Entwicklung einer mobilen Applikation für die Plattformen: Android und Windows Phone

BACHELORARBEIT IM RAHMEN DES
BACHELORSTUDIENGANGS INFORMATIK
SOFTWARE AND INFORMATION ENGINEERING

Dipl.-Ing Patrick Ritschel
Fachhochschule Vorarlberg

Vorgelegt von

Martin Münch
1110247032

Dornbirn, 17. August 2014

Zusammenfassung

In der aktuellen Post-PC-Ära verschiebt sich der Fokus der Endanwender immer mehr von statischen Desktops hin zu mobilen Geräten. Der fortgeschrittenen technischen Entwicklung sowie dem großflächigen Ausbau des Mobilfunknetzes ist es zu verdanken, dass sich diese leistungsstarken Lösungen ihren festen Platz in der Unterhaltungs- und Informationsindustrie gesichert haben. Das Einsatzgebiet der mobilen Applikationen reicht von kleineren Spielereien über nützliche Hilfsmittel im Alltag bis hin zu Business-Applikationen. Für Anbieter von digitalen Inhalten und Dienstleistungen ist die eigene mobile Applikation zum Aushängeschild ihres Unternehmens geworden. Mit der hohen Nachfrage nach mobilen Geräten ist auch der Wettbewerb unter den Herstellern gestiegen. Dies führte wiederum zu einem dynamischen Markt der zur Verfügung stehenden Plattformen

Um bei der Verbreitung von Apps einen möglichst großen Kundenkreis zu erschließen ist es notwendig, dass die Applikationen von mehr als einer der großen Plattformen unterstützt wird. Diese Arbeit beschäftigt sich mit der Herausforderung sowie den damit verbundenen Möglichkeiten, eine Applikation für die mobilen Plattformen Android und Windows Phone 8 zu entwickeln. Für diesen Zweck wurde - in Kooperation mit der Offene Jugendarbeit Dornbirn (OJAD) - ein Anforderungs-Profil ausgearbeitet. Der definierte Aufgabenbereich des Projektes liegt in der mobilen Unterstützung der SozialarbeiterInnen bei der Dokumentation von Veranstaltungen.

Ausgehend von den Projekt-Anforderungen, wird ein Kriterien-Katalog für die State of the Art-Analyse erstellt. Anhand dieser Kriterien wird eine Analyse über die aktuellen Entwicklungs-Ansätze sowie jeweils ein stellvertretendes Framework durchgeführt. Mit der Auswertung der Ergebnisse wird die State of the Art-Analyse abgeschlossen. In besagter Auswertung ist definiert, dass die Applikation auf der Basis eines nativen Ansatzes umgesetzt werden soll.
Die eigentliche Realisierung der Beispiel-Applikation ist in die beiden Kapitel

`Entwicklung` und `Implementierung` aufgeteilt. Der Inhalt des Kapitels Entwicklung beschäftigt sich mit der generellen Architektur, der Planung des User Interface (UI)-Konzeptes sowie dem Aufbau der Programmlogik. Auf die Einzelheiten der jeweiligen Software Development Kit (SDK)'s sowie ihre Verwendung in der Beispiel-Applikation im Kapitel: Implementierung eingegangen. Den Abschluss dieser Arbeit bildet eine Zusammenfassung über die Erkenntnisse und Anmerkungen, die während des Entwicklungsprozesses dieser Arbeit sowie der Beispiel-Applikation zustande gekommen sind.

Abstract

In the current post PC era the end users focus shifts from static desktops to mobile devices. It's due to the advanced technical development and the extensive expansion of mobile phone networks, these powerful solutions have secured their place in the entertainment and information industry. Mobile applications can range from small shenanigans over useful helpers in everyday life to critical business applicaitons. Providers of digital services their own mobile application is the flagship to communicate their corporate identity. With the increased demand of mobile devices competition in this market has been heightened, which leaded to a dynamic market of available platforms.

To reach the widest possible range of customers for apps it is indespensable to support more than one significant platform. This work is concerned with this challenge, as well as the associated possibilities, to develop an application for the mobile platforms Android and Windows Phone 8. For this purpose a demand profile has been developed in cooperation with OJAD. The defined task range of the project is the mobile support of social workers with the documentation of events.

By means of the defined project requirements I created a criteria catalogue for the State of the Art analysis, based on which current development approaches and representative frameworks have been studied. With the evaluation of the according results the State of the Arts analysis will be completed. According to the given results the application will be developed in a native approach. The realization of the project is split into two chapters, which are Entwicklung (Development) and Implementierung (Implementation). The chapter Entwicklung deals with the architecture of the application, the UI concept and the structure of the application logic. Details about the regarded SDKs, as well as their usage in an example application and peculiarities will be discussed in the chapter Implementierung. The final part of the work is a conclusion about the insights and annotations that came up in the process of developing this work and the example application.

Eidesstattliche 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.

Inhaltsverzeichnis

Kapitel 1

Einführung

Die Arbeitsdokumentation stellt für Organisationen einen wichtigen Faktor bei der Qualitätssicherung dar. Um aussagekräftige Analysen erstellen zu können, müssen die gesammelten Daten korrekt und konsistent sein. Leider zeigt die Erfahrung, dass vor allem im Umfeld von Vereinen und gemeinnützigen Organisationen nur selten einheitlichen Methoden zum Erfassen vorhanden sind und/oder diese nicht sachgemäß verwendet werden. In Zeiten von Smartphones liegt der Gedanke nahe, die aktuelle Technologie zu verwenden um den Arbeitsaufwand der Dokumentation zu minimieren.

Damit eine breite Masse als Zielgruppe angesprochen werden kann sollte das Programm für die geläufigsten mobilen Betriebssystem verfügbar sein. Im Moment besitzen die beiden konkurrierenden Betriebssystemen Android und iOS die größten Marktanteile im deutschsprachigen Raum. Allerdings versucht Microsoft mit Windows Mobile immer aggressiver Marktanteile zu gewinnen. Da diese Plattformen grundsätzlich nicht miteinander kompatibel sind, muss die Applikation für jedes Betriebssystem eigens implementiert werden. Die Nachteile liegen auf der Hand, neben dem Entwicklungsaufwand wird auch die Wart- und Erweiterbarkeit eingeschränkt.

Um diese Problematik zu entschärfen gibt es verschiedene Lösungsansätze. Zum einen kann das Programm in zwei Teile aufgeteilt werden. Bei diesem Vorgehen besteht die Applikation aus einen plattformunabhängigen- und einen plattformspezifischen Teil. Hierbei liegt das Ziel darin, den plattformspezifischen Code so weit wie möglich zu minimieren, wodurch der Grad der Wiederverwertbarkeit für andere Plattformen gesteigert wird. Zum anderen existieren verschiedene Frameworks für die Multiplattform-Entwicklung. Durch den Einsatz dieser Frameworks soll das Programm nur einmal für alle unterstützen Plattformen entwickelt werden. Bei diesem Vorgehen pro-

grammiert der Entwickler gegen die plattformunabhängigen Schnittstellen des jeweiligen Frameworks.

Grundsätzlich haben beide Ansätze Stärken und Schwächen, welche je nach den Bedürfnissen des Projektes anders zu gewichten sind. Das Ziel dieser Arbeit ist es, die verschiedenen Möglichkeiten zu analysieren und anhand einer Beispiel-Applikation für die Plattformen Android und Windows Phone8 zu implementieren. Auf die Berücksichtigung der iOS-Plattform wird in dieser Arbeit verzichtet. Diese Entscheidung basiert darauf, dass die Thematik „Entwicklung für Android- und iOS-Plattformen“ aus der Sicht des Autors bereits häufig in Fachzeitschriften oder dem Internet diskutiert und dokumentiert wurde[1].

