1.1) bietet Appcelerator Titanium auch native User-Interfaces an. Wie man in Abbildung 3.3 sehen kann, wandelt Titanium den erstellten Quellcode, der ganz oben dargestellt ist, schrittweise in eine native Anwendung um. Der mit Webtechnologien erstellte Quellcode kann dabei um die UI-, die Hardware-API und weitere Module ergänzt werden. Übersetzt wird die Anwendung in der Titanium Bridge, der sogenannten *Kroll*. Kroll übersetzt dabei die Aufrufe vom Javascript-Quellcode in native Aufrufe. Das bedeutet aber nicht, dass daraus ein Java-

oder Objective-C-Quellcode entsteht. Dies hat in den letzten Jahren sehr oft zu einem Missverständnis geführt und wird auch in Büchern oft falsch beschrieben.

Bei der Umwandlung über die Bridge wird ein natives Element erzeugt und es entsteht dabei zugleich ein sogenannter Proxy für diese nativen Elemente. Ein Proxy ist dabei ein JavaScript-Objekt, das ein natives Element für die jeweilige Plattform referenziert. Aus diesem Grund benötigt man auch hierzu alle SDKs der Plattformen, auf der die Anwendung laufen soll. Wird dann das Proxy-Element verändert, so sorgt Kroll dafür, dass auch das native Element verändert wird. Ist der Proxy nicht ein UI-Element, wie z. B. das Dateisystem oder ein Datenbankzugriff, so wird der native Code direkt ausgeführt und das Ergebnis in JavaScript zurückgeliefert.

Mit dem konvertierten nativen Code des Titanium SDK für die jeweilige Plattform wird die eigentliche Anwendung dann erstellt und ausgeführt. Zur Laufzeit wird dieser kompilierte plattformspezifische native Code zusammen mit dem JavaScript-Quellcode ausgeführt. Der JavaScript-Code selbst wird dabei in einem JavaScript-Interpreter, V8 (für Android) und JavaScript Core (für iOS), ausgeführt. Implementiert man eine mobile Web-Applikation, so wird diese in der JavaScript-Engine des plattform-spezifischen Browsers ausgeführt [App13a, Whi12].

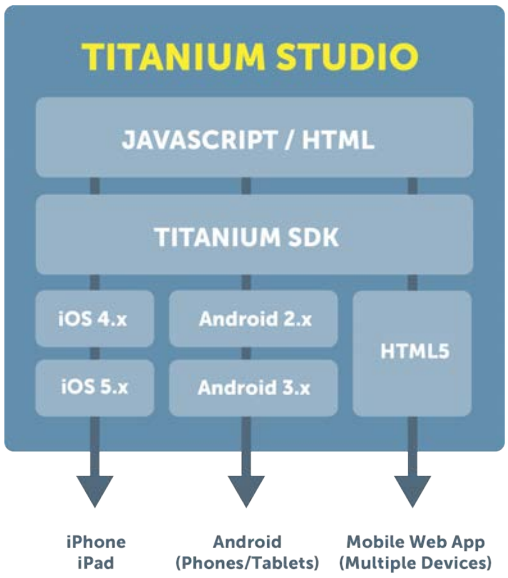


Abbildung 3.3: Appcelerator Titanium Funktionsweise (nach [App13a])

Titanium bietet ein eigenes SDK an. Die Entwicklung erfolgt dabei in dem eigenen Studio. Auch gibt es noch ein MVC-Framework, Alloy, das auch in Eclipse integriert ist, in dem man die Anwendungen erstellen kann [App14d].

Zusätzlich gibt es die Möglichkeit, auf vorgefertigte Cloud Services (z. B. ein Chat, Push Notifications, Nachrichten senden) bei der Entwicklung der Anwendung zurückzugreifen, allerdings ist dies im Vergleich zur Enterprise Edition auf eine gewisse Anzahl von Zugriffen pro Monat begrenzt [App14c].

Ferner gibt es noch einen eigenen Marketplace, auf dem bereits Module und Templates für Appcelerator den Entwicklern zur Verfügung gestellt werden, um so die eigene Anwendung erweitern oder Vorlagen für seine eigene Anwendung nutzen zu können[11].

Appcelerator Titanium bietet einen eigenen HTTP-Client an, (siehe [App14e]) und wird ständig weiterentwickelt, scheint also für die zu programmierende App geeignet zu sein.

Ein typisches Hello-World-Programm ist mit dem Alloy-Style sehr schnell erstellt, siehe folgendes Listing:

Listing 3.2: Appcelerator Titanium Hello World

```
1 <Alloy>
2   <Window class="container">
3       <Label id="helloworld">Hello World</Label>
4   </Window>
5 </Alloy>
```

Hier ist das oberste Element ein Window, in welchem sich verschiedene UI-Elemente befinden können, in diesem Fall ein Label.

3.3.2.2 Corona

Corona ist eine Erfindung der Firma Corona Labs Inc., die von Walter Luh 2008 gegründet wurde. Aktuell unterstützte Betriebssysteme sind iOS, Android, Kindle Fire und Nook. Corona hat sich dabei inzwischen einen Namen für Spielentwickler gemacht, da es auf typische Standards wie OpenGL, OpenAL, Box2D und mehr setzt.

Corona funktioniert dabei auf die ähnliche Weise wie Appcelerator Titanium (siehe Kapitel 3.3.2.1). Zunächst wird ein plattformspezifischer Code, welcher Zugriffe auf die nativen Elemente ermöglicht, mit Hilfe der Lua Engine, dem Build Tool von Corona, erstellt. Dann wird der Quellcode, der in Lua[12] geschrieben ist, auf der jeweiligen Plattform interpretiert. Dieser greift dann auf die nativen Elemente zu. Zusätzlich kann

[11] https://marketplace.appcelerator.com, aufgerufen am: 30.09.2014
[12] http://www.lua.org/about.html, aufgerufen am: 30.09.2014

auch direkt nativer Code implementiert werden. Die unterstützten nativen Sprachen hierfür sind Objective-C und C++ für iOS und Java für Android.

Schaut man sich mal aber die Dokumentation genauer an, so stellt man fest, dass es hauptsächlich um die Spielentwicklung geht, weniger um die Programmierung von Business-Apps. Für eine Business-App wäre deshalb ein Zugriff auf die nativen Bibliotheken von Android und iOS sinnvoll, was aber wiederum erhebliche Kosten mit sich bringen würde, da hierfür das kostenpflichtige Corona Enterprise Paket benötigt wird [CL14].

3.3.3 Native Frameworks mit Flash Player

Es wird eine Cross-Plattform-Technik betrachtet, die es ermöglicht, mit Hilfe des Adobe Flash Players mobile Applikationen auszuführen.

Am ehesten passt dieses Framework zu den nativen Frameworks mit einem nativem Container. Allerdings ist es eben auch mit Apache Flex möglich, Anwendungen zu schreiben, die mit Adobe Flash im Browser ausgeführt werden. Deswegen wird dieses Framework unter einer eigenen Kategorie aufgeführt.

3.3.3.1 Apache Flex

Apache Flex ist ein Open-Source-Framework, das 2004 von Macromedia gegründet wurde und seit 2006 unter dem Name Adobe Flex läuft [LTB10].

Unterstützte mobile Plattformen sind iOS, Android und Blackberry. Ebenso ermöglicht es auch noch Desktopanwendungen in Mac OS X und Windows. Ganz gewöhnlich können die erstellten Anwendungen in den jeweiligen plattformspezifischen App Stores deployt werden. Das Apache Flex SDK ermöglicht die Programmierung von RIA mit Adobe Flash.

Die Layoutkomponenten werden mit MXML, einer XML-basierten Sprache von Adobe, beschrieben. Die Anwendungslogik wird mit ActionScript programmiert. Übersetzt wird der Quellcode mit Hilfe der plattformunabhängigen Laufzeitumgebung Adobe AIR in eine native Applikation. Dabei stellt AIR eine integrierte Flash Player Umgebung bereit, in der die Anwendung schlussendlich ausgeführt wird. Dies ist insbesondere für die Ausführung in iOS gedacht, da iOS keinen Adobe Flash Player unterstützt. Alternativ ist die Umwandlung des Quellcodes in ein SWF-Format möglich, der dann direkt mit dem Adobe Flash Player aufgerufen wird [Fou14a].

Voraussetzung für die Entwicklung ist der Adobe Flash Builder[13], der jedoch nicht kostenlos verfügbar ist. Alternativ existieren mittlerweile aber auch Open-Source-Implementierungen wie FlashDevelop als IDE, mit der es möglich ist, die Anwendungen von Apache Flex zu programmieren[14]. Weitere alternativen Entwicklungsumgebungen sind der PowerFlasher FDT[15] oder IntelliJ IDEA von JetBrains[16]. Ebenso ist es möglich mit den Kommandozeilentools und einem beliebigen Texteditor die App zu entwickeln und zu testen [Fou14a].

Das Hello-World-Beispiel gleicht dabei einem XML-Dokument, wie man im folgenden Listing 3.3 sehen kann.

Listing 3.3: Apache Flex Hello World (nach [Fou14b])

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
3     xmlns:s="library://ns.adobe.com/flex/spark"
4     xmlns:mx="library://ns.adobe.com/flex/mx" width="640" height="480">
5     <s:layout>
6         <s:BasicLayout />
7     </s:layout>
8     <s:Label text="Hello World!" horizontalCenter="0" verticalCenter="0" /
          >
9 </s:Application>
```

Es werden dabei zuerst die benötigten Bibliotheken geladen. Anschließend wird die Layoutgröße und die Layoutart beschrieben, in diesem Fall ein Basic Layout (d. h. Elemente sind untereinander angeordnet). Dann wird das Label mit dem Text „Hello World“ erzeugt.

3.3.4 Native Frameworks mit Umwandlung in Binärcode

Bei dieser Technologie wird der jeweilige Quellcode in entsprechenden Bytecode der jeweiligen Zielplattform umgewandelt.

3.3.4.1 Xamarin

Xamarin, ein Projekt, das früher unter den Namen Mono bekannt war, ist ein natives Framework, welches mit C# programmiert wird. Früher war Mono ein selbständiges Tool, welches versucht hat, das .NET-Framework für nicht Windows-fähige Plattformen

[13] https://www.adobe.com/de/products/flash-builder.html, aufgerufen: 30.09.2014
[14] http://www.flashdevelop.org/, aufgerufen: 30.09.2014
[15] http://fdt.powerflasher.com/, aufgerufen: 30.09.2014
[16] https://www.jetbrains.com/idea/, aufgerufen: 30.09.2014

bereitzustellen. Seit der Version 2.0, die 2013 erschienen ist, hat Xamarin dies jedoch in ein Bundle gepackt und miteinander vereint [Fri13].

Umgewandelt wird der Quellcode eines Projekts je nach Zielplattform. Bei iOS wird dabei die programmierte Anwendung in ARM-Assemblercode umgewandelt. Bei Android wird für die implementierte Anwendung zuerst ein Zwischencode erzeugt, welcher dann zur Laufzeit anschließend in Assemblercode umgewandelt wird. Bei der Kompilierung entsteht entweder eine *app*- oder eine *apk*-Datei je nach Plattform. Diese können dann, wie gewohnt, deployt werden.

Die Runtime-Umgebung, welche für alle unterstützenden Plattformen erstellt wird, kümmert sich um Themen wie Memory Allocation, Garbage Collector etc. [Xam14b].

Als dritte Möglichkeit kann eine Windows Phone App erstellt werden. Da die Entwicklung einer Windows Phone App sowieso mit C# erfolgt und Xamarin das SDK von Windows Phone erfordert, stellt dies kein Problem dar. Kompiliert wird die Windows Phone App mit dem SDK als native Applikation ohne Verwendung von Xamarin. Jedoch kann der C# Quellcode mit Xamarin iOS, Android und Windows Phone Apps geteilt werden [Xam14b, Xam14a].

Auch verspricht Xamarin die Einbindung der Android und iOS SDKs als Standard-C# Bibliotheken, die zur Programmierung verwendet werden können. Der Shared Code wird für eine mobile Applikation unter Verwendung des .NET-Frameworks, das für alle Plattformen verwendet werden kann, geschrieben, wie in der Abbildung 3.4 zu sehen ist:

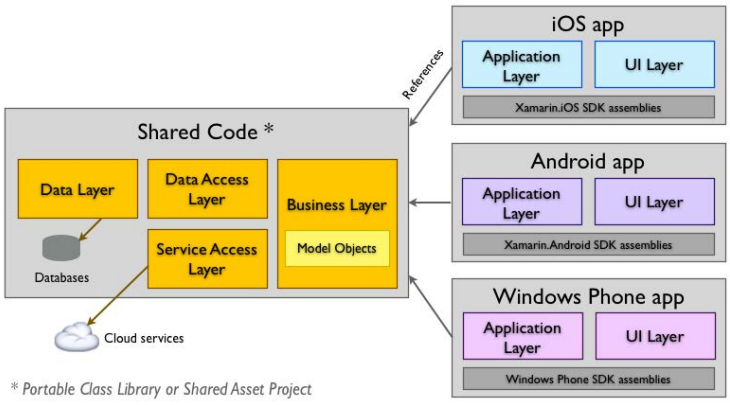


Abbildung 3.4: Xamarin Architektur (nach [Xam14c])

Die Entwicklung erfolgt entweder mit dem von Microsoft bekannten Visual Studio oder aber in einer eigenen bereitgestellten IDE namens Xamarin Studio.

Nachteilig bei der kostenlosen Version ist, dass der kompilierte Code nur 64 KB groß sein darf, diese Restriktion ist bei der kostenpflichtigen Version nicht vorhanden [Xam14a, Xam14d]. Ebenso nachteilig ist die Verwendung von plattform-spezifischen UI-Elementen, weswegen man auch meinen könnte, dass Xamarin kein Cross-Plattform-Framework ist (was diese ja von sich behaupten), da die UI-Elemente einer mobilen Anwendung einen sehr großen Teil der Programmierung ausmachen. Lediglich die Logik kann plattformübergreifend implementiert werden (siehe Abbildung 3.4) [Xam13].

3.3.5 Native Frameworks mit nativem Quellcode

Bei diesem Ansatz wird eine Programmiersprache gewählt, die es ermöglicht, auf allen Betriebssystemen einen ausführbaren Bytecode zu erzeugen.

3.3.5.1 MoSync

MoSync ist eine Open-Source Framework, das von dem gleichnamigen schwedischen Unternehmen 2004 gegründet wurde, um plattformunabhängige, mobile Applikationen zu schreiben [MoS13e].

Die Lizenzierung des Frameworks hängt von der Nutzung des MoSync SDK ab. Für den privaten Gebrauch und für die kostenlose Veröffentlichung der Anwendungen ist MoSync unter der Open-Source GPL2 lizenziert. Für den kommerziellen Gebrauch benötigt man eine kommerzielle Lizenz von MoSync. Das MoSync SDK wurde jedoch im GitHub-Verzeichnis[17] mit einer Apache 2.0 Lizenz re-lizensiert [MoS13c].

Benutzt werden dabei die nativen Programmiersprachen C und C++. Mit diesen beiden Sprachen lassen sich sowohl Anwendungen für Android, Windows Phone, Blackberry, Windows Mobile, Mobilin, Symbian als auch iOS-basierte Geräte entwickeln. Der Vorteil bei der Programmierung mit C/C++ ist, dass die Verwendung dieser Sprachen eine sehr hohe Performance bietet. Es muss lediglich der Quellcode in den plattformspezifischen Bytecode umgewandelt werden (siehe dunkelgrüne Pfeile in Abbildung 3.5). Dabei wird der GCC-Compiler genutzt und ein *Pipe-Tool*, das die MoSync-Libraries und -Ressourcen einbindet und den selbergeschrieben Code optimiert [MoS13d].

Alternativ kann eine Mischung aus C/C++, HTML5 und JavaScript zur Programmierung mit MoSync genutzt werden, welche dann im Build-Prozess in nativen Code umgewandelt wird (hellgrüne Pfeile in Abbildung 3.5).

[17] https://github.com/MoSync/MoSync, aufgerufen: 30.09.2014

MoSync Reload ist die dritte Programmiervariante des Frameworks, das ebenfalls wie PhoneGap mit den Webtechnologien HTML 5 und JavaScript programmiert wird (hellgrüne Pfeile in Abbildung 3.5).

In der folgenden Abbildung 3.5 wird der Build Prozess einer mobilen Applikation mit MoSync verdeutlicht:

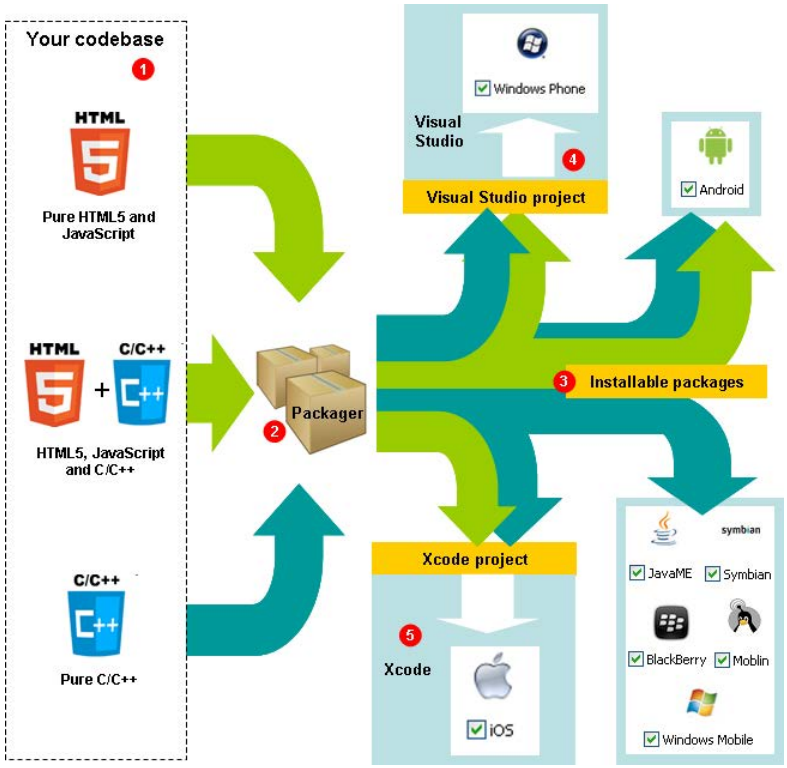


Abbildung 3.5: Build Prozess von MoSync (nach [MoS13d])

Je nach Sprache des Quellcodes und der gewünschten Zielplattform wird beim Build Prozess über den Packager entweder ein Visual-Studio- oder Xcode-Projekt oder auch direkt ein installierbares Paket für Android, Blackberry, Symbian und Windows Mobile erzeugt. Sollte auf dem zu entwickelnden Rechner Xcode und/oder Visual Studio installiert sein, so kann auch hierfür direkt der fertige Bytecode erstellt werden und muss nicht über den Aufruf der jeweiligen plattformspezifischen IDE generiert werden [MoS13d].

Das MoSync SDK enthält eine eigene Eclipse-basierte IDE, in der man die Anwendungen programmieren, kompilieren, testen und deployen kann [MoS13d]. Seit dem neusten Update des SDK ist es sogar möglich, mit geringem Aufwand Anwendungen von PhoneGap in MoSync Anwendungen zu transferieren [MoS13f].

Auch werden von MoSync die DELETE-, POST-, PUT- und GET-Methoden über ein HTTP-Request ermöglicht [MoS13b].

Ein typisches Hello World Beispiel ist im Listing 3.4 zu sehen:

Listing 3.4: MoSync Hello World (nach [MoS13a])

```
1 #include <MAUtil/Moblet.h>
2 using namespace MAUtil;
3 class MyMoblet : public Moblet{
4     public:
5         MyMoblet(){
6             maSetColor(0xFFFFFF),
7                 maDrawText(0, 32, "Hello World!");
8             maUpdateScreen();
9         }
10        void keyPressEvent(int keyCode, int nativeCode){
11            if(keyCode == MAK_0 || keyCode == MAK_BACK || keyCode ==
                  MAK_SOFTRIGHT){
12                close();
13            }
14        }
15 };
16 extern "C" int MAMain(){
17     MyMoblet myMoblet;
18     Moblet::run( &myMoblet );
19     return 0;
20 }
```

Dabei werden zuerst die nötigen Bibliotheken geladen und dann wird eine Klasse MyMoblet erstellt, die von der Klasse Moblet erbt. Diese erstellt einen DrawText (Zeile 7), welcher dargestellt wird. Ebenso wird das Verhalten von Hardwaretasten über Events gesteuert. Schließlich wird in der main-Funktion diese Klasse instantiiert und ausgeführt. Für nähere Details siehe Erläuterungen in [MoS13a].

3.4 Web Frameworks für mobile Endgeräte

Als zweite Gruppe gibt es die Web Frameworks, die versuchen, mit Hilfe von etablierten Webtechnologien wie HTML5, JavaScript und CSS3 einen nativen Look & Feel erzeugen. Häufiger findet man sie auch unter dem Namen *Mobile HTML5 Framework* wieder.

Im Folgenden werden die zwei bekanntesten Frameworks dieser Kategorie vorgestellt.

3.4.1 jQuery Mobile

Das wohl bekannteste Web Framework für mobile Endgeräte ist jQuery Mobile, das unter einer MIT Lizens veröffentlicht ist [Fou14c]. jQuery Mobile stellt dabei UI-Elemente zur Verfügung, die für alle Plattformen, Bildschirmgrößen und Geräte optimiert sind, d. h. man programmiert nur einmal ein UI-Element wie ein Button, eine Fußzeile, Auswahlbox oder Slider, etc. und es wird dann gerätespezifisch dargestellt. Auch die Animation der Übergänge wird von jQuery Mobile übernommen [FI13].

Unterstützte Betriebssysteme sind dabei iOS, Android, Windows Phone, Blackberry, Symbian, bada, MeeGo, palm webOS sowie alle gängigen Browser.

Listing 3.5: jQuery Mobile Beispiel (in Anlehnung an [Fou12b])

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.2/
           jquery.mobile-1.4.2.min.css">
5      <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
6      <script src="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1
           .4.2.min.js"></script>
7    </head>
8    <body>
9      <div data-role="page" id="mainsite">
10     <div data-role="header">
11       <h1>My first jQuery Webpage</h1>
12     </div>
13     <div data-role="main" class="ui-content">
14      <p>This is my first jQuery Mobile Webpage<p>
15     </div>
16   </body>
17 </html>
```

In dem Listing 3.5 wird ersichtlich, dass beim Programmieren die einzelnen jQuery-Mobile-Bibliotheken mit eingebunden werden, das ist zum einen die CSS-Library (Zeile 4), jQuery selbst (Zeile 5) und die spezifischen jQuery-Mobile-Funktionen (Zeile 6).