1.1 Projektbeschreibung

Im Zuge einer Kooperation mit der OJAD soll eine Applikation entwickelt werden, welche das Personal beim Dokumentieren von Veranstaltungen unterstützen soll. Bei dem Projekt soll es sich um eine verteilte Anwendung handeln. Die MitarbeiterInnen haben die Möglichkeit Veranstaltungen - für die Sie verantwortlich sind - mit Hilfe ihrer Smartphones zu dokumentieren. [2]

Bei der Dokumentation handelt es sich in erster Linie um das erfassen von statistischen BesucherInnen-Werten wie beispielsweise Altersgruppen und Geschlecht. Diese Werte sollen zentral gespeichert werden und als Grundlage für statistische Auswertungen dienen. [3]
Ein weiterer Aspekt liegt in der Erfassung des Veranstaltungsablaufes. Neben Fotos ist an dieser Stelle auch Platz für Kommentare vorgesehen. Durch die Verwendung der Kommentar-Funktion besteht die Möglichkeit ein persönliches Feedback in die Dokumentation einfließen zu lassen.

[1] Ein Anzeichen dafür zeigt das Ergebnis einer Internet-Suche. Der Term: „android and ios cross platform development“ (14.700.000 Treffer) ist wesentlich häufiger zu finden als: „android and windows phone cross platform development“ (5.270.000 Treffer)

[2] Die Kriterien der unterstützten Endgeräte können dem Abschnitt: Anforderungen entnommen werden.

[3] Das erzeugen oder erstellen der statistischen Auswertung ist nicht Bestandteil dieser Arbeit beziehungsweise des hier dokumentierten Projektes.

1.2 Anforderungen

Ziel dieses Abschnittes ist es, die Rahmenbedingungen für das Projekt zu definieren. Diese Punkte dienen als Grundlage für Entscheidungen welche im späteren Verlauf des Projektes auftreten. Um einen gewissen Spielraum für die folgende State of the Art-Analyse sowie die spätere Entwicklung zu ermöglichen, werden an dieser Stelle die Anforderungen nur auf das Nötigste begrenzt.

Applikationen

Ziel ist es eine Applikation zu entwickeln welche auf den zwei definierten Smartphone-Plattformen lauffähig ist. Auf Wunsch der OJAD wurde definiert, dass die Plattformen Windows Phone 8 sowie Android ab der Version 4.0 - Ice Cream Sandwich(Application Programming Interface (API) 14) verwendet werden sollen. In der ersten Phase der Entwicklung, welche in dieser Arbeit dokumentiert wird, soll ein lauffähiger Prototyp für jede Plattform entstehen. Der Abschnitt: Entwicklungsspezifikationen des 3. Kapitels definiert den Funktionsumfang der Applikation. Dieser Prototyp muss noch nicht den Status der Serienreife erreichen, sondern dient der OJAD als Ausgangspunkt für die weitere Entwicklung beziehungsweise Integration in ein größeres Projekt.

Kommunikation

Für die Kommunikation zwischen dem Endgerät und dem Server soll die Javascript Object Notation (JSON) eingesetzt werden. Dabei handelt es sich um ein kompaktes Datenformat, welches sich in der Kommunikation mit Webservices etabliert hat. Aus Gründen die im Abschnitt: Sicherheit angeführt werden, wird keine Verschlüsselung wie Beispielsweise Secure Sockets Layer (SSL) verwendet.

Webservice

Als Schnittstelle zwischen Server und Applikation soll ein Webservice verwendet werden. Dieser wird nach dem Representational State Transfer (REST)-Konzept realisiert. Durch die Verwendung des REST-ful Webservice soll die Applikation Zugriff auf die vom Server bereitgestellten Funktionen erhalten. Desweiteren soll - für beide Plattformen - der gleiche Webservice genutzt werden.

Server

Da die Infrastruktur der OJAD auf Microsoft-Produkte ausgelegt ist, liegt die Entscheidung nahe, den Server auf diesen Technologien aufzubauen. Für die aktuelle Entwicklung wird eine lokale MS-SQL Datenbank eingesetzt. Als Betriebssystem für den Produktiv-Server wurde ein Windows-Server 2012R2 in der 64-Bit-Version ausgewählt. Dieser verfügt über ein Intel Xeon 2,33 GHz-Prozessor sowie zwei GB-RAM.

Sicherheit

Aus Sicht der OJAD werden durch die aktuell geplante Funktionalität der Applikation keine sensiblen Daten über das Internet übertragen. Deshalb wird - in Absprach mit der OJAD - keine Verschlüsselung für die Kommunikation verwendet. In den anfänglichen Gesprächen mit der OJAD ist der Wunsch aufgekommen, dass ausschließlich die verantwortlichen MitarbeiterInnen eine ihnen zugeteilte Veranstaltung dokumentieren dürfen. Für diesen Zweck wurden die Sicherheitsziele: Authentifizierung (Pin) sowie Autorisierung (nur verantwortliche MitarbeiterInnen dürfen dokumentieren) definiert.

Kapitel 2

State of the Art

Für die Orientierung in dem weitläufigen Gebiet der mobilen Entwicklung ist es notwendig, einen Überblick über die verschiedenen Ansätze zu erhalten. Da für diese Ansätze meist mehrere Frameworks existieren wird anhand der Kriterien(siehe Abschnitt: Kriterien der Analyse) das passende Framework ausgewählt und dessen charakteristischen Eigenschaften vorgestellt. Im folgenden Abschnitt: 2.6 - Auswertung werden die projektbezogenen Vor- und Nachteile der vorgestellten Frameworks kommentiert und die Wahl des umzusetzenden Ansatzes getroffen.

2.1 Kriterien der Analyse

Sicherlich ist es während der Orientierungsphase sinnvoll, einen Blick über den Tellerrand zu wagen. Durch die Definition bestimmter Rahmenbedingungen und einer damit verbundenen eingeschränkten Auswahl soll eine zielgerichtete Analyse ermöglicht werden. Diese Rahmenbedingung orientieren sich zum größten Teil an den Projekt-Anforderungen(siehe Abschnitt 1.2 (Anforderungen)) und sind in folgenden Punkten definiert:

Unterstützte Ziel-Plattformen

Dieser Punkt wurde aus den Projekt-Anforderungen übernommen. Es müssen folgende Plattformen für das Projekt unterstützt werden.

- Android
  - ab der Version 4.0 - Ice Cream Sandwich(API 14)
- Windows Phone
  - ab der Version 8.0

Entwicklungsumgebung

Unter Entwicklungsumgebung - ist in dieser Definition - die für die Entwickeln des Projektes notwendig Infrastruktur gemeint.

- Betriebssystem
  - Die Entwicklung sowie die Wartung der Applikation muss mit einem 64Bit-Windows 8-System durchführbar sein.
- Integrated Development Environment (IDE)
  - Als IDE sollte entweder Visual Studio oder eine Eclipse-basierende IDE verwendet werden.
- Programmiersprachen
  - Als mögliche Programmiersprachen kommen C#, Java oder Javascript in Frage

Kosten

Da es sich bei der OJAD um eine Gemeinnützige Institution mit strikt definierten Ressourcen handelt, sollten die Lizenzkosten für die Evaluierung möglichst gering gehalten werden. Kostenlose- sowie OpenSource-Frameworks werden priorisiert.

Native Installation-Datei

Es muss möglich sein, dass die Applikation als native Installations-Datei vorliegt. Dies ist für die einfache und zügige Verteilung des fertigen Produktes notwendig.

2.2 Hybrider Ansatz

Der Hybride Ansatz verbindet die Entwicklung von Applikationen durch Webtechnologien mit den Vorteilen von nativen Frameworks. Die in Javascript, Hypertext Markup Language (HTML)5 und Cascading Style Sheets (CSS)3 geschriebene Web-Applikation werden mithilfe eines Hybriden-Frameworks in einen nativen Container verpackt. Durch die „Verpackung“ in den Container bieten sich im wesentlichen zwei Vorteile. Ein Vorteil besteht darin, dass der Zugriff auf Hardware, Sensorik sowie Native Funktionen gegeben ist. Die Unterstützung der Zielplattform sowie der Funktionsumfang sind dabei von dem eingesetzten Framework abhängig. Ein weiterer Vorteil besteht darin, dass die Applikation über die Plattform-Stores wie eine Native-App verteilt werden kann.

Adobe PhoneGap

Durch den Einsatz von PhoneGap können Multiplattform-Lösungen auf der Basis von Hybriden-Apps realisiert werden. Nach der Entwicklung steht ein natives App zur Verfügung das - über die jeweiligen Shops der Zielplattform - wie eine Native-Applikation verteilt werden kann. Adobe PhoneGap selbst basiert auf dem OpenSource Framework: Apache Cordova.

Von den hier vorgestellten Frameworks unterstützt PhoneGap die meisten Mobilen-Plattformen. Im Gegensatz zu Xamarin steht allerdings nicht die gesamte Funktionalität der Zielplattform zur Verfügung, sondern ausschließlich definierte Bereiche.

Merkmale

- Entwicklungssprache: Javascript (für die Definition der Grafischen Oberfläche werden HTML und CSS eingesetzt.)
- Kosten: kostenlos
- IDE: wird ohne IDE verteilt
- Unterstützte Plattformen:
  - **Android**
  - **Windows Phone**
  - Amazon Fire OS, BlackBerry 10, Firefox OS, iOS, Ubuntu, Windows 8, Tizen

PhoneGap selbst bietet keine IDE oder Grafische Oberfläche für die Entwicklung an. Sämtliche Operationen wie „neue Projekte“ anlegen oder „Projekte builden“ werden mithilfe der Kommandozeile ausgeführt.
Die Einrichtung der PhoneGap-Umgebung ist - von den hier vorgestellten Frameworks - die aufwendigste Lösung. Neben den benötigten SDK der Zielplattformen muss der Pfad zu dem lokalen ANT[1]-Verzeichnis in dem Systempfad korrekt hinterlegt sein. Desweiteren muss vor der eigentlichen Installation NodeJS lokal verfügbar sein. Mit dem Befehl `npm install -g phonegap` wird die Installation von Phonegap schlussendlich gestartet.

Nachdem das Projekt durch den `Create`-Befehl angelegt wurde, kann es zur weiteren Bearbeitung in die Eclipse-IDE importiert werden. Aus dem PhoneGap-Projekt wird durch den `Build`-Befehl eine Native-Installationsdatei erzeugt.

Ein Vorteil von PhoneGap besteht darin, dass der Code nicht an die unterschiedlichen Plattformen angepasst werden muss. Für die Verwendung von Nativen Funktionen, wird mittels Javascript auf die Bibliotheksfunktionen(PhoneGap Pluins) zugegriffen. Für diesen Zweck verwendet PhoneGap die Cordova-API (vgl. Gifford 2012, Seite 2, sowie; Ross 2013, Abschnitt: PhoneGap oder Apache Cordova)
Diese wrapped die nativen Methoden und bietet einen Zugriff via Javascript an.(Siehe Abb.: 1 (PhoneGap Architektur))

Für die Optimierung sowie Anpassung der grafischen Oberfläche können beliebige Mobile-Webframeworks wie `jQuery Mobile` oder `Sencha Touch` eingesetzt werden. Durch die Verwendung dieser Frameworks ist es möglich, dass die - mit PhoneGap entwickelte - App optisch einer Nativen-Applikation ähnelt.

Neben dem verhältnismäßigen hohen Installationsaufwand[2] besteht ein weiterer Nachteil darin, dass aktuell noch kein hochwertiger Debugger zur Verfügung steht. Somit ist es aktuell noch nicht möglich, den Ablauf des Programmes auf allen unterstützten Endgeräten nachzuverfolgen. Adobe empfiehlt daher in Ihrem Wiki, das Projekt entweder in einem Desktop-Webbrowser [3] oder je

[1] Apache ANT ist ein Programm das, durch eine zentrale Konfigurationsdatei gesteuert, Java-Quellcode kompiliert(Mehr Informationen finden Sie unter: The Apache ANT Project 2012).

[2] Im Vergleich zu den anderen Frameworks der State of the Art-Analyse.

[3] Für das Debugging wird vom Hersteller empfohlen, einen Webbrowser zu verwenden, der auf dem `Webkit`-Engine basiert.

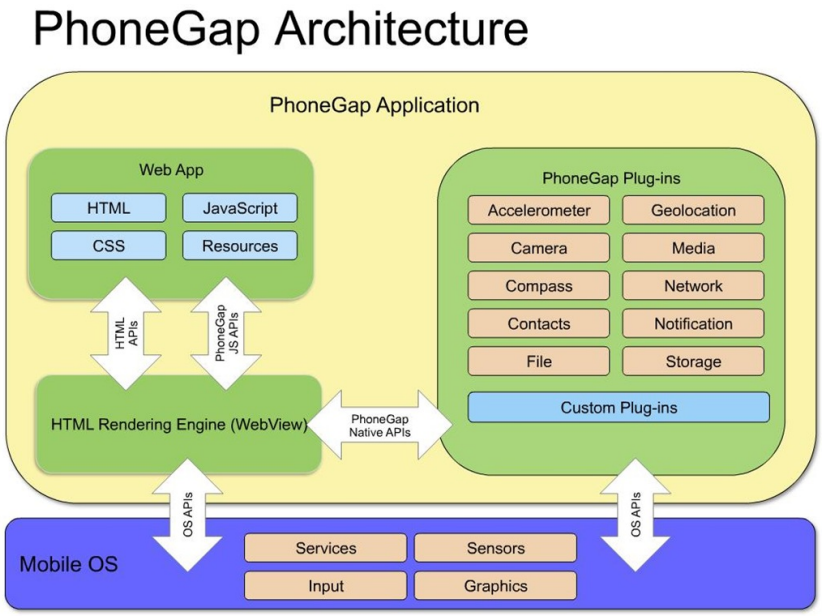

Abbildung 1: PhoneGap Architektur (Quelle: *Basic Steps In PhoneGap app Development* 2013)

nach Zielplattform, eine eigene Systemlösung zu verwenden(vgl. *Debugging in PhoneGap* 2014).

Adobe PhoneGap Build

Seit September 2012 bietet Adobe den cloud-basierten Build-Service PhoneGap Build als Teil der Creative Cloud an(vgl. *PhoneGap Build is Launched* 2012). PhoneGap Build steht zum einen kostenfrei sowie als Abonnement-Model zur Verfügung (vgl. *Adobe PhoneGap Build* 2013, ab Abschnitt: Choose your plan). Durch die Verwendung des Build-Services müssen die Zielplattformen nicht mehr lokal eingerichtet werden. Der Quelltext des Projektes wird entweder per Upload oder Link zu einem Github-Repository für den Build-Vorgang verfügbar gemacht. Die erzeugten nativen Installationsdateien können anschließend heruntergeladen werden.

2.3 Interpretierter Ansatz

Bei dem interpretierten Ansatz wird das Programm nicht in einer Maschinensprache sondern einer Interpretersprache codiert. Der Quellcode wird - zur Laufzeit - von dem Interpreter Zeile für Zeile ausgewertet und verarbeitet. Innerhalb des Framework ist der benötigte Plattform-spezifische Code implementiert. Dabei dient das verwendete Framework als Adapter zwischen dem Quellcode und der Zielplattform. Durch die ausgewerteten Anweisungen leitet der Interpreter die Ausführung der benötigten Operation des Frameworks ein.

Appcelerator Titanium

Appcelerator Titanium ist ein auf dem interpretierten Ansatz basierendes Framework. Wie bei Adobe PhoneGap wird als Entwicklungssprache Javascript eingesetzt. Im Gegensatz dazu muss bei nativen Apps kein HTML verwendet werden. Allerdings ist es möglich, auch Mobile Web Applikationen mit Titanium zu erstellen(siehe Abb.: 3 - Titanium Architektur).