Die einzelnen Seiten werden in einem div-Container erstellt, der den Wert page als data-role-Attribut erhält, siehe Zeile 9 ff.

Das Framework kann in Verbindung mit anderen Frameworks wie PhoneGap (siehe Kapitel 3.3.1.1) und Worklight (siehe Anhang A.3) verwendet werden [Fou12a, Dhu12].

Insbesondere in der UI- Programmierung der jeweiligen App bietet es eine sinnvolle Ergänzung zu den nativen Frameworks.

Weitere Web Frameworks, die ebenfalls den Ansatz von jQuery Mobile umsetzen, sind Intel's App Framework[18] und iUI[19].

3.4.2 Sencha Touch

Ein weiteres sehr bekanntes Web-Framework ist Sencha Touch, das 2010 aus den Bibliotheken Ext JS, Raphaël und jQTouch entstanden ist [Eli10].

Es kombiniert HTML5 mit CSS3- und JavaScript-Bibliotheken, die die UI-Elemente darstellen und eine unterschiedliche Darstellung für die verfügbaren Plattformen (Android, iOS, Blackberry OS) und Browser (Android Browser, Google Chrome, BlackBerry 10, Bada Mobile Browser, Kindle Fire Browser, Windows Phone 8 und Windows 8 IE10 und Mobile Safari) bieten.

Sencha Touch ist als Open-Source Framework unter der GNU GPL Lizenz Version 3 veröffentlicht. Es gibt aber auch alternativ eine kommerzielle Lizenz von Sencha Touch selbst, für den Fall, dass man seine eigene App veröffentlichen will, ohne den Quellcode offenzulegen [Sen14d]. Des Weiteren gibt es von Sencha Touch ein Bundle, welches mehrere Komponenten vereint, wie den Sencha Architect, der es erlaubt, per Drag & Drop die UI-Elemente zu erstellen, einem Sencha Touch Eclipse-Plugin als IDE, fertige Charts und Grid-Elemente und einem Support für all diese Produkte. Das Bundle ist allerdings nicht kostenlos erhältlich [Sen14c].

Listing 3.6: Sencha Touch Hello World Beispiel (nach [deH12])

```
1 new Ext.application({
2     launch: function() {
3         Ext.create("Ext.Panel", {
4             fullscreen: true,
5             html: "Hello World"
6         });
7     }
8 });
```

[18] http://app-framework-software.intel.com/, aufgerufen am: 01.10.2014
[19] http://www.iui-js.org/, aufgerufen am: 01.10.2014

In einem HTML-File wird dann auf dieses JavaScript in Listing 3.6 verwiesen. Im Gegensatz zu jQuery Mobile schreibt man nur das Grundgerüst mit den Bibliotheken in HTML, den Rest der Applikation in JavaScript. Die Programmierung folgt dabei dem bekannten MVC-Pattern. Zusätzlich zu diesem Pattern existieren sogenannte Store-Objekte, welche dafür zuständig sind, Daten in die App zu laden, und Profile-Objekte, die verschiedene Geräte für die App unterschiedlich behandeln sollen [Sen14a].

Eine Sencha Touch App besteht immer aus dem application-Objekt (siehe Zeile 1, Listing 3.6), das beschreibt, was beim Start passiert. Im Beispiel wird ein neuer leerer Container erstellt (Zeile 3), der dann HTML-Markup beinhaltet (Zeile 5). Man erkennt anhand des Codes, dass Sencha Touch sich sehr stark an der Objektorientierung orientiert [FI13].

Seit der Version 2.3 des Sencha Touch Frameworks ist es auch möglich, über PhoneGap (siehe Kapitel 3.3.1.1) auf die nativen Elemente eines mobilen Endgerätes zuzugreifen [Sen14b].

Einen ähnlichen Ansatz zu Sencha Touch nutzen auch die Frameworks Lungo[20], PhoneJS[21], Jo[22],Chocolate Chip UI[23], Enyo[24], qooxdoo[25], SproutCore[26], DHTMLX Touch[27] und The M-Project[28].

[20] http://lungo.tapquo.com/, aufgerufen am: 01.10.2014
[21] phonejs.devexpress.com, aufgerufen am: 01.10.2014
[22] http://joapp.com/, aufgerufen am: 01.10.2014
[23] http://chocolatechip-ui.com/, aufgerufen am: 01.10.2014
[24] http://enyojs.com/, aufgerufen am:01.10.2014
[25] http://qooxdoo.org/, aufgerufen am: 01.10.2014
[26] http://sproutcore.com/, aufgerufen am: 01.10.2014
[27] http://dhtmlx.com/touch/, aufgerufen am: 01.10.2014
[28] http://www.the-m-project.org/, aufgerufen am: 01.10.2014

3.5 Übersicht der Cross-Plattform-Entwicklung

Die folgende Abbildung 3.6 verdeutlich nochmals die wesentlichen Unterschiede der Funktionsweise der verschiedenen Cross-Plattformen:

Abbildung 3.6: Ausführungscontainer der Applikationen der verschiedenen Frameworks (nach [SSP+13])

Wie in der Abbildung 3.6 zu sehen ist, gibt es *Web- Applikation*, welche der Entwicklung mit einem der mobilen Web Frameworks wie jQuery Mobile oder Sencha Touch entsprechen (siehe Kapitel 3.4). Diese werden über einen Browser ausgeführt. Sie sind aber dabei auf die Funktionalität des Browsers beschränkt.

Die hier sogenannten *hybriden Applikationen - Web* bezeichnen die typischen Cross-Development-Frameworks wie PhoneGap oder Rhomobile (siehe Kapitel 3.3.1). Webtechnologien werden hier in einem nativen Container ausgeführt.

Hybride Applikationen - Mixed sind implementierte Applikationen mit Frameworks wie Appcelerator und Corona (siehe Kapitel 3.3.2). Zusätzlich zu einem nativen Container werden auch native UI-Elemente unterstützt. Man kann also auf alle nativen Elemente des mobilen Endgerätes zugreifen. Programmiert wird die App mit Webtechnologien, welche zur Laufzeit interpretiert und native Elemente über eine Schnittstellen-API aufrufen.

Die vierte Entwicklungsmöglichkeit sind die *nativen Applikationen*. Dazu gehören neben den mit den plattformspezifischen SDKs erstellten Applikationen native Frameworks wie MoSync (siehe Kapitel 3.3.5.1), das hochperformanten Code in C++ kompiliert, oder Xamarin (siehe Kapitel 3.3.4.1), das plattformspezifischen Bytecode erstellt.

Bevor man nun die mobile Applikation mit einer dieser Entwicklungsmöglichkeiten entwickelt, müssen die Anforderungen an die zu entwickelnde Applikation definiert werden. Diese werden im folgenden Kapitel vorgestellt.

4

Anforderungsanalyse

Generell gilt, dass, bevor Software (und anderes) entwickelt wird, man sich Gedanken über die Anforderungen, die diese zu erfüllen hat, machen sollte. Im sogenannten Requirements Engineering unterscheidet man hierbei zwischen funktionalen und nicht-funktionalen bzw. qualitativen Anforderungen. Diese werde im Bezug auf die zu entwickelnde Anwendung erläutert.

4.1 Funktionale Anforderungen

Die funktionalen Anforderungen legen fest, was das System tun soll, also welche Features oder Funktionen benötigt werden. Hierzu gehören auch die erforderlichen Daten und das Verhalten des Systems bzw. der Software. Funktionale Anforderungen geben jedoch grundsätzlich nicht die Art und Weise der Umsetzung vor. Typische Beispiele für solche Anforderungen sind nach [Par10]:

- Eingabe und Ausgabe
- bereitgestellte Dienste/Funktionen
- reaktives Verhalten

Für die Tourismus-Applikation wurden folgende funktionalen Anforderungen herausgearbeitet.

4.1.1 Nachrichten

Name	Server/App	Beschreibung
Kategorien	App	Es sollen die einzelnen Kategorien für die Nachrichten über die Gemeinde angezeigt werden.
Nachrichten	App	In den jeweiligen Kategorien sollen die einzelnen Nachrichten absteigend (beginnend mit dem zeitlich neuesten Eintrag) angezeigt werden. Auch soll es möglich sein, Details der einzelnen Nachrichten einzusehen. Diese Details enthalten auch dazugehörige Bilder und Dateien.
Nachrichten löschen/erstellen	Server	Nachrichten werden vom Administrator erstellt und auch gelöscht.

4.1.2 Veranstaltungen

Name	Server/App	Beschreibung
Kategorien	App	Es sollen die einzelnen Kategorien für die Veranstaltungen über die Gemeinde angezeigt werden.
Veranstaltungen	App	In den jeweiligen Kategorien sollen die einzelnen Veranstaltungen absteigend (beginnend mit dem zeitlich neusten Eintrag) angezeigt werden. Standardmäßig sollen diese für die nächsten 14 Tage angezeigt werden, es soll aber auch möglich sein, das Start- und Ende-Datum für dieses Intervall zu ändern. Durch das Klicken auf eine Veranstaltung sollen die Details einsehbar sein. Diese Details enthalten auch die dazugehörigen Bilder und Dateien.
Veranstaltungen löschen/erstellen	Server	Veranstaltungen werden vom Administrator erstellt und gelöscht.

4.1.3 Live-Reports

Name	Server/App	Beschreibung
Registrierung	App	Ein Benutzer kann sich mit seiner E-Mailadresse als Loginnamen registrieren. Benötigt wird der Vorname, Nachname sowie ein Passwort mit mindestens 4 Zeichen. Gespeichert wird dieses verschlüsselt mit SHA1 in der Datenbank. Dabei soll dem erfolgreich registrierten Benutzer sowie dem Administrator eine E-Mail zur Registrierung geschickt werden.
Login	App	Der Login erfolgt über die E-Mailadresse und das vom Bürger gewählte Passwort. Zusätzlich soll überprüft werden, für welche Kommune der Benutzer registriert ist. Ein Nutzer kann sich für die Live-Reports nur einmal anmelden (er kann den Hauptwohnsitz nur in einer Kommune haben). Ebenso soll es möglich sein, den Login auf dem Endgerät zu speichern, d. h. solange man sich nicht abmeldet, gelangt man automatisch in die Übersicht der Live-Reports.
Profil ändern	Server	Profil des Anwenders soll nur durch einen Administrator geändert werden können.
Anwender löschen	Server	Anwender sollen nur durch einen Administrator gelöscht werden können.
Übersicht	App	In der Übersicht soll man die Möglichkeit haben, sich abzumelden, eine neue Bürgermeldung zu erstellen und sich offene Bürgermeldungen anzusehen. Die Darstellung erfolgt in Tabellenform, sowie auf einer Karte.
Bürgermeldung erstellen	App	Es soll bei der Erstellung einer neuen Bürgermeldung möglich sein, ein Bild mit der Kamera des Endgerätes zu machen oder ein bestehendes Bild aus der Galerie hochzuladen. Ebenso soll der aktuelle Standort bestimmt werden können.

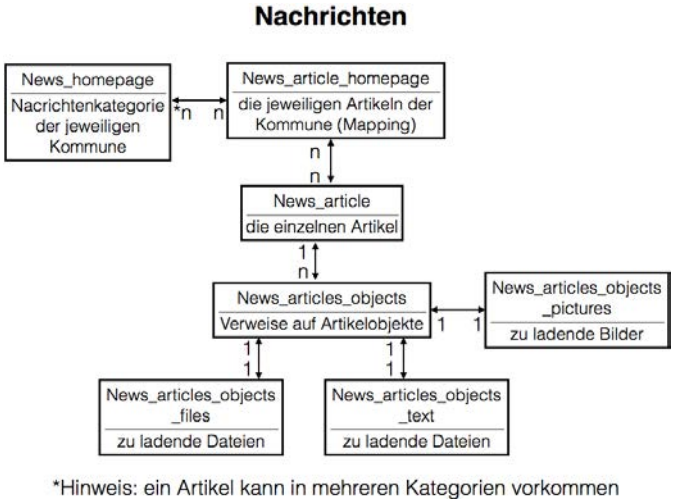
		Der Nutzer kann weiter den Titel, Ort und eine Beschreibung der Bürgermeldung hinzufügen. Dem zuständigen Ansprechpartner der Gemeinde soll eine E-Mail über die erstellte Bürgermeldung zugehen.
Bürgermeldung anzeigen	App	Es sollen in einer Tabelle die aktuellen, offenen Bürgermeldungen aufgelistet sein. Durch Klicken soll man die Möglichkeit haben, die Details der einzelnen Bürgermeldung zu erhalten. Der Nutzer soll zudem die Möglichkeit haben, eine Bürgermeldung zu kommentieren.
Status einer Meldung ändern	Server	Der Status einer Bürgermeldung soll von einem Mitarbeiter der Kommune geändert werden können. Dabei gibt es die drei Status: neu, in Bearbeitung, geschlossen.
Bürgermeldung löschen/ändern	Server	Bürgermeldungen sollen nur durch Administrator gelöscht oder abgeändert werden können.

4.1.4 Karte

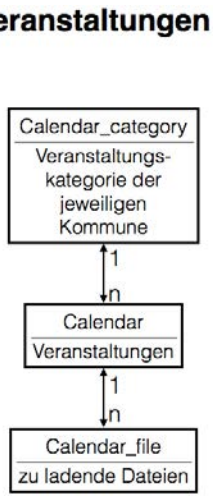
Name	**Server/App**	**Beschreibung**
Kategorien	App	Es sollen die einzelnen Kategorien für die Karte der jeweiligen Kommune angezeigt werden.
Kategorien hinzufügen/ändern	Server	Ausschließlich dem jeweiligen Administrator ist es vorbehalten, Kategorien hinzuzufügen und zu ändern.
POIs anzeigen	App	Die Karte soll die Points of Interest der jeweiligen Kategorie anzeigen. Die zuletzt ausgewählten Einträge sollen beim nächsten Aufruf wieder angezeigt werden.
POIs ändern/löschen	Server	Die Änderung und Löschung von Einträgen soll nur durch den Administrator erfolgen.
Routen anzeigen	App	Die Karte soll neben den Kategorien auch Routen/Wanderwege für die jeweilige Gemeinde anzeigen.
Routen ändern/löschen	Server	Die Änderung und Löschung von Einträgen soll nur durch den Administrator erfolgen.

4.1.5 Daten

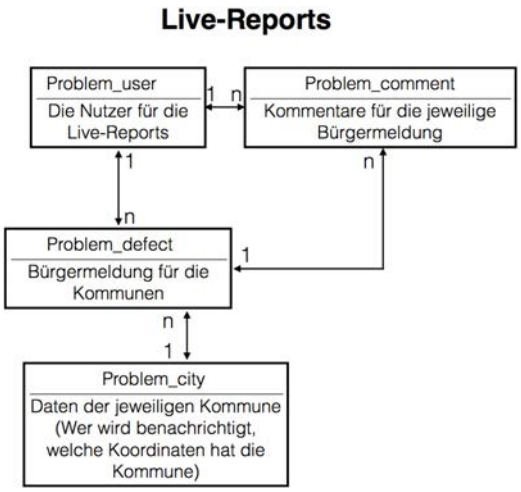
Die Daten selbst sind in einer MySQL-Datenbank, also in einem relationalen Schema hinterlegt, welche von einem Dienstleistungsunternehmen für Kommunen gepflegt wird. Das Datenschema selbst entspricht keiner der bekannten Normalformen (mit Ausnahme der Map-/Karten-Tabellen, die alle in 3. NF sind). Da jedoch die zu entwickelnde Anwendung abwärtskompatibel sein soll, muss die API für die mobilen Endgeräte auf dieser Datenbank aufgebaut werden. Für das Testen der Applikation steht eine Testdatenbank zur Verfügung. Der Aufbau der vorgegeben Datenbank folgt dabei grob dem Schema der folgenden Abbildung 4.1:



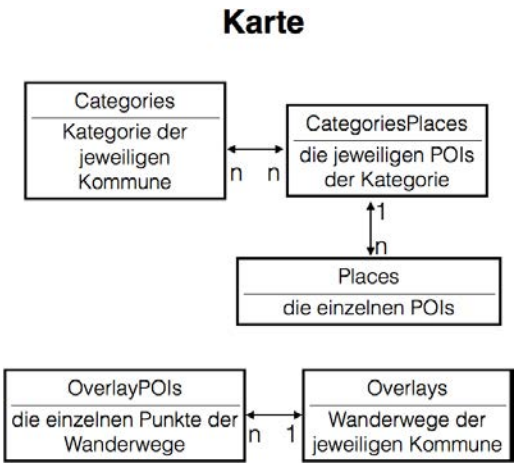
(a) Nachrichten



(b) Veranstaltungen



(c) Bürgermeldungen



(d) Karte

Abbildung 4.1: grobes Datenbankschema

4.2 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben hingegeben die qualitativen Aspekte des Systems bzw. der Software. Sie beschreiben also, wie etwas gemacht werden soll, mit welcher Qualität, mit welcher Verfügbarkeit, mit welcher Performance, etc. [Par10].

Der ISO-Standard 9126, der die Qualitätsmerkmale einer Software beschreibt und gerne für die nicht-funktionalen Anforderungen herangezogen wird, wurde inzwischen durch den ISO-Standard 25010:2011 abgelöst. Der neue Standard wurde dabei um weitere Qualitätsmerkmale erweitert. Die aktuellen Eigenschaften sind nach [ISO11]:

- Änderbarkeit (Maintainability)
- Funktionalität (Functional suitability)
- Zuverlässigkeit (Reliability)
- Benutzbarkeit (Usability)
- Effizienz (Performance efficiency)
- Übertragbarkeit (Portability)
- Sicherheit (Security)
- Kompatibilität (Compatibilty)

Hinsichtlich der zu implementierenden mobilen Tourismus-Applikation gibt es folgende nicht-funktionale Anforderungen.