Merkmale

- Entwicklungssprache: Javascript
- Kosten: kostenlos
- IDE: Titanium Studio
- Unterstützte Plattformen:
  - **Android**
  - **Windows Phone** derzeit noch nicht verfügbar, allerdings angekündigt (vgl. *Titanium Support Plans for Windows 8* 2013)
  - Tizen, BlackBerry

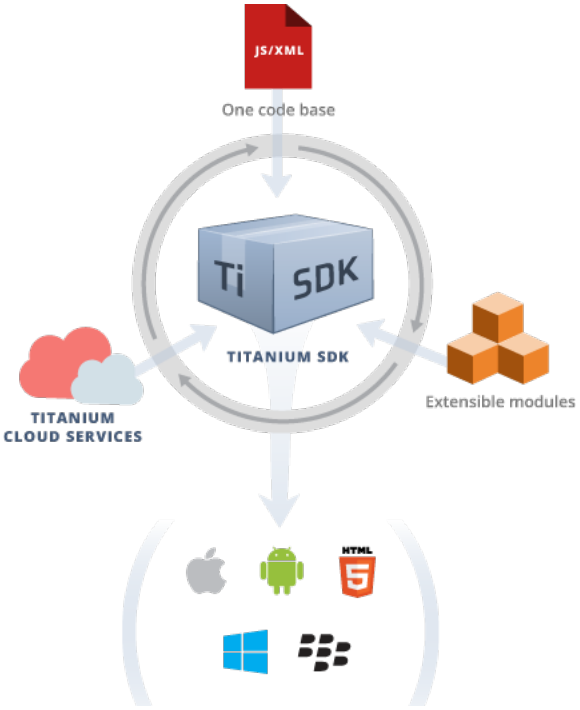

Abbildung 2: Übersicht des Titanium-SDK's (Quelle: *Titanium SDK* 2014)

Titanium SDK

Das Titanium-SDK, welches in der nativen Sprache des Zielsystems entwickelt wurde, kann auf verschiedene Weisen erweitert werden. Neben Marketplace- sowie selbst entwickelten Modulen kann auch auf die Unterstützung von Cloud Services zugegriffen werden(siehe Abb.: 2 - Titanium SDK). Diese Erweiterungen werden direkt über die IDE Titanium Studio in das Projekt eingebunden. Es handelt sich nicht um einen Cross-Compile Vorgang wie er bei dem Xamarin-Framework stattfindet. Stattdessen wurde der Javascript-Quellcode mit dem Titanium-SDK zur Laufzeit durch den Interpreter der Endgerätes ausgeführt.

Titanium Studio

Während der Entwicklung an einem Titanium-Projekt findet die Arbeit innerhalb der IDE Titanium Studio statt. Titanium Studio basiert auf dem für Webentwicklung spezialisierten IDE Aptana Studio. Als Abhängigkeit wird - wie bei dem Adobe PhoneGap-Framework - NodeJS benötigt, allerdings wird dies automatisch bei der Installation bezogen.

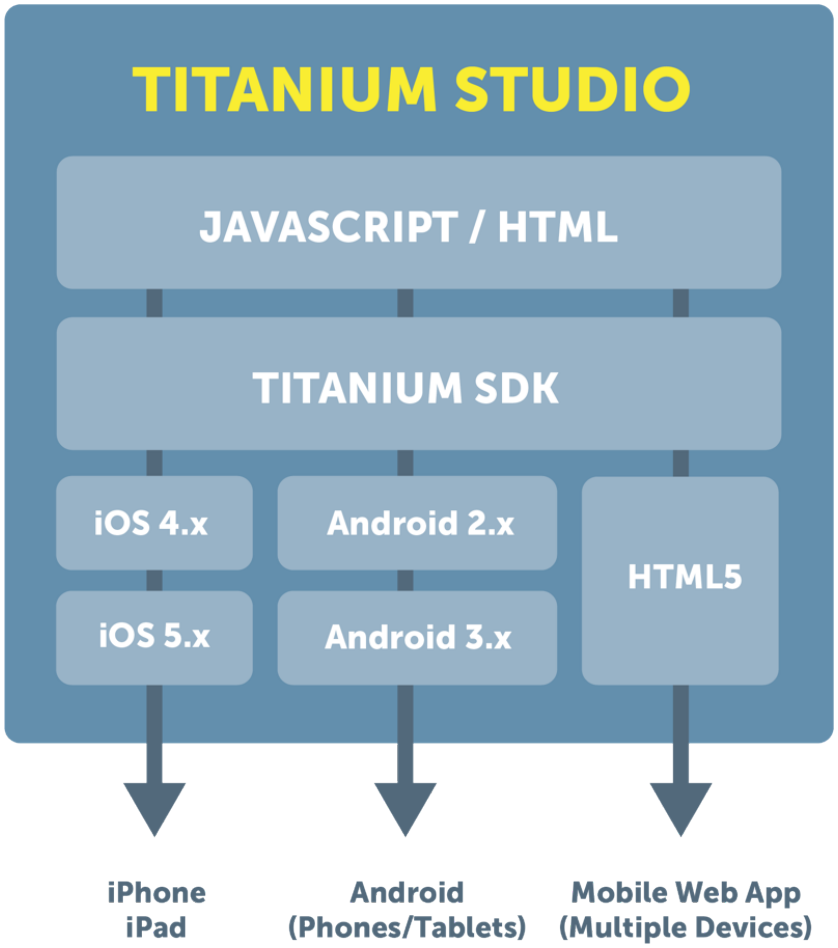

Abbildung 3: Titanium Architektur (Quelle: *Titanium Platform Overview* 2014)

Für den Bezug sowie den Start von Titanium Studio ist ein kostenloser Appcelerator-Benutzeraccount notwendig. Beim ersten Start führt ein Dialog durch die Einrichtung der gewünschten Plattformen. Über den integrierten Marketplace können die im Abschnitt 2.3 (Titanium SDK) erwähnten Module und Cloud Services bezogen werden. Die Integration der SDK-Erweiterungen erfolgt in einer Extensible Markup Language (XML)-basierten Konfigurations-Datei des Projektes. Ein Vorteil gegenüber Adobe PhoneGap besteht in dem von Titanium Studio integrierten Debugger sowie der Unterstützung beim Einrichten der Zielplattformen.

2.4 Übersetzender Ansatz

Dieser Ansatz bezieht eine besondere Position. Im Gegensatz zu den vorhergegangenen Ansätzen wird der Quellcode nicht interpretiert sondern mit Hilfe von Plattform-Bibliotheken in kompilierten Maschinencode - für das jeweilige System - übersetzt.

Xamarin

Bei Xamarin handelt es sich um ein Framework, dass verschiedene Plattformen für die Entwicklung anbietet. Jede dieser Framework-Plattformen sind jeweils für ein Zielsystem wie beispielsweise Android ausgelegt. Durch den Einsatz von Xamarin kann mittels C# auf die gesamte native API zugegriffen werden (vgl. Xamarin Inc. 2014a, S. 10).

Merkmale

- Entwicklungssprache: C#
- Kosten: 0 - 1.899 $
- IDE: Xanmarine Studio (Eclipse) oder Visual Studio
- Unterstützte Plattformen:
  - **Android**
  - Mac
  - **Windows Phone**

Durch die Möglichkeit, die gesamte API des Zielsystemes zu verwenden(siehe Abb.: 4 - Titanium SDK), steht eine mit Xamarin erzeugte Applikation in den Punkten Aussehen sowie Funktionalität einer nativen Applikation in nichts nach. Dadurch ist es möglich, Teile des Quellcodes aus einer bestehenden C#-Applikation plattformübergreifend zu verwenden. Allerdings müssen die systemspezifische Faktoren, wie beispielsweise die grafische Oberfläche, weiterhin für jede Zielplattform separat angepasst werden. Durch den Einsatz des - teilweise kostenpflichtigen - Component Store lässt sich das eigene Programm mit zusätzlichen Modulen und Bibliotheken wie beispielsweise Azure Mobile Services erweitern.