Änderbarkeit Ein gutes Softwaredesign ist in einzelne Module aufgeteilt, um einfacher verändert, erweitert und gewartet werden zu können. Dies soll auch hier in der mobilen Applikation umgesetzt werden. Ebenso sollte in der mobilen Applikation eine Trennung von Logik, Inhalt und Layout genutzt werden, sofern dies möglich ist. Dies ist auch in dem zu erstellenden Framework möglich. Appcelerator Titanium implementiert eine mobile Applikation in einem sogenannten MVC-Pattern, siehe auch Kapitel 5.5.2.

Funktionalität Es erscheint sehr verwirrend, wenn in den nicht-funktionalen Anforderungen die Funktionalität zu beschreiben ist. Gemeint ist aber hier nur, ob die funktionalen Anforderungen den erforderlichen Anforderungen der mobilen Applikation entsprechen.

Dabei gibt es zwei wesentliche Aspekte zu berücksichtigen.

- Zum einen ist in der mobilen Applikation darauf zu achten, dass ein Benutzer nur mit den nötigen Funktionen konfrontiert wird; er soll auch nur die dafür erforderlichen Daten sehen.
- Ebenso ist der Einsatz von externen Diensten zur Erfüllung der funktionalen Anforderungen so gering wie möglich zu halten.

Zuverlässigkeit Eine Software bzw. eine mobile Applikation sollte unter bestimmten Bedingungen in einer bestimmten Zeit bestimmte Funktionen ausführen. Dies betrifft hier vor allem die externen Dienste der mobilen Applikationen. Diese sollten nicht abstürzen und auch sonst sehr zuverlässig funktionieren. Aber auch das Abfangen von falschen Eingaben muss hierbei berücksichtigt werden. Insbesondere bei mobilen Applikationen ist es sehr wichtig, diese abzufangen und entsprechende Fehlermeldungen für den Anwender auszugeben.

Benutzbarkeit Für die Programmierung einer App sollten bereits bekannte Navigations- und Design-Konzepte verwendet werden. Hierbei sollen die unterschiedlichen Eigenschaften von mobilen Geräten (Bildschirmauflösung, Orientierung) berücksichtigt werden.

Die Applikation soll dem Benutzer auch Fehler anzeigen, insbesonders, da die Applikation eine Internetverbindung, bei der möglicherweise schnell Verbindungsfehler etc. auftreten, nutzt.

Effizienz Die Effizienz bzw. die Performance bezeichnet die Aspekte Antwortzeitverhalten, Rechenaufwand und Datendurchsatz. Die Anwendung sollte schnell auf Benutzereingaben reagieren und rechenaufwendige Operationen sollten vermieden werden. Dies gilt insbesondere für mobile Applikationen, da die Rechenleistung der CPU beschränkt ist, und eine hohe Leistung auch den Akku schnell aufbraucht.

Unnötige Datentransfers sind ebenso zu vermeiden und der Datendurchsatz ist so gering wie möglich zu halten. Das (begrenzte) Datenvolumen der mobilen Funkübertragung soll nicht unnötig aufgebraucht werden.

Deshalb sollten Caching-Mechanismen benutzt werden, die diesem Problem entgegenwirken.

Übertragbarkeit Die Übertragbarkeit kann zum einen so betrachtet werden, dass das System, bzw. die mobile App auf mehreren Plattformen laufen kann, was ja gerade bei der Cross-Plattform-Entwicklung gegeben ist.

Zum anderen wird berücksichtigt, wie gut der Quellcode in eine andere Entwicklungsumgebung übertragen werden kann.

Sicherheit Da diese Applikation auch sicherheitskritische Informationen speichert, muss sichergestellt werden, dass von außen kein Zugriff möglich ist. Kritischen Benutzerdaten müssen verschlüsselt abgespeichert und übertragen werden. Auf dem Applikationsserver selbst, der die Daten enthält, darf nur ein autorisierter Zugriff erfolgen.

Kompatibilität Hier geht es um den Datenaustausch zwischen Systemen und Komponenten.

Bei der mobilen Applikation ist darauf zu achten, dass die Daten in einem kompatiblen Format abgespeichert und ausgetauscht werden. Als Standard dient hierbei das JSON-Format. Dieses kann sehr leicht über einen HTTP-Request übertragen werden. Des Weiteren kann es auch immer einheitlich geparst und erstellt (stringify) werden.

Bei den nicht-funktionalen Anforderungen werden gesondert noch die Anwendungsinformationen sowie die technische Anforderungen betrachtet.

4.2.1 Anwendungsinformationen

Name	Beschreibung
Sprache	Die Sprache der Applikation ist Deutsch.
Design	Das Logo und die Icons der Anwendung werden von dem Dienstleistungsunternehmen für Kommunen bereitgestellt. Das Aussehen der eigentlichen Anwendung soll auf Android und iOS gleich sein.

4.2.2 Technische Anforderungen

Name	Beschreibung
Plattform	Die Anwendung soll sowohl auf iOS-, als auch auf Android-Geräten ausgeführt werden können.
Gerätekategorie	Die Anwendung soll jeweils für Smartphones und Tablets geeignet sein.

5

Eine Tourismus-Anwendung mit dem Framework Appcelerator Titanium

Da es eine große Anzahl von in zu Betracht ziehenden Frameworks für die plattformunabhängige Entwicklung der Tourismus-Anwendung gibt, fiel die Wahl für diese Arbeit auf das Framework Appcelerator Titanium, siehe Kapitel 3.3.2.1.

Wie bereits in Kapitel 3 beschrieben, gibt es keine eindeutige Begründung für die Auswahl dieser Cross-Plattform. Dies muss nach Anwendungsfall entschieden werden. Es gab jedoch ein paar Anhaltspunkte für diese Entscheidung.

Zum einen muss diese mobile Applikation natürlich auf native Hardware, wie Kamera, GPS etc. zugreifen können. Des Weiteren soll es auch möglich sein, die Applikation für unterschiedliche mobile Endgeräte unterschiedlich zu programmieren, d. h. die grafischen Elemente sollen sich in den Anwendungsplattformen unterscheiden (siehe Kapitel auch 4). Deshalb fiel die Wahl auf ein natives Framework. Eine zusätzliche Intention dieser Arbeit ist auch, eine alternative Entwicklungsmöglichkeit zum weit verbreiteten Framework PhoneGap/Cordova (Kapitel 3.3.1.1) aufzuzeigen.

5.1 REST

Die Representational-State-Transfer-Architektur, kurz REST-Architektur, ist ein von Roy Fielding in seiner Dissertation [Fie00] geprägtes Konzept für die einheitliche Übertragung

von Ressourcen für Web Services. Das Konzept ist dabei an kein bestimmtes Protokoll gebunden. Folgende Eigenschaften/Restriktionen werden dabei an eine solche REST-Architektur gestellt:

Das grundlegende Prinzip von REST ist die Übermittlung von Ressourcen. Ressourcen sind abstrakte Objekte, die einen bestimmten Typ, Daten und Beziehungen zu anderen Ressourcen aufweisen (z. B. ein Bild, Dokument, Kunde, eine Person). Jede Ressource liegt einer eindeutigen URI zugrunde. Wie diese Ressource repräsentiert wird, ob als XML-, XHTML und/oder JSON ist ebenso wenig festgelegt, wie, die Übertragung zwischen Server und Client selbst stattfinden soll. Die Ressource kann also in mehreren Varianten repräsentiert werden und je nach Anforderungsspezifikation in verschiedenen Datenformaten übertragen werden.

Die Kommunikation bei REST ist zustandslos, d. h. es existieren keine Sessions oder Cookies auf dem Server. Die Ressource wird aber bei jedem Zugriff neu angefordert. Sollte jedoch eine Cache client-seitig verwendet werden, so sollen die erhaltenen Daten als cacheable oder non-cacheable gekennzeichnet sein.

Jede Repräsentation einer Ressource sollte über übliche Methoden wie z. B. die von dem HTTP-Protokoll gewohnten *GET-, PUT-, POST-, DELETE*-Requests zugreifbar sein. Hierzu ist eine einheitliche abstrakte Schnittstelle erforderlich, die wiederum selbst Anforderungen mit sich bringt. Diese Schnittstelle erfordert die Identifikation der verschiedenen Ressourcenformate sowie die Manipulation dieser Ressourcen, also der erhaltenen Daten. Ebenso soll die Schnittstelle das von Fielding genannte „Hypermedia as the Engine of Application State “ implementieren, d. h. der Client kommuniziert mit dem Web Service nur über Hypermedia[29], die der Server bereitstellt. Er muss also nicht wissen, wie er mit dem Web Service auf Protokollebene kommunizieren soll, wie z. B. bei SOAP, sondern wie er mit Hypermedia umgeht. Dabei navigiert er mit den eindeutigen URIs die REST-Schnittstelle an.

Dazu gibt es die Restriktion, dass das Client-Server Konzept durch ein „Layered System“ , also durch eine Schichtenarchitektur getrennt ist. So wird nur die aktuell verbundene Komponente erkannt und nicht die darüber hinaus. Dies hat den Vorteil, dass man Funktionen einfach implementieren, austauschen und Services auf verschiedenen Server verteilen kann.

[29] Hypermedia bezeichnet einen Hypertext, also Information in textueller Art, der um Grafiken, Audio und Video erweitert wird [Bol98].

Die letzte Beschränkungen der REST-Architektur ist optional, das sogenannte *Code on Demand*. Damit wird Code auf dem Client in Form von Applets oder Skripte vom Server heruntergeladen und ausgeführt.

In den meisten Fällen wird das HTTP-Protokoll für die REST-Schnittstelle verwendet, da dieses ein bereits vertrautes und bewährtes Konzept des WWW ist [Hel13, Fie00].

Für die serverseitige Entwicklung der REST-API mit PHP wird in dieser Arbeit das Framework Slim verwendet, da der Einarbeitungsaufwand gering und das Framework nicht überladen ist.

5.2 Slim Framework

Slim ist ein PHP-micro-Framework, welches die Implementierungen Web-Services und APIs ermöglicht, das 2013 von dem Entwickler Josh Lockhart als Open-Source-Version – unter einer MIT Lizenz veröffentlicht – entstanden ist. Es ist sehr schlank gehalten und enthält nur die wichtigsten Funktionen für ein REST-Framework. Das ist wiederum der Vorteil von Slim, denn so ist es ein sehr leichtgewichtiges und schnelles Framework.

Slim verarbeitet typischerweise einen HTTP-Request, welcher an einen Controller weitergeroutet wird. Dieser erzeugt als Antwort ein HTTP-Response, meist in Form eines Views. Natürlich enthält Slim auch weitere Funktionen wie Caching, Logging, Statusänderungen und Weiterleitungen für den HTTP-Statuscode [Sch14, Loc14].

Listing 5.1: Slim REST Hello World Beispiel (nach [Loc12])

```
1  <?php
2  //load slim libraries
3  require 'vendor/autoload.php';
4  //create a new slim object
5  $app = new \Slim\Slim(array('debug' => true));
6  //define uri with function
7  $app->get('/hello/:name', function($name) {
8      echo "Hello, $name";
9  });
10 //start slim app
11 $app->run();
12 ?>
```

Wie man im Listing 5.1 sehen kann, werden zuerst automatisch die benötigten Klassen geladen, die für die Slim-Applikation benötigt werden. Dies übernimmt die Datei *autload.php* (Zeile 2).

Eine Anwendung wird als neues Objekt mit der Eigenschaft debug=true instantiiert (Zeile 3). Das Debugging liefert extra Fehlerinformationen während der Entwicklung. In der Produktionsphase sollte der Debugging-Modus ausgeschaltet werden.

Diese minimale Anwendung definiert einen *GET*-Request (Zeile 5). Es wird zuerst der Pfad angegeben, wie die URI aufgerufen werden kann. Dabei gibt es einen sogenannten *named Parameter* in der URI (gekennzeichnet durch Voranstellen eines Doppelpunktes), welcher ein Argument (hier ein Name) in der URI entgegennimmt. Als zweites Argument erhält der *GET*-Request eine Funktion. In diesem einfachen Beispiel wird einfach ein *Hello, $name* ausgegeben.

Will man nun die angelegte Ressource anfordern, so kann man dies durch einen Aufruf – vorausgesetzt localhost ist der aktuelle Server – von der URL *localhost/hello/Daniel* im Browser machen. Das Ergebnis ist die Ausgabe: *Hello, Daniel*.

5.3 BoxResizer

Eines der größeren Probleme bei der Programmierung von mobilen Applikationen ist das Aufrufen von großen Bildern. Diese werden oft originalgetreu in voller Qualität vom Server geladen und erst nachträglich auf dem Client, also dem mobilen Endgerät, skaliert und angepasst. Dies verursacht einen unnötig großen Datenverkehr zwischen Server und Client. Skalierte Bilder müssen für mobile Endgeräte jedoch nicht so hochauflösend sein. So geht Bandbreite verloren.

In der zu erstellenden Applikation wird deswegen ein anderer Ansatz verfolgt. Hierbei ist der Service von BoxResizer sehr nützlich. Die URL des Imagefile wird der REST–Schnittstelle übergeben, welche dann BoxResizer aufruft und ein skaliertes Bild zurückschickt.

BoxResizer agiert dabei als Proxy, der die URI des Bildes entgegennimmt und entsprechend skaliert, dreht, in ein anderes Format umwandelt oder die Qualität des Bildes ändert. Es schickt dann das Bild bearbeitet wieder zurück.

Der kostenlose Service von BoxResizer basiert auf der Amazon Elastic Compute Cloud, kurz EC2. Amazon EC2 stellt dabei Rechenkapazität in der Cloud zur Verfügung. Diese garantiert dabei ein Service-Level-Agreement von 99,95% Verfügbarkeit [PRS11].

Am folgenden Beispiel sieht man, wie der BoxResizer-Service anzuwenden ist:

Listing 5.2: BoxResizer Beispiel

```
1 $source="http://www.uni-ulm.de/in/dbis/example.png";
2 //definiere Optionen
3 $options="resize=100x100";
4 $proxyurl='http://proxy.boxresizer.com/convert?'.$options.'&source=';
5 $proxyurl.=$source;

7 $image=@imagecreatefrompng($proxyurl);
8 imagepng($image);
```

In Listing 5.2 wird zuerst die URI für das Bild definiert. Dann versieht man seine *proxyurl* mit Optionen, wie das Bild bearbeitet werden soll. In diesem Fall wird einfach die Skalierung auf 100 × 100 Pixel (Zeile 3) gesetzt. Anschließend fügt man die URI zusammen und erzeugt das Bild (Zeile 7-8).

5.4 Serverseitige Entwicklung, Architektur

Für den REST-Service gilt es zunächst einmal, sich passende URIs auszudenken – bzw. genauer gesagt, sich deren Struktur zu überlegen.

Im folgenden wird dargestellt, wie diese REST-API für die Tourismus-Applikation aufgebaut ist. Die Grundstruktur folgt dabei dem Schema „http://<yourserver.de>/api/“ .

5.4.1 Nachrichten

Wie in den Anforderungen (siehe auch Kapitel 4) bereits ersichtlich wurde, soll es möglich sein, zuerst die Kategorien für die Nachrichten der jeweiligen Gemeinde oder Stadt zu erhalten. Hierzu wird die URI in Zeile 3 des Listings 5.3 definiert. Sollte ein Wort durch einen Doppelpunkt vorangestellt sein, so bedeutet dies in Slim, dass es sich hierbei um eine Variable handelt, die hier eingesetzt werden muss, in diesem Fall also die communityId. Eine gültige URL lautet somit: „http://<server>/api/4711/news/“ . Dieser HTTP-GET-Request liefert die Kategorien der Nachrichten für die Gemeinde mit der ID 4711 zurück.

Möchte man nun die dazugehörigen Nachrichten für die jeweilige Kategorie erhalten, so folgt man diesem Schema und ruft diese URI durch Ergänzung der Kategorie auf.

Will man sich die Details (also den Inhalt) einer einzelnen Nachricht anzeigen lassen, dann folgt man der Struktur, wie in Zeile 8 des Listings 5.3.

Des Weiteren wurden noch GET-Aufrufe definiert, die es ermöglichen, Nachrichten von einem bestimmten Datum (Zeile 11), Nachrichten der letzten zwei Wochen (Zeile 13) oder eine gewisse Anzahl der neuesten Nachrichten (Zeile 15) zu erhalten.

Listing 5.3: News URIs

```
1  <?php
2  //get categories for a communityid
3  $app->get('/api/:communityid/news/','getNewsCategories') ;

5  //get all news for a communityid for a specific category
6  $app->get('/api/:communityid/news/:category/','getAllNews') ;
7  //get News Details
8  $app->get('/api/news/articles/:articleid','getNewsDetails') ;

10 //find news for a communityid for a specific date
11 $app->get('/api/:communityid/news/date/?:query','findbyDate');
12 //get news of the last 14 days
13 $app->get('/api/:communityid/news/:category/latest/','getLatestNews') ;
14 //get the last x(count) entries of the news
15 $app->get('/api/:communityid/news/:category/latest/:count','
       getAmountLatestNews') ;
16 ?>
```

5.4.2 Veranstaltungen

Die Aufrufe der Veranstaltungen sind den Aufrufen der Nachrichten sehr ähnlich.

Hier beginnt man auch mit dem Aufruf, der die Kategorien für die Veranstaltungen zurückliefert, siehe Zeile 3 Listing 5.4.

Möchte man nun alle Veranstaltungen für eine Kategorie zwischen einem bestimmten Start- und Enddatum, so folgt man dem Schema in Zeile 6 des Listings 5.4. Hierbei gibt es eine Besonderheit. Da es Veranstaltungen geben kann, die bis zum Enddatum gehen können, aber bereits vor dem Startdatum beginnen, muss man dies in der Implementierung berücksichtigen und diese auch in das Ergebnis mit einbeziehen. Ebenso kann es Veranstaltungen geben, die bereits in dieser Zeit zwischen Start- und Enddatum beginnen, jedoch länger andauern. Diese müssen ebenfalls im Ergebnis auftauchen. Für das Start- und Enddatum kann man auch das Attribut *today* benutzen, welches dann auf den jeweiligen Tag umgerechnet wird.

Beschränkt man sich bei dem Zeitraum auf die von heute ausgehenden 14 Tage , so kann man dies durch den Aufruf mit /latest anstelle von Start- und Enddatum erhalten, siehe Zeile 11, Listing 5.4.

Will man dieses Ergebnis nun noch ein bisschen mehr einschränken, so kann man eine Anzahl von neuesten Einträgen für die Veranstaltungen der zukünftigen 14 Tage durch die URI in Zeile 13 auswählen, also z. B. die neuesten 10 Einträge der zukünftigen 2 Wochen der Gemeinde mit der ID 4711 und der Veranstaltungskategorie 1 durch „http://server/api/4711/events/1/latest/10/“ .

Soll das Ergebnis nicht nur eine Kategorie, sondern alle Veranstaltungen für die Kommune in einem bestimmten Zeitraum enthalten, so gelingt dies durch den Aufruf in Zeile 9, Listing 5.4.

Die Details einer Veranstaltung erhält man hier durch die URI /api/events/event/:eventid, siehe Zeile 15, Listing 5.4

Listing 5.4: Events URIs

```
1  <?php
2  //get event categories
3  $app->get('/api/:communityid/events/','getEventCategories') ;

5  // get all event category entries between two dates
6  $app->get('/api/:communityid/events/:categoryid/:start/:end','
       getCategoryEventsBetween') ;

8  // get all events between a start and a end date for a communityid
9  $app->get('/api/:communityid/events/:start/:end','getEventEntriesBetween')
       ;
10 //get the newest events of the future 14 days
11 $app->get('/api/:communityid/events/:categoryid/latest/','getLatestEvents'
       ) ;
12 // get amount(x) of newest events
13 $app->get('/api/:communityid/events/:categoryid/latest/:count/','
       getAmountLatestEvents') ;
14 // get details for an event
15 $app->get('/api/events/event/:eventid/','getEventDetails') ;
16 ?>
```

5.4.3 Live-Reports

Wie aus den Anforderungen ersichtlich, gibt es einige Funktionen, die für die Live-Reports umzusetzen sind.

Bevor man die Bürgermeldungen sieht oder welche erstellen kann, muss man sich zuerst einloggen. Dabei wird zusätzlich zu dem Benutzernamen und Passwort auch überprüft, ob sich dieser Nutzer nicht bereits für eine andere Gemeinde registriert hat.

Sollte noch kein Login vorhanden sein, so kann man sich auch über den POST-Request (in Zeile 20 des Listings 5.5) registrieren.
Möchte man sich wieder von den Bürgermeldungen abmelden, so gibt es hier eine eigene URI (siehe Zeile 22).

Wenn man sich nun angemeldet hat, möchte man alle erstellten Bürgermeldungen für eine Gemeinde erhalten. Dies erfolgt durch die URI /api/:communityID/reports/ (Zeile 3 Listing 5.5). Die Details zu einer spezifischen Bürgermeldung erhält man über die URI in Zeile 5.

Eine neue Bürgermeldung kann man mit der URI des POST-Requests /api/:communityID/reports/create erzeugen. Hierbei muss der Titel, der Ersteller, der Ort, die Beschreibung und optional ein Bildname übertragen werden. Das Bild einer zu erzeugenden Bürgermeldung muss jedoch über eine eigene URI hochgeladen werden (Zeile 16 Listing 5.5), da von dem ausgewählten Framework Titanium das Hochladen eines Multidata-HTTP-Requests nicht funktioniert. Mehr dazu wird bei der Umsetzung in Kapitel 5.5.6 erläutert.

Bei den Bürgermeldungen soll es möglich sein, dass Bürger Kommentare schreiben können (siehe Anforderungen Kapitel 4.1.3). Die bisherigen Kommentare für eine Bürgermeldung, die man über die URI in Zeile 7 des Listings 5.5 bekommt, werden absteigend nach dem Datum (beginnend mit dem Neuesten) sortiert ausgegeben. Mit einem POST-Request kann man einen neuen Kommentar erstellen, indem man die entsprechende URI aufruft und dabei einmal den Nutzer und den Kommentar als Text als POST-Parameter übergibt.

Listing 5.5: Reports URIs

```
1 <?php
2 //get reports for community
3 $app->get('/api/:communityid/reports/','getLiveReports');
4 //get report details
5 $app->get('/api/reports/:reportid','getReportDetails');
```

```
6  //get comments for report
7  $app->get('/api/reports/:reportid/comments/','getCommentsForReport');
8  //create a new comment
9  $app->post('/api/reports/:reportid/comment','createComment');
10 //get comment details
11 $app->get('/api/comments/:commentid','getCommentDetails');

13 //create a new report
14 $app->post('/api/:communityid/reports/create','createReport');
15 //upload picture for report
16 $app->post('/api/:communityid/reports/uploadImage','uploadReportImage');
17 // login
18 $app->post('/api/:communityid/reports/login','makeLogin');
19 // register new user
20 $app->post('/api/:communityid/reports/register','registerUser');
21 // logout
22 $app->post('/api/:communityid/reports/logout','makeLogout');
23 ?>
```

5.4.4 Karte

Ein weiteres Modul ist die Karte. Auch hier gibt es verschiedene URIs, die im folgenden erläutert werden.

Zunächst wurde die Möglichkeit implementiert, alle Sehenswürdigkeiten (Points of Interest) über die URI in Zeile 3 des Listings 5.6 abzurufen. Man erhält hierbei alle POIs der jeweiligen Gemeinde. Mit der URI in Zeile 5 erhält man nun die Details eines einzelnen Punktes. Neben den POIs sollen auch noch, wie aus den Anforderungen aus Kapitel 4.1.4 hervorgeht, Wanderwege/Routen einer Gemeinde dargestellt werden. Dies wird über die URI in Zeile 7 ermöglicht. Es werden alle Wanderwege und Routen ausgegeben, die für die Gemeinde angelegt sind.

Möchte man erst einmal die jeweiligen Kategorien, die es für die Gemeinde gibt, abrufen, so kann man diese über die URI in Zeile 11 bekommen. Wenn man eine bestimmte Kategorie hat, versucht man mit der URI in Zeile 13 die einzelnen Points of Interest für diese Kategorie abzufragen. Auch kann man alle Wegpunkte einer Wanderstrecke/Route über die URI in Zeile 15 erhalten.

All diese Funktionen liefern das Ergebnis jedoch nicht als JSON-Objekt, sondern als gzip-komprimierten String zurück, da dies sonst erheblichen Datenverkehr verursachen würde.

Für den Live-Report gibt es noch eine URI, die hier bei der Karte auftaucht. Diese passt eher zu den Karten, weswegen sie auch hier in Zeile 9 des Listings 4.1.4 zu finden ist. Damit kann die Position der jeweiligen Gemeinde bestimmt werden. Dies ist vor allem dann interessant, wenn für eine Bürgermeldung keine Koordinaten vorhanden sind, denn dann können wenigstens die Koordinaten der Gemeinde angezeigt werden (siehe auch Kapitel 5.5.6).

Listing 5.6: Maps URIs

```
1 <?php
2 //get all POI for a communityid
3 $app->get('/api/:communityid/maps/POI','getAllPOI');
4 //get Details for a POI
5 $app->get('/api/:communityid/maps/POI/:poi','getPOIDetail');
6 //get all routes for a communityid
7 $app->get('/api/:communityid/maps/routes','getAllRoutes');
8 //get community position
9 $app->get('/api/:communityid/maps/getactualPosition','getCommunityPosition
      ');
10 //get maps categories for a communityid
11 $app->get('/api/:communityid/maps','getMapsCategories') ;
12 //get all poi for a category for a communityid
13 $app->get('/api/:communityid/maps/:category','getAllCategoryPOI');
14 //get all POI along a routeid
15 $app->get('/api/:communityid/maps/routes/:route','getRoutePOI');
16 ?>
```

5.4.5 Bilder und Dateien

Da für alle verschiedenen Kategorien immer wieder Bilder und Dateien geladen werden, wurden eigene URIs für das Laden von diesen implementiert, wie man im folgenden Listing 5.7 sieht.

Listing 5.7: Utils URIs

```
1 <?php
2 /*
3  * FOR ALL APIS: GET PICTURES
4  */
5 //get Image in the specified size
6 $app->get('/api/:module/img/:url/:size','getResizedImage');
7 //get Image in a default size
8 $app->get('/api/:module/img/:url','getDefaultImageSize');
9 /*
```

```
10   * FOR ALL APIS: GET FILES
11   */
12 $app->get('/api/:module/file/:url','getFile');
13 ?>
```

Es gibt die Möglichkeit, ein Bild einheitlich zu laden. Wie im Kapitel 5.3 erläutert, wird dabei der Proxy BoxResizer genutzt. Dabei gibt es zwei Möglichkeiten, wie man ein Bild bekommt.

Zum einen übergibt man den Bildnamen in Zeile 6 des Listings 5.7 mit Angabe der Größe. Also es wird z. B. die URL *„http://server.uni-ulm.de/api/events/image.png/100x100“* aufgerufen, was das Bild in dem Format 100×100 Pixel vom Server für das Modul Veranstaltungen zurückliefert. Auch ist es möglich das Format *tablet* für Tablet Geräte und das Format *handheld* für Smartphones zu verwenden, welches intern auf eine jeweilige Standardgröße umgewandelt wird.

Die zweite Möglichkeit (Listing 5.7, Z.8) bietet der Aufruf ohne Format Angabe. Dieser liefert das Bild in einem Standardformat der Größe 200×200 zurück.

Neben Bildern werden auch Dateien von allen Modulen genutzt. Deswegen wurde das Herunterladen von diesen auch ausgelagert. Durch einen Aufruf von bspw. *„http://server.uni-ulm.de/api/news/file/file.pdf“* wird die Datei file.pdf für das Modul Nachrichten geladen.

5.5 Clientseitige Entwicklung

Aufbauend auf die Entwicklung des Servers kann nun die Entwicklung des Clients, also der mobilen Tourismus-Applikation, erfolgen, die anschließend auch beschrieben wird.

5.5.1 Die Entwicklung mit Appcelerator Titanium

Das Kernstück von Appcelerator ist das Titanium SDK. Es enthält neben der Möglichkeit zur Implementierung von nativen, hybriden und mobilen Web-Apps auch eine eigene Entwicklungsumgebung, die als Eclipse-Plugin unter dem Namen Titanium Studio konzipiert wurde.

Ein Projekt ist dabei immer folgendermaßen aufgebaut:

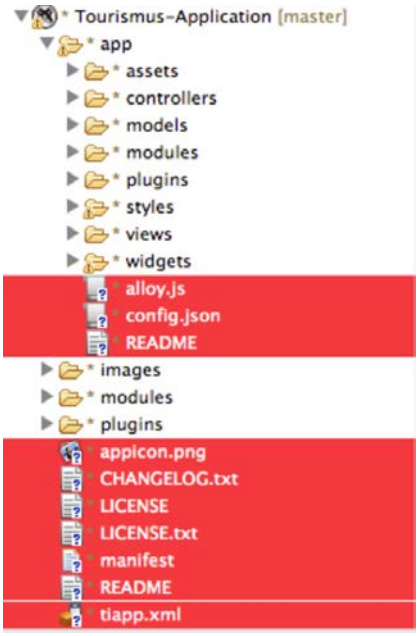


Abbildung 5.1: Titanium Projektstruktur

Jedes Projekt enthält die Datei tiapp.xml. Diese beschreibt die Konfiguration der Applikation, z. B. welche Plugins erforderlich sind, welche globalen Eigenschaften diese Applikation unterstützen sollte, bspw. welche Bildschirmausrichtung für welches Endgerät unterstützt wird oder welche Zugriffsrechte diese Applikation in Android erhalten soll (analog zum Android Manifest bei der Entwicklung mit dem Google SDK) und welche verschiedenen Plattformen unterstützt werden sollen.

In der Programmierung einer Applikation ohne Verwendung von Alloy (siehe folgendes Kapitel 5.5.2), gibt es einen Ressourcen-Ordner, in welchem die App geschrieben wird. Dort wird dann eine Datei namens *app.js* angelegt, welche als erstes bei dem Start der Anwendung aufgerufen wird. Bei Verwendung von Alloy, so wie man es auch hier in der Abbildung 5.1 sehen kann, wird in dem Ordner app jeweils eine View, ein Controller und ein Style definiert. Die View beschreibt dabei die Elemente für einen Controller. In dem Controller wird dabei das Verhalten dieser Views definiert und der Style beschreibt das Aussehen der Views.

Ebenso können Module, Plugins und Widgets integriert werden, welche sich in dem entsprechenden Ordner wiederfinden. Diese müssen dann jedoch in der tiapp.xml oder in der config.json Datei des Projekts verknüpft werden.

Dabei beginnt in Alloy die Applikation immer mit dem Controller *index.js*. [Cop13]

5.5.2 Das Alloy-Prinzip von Appcelerator Titanium

Ein großer Vorteil bei der Implementierung ist die Nutzung von Alloy. Alternativ gibt es auch hierzu die Möglichkeit, die mobile Applikation nur in Javascript zu schreiben und von dort auf die UI-Elemente und das Aussehen einzugehen. Seit der Version 3.0 von Appcelerator Titanium gibt es allerdings dieses MVC-Pattern, welches eine erhebliche Vereinfachung der Programmierung erlaubt.

Alloy ist ein MVC-Pattern, das es erlaubt, Logik, Daten und das Aussehen zu trennen. Somit kann für das Layout auf unterschiedliche Plattformen eingegangen und das

Layout nach Größe des Endgerätes und nach dem plattformspezifischen Betriebssystem verändert werden.

Das Aussehen wird in einer eigenen Stylesheet-Sprache definiert, der sogenannten *Titanium Stylesheet*-Sprache. Sie gleicht dabei der CSS-Sprache.

Die Views selbst werden mit XML beschrieben. Dabei können die gleichen UI-Elemente benutzt werden, die man auch in der klassischen Programmierung von Titanium verwenden kann.

Schließlich wird die Logik, wie sich die Views verhalten und welche Daten sie enthalten sollen, in einem Controller niedergeschrieben. Hierbei kann der komplette Umfang von dem normalen Titanium Javascript verwendet werden.

5.5.3 Menü

Beim Start der Applikation sollte man einen Startbildschirm, also eine sogenannte Landing-Page anzeigen. Da es 15 Kategorien/Module gibt, wurde dieser als 3×5 Grid-Layout implementiert. Dreht sich der Bildschirm, wird dieses horizontal auf 5×3 Zellen transponiert. Ausgehend von diesem Startbildschirm, welcher in der Abbildung 5.2 zu sehen ist, wird dann durch ein Klick auf das entsprechende Bild das jeweilige Modul aufgerufen.

(a) Menu Portrait

(b) Menü Landscape

Abbildung 5.2: Hauptmenü der Tourismus-Applikation

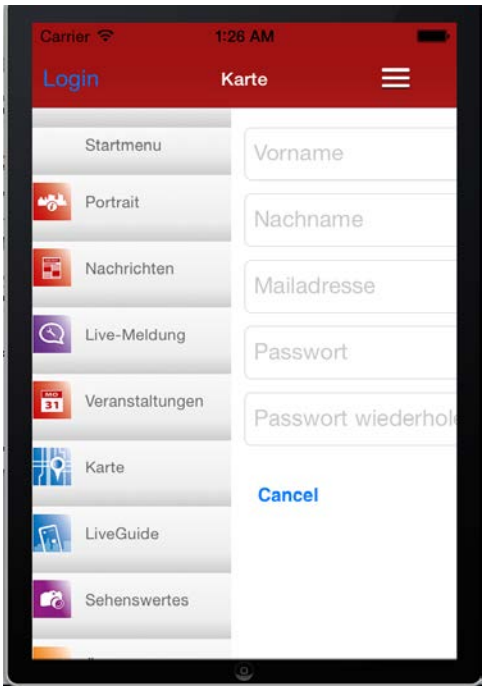


Abbildung 5.3: Slidemenü

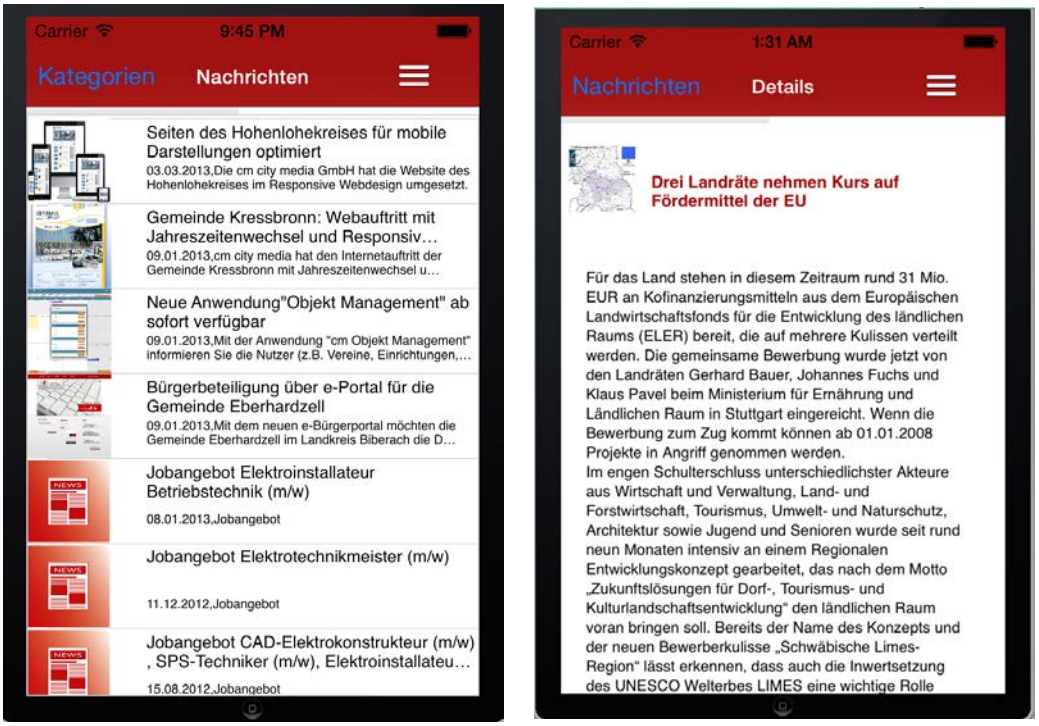
Damit man jedoch nicht jedes Mal zu diesem Startmenü zurückkehren muss, wenn man zwischen den Modulen wechseln möchte, wurde ein Slidemenü implementiert. Dieses kann durch Klicken auf den Menü-Button rechts oben (bei Android funktioniert dies durch Klicken auf die Action Bar) oder durch Ziehen von links nach rechts im Bildschirm. Bei Android wurde dabei ein etwas anderes Slidemenü implementiert[30], wie bei iOS, was daran liegt, dass die Applikation die unterschiedlichen Hardwareelemente berücksichtigt. Der Hardware Zurück-Button von Android ist bei iOS nicht vorhanden. Dort muss mit einem sogenannten Navigationsbutton navigiert werden. Bei dem Slidemenü öffnet sich auf der linken Seite des Bildes das Menü, in dem man nun die einzelnen Module auswählen kann.

5.5.4 Nachrichten

Befindet man sich in dem Modul Nachrichten, so erscheint eine Übersicht der Nachrichtenkategorien. Durch Auswahl eines Eintrages gelangt man dann zur Nachrichtenübersicht, wie man auch in Abbildung 5.4 (a) sehen kann. Dabei wird auch ein Teaserbild, sofern eines vorhanden ist, geladen (siehe 5.3), sowie der Nachrichtentitel und ein Teasertext, sofern einer verfügbar ist. Die Nachrichten sind absteigend, beginnend mit dem neusten Eintrag, sortiert.

In den Nachrichtendetails werden dann zusätzlich noch weitere Bilder, sowie Texte und Dateien für die jeweilige Nachricht geladen. Auch sind die Bilder in dieser Detailansicht klickbar, um sie in einer vergrößerten Ansicht zu sehen.

[30] https://github.com/viezel/NappDrawer/tree/master/android, aufgerufen am: 26.09.2014



(a) Nachrichten

(b) Nachrichtendetails

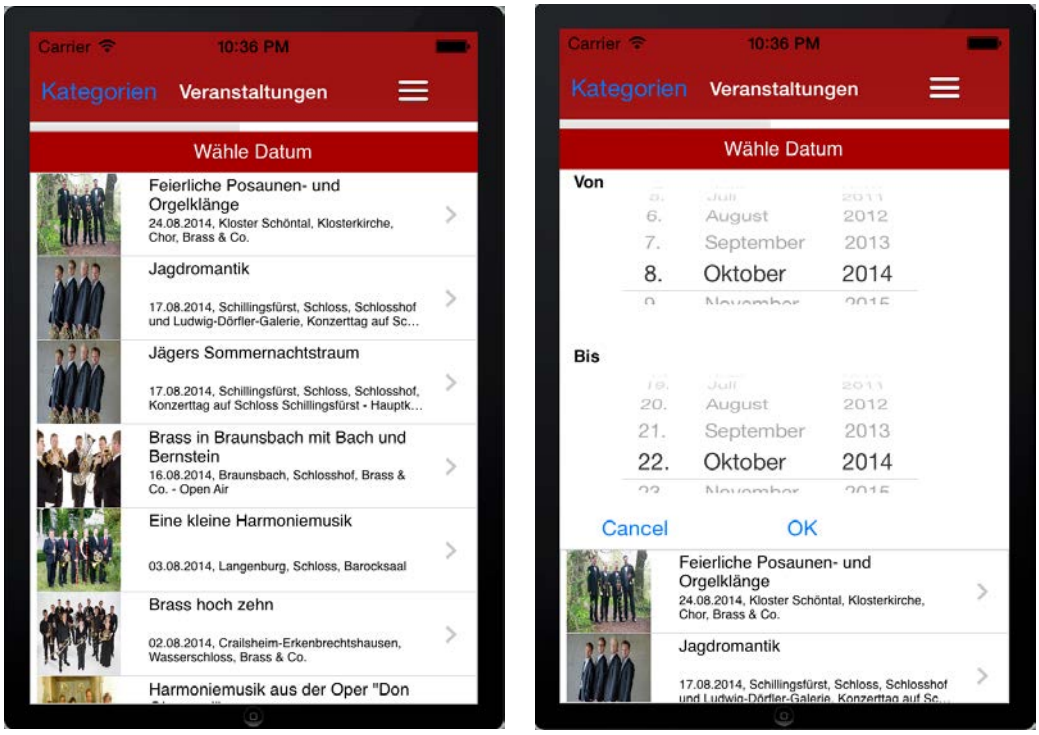
Abbildung 5.4: Nachrichten

5.5.5 Veranstaltungen

Die Veranstaltungen folgen in der Umsetzung dem gleichen Prinzip wie bei den Nachrichten. Auch hier gibt es Veranstaltungskategorien, in welche sich die jeweiligen Veranstaltungen teilen lassen.