Abbildung 4: Xamarin Plattform (vgl. Xamarin Inc. 2014a, S. 8; Grafik wurde durch Autor angepasst)

Das Manko liegt in den verhältnismäßig hohen Lizenzkosten von Xamarin[4]. Je nach gewähltem Finanzierungsplan fallen zwischen 0 - 1.899 $ pro Jahr, Entwickler und Plattform an (vgl. Xamarin Inc. 2014b). Es ist zwar eine kostenlose Option verfügbar, allerdings ist diese stark limitiert.[5] Somit muss mit jährlichen Kosten (pro EntwicklerInn) von mindestens 299$, beziehungsweise 999$ wenn die Visual Studio-Unterstützung gewünscht ist, gerechnet werden.

[4]Diese Aussage bezieht sich auf die Kosten von Xamarin im Vergleich zu den alternativen Frameworks.

[5]Eine Restriktion besteht beispielsweise in der maximalen Größe des Quellcodes. (vgl. Xamarin Inc. 2014c)

2.5 Nativer Ansatz

Bei dem nativen Ansatz muss die Applikation für jede Plattform eigens implementiert werden. Dafür wird jeweils das native SDK verwendet. Aufgrund der Spezifikation (siehe Abschnitt: 2.1 - Unterstützte Ziel-Plattformen) werden an dieser Stelle die beiden nativen SDK's vorgestellt.

Android Entwicklung

Der Einstieg in die Android-Entwicklung ist denkbar einfach und ohne finanzielle Aufwände durchführbar. Für die Einrichtung der Entwicklungsumgebung ist es ausreichend, das Eclipse ADT bundle herunter zu laden und zu entpacken.

Merkmale

- Entwicklungssprache: Java
  - C/C++ (Android Native Development Kit (NDK))
- Kosten: kostenlos
- IDE: Eclipse Android Developer Tools (ADT) oder Android Studio
- Unterstützte Plattformen:
  - **Android**

Darin ist neben der - für Android angepassten - Eclipse-IDE das SDK sowie ein Abbild für den Android Emulator enthalten (siehe Android Developer 2014e). Dies ist alles was für den Start notwendig ist. Der Einstieg in die Entwicklung soll durch das auf der Android-Entwicklerseite bereitgestellte Training erleichtert werden (siehe Android Developer 2014f). Anhand der Entwicklung einer Beispiel-Applikation werden die verschiedenen Elemente der Android-API erläutert. Für weiterführende Fragen befinden sich auf der Android-Entwicklerseite die API-Guides, sowie die - mit Beispielen dokumentierten - API-Referenzen (vgl. Android Developer 2014a; sowie Android Developer 2014b). EntwicklerInnen, die ihre Applikation über den Play-Store verteilen möchten, benötigen dafür eine Entwicklerlizenz. Dies ist durch eine Registrierung über die „Google Play-Developer Console“ möglich. Für die Registrierung fällt eine einmalige Gebühr in der Höhe von 25$ an (vgl. Android Developer 2014d)

Windows Phone Entwicklung

Der Download der „Developer Tools“ im „Windows Phone Dev Center“ ist der erste Schritt zur Windows Phone Entwicklung. In dem Software-Paket ist neben der IDE: Visual Studio auch das benötigte Windows Phone SDK enthalten. Des weiteren kann von dem „Windows Phone Dev Center“ der Windows Phone Emulator bezogen werden.

Merkmale

- Entwicklungssprache: C#
- Kosten: kostenlos
- IDE: Visual Studio
- Unterstützte Plattformen:
  - **Windows Phone**

Um den Einstieg zu erleichtern bietet Microsoft auf der hauseigenen Bildungsplattform Microsoft Virtual Academy (MVA) Lernvideos und Material für die Windows Phone Entwicklung an. Der Vorteil der Windows Phone Entwicklung liegt in der Verfügbarkeit des `.NET`-Frameworks. Dadurch haben die Entwickler Zugriff auf Techniken wie Databinding, Extensible Application Markup Language (XAML) und das Model View ViewModel (MVVM)-Konzept, was die Entwicklung und Implementierung unterstützen soll.

Für das Ausführen und Testen der Applikation werden zwei Optionen angeboten. Zum einen kann das Programm, direkt aus Visual Studio, auf das Windows Phone 8-Gerät kopiert und im Debug-Modus gestartet werden. Für diesen Vorgang muss das Endgerät allerdings registriert sein. Um diese Registrierung durchführen zu können, benötigt der/die EntwicklerIn wiederum Zugriff auf einen aktiven „developer account“, der mit jährlichen Kosten verbunden ist (vgl. Microsoft MSDN 2014a). Zum anderen kann das Programm, ebenfalls im Debug-Modus, über Visual Studio auf dem Emulator ausgeführt werden. Für diese Option wird kein „developer account“ benötigt. Allerdings verfügt der Emulator über relativ hohe Systemanforderung. Diese sind - unter anderen - Hyper-V, Second Level Address Translation (SLAT), mindestens Windows 8 in der 64-Bit Pro-Edition und vier Gigabyte (GB) Random-Access Memory (RAM) (vgl. Microsoft MSDN 2014d).

2.6 Auswertung

Nach der Analyse der verschiedenen Frameworks soll an dieser Stelle die Auswahl des - im Projekt eingesetzten - Frameworks getroffen und begründet werden. Die Aufzählungen der Begründung findet in der selben Reihenfolge wie die Analyse statt und soll keine Reihung darstellen. Die Begründungen für diese Entscheidungen sind nicht im Allgemeinen (für jedes Projekt) gültig, sondern basieren - in erster Linie - auf den Projekt-Anforderungen des 1. Kapitels, sowie den - davon abgeleiteten und erweiterten - Kriterien der Analyse des 2. Kapitels.

Adobe PhoneGap

Adobe PhoneGap beeindruckt durch die schnelle und komfortable Entwicklung mittels Webtechnologien, sowie dem Vorteil, dass der Code nur einmal implementiert wird und nicht auf die verschiedenen Plattformen angepasst werden muss. Die geringeren Probleme mit der Einrichtung der Entwicklungsumgebung (Abhängigkeiten) können mit der Verwendung von Adobe PhoneGap Build umgangen werden. Adobe PhoneGap ist in der engeren Auswahl und wird - bei zukünftigen Projekten -als Kandidat in Betracht gezogen.

Appcelerator Titanium

Die Vorzüge der komfortablen IDE mit dem inkludierten Debugger sowie die einfache Erweiterbarkeit des Titanium SDK zeichnen Appcelerator Titanium aus. Leider scheidet es aufgrund des Punktes: Unterstützte Ziel-Plattformen aus der Definition: Kriterien der Analyse des 2. Kapitels, aus. Es ist zwar die Unterstützung für Windows Phone 8 angemeldet (siehe: *Titanium Support Plans for Windows 8* 2013), allerdings ist diese - zum aktuellen Zeitpunkt - noch nicht verfügbar.

Xamarin

Der Hauptvorteil von Xamarinliegt in der Wiederverwendbarkeit von existierenden C#-Codes. Dieser Vorteil kommt in dem aktuellen Projekt nicht zum Tragen, da kein bestehender Code existiert. Nichtsdestotrotz muss die Oberfläche für jede Applikation separat entwickelt werden. Allerdings kann der plattformunabhängige Teil (Logik und Kommunikation) der Windows Phone Applikation im Android-Projekt wiederverwendet werden, wenn Xamarin

verwendet wird. Dennoch rechtfertigen die Vorteile von Xamarin nicht die notwendige Investition[6] (siehe: Xamarin Inc. 2014b).