Eine Besonderheit gibt es jedoch für die Veranstaltungsübersicht (Abbildung 5.5 (a)), wie man in den Abbildung 5.5 (b) sehen kann. Wie man den Anforderungen in Kapitel 4.1.2 entnehmen kann, werden normalerweise die Veranstaltungen der nächsten 14 Tagen angezeigt. Da man aber eventuell einen anderen Zeitraum auswählen möchte, wurde hier ein Datepicker implementiert. Dieser erlaubt es, das Von- und Bis-Datum der anzuzeigenden Veranstaltungen zu ändern.

Klickt man wiederum auf eine Veranstaltung in der Tabelle der Veranstaltungsübersicht, so gelangt man auch hier zu den Veranstaltungsdetails, in welcher auch weitere Bilder und Dateien für diese Veranstaltung geladen werden.



(a) Veranstaltungen (b) Wähle anderes Datum

Abbildung 5.5: Veranstaltungen

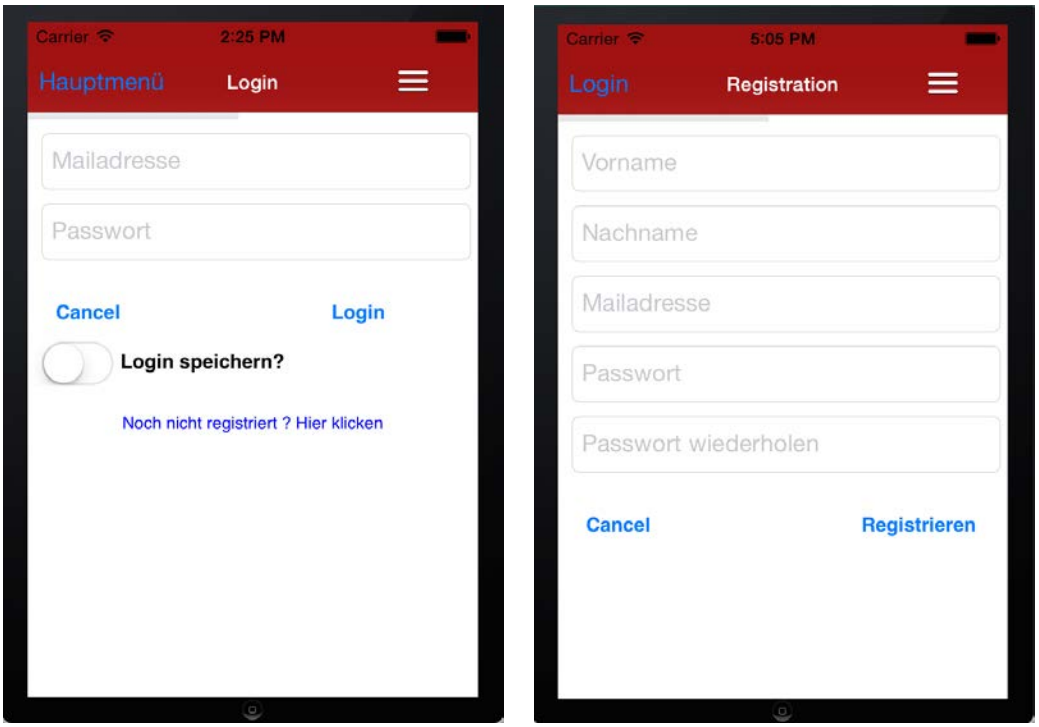
,

5.5.6 Live-Reports

Ein weiteres Modul sind die Bürgermeldungen, auch Live-Reports genannt. Die Intention hinter diesem Modul ist, dass Bürger der Kommune mitteilen können, wenn öffentliches Gut betroffen ist, bspw. ein Mülleimer der Stadt nicht geleert wurde, Straßenlaternen defekt oder die Jalousien der Stadthalle kaputt sind.

Doch bevor man diese Meldungen sehen kann, oder neue erstellen kann, muss der Bürger eingeloggt sein. Dazu erscheint der Startbildschirm wie in Abbildung 5.6 (a) zu sehen ist. Der Radiobutton fragt, ob der Login für die Dauer der Applikation gespeichert werden soll. Sollte dieser Wert auf Ja gesetzt sein, so werden die Logindaten verschlüsselt in den Anwendungsdaten der Applikation abgespeichert. Nur wenn man sich dann explizit von den Live-Meldungen abmeldet, werden diese Daten wieder gelöscht. Sollte man noch nicht für die Live-Reports registriert sein, so kann man dies durch den Klick auf den entsprechenden Text auf der Loginseite tun. Dabei gelangt man zu einer neuen Seite, in welcher man die benötigen Felder Vorname, Nachname, Benutzername und Passwort eingibt. Dabei wird natürlich auch auf die Anforderung eingegangen, dass ein

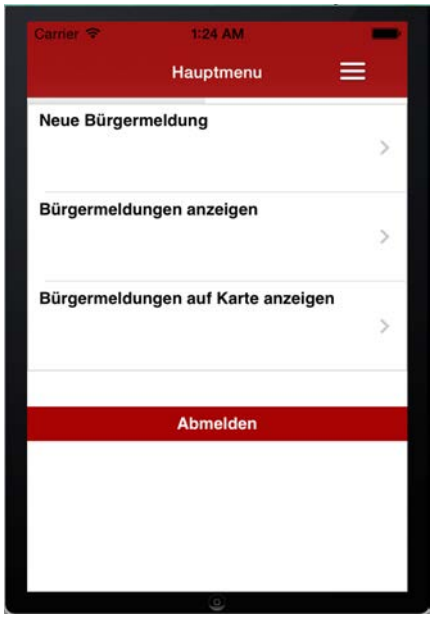
Passwort mindestens 4 Zeichen lang sein soll. Das Passwort muss ebenfalls bestätigt werden, damit weitere Fehler vermieden werden. Bei einer erfolgreichen Registrierung wird eine E-Mail an den Benutzer selbst, sowie an den jeweiligen Betreuer der Gemeinde geschickt. Nach der Registrierung wird man wieder zurück zur Loginseite geleitet, wo man sich nun anmelden kann.



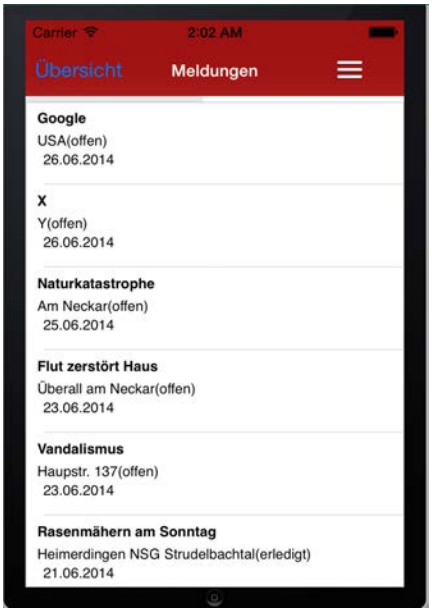
(a) Login (b) Registration

Abbildung 5.6: Live-Reports Anmeldung

Nach einer erfolgreichen Anmeldung gelangt man in die Übersicht der Bürgermeldungen, wo man neue Bürgermeldungen erstellen oder sich die bisherigen in einer Tabelle oder auf der Karte anschauen kann, siehe Abbildung 5.7 (a).

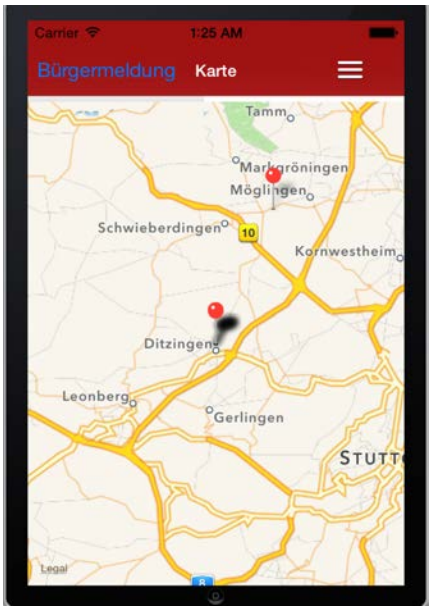


(a) Reports



(b) Reportdetails

(c) Reports



(d) Reports in der Karte

Abbildung 5.7: Live-Reports

Bei *Bürgermeldung anzeigen* gelangt man in die klassische Übersicht in Tabellenform, in der man alle Meldungen sehen kann, siehe Abbildung 5.7 (b). Klickt man hierauf, gelangt man wiederum in die Detailübersicht (Abbildung 5.7 (c)), in der alle Informationen vorhanden sind. Auch kann man sich die jeweilige Meldung in der Kartenansicht anschauen (siehe Abbildung 5.7 (d)). Man hat außerdem die Möglichkeit, Kommentare hinzuzufügen. Diese werden unter der jeweiligen Bürgermeldung angezeigt.

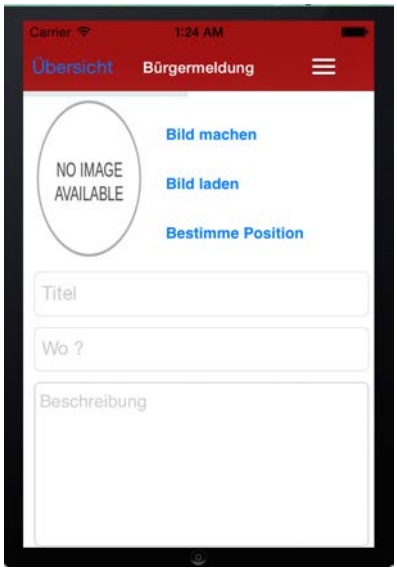


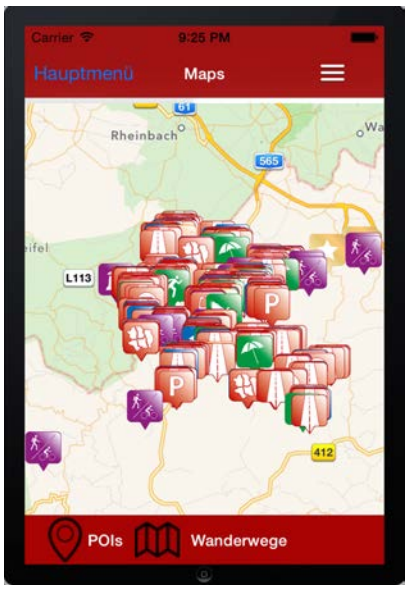
Abbildung 5.8: Erstellung Bürgermeldung

Eine weitere Seite in diesem Modul dient der Erstellung einer neuen Bürgermeldung. Man kann hier durch Beschreibung eines Titels, eines Orts und einer detaillierten Beschreibung solch eine erstellen. Optional kann noch ein Bild gemacht oder ein Bild aus der Galerie des mobilen Geräts hochgeladen werden. Das Bild wird dabei skaliert auf den Server hochgeladen, um unnötigen Datentransfer und hohe Speicherkapazität zu vermeiden. Hat man außerdem sein Internet und GPS aktiviert, so kann man seine aktuelle Position durch Klicken auf dem Button *Bestimme Position* bestimmen lassen. Die Koordinaten werden dabei vom Gerät ermittelt und an das Reverse Geocoding, also die Bestimmung einer gültigen Adresse zu den entsprechenden Koordinaten, der Google Maps API[31] geschickt, welches das Ergebnis zurückliefert. Alternativ existiert die Möglichkeit das Reverse Geocoding, welches von Titanium angeboten wird, zu verwenden. Diese greifen dabei auf den Service von MapQuest Open Nominatim[32] zurück, welcher für Deutschland ungenügende Ergebnisse liefert.

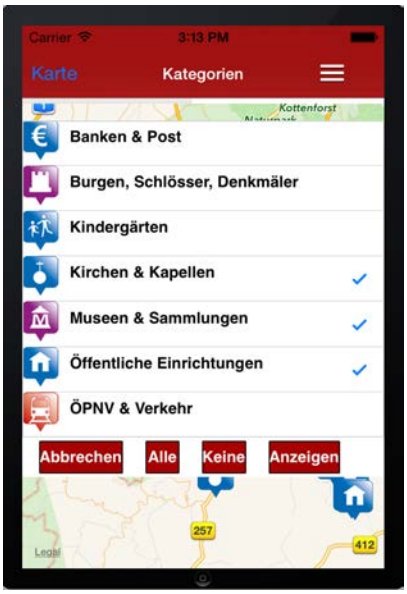
5.5.7 Karte

Das vierte Modul ist die Karte. Auf dieser kann man sich sowohl Routen/Wanderwege als auch Sehenswürdigkeiten (Points of Interest) anschauen. Standardmäßig werden dabei beim Start alle Sehenswürdigkeiten der verschiedenen Kategorien geladen. In iOS findet man unten auf dem Bildschirm eine Tabbar (Abbildung 5.9 (a)), in welcher man dann einerseits die Kategorien für die POIs auswählen kann (Abbildung 5.9 (b)). Dabei können mehrere oder auch keine Kategorien ausgewählt werden. Man kann dabei die jeweiligen Kategorie durch Tippen ab- oder auswählen. Klickt man nun auf den Button *Anzeigen*, so wird die Auswahl der Kategorien als Datei im Applikationsfolder des Endgerätes hinterlegt. Beim nächsten Start der Anwendung wird die gespeicherte Auswahl übernommen.

[31] https://developers.google.com/maps/?hl=de, aufgerufen am: 03.10.2014
[32] https://open.mapquestapi.com/nominatim/, aufgerufen am: 03.10.2014



(a) POIs



(b) POIs Kategorien

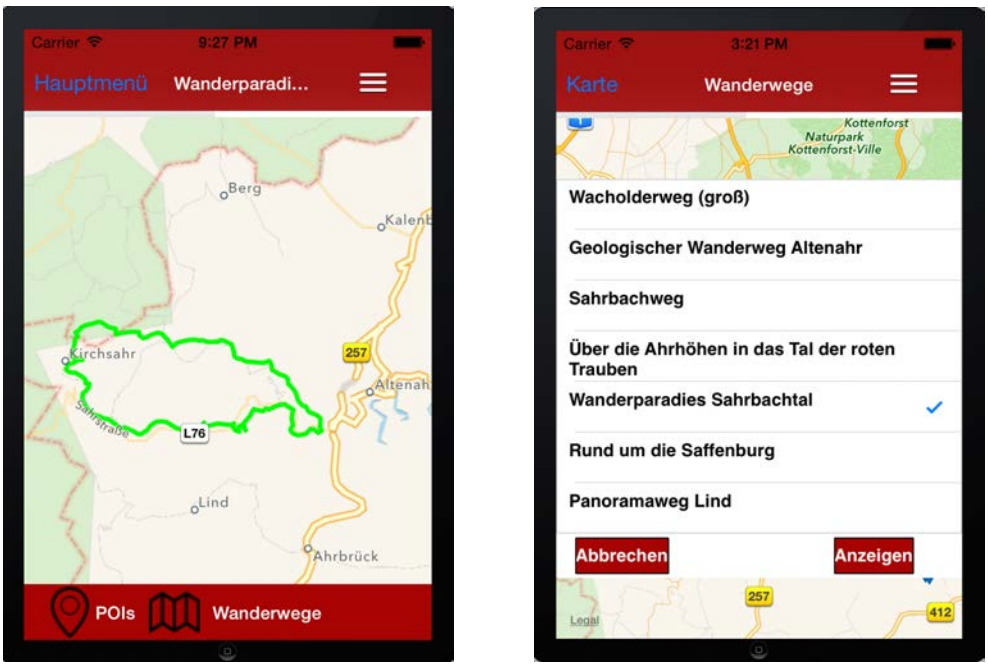
(c) Details eines POIs

Abbildung 5.9: Maps POIs

Klickt man nun auf der Karte auf einen bestimmten POI, so erscheint dort die Annotation mit dem Titel, den Koordinaten, Höheninformatione und einem Infobutton. Klickt man bei iOS auf den Infobutton, so erscheinen die Details zu einem POI, wie Bilder, Adresse und Telefonnummer und weitere Informationen, je nach Point of Interest (siehe auch Abbildung 5.9 (c)). Bei Android kann man nicht nur auf den Infobutton, sondern auch einfach auf die Annotation klicken um die Details anzuzeigen, da der Infobutton bei Android sehr klein dargestellt wird.

Andererseits kann man in der Tabbar bei iOS auch Wanderwege/Routen auswählen. Bei der Auswahl kann dabei, nicht wie bei den POIs mehrere, sondern nur eine Route ausgewählt werden, welche dann anschließend angezeigt wird (siehe auch Abbildung 5.10 (a)). Diese Route wird ebenfalls als Datei abgespeichert und sollte beim nächsten Aufrufen der Karte geladen werden.

Bei Android gibt es keine Tabbar, sondern die Buttons wurden als Menü implementiert, welche aus zwei Icons rechts in der Action Bar bestehen.



(a) ausgewählte Route (b) Routen/Wanderwege

Abbildung 5.10: Maps Routen

5.6 Unterschied zur Entwicklung einer mobilen Cross-Plattform-Applikation mit PhoneGap

Beide Frameworks erlauben die Programmierung plattformunabhängiger mobilen Applikationen. Beide Frameworks sind als Open-Source Framework frei verfügbar. Auch nutzen beide gewisse Webtechnologien zur Implementierung einer mobilen Applikation. Jedoch unterscheidet sich eine Umsetzung mit Appcelerator Titanium wesentlich von der mittels PhoneGap.

Während PhoneGap mittels eines Wrappers bzw. eines nativen Containers eine mobile Web-App ausführt, die mit den Webtechnologien HTML5, CSS3 und JavaScript programmiert wurde (siehe auch Kapitel 3.3.1.1), versucht Appcelerator Titanium, Ja-

vaScript Proxy-Objekte (siehe auch Kapitel 3.3.2.1) in der nativen Umgebung mittels eines JavaScript-Interpreters zu interpretieren. PhoneGap stellt dabei eine native API bereit, die es erlaubt, auf gewisse Hardwarefunktionen zuzugreifen. Der Zugriff auf die Hardware ist jedoch bei PhoneGap beschränkt. Bei Titanium hingegen hat man Zugriff auf alle nativen Hardwareelemente.

Ein Vorteil von PhoneGap ist jedoch, dass jede Plattform, welche ein WebView unterstützt, eine PhoneGap Applikation ausführen kann. Die Portabilität ist somit sehr hoch, da viele mobile Geräte heutzutage eine WebView unterstützen.

Auch muss man bei der Implementierung von nativen Elementen nicht wissen, wie es intern umgesetzt wird. Der Aufruf eines nativen Elements erscheint wie eine Black-Box für den Benutzer. Jeder der eine Webseite programmieren kann, kann mit PhoneGap eine mobile Applikation erstellen.

Weiterhin ermöglicht PhoneGap native Plugins einfach zu implementieren und in PhoneGap zu integrieren.

Nachteilig ist, dass die auszuführende Applikation abhängig von dem Rendering des Browsers ist, der eine WebView erzeugt. Die Erfahrungswerte zeigen, dass das Rendering bei iOS und Android sehr gut funktioniert. Auch muss berücksichtigt werden, dass die Browser den Quellcode unterschiedlich parsen und darstellen.