Nativer Ansatz

Der native Ansatz bietet den Entwicklern die Freiheit, dass gesamte SDK der Plattform zu nutzen. Neben den Vorteilen der nativen Oberflächen Gestaltung sowie der Performance entfallen Abhängigkeiten gegenüber Dritten (Frameworks). Der Einsatz von nativen Oberflächen der jeweiligen Plattformen ist in diesem Projekt aufgrund der definierten Ziele der Gestaltung (siehe Abschnitt: 3.2 - Ziele der Gestaltung) notwendig. An dieser Stelle ist anzumerken, dass die zuvor getroffene Aussage bezüglich der Performance nicht generell gültig ist. Durch Fehlentscheidungen in der Entwicklung sowie der Implementierung, kann die Performance einer nativen Applikation stark abfallen.

Es ist möglich, dass der Einsatz von Abhängigkeiten (Frameworks) - in gewissen Fällen - zu Problemen führen kann. Daher gilt es unter anderem zu beachten, inwiefern die Frameworks gegenüber neuen Plattform-Updates kompatibel sind. Diese Thematik spielt dann eine Rolle, wenn das Projekt in der Zukunft mit einem neuen Feature erweitert werden soll.

Trotz des Nachteils, der fehlenden Codebasis für beide Ziel-Plattformen, fällt die Entscheidung auf Grund der erwähnten Vor- und Nachteile zu Gunsten des **nativen Ansatzes** aus[7].

[6] Diese Aussage betrifft die Meinung des Autors zu dem vorliegenden Projekt.

[7] Ein weiterer Grund für die Entscheidung, ist der Wunsch einer nativen App von Seiten der OJAD

Kapitel 3

Entwicklung

Nach Abschluss der State of the Art-Analyse wurde in der Auswertung die Entscheidung getroffen, den nativen Ansatz für das Projekt zu wählen. Die Entwicklung ist in die drei Abschnitte: Entwicklungsspezifikationen, Design-Entwurf und Architektur aufgeteilt. In der ersten Phase der Entwicklung werden die Entwicklungsspezifikationen festgelegt. Diese definieren, welche Interaktionen zwischen dem Benutzer und dem System möglich sind und wie das System auf den Input reagiert. In der zweiten Phase des Projektes: Design-Entwurf, werden Entscheidungen bezüglich der Benutzeroberfläche getroffen. Anhand dieser Entscheidungen sowie derEntwicklungsspezifikationen werden die ersten Mock-ups erstellt. Die dritte Phase: Architektur bildet den Abschluss diese Kapitels. Darin befinden sich Entscheidungen bezüglich des Aufbaues der Software-Architektur. Dabei bildet die Architektur das Fundament für die anschließende Implementierung, welche in Kapitel 4 besprochen wird.

3.1 Entwicklungsspezifikationen

In den Entwicklungsspezifikationen wird zum einen die Funktionalität des Programmes festgelegt und zum anderen definiert, wie sich das System in den jeweiligen Zuständen beziehungsweise Situationen verhalten soll. Die Entwicklungsspezifikationen sind plattformunabhängig und beschreiben in erster Linie das: „WIE“ (...etwas abläuft). Jede Entwicklungsspezifikation bildet ein definiertes Szenario des Programmablaufes ab. Anhand des Szenarios `Login` wird später (siehe 4. Kapitel - Implementierung) die Architektur des Projektes erläutert. Innerhalb eines Szenarios werden die Elemente Ablauf, Vor- und Nachbedingung sowie Auslöser definiert. Der Auslöser gibt an, durch welches Verhalten oder ausgeführte Aktion ein spezifiziertes Szenario

ausgelöst wird. Auslöser können von Eingaben der Benutzer sowie systeminternen Vorkommnissen stammen. Dies kann unter anderem auch der erfolgreiche Abschluss eines anderen Szenarios sein. Bei dem Start (Vorbedingung) sowie dem Ende (Nachbedingung) eines Szenarios muss sich das System in einem konsistenten Zustand befinden.

Login-Szenario

Durch einen Klick wird die Eingabe der Zugangsdaten abgeschlossen und das Login-Szenario gestartet. Dabei werden die Daten aus der Eingabemaske ausgelesen, wobei das Passwort in ein nicht-Klartext-Format umgewandelt wird. Ist dieser Vorgang erfolgreich abgeschlossen, wird der Server für die Authentifizierung kontaktiert.

Sollten die Prüfung der Daten auf der Serverseite positiv verlaufen, so wird für den Client ein Identitätsnachweis erzeugt. Dieser Identitätsnachweis ist mit einer Gültigkeitsdauer versehen und enthält keine Information über den Benutzername oder das Password. Anschließend legt der Server den Identitätsnachweis ab und sendet eine Kopie an den Client.

Sollte die Prüfung der Daten negativ sein, so übermittelt der Server eine entsprechende Nachricht an den Client. Im Fall einer negativen Rückmeldung informiert das System die Benutzer und bricht das Szenario ab. Andernfalls wird, nach Erhalt des Identitätsnachweis, die Startansicht erzeugt, welche wiederum ein eigenes Szenario darstellt.

3.2 Design-Entwurf

Dieser Abschnitt beschäftigt sich mit dem Design und der Modellierung der Benutzeroberfläche. Da die Nutzung des nativen Ansatzes gewählt wurde, sollte sich der Entwurf an den Referenzen der jeweiligen Plattform orientieren. In dem Abschnitt Ziele der Gestaltung wurde erläutert, inwiefern die Usability für den Erfolg des Programmes verantwortlich ist. Aus den daraus gewonnen Erkenntnissen wurden drei Ziele für die Gestaltung der Benutzerschnittstelle festgelegt. Um diese weitgehendst zu erfüllen, ist in dem Absatz: Mock-Ups - Prototyp Entwicklung beschrieben, wie ein `Paper Prototyping`-Prozess zur Steigerung der Usability durchgeführt werden kann.

Ziele der Gestaltung

Um die Ziele der Gestaltung definieren zu können muss zuerst analysiert werden, wie die Zielgruppe ihre Wünsche und Erwartungen an die Applikation definiert.
Wie in der Einführung erwähnt wurde, besteht die Hauptaufgabe der Applikation in der Dokumentation von Veranstaltungen durch die verantwortlichen BetreuerInnen. Um das bestmögliche Ergebnis zu erzielen, sollte die Dokumentation während oder kurz nach der Veranstaltung getätigt werden. Um einen Akzeptanzverlust der Applikation vorzubeugen, muss die Dokumentation schnell und einfach durchgeführt werden können. Wenn dies nicht möglich ist, besteht die Gefahr der Rückfälligkeit zu den beschriebenen „alten Gewohnheiten“. Daraus kann geschlussfolgert werden, dass der Erfolg im Praxistest von der Usability des Programmes abhängt. Auf der Basis dieser Erkenntnis werden die folgenden Punkte als Grundlage für den weiteren Design-Prozess angenommen:

Ziel: Übersichtlichkeit

Eine klare und aufgeräumte Oberfläche ist notwendig, um die Unterstützung der Benutzer zu maximieren. Dazu zählt das Aufteilen der Inhalte in sinnvolle und logische Gruppen unter der Berücksichtigung der Arbeitsabläufe. Des weiteren wird auf die Integration eines Corporate Identity-Design in der Oberfläche des Programmes verzichtet.

Ziel: Vertraute Umgebung

Durch das Erzeugen einer vertrauten Umgebung sollen die Nutzer bekannte Elemente ihrer Plattform in der Applikation wiedererkennen. Dadurch sollen die Einarbeitungszeiten in die Applikation minimiert, sowie der Wiedererkennungswert von plattformtypischen Elementen wie beispielsweise die Navigation durch das Programm, maximiert werden. Es wird versucht, dass die Benutzer jeder Zeit das Gefühl haben, dass sie das Programm vollständig kontrollieren. Dadurch soll den Nutzern die Angst genommen werden „etwas falsches zu machen“. Für die Realisierung dieses Zieles muss sich die Applikation an dem Standartverhalten der jeweiligen Plattform orientieren.