Ein weiterer Nachteil von PhoneGap ist, dass keine nativen UI-Elemente genutzt werden können. Die Bedienelemente können nur innerhalb dieses WebViews erstellt werden. Die Performance von den Bedienelemente reicht noch nicht an die nativen Elemente heran, auch wenn mit der Webkit Engine die Browser immer besser werden. Das ist vermutlich eines der größten Probleme für mobile Web-Applikationen.

Ein großer Vorteil bei der Entwicklung mit Titanium ist die Verwendung von nativen UI-Elementen und Funktionen. Titanium versucht sich hier näher an der nativen Entwicklung der einzelnen Plattformen zu orientieren. Es kann fast jedes native Bedienelement erstellt und verwendet werden. Wenn man also eine TabGroup implementiert, gibt es keine visuellen Emulationen, die versuchen einen nativen Look & Feel herzustellen, so wie bei PhoneGap. Es können tatsächlich die Eigenschaft des nativen Elements benutzt werden. Man hat also das Look & Feel einer nativen Applikation.

Auch ist die Einarbeitung wesentlich einfacher, da man sich nicht in die nativen Programmiersprachen einarbeiten muss. Diese Verwendung von JavaScript anstatt Objective-C

bzw. Java wird auch als *Atwood's Law* bezeichnet. Dieses besagt, dass wenn man eine Anwendung mit JavaScript programmieren kann, man dies auch tun soll [Atw07].

Ein Nachteil von Titanium ist allerdings, dass die Unterstützung von Plattformen auf Android, iOS und Blackberry OS beschränkt ist. Die Unterstützung für eine neue native Plattform (z. B. Windows Phone 8) stellt für Appcelerator einen enormen Entwicklungsaufwand dar. Nicht nur der Zugriff auf alle Hardware-Elemente, sondern auch native UI-Elemente müssen hierfür entwickelt werden.

Auch scheint die Implementierung für die Programmierung einer mobilen Web-App noch nicht komplett ausgereift zu sein.

Ein weiterer Nachteil ist, dass manche UI-Elemente von Titanium noch nicht so gut funktionieren wie die nativen Elemente der jeweiligen Plattform [Whi12].

5.7 Fazit der Entwicklungsmöglichkeit mit Appcelerator Titanium

Es ist nicht immer der Fall, dass eine Methode auch für alle verschiedenen Plattformen zur Verfügung stehen. Oft gibt es nur spezifische Methoden für iOS-Geräte, die Android aber nicht unterstützt. Aus diesem Grund muss oft nach alternativen Entwicklungsmöglichkeiten gesucht oder je nach Plattform eine unterschiedliche Logik verwendet werden, um eine Anforderung zu implementieren.

Das ist auch der größte Nachteil, denn für unterschiedliche UI-Elemente muss man oft unterschiedliche Views für die einzelnen mobilen Betriebssysteme entwickeln. Insofern ist vermutlich der Aufwand für die Programmierung der UI-Elemente im Vergleich zur nativen Programmierung nicht wesentlich geringer.

Ein weiterer Nachteil gegenüber der nativen Entwicklung besteht in der Performance. Insbesondere hat sich während der Implementierung gezeigt, dass die Ausführung der Applikation mit Android oft zu kleinen Verzögerungen führt, was in einer subjektiv langsamen App resultiert.

Bei dem Testen auf dem echten mobilen Gerät merkt man überdies, dass man auf die Speicherverwaltung aufpassen muss je komplexer eine Applikation wird, da dies sonst zu Abstürzen führen kann.

Die Größe eines Android Builds ist zudem erheblich groß. Der Grund liegt darin, dass Titanium die V8 JavaScript Engine für jeden Build erstellt. In iOS wird die JavaScript Engine des mobilen Endgerätes genutzt.

Möchte man seine Anwendung auch für iOS mit dem Appcelerator Titanium SDK kompilieren, so gilt es zu beachten, dass in der kostenlosen Version von Appcelerator ein Mac OS-fähiges Gerät notwendig ist, da nur solche das SDK für iOS besitzen. Möchte man dann die erstellte Anwendung auf einem echten Gerät testen, so fallen hier zusätzliche Kosten für die Mitgliedschaft als Developer bei Apple an. Dies muss während der Entwicklung berücksichtigt werden.

Vorteilhaft ist aber die einmalige Verwendung der Logik, vorausgesetzt die Methoden existieren auf allen Plattformen. Diese muss nur einmal implementiert werden und kann für alle Betriebssysteme genutzt werden. Ebenso die Verwendung von Medien-Aufrufen wie Kamera, Galerien, Maps und das Dateisystem. Hier ist aufgerufener Code mit Titanium wesentlich kürzer als ein nativer Quellcode in Objective-C oder Java. Sollte einmal die Appcelerator Titanium API keine ausreichenden Entwicklungsmöglichkeiten bereitstellen, so kann man hier auch auf die nativen SDKs von Android und iOS zurückgreifen. Diesen Quellcode in Objective-C oder Java kann man in Form eines externen Moduls in das Projekt einbinden.

Ebenso muss man nur JavaScript und die spezifische API von Titanium erlernen und muss sich nicht mit den jeweiligen Programmiersprachen Java für Android und Objective-C für iOS beschäftigen. Dies ist vor allem für Web-Programmierer sehr vorteilhaft.

Auch der geringere Wartungs- und Entwicklungsaufwand macht sich positiv bemerkbar. Man muss nämlich nur noch einen gemeinsamen Quellcode für alle Plattformen pflegen, die sogenannte *Shared Code Base* (siehe auch Kapitel 3).

Des Weiteren können die nativen Vertriebsmöglichkeiten genutzt werden. Die entwickelte Anwendung kann, wie gewöhnlich, im Google Play Store und dem Apple Store veröffentlicht und vertrieben werden.

Als Fazit lässt sich für die Entwicklung feststellen, dass auch Appcelerator Titanium einige Vor- und Nachteile hat. Die Entwicklung ist aber wesentlich vereinfacht im Vergleich zur nativen Entwicklung in Android oder iOS. Es bietet auf jeden Fall eine Alternative zu all den anderen Cross-Plattformen wie PhoneGap, Rhomobile, oder anderen. Insbesondere interessant ist die Verwendung von nativen UI-Elementen. Damit unterscheidet sich die mobile Applikation vom Look & Feel von anderen Applikationen. Sehr oft wird für diese Art von mobilen Applikation PhoneGap und jQuery Mobile verwendet.

Jedoch wird es die native Entwicklung nicht komplett ersetzen können. Für performancekritische mobile Applikationen eignet sich Titanium nämlich nicht.

6

Anforderungsabgleich

Im Folgenden werden die Anforderungen (Kapitel 4) an die zu entwickelnde Applikation mit der Implementierung abgeglichen.

6.1 Funktionale Anforderungen

Es werden zuerst die funktionalen Anforderungen abgeglichen. Folgende Zeichen werden hierbei verwendet:

✓ kennzeichnet eine erfüllte Anforderung.

× kennzeichnet eine nicht erfüllte Anforderung.

Bei teilweise erfüllten Anforderungen sind beide Zeichen zu erkennen.

6.1.1 Nachrichten

Anforderung	✓/×	**Kommentar**
Kategorien anzeigen	✓	konnte über eine Tabellenansicht umgesetzt werden (siehe Kapitel 5.5.4).
Nachrichten anzeigen	✓	konnte ebenfalls über eine Tabellenansicht implementiert werden (siehe Kapitel 5.5.4).
Nachrichten löschen/erstellen	✓	bleibt dem Administrator der Seite vorbehalten.

6.1.2 Veranstaltungen

Anforderung	✓/×	**Kommentar**
Kategorien anzeigen	✓	konnte über eine Tabellenansicht umgesetzt werden (siehe Kapitel 5.5.5).
Veranstaltungen anzeigen	✓	Zusätzlich zur Tabellenansicht musste noch ein Datepicker umgesetzt werden. Das Überblenden der Tabelle bei der Auswahl eines Datums war bei Android jedoch eine Herausforderung (siehe Kapitel 5.5.5).
Veranstaltungen löschen/erstellen	✓	bleibt dem Administrator der Seite vorbehalten.

6.1.3 Live-Reports

Anforderung	✓/×	**Kommentar**
Registrierung	✓	konnte gut erfüllt werden, da Titanium eine API zum Speichern sowie zum Verschlüsseln der Daten bereitstellt. Ebenso werden verschiedene Tastaturlayouts zur Eingabe von Texten unterstützt (siehe Kapitel 5.5.6).
Login	✓	Sowohl das automatische Einloggen als auch das Einloggen auf dem Server konnte problemlos umgesetzt werden (siehe Kapitel 5.5.6).
Profil ändern	✓	bleibt dem Administrator der Seite vorbehalten.
Anwender löschen	✓	bleibt dem Administrator der Seite vorbehalten.
Übersicht anzeigen	✓	konnte umgesetzt werden (siehe Kapitel 5.5.6).
Bürgermeldung erstellen	✓	Mit dem verwendeten Framework kann sehr einfach auf die nativen Elemente wie Kamera, Gallerie und Geolocation zugegriffen werden. Die Daten können über ein POST-Request an den Server geschickt werden. Lediglich die fehlende Unterstützung für einen Multidata-Request erforderte einen zusätzlichen Aufruf für das Hochladen des optionalen Bildes (Kapitel 5.5.6).

Bürgermeldung anzeigen	✓	Die Übersicht sowie die Details einer Bürgermeldung konnten einfach umgesetzt werden. Das Kommentieren einer Bürgermeldung war ebenfalls mit bestehenden Funktionen realisierbar (siehe Kapitel 5.5.6).
Status einer Meldung ändern	✓	bleibt dem Administrator der Seite vorbehalten.
Bürgermeldung löschen/ändern	✓	bleibt dem Administrator der Seite vorbehalten.

6.1.4 Karte

Anforderung	✓/×	**Kommentar**
Kategorien	✓	wurde über eine Tabellenansicht umgesetzt (siehe Kapitel 5.5.7).
Kategorien hinzufügen/ändern	✓	bleibt dem Administrator der Seite vorbehalten.
POI anzeigen	✓	Das Anzeigen der POIs der jeweiligen Kategorien kann einfach über Annotation implementiert werden. Das Abspeichern der zuletzt aufgerufenen Kategorien konnte ebenso umgesetzt werden (siehe Kapitel 5.5.7).
POI ändern/löschen	✓	bleibt dem Administrator vorbehalten.
Routen anzeigen	✓	Das Darstellen einer Route kann über die vom bereitgestellten Framework Maps-Routen-Funktion realisiert werden. Ebenso konnte das Speichern der zuletzt ausgewählten Route implementiert werden (siehe Kapitel 5.5.7).
Route ändern/löschen	✓	bleibt dem Administrator der Seite vorbehalten.

6.2 Nichtfunktionale Anforderungen

Ebenso werden die qualitativen Anforderungen abgeglichen.

Anforderung	✓/×	**Kommentar**
Änderbarkeit	✓	Durch die Verwendung des Alloy Konzepts von Titanium kann der Code in voneinander getrennte Module aufgeteilt werden.
Funktionalität	✓	Der Benutzer bekommt nur soviel Daten zu sehen wie es tatsächlich nötig ist.
Zuverlässigkeit	✓/×	BoxResizer ist, obwohl in den SLA beschrieben, nicht immer verfügbar.
Benutzbarkeit	✓/×	Es werden bekannte Design- und Navigationskonzepte der Plattformen verwendet. Auch auf Unterschiede bei Android und iOS wurde eingegangen, ebenso wie auf die Orientierung des Bildschirms. Jedoch benötigt Android manchmal lange zum Laden der Daten.
Effizienz	✓/×	Applikation braucht auf Android sehr lange bis das Startmenü geladen ist. Das liegt allerdings am Javascript Interpreter und lässt sich leider nicht vermeiden.
Übertragbarkeit	✓/×	Die Tourismus Applikation kann von Appcelerator Titanium nicht in ein anderes Framework portiert, sondern nur mit Appcelerator Titanium selbst weiterentwickelt werden.
Sicherheit	✓	konnte gut erfüllt werden, weil die benötigten Verschlüsselungsalgorithmen und das Speichern verschlüsselter Information von dem Framework Titanium unterstützt werden.
Kompatibilität	✓	Der Austausch von Daten über gängige Formate wird mit Titanium gut unterstützt. Als Grundlage für die Kommunikation zwischen dem Server und der mobilen Applikation wird JSON genutzt.

6.2.1 Anwendungsinformationen

Anforderung	✓/×	**Kommentar**
Sprache Deutsch	✓	
Design	✓/×	fast gleiches Aussehen auf iOS und Android. Aufgrund unterschiedlicher Hardware der Zielplattformen mussten hier teilweise unterschiedliche Konzepte implementiert werden, insbesondere für das Slidemenü und die Karte.

6.2.2 Technische Anforderungen

Anforderung	✓/×	**Kommentar**
Plattform	✓	App für Android & iOS entwickelt, auch wenn oft unterschiedliche Konzepte und UI-Elemente für die Zielplattformen verwendet werden musste.
Gerätekategorie	✓	Tablets & Smartphones werden unterstützt. Das Alloy Konzept von Titanium ermöglicht es, das Aussehen der Anwendung für die entsprechenden Geräte zu optimieren.

7

Fazit & Ausblick

Ziel dieser Arbeit war es, unterschiedliche Möglichkeiten der Cross-Plattform-Entwicklung aufzuzeigen und eine aktuelle Referenz der verschiedenen Plattformen zu erstellen. Darüber hinaus galt es, einen Prototypen mit Hilfe einer ausgewählten Cross-Plattform-Entwicklungsumgebung zu implementieren.

In diesem letzten Kapitel wird ein Fazit zu der Entwicklung mit der Cross-Plattform-Technologie gezogen und auf zukünftige Entwicklungstrends von mobilen Applikationen sowie einen Ausblick auf die Verbesserung der implementierten App eingegangen.

7.1 Cross-Plattform-Entwicklung

Die Cross-Plattform-Entwicklung wird immer mehr in Betracht gezogen. Dabei wurde eine Cross-Plattform-Pyramide von dem Unternehmen AllianceTek ([All12]) entwickelt, die sehr deutlich die Unterschiede der verschiedenen Arten der Cross-Plattform-Entwicklung und ihre Vor- und Nachteile aufzeigt.

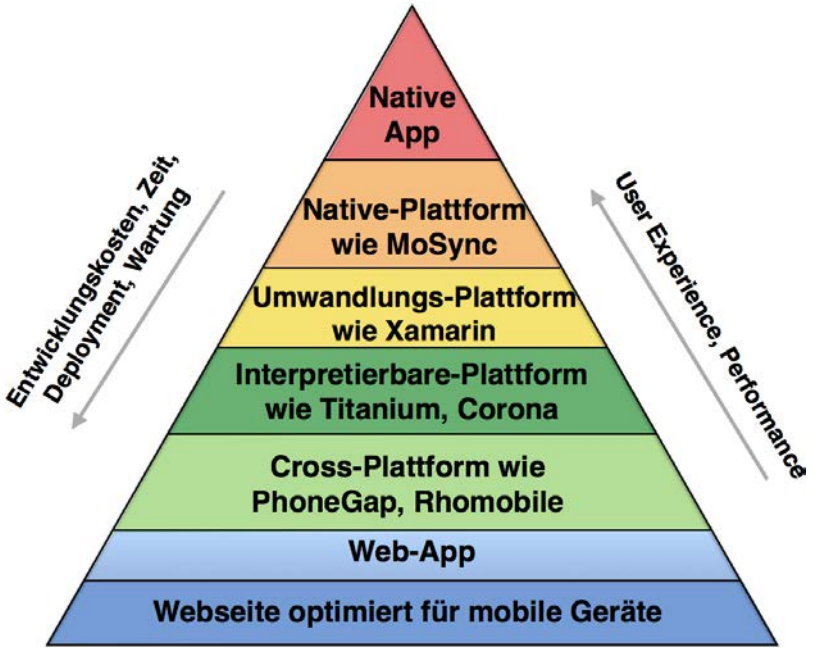


Abbildung 7.1: Mobile App Pyramide (in Anlehnung an [All12])

Ganz unten befinden sich Webseiten, die für mobile Geräte optimiert sind. Dann geht es nach oben in der Pyramide mit dem mobilen Web-Apps, die sich mit Hilfe von CSS3, HTML5 und JavaScript programmieren lassen, um einen nativen Look zu erzeugen. Diese bieten jedoch noch keinen Zugriff auf native Elemente. Dies ermöglichen erst Cross-Plattformen wie PhoneGap und Rhomobile. Eine Stufe höher findet man dann die interpretierten Applikationen, die auch native Bedienelemente verwenden. Xamarin und MoSync erlauben auch eine native Ausführung, da die Umwandlung direkt in plattformspezifischen Bytecode erfolgt.

In dieser Pyramide steigt von unten nach oben die User Experience, also die Qualität der Interaktion zwischen Nutzer und Applikation. Benutzt der Anwender diese Applikation gerne, da sie nicht abstürzt, Fehlermeldungen anzeigt oder auch sich leicht bedienen lässt, sind Aspekte, die in die User Experience einfließen.

Ebenso erhöht sich die Performance, da auf die nativen Elemente zugegriffen werden kann und Quellcode nicht mehr interpretiert wird. Stadtessen wird der Quellcode in plattformspezifischen Binärcode umgewandelt und dann direkt ausgeführt.

Jedoch erhöhen sich von der optimierten Webseite hin zu einer nativen Applikation die Entwicklungskosten und die Zeit. Bei den mobilen Apps können noch Webtechnologien eingesetzt werden. Bei der nativen oder Cross-Plattform-Entwicklung mit Umwandlung in Binärcode ist jedoch immer ein spezifisches Know-How der jeweiligen Technologie und der Zielplattformen nötig. Man benötigt also hier mit der Einarbeitung mehr Zeit.

Auch die Kosten erhöhen sich, da zum einen mehr Zeit für die Entwicklung benötigt wird. Zum anderen entstehen aber auch eventuelle Kosten für die Entwicklung in einer nativen Sprache.

Auch ist es einfacher, eine Webseite auf allen verschiedenen Plattformen zu deployen als eine native App. Diese muss auf den einzelnen Stores vertrieben werden und bedarf bei iOS auch noch einer Zustimmung von Apple. Ferner müssen Updates bei nativen Applikationen erst wieder auf dem Zielgerät neu installiert werden. Bei mobilen Web-Apps wird hingegen bei jedem Aufruf automatisch die aktuelle Version auf dem Endgerät geladen.

Ein weiterer Aspekt ist die Wartung der Applikation, denn der Quellcode einer nativen Applikation muss für alle verschiedenen Plattformen gewartet werden. Änderungen müssen immer mehrfach für die jeweiligen Zielplattformen vorgenommen werden. Dies ist bei einer einheitlichen Sprache wie den Webtechnologien nicht der Fall.

7.2 Zukünftige Trends bei der Entwicklung mobiler Applikationen

Die App-Entwicklung ist auch weiterhin attraktiv, da immer mehr neue Ideen auf den Markt kommen und die Menschen versuchen, ein vernetzteres Zuhause anzustreben (“Internet of Things“). Es geht darum, dass der Rauchmelder, die Waschmaschine oder der Kühlschrank mit mobilen Gerät kommunizieren kann um Status-Informationen mitzuteilen.

Gartner schätzt zudem einen großen Anstieg von bis zu 310 Millionen mobiler Applikation in allen App Stores bis zum Jahr 2016 [MyF13].

Auch neue mobile Betriebssysteme kommen auf den Markt. Firefox OS[33], Ubuntu OS[34] und Sailfish OS[35], ein proprietäres Betriebssystem von Nokia, sind bereits im Jahr 2014 erschienen.

Inzwischen überlegen sich viele Entwickler bei der Entwicklung selber eine Cross-Plattform einzusetzen, denn dies ermöglicht eine kostengünstige Entwicklung. PhoneGap/Cordova scheint dafür sehr oft eingesetzt zu werden, da es bei Webprogrammierern sehr beliebt ist [FBP+13, Set14].

[33] https://www.mozilla.org/en-US/firefox/os/, aufgerufen am: 24.09.2014
[34] http://www.ubuntu.com/phone, aufgerufen am: 24.09.2014
[35] https://sailfishos.org/, aufgerufen am: 24.09.2014

Laut dem Magic Quadrant aus dem Jahr 2014, siehe auch Abbildung 7.2, von Gartner befinden sich die typischen Cross-Plattform-Technologien in dem Quadranten *Visionäre* und *Leaders* (PhoneGap taucht hier unter Adobe auf). Ein Magic Quadrant ist dabei entlang der Achse *Completeness of Vision*, welche beschreibt, wie gut der Anbieter eine eigene Vision hat oder dem Markt folgt, und der Achse *Ability to Execute* angeordnet, welche die finanzielle Rentabilität, die Größe des Kundenstamms des Unternehmens, etc. betrachtet. Unter einem Visionär versteht Gartner also ein Unternehmen, das eine neue Technologie für einen bestimmten Markt entwickelt und sich darin noch behaupten muss. Ein Leader bietet laut Gartner eine ausgereifte Technologie und ist auch weiterhin bestrebt die Marktposition zu behalten. Leaders zeichnen sich durch ein sehr hohen Kundenstamm (relativ zum Markt) aus [Leh08].
Obwohl Appcelerator ein kleiner Anbieter im Vergleich zu anderen Cross-Plattformen ist, hat sich *Appcelerator Titanium* im Vergleich zur Einschätzung des Magic Quadrants aus dem Jahr 2013 von einem Visionär zu einem Leader entwickelt. Einer der Gründe dafür ist, dass Appcelerator Enterprise-Kunden nun mehr Dienste (wie die Cloud, Mobile Backend as a Service, Testing und Analysis Services) bietet. Ebenso erhöhte sich die Anzahl der Entwickler von 450.000 im Jahr 2013 auf 600.000 im Jahr 2014 [WMBV14, FPS+13].

Abbildung 7.2: Gartner Magic Quadrant Mobile Application Development 2014 (nach [WMBV14])

Gartner prognostiziert, dass bis zu 50 % der mobilen Applikationen bis 2016 als hybride App, also unter Verwendung einer Cross-Plattform-Technologie, entwickelt werden [Riv13].

Ob jetzt aber PhoneGap, Titanium, MoSync, JQuery Mobile, Sencha Touch oder andere Frameworks für die Entwicklung benutzt werden, spielt für den Anwender keine Rolle. Am Ende interessiert den Nutzer von mobilen Applikationen nur, wie schnell die App läuft und wie gut sich das Look & Feel anfühlt und nicht, wie diese Applikation programmiert wurde.

7.3 Ausblick

Der Ausblick beschäftigt sich mit den Themen, die während der Entwicklung der Tourismus-Applikation aufgekommen sind. Es zeigt Verbesserungsvorschläge für die implementierte Applikation und die Auswahl des Cross-Plattform-Frameworks auf.

7.3.1 Verbesserung der Auswahlentscheidung

Um eine wirklich fundierte Auswahl der zu entwickelnden Applikation entsprechenden Cross-Plattform zu treffen, ist es sehr wichtig, Vergleiche der verschiedenen Implementation vorzunehmen. Diese mobile Applikation sollte, um eine fundierte Auswahl der zu wählenden Cross-Plattform zu treffen, auch beispielhaft mit anderen Frameworks außer Appcelerator Titanium implementiert werden. Dies war jedoch nicht die Intention dieser Arbeit. Es gibt bereits mehrere Arbeiten, die sich damit beschäftigen, aufgrund mehrfacher Implementation eine Entscheidung der Cross-Plattform zu treffen.

7.3.2 Unterstützung weiterer Displaygrößen und Betriebssysteme

Die Tourismus-Applikation wurde hauptsächlich für iOS und Android implementiert. Das Testen unter iOS konnte nur in dem Simulator stattfinden, da kein verfügbares Gerät zum Testen und keine Apple Developer Mitgliedschaft vorhanden war. Für die Entwicklung und das Testen in Android wurde ein Samsung Galaxy S3 verwendet, welches eine sehr große Bildschirmauflösung hat. Da es aber sehr viele kleinere Smartphones gibt, sollte man vielleicht in der Entwicklung noch auf die Layout-Elemente für kleinere Bildschirme eingehen, da diese eventuell nicht so dargestellt werden, wie gewünscht. Dies wurde in der implementierten Applikation nur teilweise berücksichtigt.

Um die Erreichbarkeit zu erhöhen, könnte noch Windows Phone als zu unterstützende Plattform berücksichtigt werden, dies war allerdings mit Appcelerator nicht möglich, da

der angekündigte Support für Windows Phone Applikationen voraussichtlich erst im Winter 2014 möglich ist.

7.3.3 Verbesserung der Anforderungen

Bei der Implementation sind zwei nicht berücksichtigte Anforderungen für die Bürgermeldungen aufgefallen.

Zum einen besteht bei der Registration die Möglichkeit, sich mit einer fremden E-Mailadresse zu registrieren. Für eine sichere Registration sollte die sogenannte *Zwei-Faktor-Authentifizierung* genutzt werden. Es sollte also dem neu registrierten Benutzer eine Bestätigungsmail zugehen, die er dann bestätigen muss, um sich erfolgreich zu registrieren.

Zum anderen gibt es die Möglichkeit, Spam als neue Bürgermeldung zu erstellen. Dies sollte ebenfalls verhindert werden. Das einfachste wäre es, diese Aufgabe dem Administrator der jeweiligen Kommune zu überlassen, der einen Benutzer nach mehrmaligen Missbrauch der Bürgermeldungen sperrt.

7.4 Schlussfolgerung

Man hat gesehen, dass es einen sehr großen Vorteil bei der Entwicklung einer Applikation mit einem Cross-Plattform-Framework gibt. Aber so wie bei allen Dingen im Leben, muss man immer zwischen den Vorteilen und Nachteilen einer neuen Idee, hier also die vereinfachte Entwicklung einer App mit Hilfe einer Cross-Plattform, abwägen. Auch lässt sich dies nicht pauschal beantworten. Man sollte dies immer auf den Einzelfall herunter brechen und spezifisch entscheiden.

Mit Sicherheit kann man aber sagen, dass die Cross-Plattform- Entwicklung immer mehr und mehr im Kommen ist, denn viele Entwickler überlegen, eine App mit einem solchen Framework zu schreiben; und dass es eine Alternative für alle diejenigen bietet, die sich mit Webtechnologien auskennen. Doch auch eine plattformspezifische Programmiersprache kann leicht erlernt werden. Dies sollte kein Hinderungsgrund sein.

Ganz wird es aber die ursprüngliche Entwicklung einer App mit den jeweiligen plattformspezifischen SDKs nicht ersetzen. Insbesondere dann, wenn performancekritische Applikationen, wie z. B. Spiele, implementiert werden müssen. Für eine mobile App, die ohne oder mit nur wenig performancekritischen nativen Elementen auskommt, bietet es hingegen eine kostengünstige und zeitsparende alternative Entwicklungsmöglichkeit.

A

Anhang

A.1 Quellcode

Der Quellcode befindet sich auf der beigefügten CD.
Die Ordnerstruktur ist dabei folgendermaßen aufgebaut:

rest/ In diesem Ordner befindet sich der Quellcode des REST-Services des Servers

Tourismus-Application In diesem Ordner befindet sich die clientseitige Entwicklung der mobilen Applikation

In beiden Ordnern ist eine README Datei vorhanden, welche den weiteren Aufbau des Ordners und der REST-API respektive der mobilen Applikation beschreibt.

A.2 Dalvik Virtual Machine

Das folgende Schaubild veranschaulicht, wie Java-Code in der Runtime von Android mit Hilfe der Dalvik Virtual Machine in ausführbaren Code übersetzt wird:

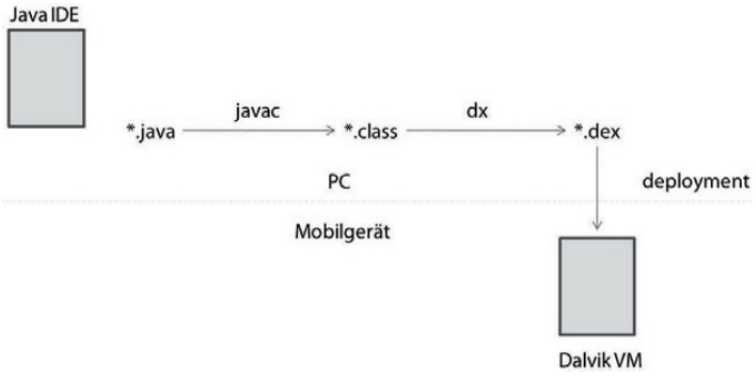


Abbildung A.1: Dalvik VM (nach[BP09] S.17)

A.3 Frameworks

Hier sind diejenigen Frameworks aufgelistet, die nicht den Anforderungsbedingungen entsprachen, erst im Nachhinein entdeckt wurden oder weniger bekannt und geeignet für die Cross-Plattform-Entwicklung dieser App sind:

Framework	**Anmerkungen**
Application Craft [36]	Web Framework, das es ermöglicht in der Cloud per Drag & Drop eine Anwendung zu entwickeln. Es ist allerdings nicht kostenlos.
Baker Framework[37]	HTML5 basiertes Framework, spezialisiert auf interaktive Bücher und Magazine für iOS.
DaVinci SDK [38]	IDE für die Entwicklung einer mobilen Web-App. Es enthält offene JavaScript Bibliotheken wie jQuery Mobile, Backbone, Underscore, Knockout und Handlebars.
EmbedJS [39]	JavaScript Framework, wurde allerdings seit 2012 nicht mehr weiterentwickelt.
eMobc[40]	Open-Source Framework, das von Spaniern gegründet wurde, scheint nicht mehr weiterentwickelt zu werden. Auf Anfrage bei Entwickler keine Reaktion.
Foundation 5 [41]	mobiles Web Framework, erst im Nachhinein entdeckt.

[36] www.applicationcraft.com, aufgerufen: 08.10.2014
[37] http://www.bakerframework.com/, aufgerufen: 08.10.2014
[38] http://www.davincisdk.com, aufgerufen: 08.10.2014
[39] http://uxebu.github.io/embedjs/, aufgerufen: 08.10.2014
[40] http://www.emobc.com/, aufgerufen: 08.10.2014
[41] http://foundation.zurb.com/, aufgerufen: 08.10.2014

Gideros [42]	native Cross-Plattform ähnlich wie Marmalade, ist für Spieleprogrammierung konzipiert.
IBM Worklight[43]	ähnlich wie Rhomobile, jedoch keine kostenlose Version verfügbar.
Ionic [44]	erst nachträglich entdecktes Web Framework.
jQTouch [45]	das bekannte jQTouch Framework ist nun Bestandteil von Sencha Touch.
Junior [46]	mobiles Web Framework, das eine Sammlung von bereits bestehenden JavaScript Bibliotheken ist.
Kendo UI Complete[47]	Web Framework basierend auf HTML5 und JavaScript, jedoch nicht kostenlos, nur 30 Tage Testversion erhältlich.
Kony [48]	Plattform, die je nach Wunsch eine hybride, native oder Web-App erstellt und bekannte Frameworks und alle SDKs einbindet. Kony ist jedoch nicht kostenlos verfügbar.
Laker Compedium[49]	HTML5 basiertes Framework, das auch auf digitale Magazine spezialisiert ist.
LimeJS [50]	mobiles Web Framework für die Spielentwicklung.
Marmalade [51]	natives Framework, das auf die Spielentwicklung spezialisiert ist, erst seit kurzem frei verfügbar.
Moai [52]	Framework für die Spielentwicklung.
Moobile [53]	Web Framework basierend auf MooTools. Wurde seit über einem Jahr nicht mehr weiterentwickelt.
Next Interface[54]	Open-Source Java GWT UI Framework das 2012 begonnen wurde zu entwickeln, seither jedoch nicht weiterentwickelt wurde und demzufolge nicht ausgereift ist.

[42] http://giderosmobile.com/, aufgerufen: 08.10.2014
[43] http://www-03.ibm.com/software/products/de/worklight/, aufgerufen: 08.10.2014
[44] http://ionicframework.com/, aufgerufen: 08.10.2014
[45] http://jqtjs.com/, aufgerufen: 08.10.2014
[46] http://justspamjustin.github.io/junior/#home, aufgerufen: 08.10.2014
[47] http://www.telerik.com/kendo-ui, aufgerufen: 08.10.2014
[48] http://www.kony.com/, aufgerufen: 08.10.2014
[49] http://www.lakercompendium.com/, aufgerufen: 08.10.2014
[50] http://www.limejs.com/, aufgerufen: 08.10.2014
[51] https://www.madewithmarmalade.com/, aufgerufen: 08.10.2014
[52] http://getmoai.com, aufgerufen: 08.10.2014
[53] http://moobilejs.com/, aufgerufen: 08.10.2014
[54] http://nextinterfaces.com/b/, aufgerufen: 08.10.2014

Paradise [55]	Lediglich nur ein Anbieter, der die Apps dann selbst für die Plattformen programmiert.
Qt[56]	natives Framework mit nativem Quellcode in C++ oder QML, jedoch nicht kostenlos verfügbar, nur 30 Tage Testversion.
QuickConnectFamily Hybrid [57]	wird nicht mehr weiterentwickelt, letzter Stand von 2011.
Rikulo [58]	mobiles Web Framework, erst im Nachhinein entdeckt.
Sidetap [59]	mobiles Web Framework, wurde seit über einem Jahr nicht mehr weiterentwickelt.
Telerik Platform [60]	mobiles Web Framework, ist nicht kostenlos verfügbar.
Trigger.io [61]	mobiles Web Framework, ist nicht kostenlos verfügbar.
ViziApps [62]	mobiles Web Framework, ist nicht kostenlos verfügbar, lediglich die 30 Tage Testversion.
V-Play [63]	Framework ebenfalls für die Spielentwicklung.
Wijmo [64]	mobileWeb Framework, das jQuery und jQuery Mobile Objekte als UI-Elemente nutzt, ist jedoch nicht kostenlos verfügbar.
Wink [65]	Open-Source Web Framework, wurde jedoch seit gut einem Jahr nicht mehr weiterentwickelt.
xui [66]	Open-Source Web Framework. Es wurde jedoch seit 2012 nicht mehr weiterentwickelt.

[55] https://www.paradiseapps.net/, aufgerufen: 08.10.2014
[56] https://qt-project.org/doc/qt-5/mobiledevelopment.html, aufgerufen: 08.10.2014
[57] http://www.quickconnectfamily.org/qc_hybrid/, aufgerufen: 08.10.2014
[58] http://rikulo.org/, aufgerufen: 08.10.2014
[59] http://sidetap.it/, aufgerufen: 08.10.2014
[60] http://www.telerik.com/platform#overview, aufgerufen: 08.10.2014
[61] http://trigger.io, aufgerufen: 08.10.2014
[62] http://www.viziapps.com/, aufgerufen: 08.10.2014
[63] http://v-play.net/, aufgerufen: 08.10.2014
[64] http://wijmo.com/, aufgerufen: 08.10.2014
[65] http://www.winktoolkit.org, aufgerufen: 08.10.2014
[66] https://github.com/xui/xuijs.com, aufgerufen: 08.10.2014

A.4 Icons

Für die Tourismus-Applikation wurden zusätzlich zu den bereitgestellten Icons frei verfügbare und/oder für den kommerziellen Gebrauch erlaubte Icons verwendet:

Maps Icon Android: POI Kategorien, mit der CC Attribution 4.0 Lizenz für den kommerziellen Gebrauch erlaubt, http://www.iconarchive.com/show/ecommerce-business-icons-by-designcontest/maps-icon.html[67].

Maps Icons iOS und Android: POI Kategorien und Wanderwege (iOS, sowie Info-Button (Android), veröffentlicht unter einer GPL / CC BY 3.0 Lizenz, https://icomoon.io/preview-free.html[68].

Live-Reports Icon: No Image available, bedarf keinem Copyright, https://commons.wikimedia.org/wiki/File:No_image_available.svg[69].

[67] zuletzt aufgerufen am: 06.10.2014
[68] zuletzt aufgerufen am: 06.10.2014
[69] zuletzt aufgerufen am: 06.10.2014

Abkürzungsverzeichnis

ADT	Android Development Tool
AIR	Adobe Integrated Runtime
API	Application Programming Interface
App	Applikation
AVD	Android Virtual Device
bspw.	beispielsweise
bzw.	beziehungsweise
ca.	circa
CLI	Command Line Interface
CPU	Central processing unit
CSS	Cascading Style Sheets
d. h.	das heißt
DOM	Document Object Model
DVM	Dalvik Virtual Machine
etc.	et cetera
ff.	fort folgende
GPS	Global Positioning System
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDC	International Data Corporation
IDE	Integrated Developement Environment
IPC	Inter-Process Communication

JSON JavaScript Object Notation

LOC Lines of Code

MVC Model View Controller

NF Normalform

OS Operating System

ÖPNV öffentlicher Personennahverkehr

PHP Hypertext Preprocessor

POI Point of Interest

QML Qt Meta Language oder Qt Modeling Language

REST Representational State Transfer

RIA Rich Internet Application

SDK Software Development Kit

SLA Service Level Agreement

SQL Structured Query Language

SWF Shockwave Flash

UI User Interface

URI Uniform Ressource Identifier

URL Uniform Ressource Locator

VM Virtual Machine

WWW World Wide Web

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

z. B. zum Beispiel

Abbildungsverzeichnis

Literaturverzeichnis

[All12] ALLIANCETEK.COM: *Comparison Chart for Mobile App Development Methods*. http://www.alliancetek.com/downloads/article/comparison-chart-for-mobile-app.pdf. Version: 2012, Abruf: 2014-10-04

[App13a] APPCELERATOR, Inc.: *Titanium Platform Overview*. http://docs.appcelerator.com/titanium/3.0/#!/guide/Titanium_Platform_Overview. Version: 2013, Abruf: 2014-10-04

[App13b] APPLE, Inc.: *Cocoa Touch Layer*. http://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1. Version: 2013, Abruf: 2014-10-04

[App13c] APPLE, Inc.: *Core OS Layer*. https://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html#//apple_ref/doc/uid/TP40007898-CH11-SW1. Version: 2013, Abruf: 2014-10-04

[App13d] APPLE, Inc.: *Core ServiceLayer*. http://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.html#//apple_ref/doc/uid/TP40007898-CH10-SW5. Version: 2013, Abruf: 2014-10-04

[App13e] APPLE, Inc.: *iOS Technology Overview-About the iOS Technologies*. https://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iOSTechOverview.pdf. Version: 09 2013, Abruf: 2014-10-04

[App13f] APPLE, Inc.: *Media Layer*. http://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html#//apple_ref/doc/uid/TP40007898-CH9-SW4. Version: 2013, Abruf: 2014-10-04

[App14a] APPCELERATOR, Inc.: *The Appcelerator Platform*. http://www.appcelerator.com/platform/appcelerator-platform/. Version: 2014, Abruf: 2014-10-04

[App14b] APPCELERATOR, Inc.: *Enterprise Connectors*. http://www.appcelerator.com/platform/apis/. Version: 2014, Abruf: 2014-10-04

[App14c] APPCELERATOR, Inc.: *Plans & Pricing*. http://www.appcelerator.com/plans-pricing/. Version: 2014, Abruf: 2014-10-04

[App14d] APPCELERATOR, Inc.: *Titanium Mobile Development Environment*. http://www.appcelerator.com/titanium/. Version: 2014, Abruf: 2014-10-04

[App14e] APPCELERATOR, Inc.: *Titanium.Network.HTTPClient*. http://docs.appcelerator.com/titanium/latest/#!/api/Titanium.Network.HTTPClient. Version: 2014, Abruf: 2014-10-04

[App14f] APPLE, Inc.: *Apple erfindet mit dem iPhone das Mobiltelefon neu*. https://www.apple.com/de/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html. Version: 2014, Abruf: 2014-10-04

[App14g] APPLE, Inc.: *iOS Developer Program*. https://developer.apple.com/programs/ios/. Version: 2014, Abruf: 2014-10-04

[App14h] APPLE, Inc.: *Xcode*. https://developer.apple.com/xcode/index.php. Version: 2014, Abruf: 2014-10-04

[AS14a] ADOBE SYSTEMS, Inc.: *About the Project*. http://phonegap.com/about/. Version: 2014, Abruf: 2014-10-04