Mock-Ups - Prototyp Entwicklung

Um schon in einem frühen Stadium des Entwicklungsprozesses Rückmeldungen über die Bedienbarkeit des Programmes zu erhalten, empfiehlt es sich, in die Entwicklung von Mock-ups zu investieren. Unter Mock-Ups versteht man einen nicht funktionellen Prototypen. Dies bedeutet, dass der Prototyp nicht programmiert, sondern mit einem Bildbearbeitungsprogramm erzeugt wird. Dabei wird für jede Ansicht ein eigener Prototyp „gezeichnet“. Anschließend werden mit Personen aus der Zielgruppe die einzelnen Szenarios der Applikation simuliert. Während des Testes sollen die NutzerInnen den Ablauf des Programmes, die erwarteten ihrer Ergebnis von Interaktionen sowie ihre Emotionen so detailliert wie möglich kommentieren. Mithilfe des Testes lässt sich feststellen, wie schlüssig die Arbeitsabläufe sind, ob die Funktionalität der Oberfläche verständlich ist und ob die Reaktionen des Programmes mit den Erwartungen der Benutzer übereinstimmen. Der Prozess des `Paper Prototyping` sollte nicht einmalig, sondern in Form eines iterativen Entwicklungsprozess durchgeführt werden. Dies bedeutet, dass auf der Basis der erhaltenen Rückmeldungen die Mock-Ups optimiert und die Test wiederholt werden. Der Test kann als positiv betrachtet werden, wenn die Rückmeldungen der Probanden mit den zuvor definierten Design-Zielen (siehe Abschnitt: Ziele der Gestaltung) übereinstimmen.

Die Mock-Ups der Beispiel-Applikationen befinden sich im Anhang: Diagramme und Bilder unter dem Abschnitt: Mock-Ups.

3.3 Architektur

Plattformunabhängig

Dieser Abschnitt beschäftigt sich mit der Frage, welche Teile des Projektes plattformunabhängig entwickelt werden können. Durch die Maximierung des unabhängigen Anteils verringert sich die Entwicklung- und Implementierungs-Dauer. Desweiteren wird hierdurch - nach Abschluss der Projektphase - die Wartung und Erweiterbarkeit deutlich erleichtert.

Webserver

Die Entwicklung des Webserver als Backend-System ist vollständig losgelöst von den eingesetzten Plattformen, da die Applikationen keinen direkten Zugriff auf das Backend haben. Für die Kommunikation zwischen Applikation und Backend wird ausschließlich der Webservice verwendet[1]. Dabei dient der Webserver als Umgebung in die der Webservice eingebettet ist. Innerhalb dieser Infrastruktur werden zum einen Stammdaten und zum anderen Erfassungsdaten verwaltet. Bei den Stammdaten handelt es sich um fixe Daten, die nicht regelmäßig geändert werden. Die Erfassungsdaten, wie Veranstaltungen, Besucher-Daten und eingeteilte Mitarbeiter, sind so ausgelegt, dass diese nicht durch unterschiedliche AnwenderInnen veränderbar sind. Diese Daten können ausschließlich durch die verantwortlichen Mitarbeiter verändert werden. Dadurch ist sichergestellt, dass die Daten nicht von verschiedenen Anwendern überschrieben werden können. Des weiteren stellt der Webserver dem Webservice eine Fassade für die Durchführung von Datenbank-Operationen zur Verfügung. Durch dieses Vorgehen ist der Webservice weitgehendst von dem Webserver entkoppelt. Somit kann der Webserver erweitert werden, ohne dass der Webservice verändert werden muss. Dies ist notwendig, da geplant ist, die Daten des Webserver durch einen Microsoft Exchange-Server zu aktualisieren[2].

Webservice

Der Webservice dient als Schnittstelle zwischen dem Client und dem Webserver und orientiert sich an dem REST-Paradigma. Dabei kann jede verfügbare Operation des Webservice über ihre eigene Uniform Resource Locator (URL)-Adresse erreicht werden. Als Datenformat für die Kommunikation wird JSON

[1] Die Funktionsweise des Webservice wird im Abschnitt: Webservice erklärt.

[2] Dieses Feature ist allerdings nicht Bestandteil des aktuellen Projektes sondern soll nur die Notwendigkeit der Erweiterbarkeit demonstrieren.

eingesetzt. Da der Webservice Zustandslos ist, werden serverseitig keine Verbindungsdaten dauerhaft gehalten. Dies bedeutet, dass der Client bei jeder Anfrage (request) alle benötigten Daten mitsenden muss. Die eingehenden Daten werden Mithilfe des `StringToObjectParser` in die gewünschten Objekte übersetzt. Anschließend wird, je nach kontaktierter URL, die entsprechende Operation gestartet und die erzeugten Objekte für die Bearbeitung übergeben. Nachdem die Operation fertiggestellt ist wird das Ergebnis durch den `JSON-Parser` in eine Zeichenkette umgewandelt. Die erzeugte Nachricht wird anschließend an den Client returniert.

Durch den Einsatz eines REST-ful Webservice kann eine leicht erweiterbare und plattformunabhängig Anbindung an das Backend sichergestellt werden.

Plattformabhängig

Da die beiden Systeme Windows Phone und Android nicht kompatible Programmiersprachen einsetzen, ist es notwendig, dass für jede Plattform eine eigenständige Applikation entwickelt wird. Die Entwicklung einer einheitliche Architektur ist auf Grund der unterschiedlichen Entwicklungs-Konzepte beziehungsweise der einzelnen Plattformen[3] nur bis zu einem gewissen Grad realisierbar. Allerdings kann definiert werden, dass sich beide Applikation die Logik der einzelnen Szenarios in den „Usecase-Controllern“ teilen können. Diese werden von einem zentralen Applikationsservice ausgelöst und sind so von der Grafischen Oberfläche entkoppelt. Für Operationen von zeitintensiven Aufgaben, wie beispielsweise Datenbank- oder Webservice-Zugriffen, werden „asynchrone Tasks“ eingesetzt. Diese Tasks laufen nicht innerhalb des Hauptthreads, wodurch das Blockieren der Benutzeroberfläche vermieden wird. Für die lokale Datenbank soll SQLite als Datenbank-System eingesetzt werden. Die Aufgabe der lokalen Datenbank besteht darin, die vom Server geladenen Daten für die spätere Verwendung aufzubewahren. Des weiteren werden dokumentierte Veranstaltungen, welche nicht erfolgreich an den Server übermittelt wurden, in der lokalen Datenbank gesichert.

[3] Beispielsweise das MVVM- und Databinding-Konzept von Windows Phone

Kapitel 4

Implementierung

Ziel dieses Kapitel ist es, einen Überblick über die Implementierung der verschiedenen Plattformen zu geben. Hierzu wurde das Kapitel in zwei Abschnitte unterteilt, die stellvertretend für die zwei Subsysteme Android Applikation und Windows Phone Applikation stehen, aus welchen sich das Projekt zusammensetzt. Anhand des definierten Szenarios Login (siehe Abschnitt: 3.1 - Entwicklungsspezifikationen des 3. Kapitels) soll die Implementierung durch die einzelnen Schichten der Applikationen aufgezeigt werden. Des weiteren werden in den Abschnitten 4.1 - Android Applikation und 4.2 - Windows Phone Applikation die plattformspezifischen Besonderheiten der grafischen Oberfläche behandelt.