[AS14b] ADOBE SYSTEMS, Inc.: *Getting Started with Android*. http://docs.phonegap.com/en/1.9.0/guide_getting-started_android_index.md.html. Version: 2014, Abruf: 2014-10-04

[AS14c] ADOBE SYSTEMS, Inc.: *Supported Features*. http://phonegap.com/about/feature/. Version: 2014, Abruf: 2014-10-04

[AT11] ALCALA TOCA, F.: *Cross-Platform-Entwicklung unter iOS und Android:Technologieüberblick und Prototyp-basierte Bewertung*. http://wwwiti.cs.uni-magdeburg.de/iti_db/publikationen/ps/12/thesisAlcalatoca.pdf. Version: 2011, Abruf: 2014-10-04

[Atw07] ATWOOD, Jeff: *The Principle of Least Power*. http://blog.codinghorror.com/the-principle-of-least-power/. Version: 2007, Abruf: 2014-10-04

[Bid13] BIDELMAN, E.: *Leitfaden für die ersten Schritte bei der Verwendung des Anwendungscaches*. Apache 2.0 License. http://www.html5rocks.com/de/tutorials/appcache/beginner/. Version: 2013, Abruf: 2014-10-04

[Bol98] BOLES, D.: *Begleitbuch zur Vorlesung Multimedia-Systeme: Definition von Hypertext und Hypermedia-Systemen*. Carl von Ossietzky Universität Oldenburg. http://www-is.informatik.uni-oldenburg.de/~dibo/teaching/mm/buch/node52.html. Version: 1998, Abruf: 2014-10-04

[BP09] BECKER, A. ; PANT, M.: *Android-Grundlagen und Programmierung*. 1. Auflage. dpunkt.verlag, 2009. – 15–21 S. – ISBN 978–3–89864–574–4

[CL14] CORONA LABS, Inc.: *Develop Cross Platform Mobile Apps and Games | Corona Labs*. http://coronalabs.com/products/corona-sdk/. Version: 2014, Abruf: 2014-10-04

[Cop13] COPE, Darren: *Appcelerator Titanium Application Development by Example*. Packt Publishing, 2013. – ISBN 978–1849695008

[deH12] DEHAAN, P.: *Hello World*. Sencha,Inc. http://www.sencha.com/learn/hello-world/. Version: 2012, Abruf: 2014-10-04

[Dhu12] DHUYVETTER, D.: *Using jQuery Mobile with Worklight 5.0.5*. IBM Corp. 2013. https://www.ibm.com/developerworks/community/blogs/dhuyvett/entry/using_jquery_mobile_with_worklight_5_0_0?lang=en. Version: 2012, Abruf: 2014-10-04

[Eli10] ELIAS, A.: *Ext JS + jQTouch + Raphaël = Sencha*. 2014 Sencha,Inc. http://www.sencha.com/blog/ext-js-jqtouch-raphael-sencha. Version: 2010, Abruf: 2014-10-04

[FBP+13] FINLEY, Ian ; BAKER, Van L. ; PARMELEE, Ken ; SMITH, David M. ; VALDES, Ray ; VAN HUIZEN, Gordon: *Magic Quadrant for Mobile Application Development Platforms*. http://www.gartner.com/technology/reprints.do?id=1-1I9XO1F&ct=130808&st=sb. Version: 2013, Abruf: 2014-10-04

[Fel14] FELONEY, S.: *Windows 8 Support, What's Going On?* Appcelerator Inc. http://www.appcelerator.com/blog/2014/01/windows-8-support-whats-going-on/. Version: 2014, Abruf: 2014-10-04

[FI13] FRANKE, F. ; IPPEN, J.: *Apps mit HTML5 und CSS3: für iPad, iPhone und Android*. 2. aktualisierte und erweiterte Auflage. Galileo Press,Bonn, 2013 (Galileo Computing). – 24–25,39–44,317–325,341–345 S. – ISBN 978–3–8362–2237–2

[Fie00] FIELDING, R. T.: *CHAPTER 5i-Representational State Transfer (REST)*. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Version: 2000, Abruf: 2014-10-04

[Fla11] FLANAGAN, D.: *JavaScript: the definitive guide; [activate your web pages; covers ECMAScript 5 & HTML5]*. 6. ed. Beijing : O'Reilly, 2011. – XVI, 1078 S. http://deposit.d-nb.de/cgi-bin/dokserv?id=3480223&prov=M&dok_var=1&dok_ext=htm. – ISBN 978–0–596–80552–4

[Fou12a] FOUNDATION, The jQuery: *Building PhoneGap apps with jQuery Mobile*. https://demos.jquerymobile.com/1.1.0/docs/pages/phonegap.html. Version: 2012, Abruf: 2014-10-04

[Fou12b] FOUNDATION, The jQuery: *Getting Started with jQuery Mobile*. https://demos.jquerymobile.com/1.2.0/docs/about/getting-started.html. Version: 2012, Abruf: 2014-10-04

[Fou13] FOUNDATION, Apache S.: *Apache Cordova*. Apache License, Version 2.0. http://cordova.apache.org/. Version: 2013, Abruf: 2014-10-04

[Fou14a] FOUNDATION, Apache S.: *About Apache Flex*. Apache License, Version 2.0. http://flex.apache.org/about-whatis.html. Version: 2014, Abruf: 2014-10-04

[Fou14b] FOUNDATION, Apache S.: *Getting Started with Apache Flex*. Apache License, Version 2.0. https://flex.apache.org/doc-getstarted.html. Version: 2014, Abruf: 2014-10-04

[Fou14c] FOUNDATION, The jQuery: *License*. https://jquery.org/license/. Version: 2014, Abruf: 2014-10-04

[FPS+13] FINLEY, V.L. I.AND B. I.AND Baker ; PARMELEE, K. ; SMITH, D.M. ; VALDES, R. ; VAN HUIZEN, G.: *Magic Quadrant for Mobile Application Development Platforms*. Gartner Inc. http://www.gartner.com/technology/reprints.do?id=1-1I9XO1F&ct=130808&st=sb. Version: 2013, Abruf: 2014-10-04

[Fri13] FRIEDMAN, N.: *Announcing Xamarin 2.0*. Xamarin,Inc. http://blog.xamarin.com/announcing-xamarin-2.0/. Version: 2013, Abruf: 2014-10-04

[Goo13a] GOOGLE: *System-Architecture*. Creative Commons Attributions 2.5. http://developer.android.com/images/system-architecture.jpg. Version: 2013, Abruf: 2014-10-04

[Goo13b] GOOGLE, Inc.: *Vereinbarung für den Entwicklervertrieb*. https://play.google.com/about/developer-distribution-agreement.html. Version: 2013, Abruf: 20014-04-10

[Goo14a] GOOGLE: *Android NDK*. Creative Commons Attributions 2.5. https://developer.android.com/tools/sdk/ndk/index.html. Version: 2014, Abruf: 2014-10-04

[Goo14b] GOOGLE: *Get Started with Publishing*. Creative Commons Attributions 2.5. https://developer.android.com/distribute/googleplay/start.html. Version: 2014, Abruf: 2014-10-04

[Goo14c] GOOGLE: *Tools Help*. Creative Commons Attributions 2.5. http://developer.android.com/tools/help/index.html. Version: 2014, Abruf: 2014-10-04

[Hel13] HELMICH, M.: *RESTful Webservices (1): Was ist das überhaupt?* Mittwald Blog. http://blog.mittwald.de/webentwicklung/restful-webservices-1-was-ist-das-uberhaupt/. Version: 2013, Abruf: 2014-10-04

[HHM13] HEITKÖTTER, Henning ; HANSCHKE, Sebastian ; MAJCHRZAK, Tim A.: *Web Information Systems and Technologies*. Springer-Verlag Berlin Heidelberg, 2013. – 120–138 S. http://link.springer.com/chapter/10.1007%2F978-3-642-36608-6_8. – ISBN 978–3–642–36608–6

[Hol10] HOLISTER, S.: *Microsoft prepping Windows Phone 7 for an October 21st launch? (update:US on Nov.8?)*. http://www.engadget.com/2010/09/26/microsoft-prepping-windows-phone-7-for-an-october-21st-launch/. Version: 2010, Abruf: 2014-10-04

[IDC13] IDC: *Apple Cedes Market Share in Smartphone Operating System Market as Android Surges and Windows Phone Gains, According to IDC*. https://www.idc.com/getdoc.jsp?containerId=prUS24257413. Version: 2013, Abruf: 2014-10-04

[ISO11] ISO/IEC: *ISO/IEC 25010:2011(en)*. https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en. Version: 2011, Abruf: 2014-10-04

[Kim08] KIM, A.: *iPhone, iPod Touch SDK Details March 6th*. http://www.macrumors.com/2008/02/27/iphone-ipod-touch-sdk-details-march-6th/. Version: 2008, Abruf: 2014-10-04

[Kuh12] KUHN, P.: *Windows Phone 8: Introduction to the Platform*. http://www.silverlightshow.net/items/Windows-Phone-8-Introduction-to-the-Platform.aspx. Version: 2012, Abruf: 2014-10-04

[Kün12] KÜNNETH: *Android 4-Apps entwickeln mit dem Android SDK*. 2. Auflage. Galileo Computing, 2012. – 21–33 S. – ISBN 978–3836219488

[Leh08] LEHMANN, J.: *Magic Quadrants and MarketScopes: How Gartner Evaluates Vendors Within a Market*. Gartner,Inc. https://www.gartner.com/doc/486094#a-2024708249. Version: 2008, Abruf: 2014-10-04

[Lic13] LICENSE, Apache 2.: *Kernel Overview*. https://source.android.com/devices/tech/datausage/kernel-overview.html. Version: 2013, Abruf: 2014-10-04

[Loc12] LOCKHART, J.: *Slim*. MIT Public License. http://slimframework.com/. Version: 2012, Abruf: 2014-10-04

[Loc14] LOCKHART, J.: *Slim Framework Documentation*. MIT Public License. http://docs.slimframework.com/. Version: 2014, Abruf: 2014-10-04

[LTB10] LABRIOLA, M. ; TAPPER, J. ; BOLES, M.: *Adobe Flex 4: Training from the Source*. Pearson Education, 2010 (Training from the Source Bd. 1). – Foreword S. http://books.google.de/books?id=Jz1KWU5tc3YC. – ISBN 9780321694423

[Mic14a] MICROSOFT: *Kontotypen, Standorte und Gebühren*. http://msdn.microsoft.com/de-de/library/windows/apps/jj863494.aspx. Version: 2014, Abruf: 2014-10-04

[Mic14b] MICROSOFT: *Native code on Windows Phone 8*. http://msdn.microsoft.com/en-US/library/windowsphone/develop/jj681687(v=vs.105).aspx. Version: 2014, Abruf: 2014-10-04

[Mic14c] MICROSOFT: *Vereinbarung für App-Entwickler*. http://msdn.microsoft.com/de-de/library/windows/apps/hh694058.aspx. Version: 2014, Abruf: 2014-10-04

[Mic14d] MICROSOFT: *Windows Phone API reference*. http://msdn.microsoft.com/library/windowsphone/develop/ff626516(v=vs.105).aspx#BKMK_WindowsRuntimestyleAPIforWindowsPhone. Version: 2014, Abruf: 2014-10-04

[Mic14e] MICROSOFT: *Windows Phone versions*. http://msdn.microsoft.com/en-us/library/windows/apps/hh202996(v=vs.105).aspx. Version: 2014, Abruf: 2014-10-04

[Mic14f] MICROSOFT: *Windows.ApplicationModel Namespace*. http://msdn.microsoft.com/de-de/library/windows/apps/windows.applicationmodel.aspx. Version: 2014, Abruf: 2014-10-04

[MoS13a] MOSYNC, AB: *HelloWorld, Deconstructed*. http://www.mosync.com/docs/sdk/cpp/guides/hello-world-deconstructed/index.html. Version: 2013, Abruf: 2014-10-04

[MoS13b] MOSYNC, AB: *HTTP Connections*. http://www.mosync.com/docs/sdk/cpp/guides/communication/http-connections/index.html. Version: 2013, Abruf: 2014-10-04

[MoS13c] MOSYNC, AB: *Licensing and Subscriptions*. http://www.mosync.com/mosync-dual-licence-model. Version: 2013, Abruf: 2014-10-04

[MoS13d] MOSYNC, AB: *The MoSync Toolchain*. http://www.mosync.com/docs/sdk/tools/guides/architecture/toolchain/index.html. Version: 2013, Abruf: 2014-10-04

[MoS13e] MOSYNC, AB: *The Team*. http://www.mosync.com/the_company. Version: 2013, Abruf: 2014-10-04

[MoS13f] MOSYNC, AB: *What's New in MoSync SDK 3.0*. http://www.mosync.com/docs/sdk/release-notes/whats-new-mosync-sdk-30/index.html. Version: 2013, Abruf: 2014-10-04

[MS12] MOTORLA SOLUTIONS, Inc.: *Rhodes Architecture: Frequently Asked Questions*. http://docs.rhomobile.com/en/2.2.0/architecturefaq. Version: 2012, Abruf: 2014-10-04

[MS14a] MOTOROLA SOLUTIONS, Inc.: *Rhomobile suite*. http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite. Version: 2014, Abruf: 2014-10-04

[MS14b] MOTOROLA SOLUTIONS, Inc.: *Using native UI elements*. http://docs.rhomobile.com/en/4.0.0/guide/native_ui_elements. Version: 2014, Abruf: 2014-10-04

[MS14c] MOTOROLA SOLUTIONS, Inc.: *Welcome To RhoMobile Suite*. http://docs.rhomobile.com/en/4.0.0/guide/welcome. Version: 2014, Abruf: 2014-10-04

[MyF13] MYFIRSTMOBILEAPP.COM: *MOBILE APPLICATION DESIGN & DEVELOPMENT TRENDS -2013*. http://www.apps-world.net/media/docs/resources/Whitepaper.pdf. Version: 2013, Abruf: 2014-10-04

[Nat14] NATIONS, D.: *Web Applications*. About.com. http://webtrends.about.com/od/webapplications/a/web_application.htm. Version: 2014, Abruf: 2014-10-04

[OR11] OFFERMANN, S. ; RIOS, V.: *Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap*. Adobe Systems Incorporated. http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html. Version: 2011, Abruf: 2014-10-04

[Par10] PARTSCH, H.: *Requirements-Engineering systematisch: Modellbildung für softwaregestützte Systeme*. 2.überarb. u. erw. Auflage. Springer Verlag, 2010. – 27–28 S. – ISBN 978–3642053573

[pho14] *Plugin Development Guide*. Apache License, Version 2.0. http://docs.phonegap.com/en/3.4.0/guide_hybrid_plugins_index.md.html#Plugin%20Development%20Guide. Version: 2014, Abruf: 2014-10-04

[PRS11] PREEZ, M. du ; REVELL, A. ; SMITH, S.: *BoxResizer - A simple image resizer service*. Boxcss. http://boxresizer.com/. Version: 2011, Abruf: 2014-10-04

[Riv13] RIVERA, Janessa: *Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid*. Gartner,Inc. http://www.gartner.com/newsroom/id/2324917. Version: 2013, Abruf: 2014-10-04

[Rub12] RUBINO, D.: *Overview and review of Windows Phone 8*. http://www.wpcentral.com/overview-and-review-windows-phone-8. Version: 2012, Abruf: 2014-10-04

[Sch14] SCHOSSOW, C.: *Interview: Creator of the Slim Framework*. New Media Campaigns. http://www.newmediacampaigns.com/blog/an-interview-with-the-founder-of-slim-php-framework-our-josh-lockhart. Version: 2014, Abruf: 2014-10-04

[Sen14a] SENCHA, Inc.: *Intro to Applications with Sencha Touch*. http://docs.sencha.com/touch/2.3.1/#!/guide/apps_intro. Version: 2014, Abruf: 2014-10-04

[Sen14b] SENCHA, Inc.: *Sencha Touch-Build Mobile Web Apps with HTML5*. https://www.sencha.com/products/touch/. Version: 2014, Abruf: 2014-10-04

[Sen14c] SENCHA, Inc.: *Sencha Touch Bundle*. http://www.sencha.com/products/touch-bundle/. Version: 2014, Abruf: 2014-10-04

[Sen14d] SENCHA, Inc.: *Sencha Touch Licensing Options*. https://www.sencha.com/products/touch/license/. Version: 2014, Abruf: 2014-10-04

[Set14] SETHURAMAN, Anusha: *Hot Trends in Mobile App Development for 2014*. http://blog.newrelic.com/2014/02/04/2014-mobile-dev-trends/. Version: 2014, Abruf: 2014-10-04

[SHS13] SPIERING, M. ; HAIGES, S. ; SCHOLZE, R.: *HTML5-Apps für iPhone und Android: HTML5, CSS3 und jQuery Mobile; Design, Programmierung und Veröffentlichung plattformübergreifender Apps*. 3., aktualis. Aufl. Haar bei München : Franzis, 2013. – 263–266 S. http://deposit.d-nb.de/cgi-bin/dokserv?id=4091784&prov=M&dok_var=1&dok_ext=htm. – ISBN 978–3–645–60201–3

[SSP+13] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. ; NIENHAUS, H. ; REICHERT, M.: Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In: *9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps*, 2013, 509–518

[TCLC14] THE COMPUTER LANGUAGE COMPANY, Inc.: *cross platform Definition from PC Magazine Encyclopedia*. http://www.pcmag.com/encyclopedia/term/40495/cross-platform#fbid=aHfb3ldkqPq. Version: 2014, Abruf: 2014-10-04

[Tec14] TECHTERMS.COM: *Crossplatform*. http://www.techterms.com/definition/crossplatform. Version: 2014, Abruf: 2014-10-04

[Vis14] VISWANATHAN, P.: *Native Apps vs. Web Apps – What is the Better Choice?* About.com. http://mobiledevices.about.com/od/additionalresources/a/Native-Apps-Vs-Web-Apps-Which-Is-The-Better-Choice.htm. Version: 2014, Abruf: 2014-10-04

[W3C13a] W3C: *HTML & CSS*. http://www.w3.org/standards/webdesign/htmlcss. Version: 2013, Abruf: 2014-10-04

[W3C13b] W3C: *HTML 5.1*. http://www.w3.org/TR/html51/. Version: 2013, Abruf: 2014-10-04

[W3C14] W3C: *CSS Syntax Module Level 3*. http://www.w3.org/TR/2014/CR-css-syntax-3-20140220/. Version: 2014, Abruf: 2014-10-04

[Whi12] WHINNERY, Kevin: *Comparing Titanium and PhoneGap*. http://www.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap/. Version: 2012, Abruf: 2014-10-04

[WMBV14] WONG, J. ; MARSHALL, R. ; BAKER, V.L. ; VALDES, R.: *Magic Quadrant for Mobile Application Development Platforms*. Gartner Inc. http://www.gartner.com/technology/reprints.do?id=1-20TPY1B&ct=140903&st=sb. Version: 2014, Abruf: 2014-10-04

[Xam13] XAMARIN, Inc.: *Part 3 - Setting up a Xamarin Cross Platform Solution*. http://docs.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_3_-_setting_up_a_xamarin_cross_platform_solution/. Version: 2013, Abruf: 2014-10-04

[Xam14a] XAMARIN, Inc.: *Frequently Asked Questions - Xamarin*. https://xamarin.com/faq. Version: 2014, Abruf: 2014-10-04

[Xam14b] XAMARIN, Inc.: *Part 1 - Understanding the Xamarin Mobile Platform*. http://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_1_-_understanding_the_xamarin_mobile_platform/. Version: 2014, Abruf: 2014-10-04

[Xam14c] XAMARIN, Inc.: *Part 3 -Setting Up A Xamarin Cross Platform Solution*. http://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_3_-_setting_up_a_xamarin_cross_platform_solution/. Version: 2014, Abruf: 2014-10-04

[Xam14d] XAMARIN, Inc.: *What is Xamarin? Take the tour and start building apps - Xamarin*. https://xamarin.com/tour. Version: 2014, Abruf: 2014-10-04

Name: Daniel Glunz Matrikelnummer: 702033

Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben wird.

Ulm, den .....

Daniel Glunz