4.1 Android Applikation

Grafische Oberfläche

Android bietet über das SDK ein reichhaltiges Angebot für die Gestaltung des UI an. Durch die kontinuierlichen Erweiterungen der UI-Bibliotheken werden regelmäßig neue Konzepte (siehe Abschnitt Fragments) sowie Features vorgestellt. Diese sind neben weiteren Änderungen Bestandteil der jeweiligen API-Version des SDK. Das Mindest- sowie das Ziel-API-Level werden im Manifest[1] des Projektes festgelegt. Dabei gibt das Ziel-API-Level an, für welche Android Version die Applikation optimiert ist. Dagegen definiert das Mindest-API-Level die älteste Android Version, unter welcher die Applikation vollständig lauffähig ist. Je nach gewählter Spezifikation wird die Auswahl der möglichen Optionen durch das API-Level eingeschränkt. Des-

[1] Das Manifest ist eine XML-Datei in welcher die Rahmenbedingungen der App definiert werden.

halb gilt es abzuwägen, ob eher die `breite Masse` (niedriges API-Level) erreicht werden soll, oder ob der Fokus auf der Verwendung von modernen Gestaltungselemente (hoher API-Level) liegt. Des weiteren wird für die UI-Entwicklung eine Support-Bibliothek durch das Android-SDK zur Verfügung gestellt. Mithilfe derer ist es möglich, eine Abwärtskompatibilität bis zu API-Level 4 (Android 1.6) sicherzustellen. (vgl.: Android Developer 2014i)

Activities

Eine Activity ist eine in sich geschlossene Komponente, welche eine grafische Oberfläche an die Applikation anbindet. Eine Activity besteht aus einer Java- sowie XML-Datei. Zum einen erfolgt in der Java-Datei die Anbindung der Activity an die Applikation. Zum anderen kann auf die einzelnen UI-Elemente zugegriffen und deren Verhalten definiert werden. Die XML-Datei dient der Gestaltung der grafischen Benutzeroberfläche. Neben der Spezifikation der anzuzeigenden UI-Elementen können an dieser Stelle auch Designeinstellungen vorgenommen werden. Die XML-Datei kann entweder mit einem Text-Editor oder mithilfe eines UI-Designers der jeweiligen IDE erzeugt und bearbeitet werden.

In der Beispiel-Applikation wurde für jedes Szenario (siehe Abschnitt: Entwicklungsspezifikationen des 3. Kapitels) eine eigene Activity erstellt.[2]. Damit ergibt sich ein modularer Aufbau, wodurch der Funktionsumfang der Applikation einfach erweitert werden kann.

Fragments

Das Konzept der Fragments wurde offiziell mit dem Erscheinen von Android3 in das SDK aufgenommen. Durch den Einsatz der `Support Library` ist es möglich, Fragments auch in älteren Android-Versionen einzusetzen. Mithilfe von Fragments lässt sich der Inhalt beziehungsweise das Verhalten in einzelne Komponenten aufteilen. Dadurch besteht die Möglichkeit, eigenständige und wiederverwertbare Inhalte zu erzeugen. Da Fragments die Teilmenge einer Activity bilden, sind sie neben ihrem eigenen `Lifecycle` auch von dem `Lifecycle` der jeweiligen Activity abhängig.

[2]Details zur Verwendung von Activities in der Beispiel-Applikation befinden sich in dem Abschnitt: Login Szenario - Android Applikation des 4. Kapitels

Für die Kommunikation zwischen verschiedenen Fragments ist es zu empfehlen, einen `Callback`-Mechanismus in die übergeordnete Activity zu implementieren. (vgl.: Meier 2012, Seite 96: Fundamental Android UI Designs; sowie: Android Developer 2014c)

In der Beispiel-Anwendung wurden in jeder Activity Fragments eingesetzt. Der Sinn von Fragments wird an folgenden Beispiel klarer ersichtlich.

Nach dem erfolgreichen Login (Activity) wird in der Beispiel-Applikation der Startbildschirm (Activity) aufgebaut, dieser wird durch zwei Fragments dargestellt. Hierbei werden in dem ersten Fragment die offenen- und im zweiten Fragment die bearbeiteten Veranstaltungen angezeigt (siehe Abb.: 8 - Startbildschirm - Android (Mock-Ups) im Anhang: Diagramme und Bilder). Zwischen den beiden Fragments kann mit einer Wischbewegung gewechselt oder eine zu bearbeitende Veranstaltung ausgewählt werden.

Login Szenario - Android Applikation

In diesem Abschnitt soll die Implementierung des Login-Szenarios (siehe Abschnitt: Entwicklungsspezifikationen des 3. Kapitels) für die Android-Applikation demonstriert werden. Zum einen werden dabei die verwendeten Android-Werkzeuge und zum anderen die implementierte Architektur beschrieben.

Service

Grundsätzlich gibt es zwei Arten von Services in Android: `Local`- und `Remote`-Services. Der `Local`-Service kann wiederum in die zwei Unterkategorien `Started` und `Bound` eingeteilt werden. Ein `Started`-Service wird explizit von der Anwendung gestartet und beendet[3], während ein `Bound`-Service nur ausgeführt wird solang eine Komponente an ihn gebunden ist (siehe Abb.: 12 - Service Lifecycle im Anhang: Diagramme und Bilder). Der Vorteil eines `Remote`-Service liegt darin, dass fremde Android-Applikation auch auf diesen zugreifen können. (vgl.: Android Developer 2014h; sowie: Satya Komatineni 2012, ab Seite: 409ff)

Innerhalb der Beispiel-Applikation wird ein `Bound`-Service für Datebank- und Webservice-Zugriffe eingesetzt. Nach dem Starten der Login-Activity bindet sich die Activity mithilfe des bereitgestellten Interfaces auf den `EventDokuService`. Sobald der Sende-Button gedrückt wurde, ruft die

[3]Das Verhalten des Start- und Endzeitpunkt lassen sich durch Parameter anpassen.

Login-Activity die Methode `login` des Services auf und übergibt diesem die entsprechenden Daten. Der Service gibt darauf hin die Daten an den Login-Usecase-Controller weiter. Dieser bearbeitet die Anfrage und leitet, nach der Fertigstellung, das Ergebnis direkt an die Login-Activity weiter.

Usecase-Controller

Ein Usecase-Controller implementiert die Logik und den Ablauf eines definierten Szenarios (siehe Abschnitt: Entwicklungsspezifikationen des 3. Kapitels). Der Ablauf des Usecase-Controller wird über eine Fassade von dem Service gestartet. Dabei wird in der Fassade jeder Controller nach dem `Singleton`-Muster gehalten. Zu Beginn des Ablaufes eines Usecase-Controllers wird ein Container-Objekt mit den erhaltenen Daten erzeugt. Dieser Container wird, solange kein Fehler auftritt, durch die Liste der abzuarbeitenden Aufgaben gereicht. Bei jedem Schritt (Aufgabe) werden entsprechenden Informationen dem Container hinzugefügt oder abgeändert. Das Schema des Usecase-Controller ist Bestandteil der entwickelten Architektur und nicht des Android-SDK's.

Im Falle des Login-Szenarios verfügt der Controller über zwei Aufgaben, welche asynchron ausgeführt werden. Im ersten Schritt `getRandomNumber` wird eine Zufallszahl über den Webservice von dem Server angefordert. Diese Zahl ist für das `hashen` des Passworts notwendig, welches wiederum im nächsten Schritt benötigt wird. Nach Erfolg des ersten Schrittes wird der zweite eingeleitet. Dabei wird wiederholt auf den Webservice zugegriffen. Zusätzlich werden diesmal die benötigten Zugangsdaten mitgesendet um sich am Server zu authentifizieren. Sollte die Authentifikation erfolgreich verlaufen sein, so wird der retourniert Token in dem Laufzeitspeicher der Applikation abgelegt. Die Login-Activity wird anschließend durch ein `Callback-Interface` über den positiven- oder negativen Verlauf des Szenarios informiert und kann dementsprechend fortfahren.

Asynchrone Tasks

Da ein Service standardmäßig im `Haupt-Thread` läuft, wird während der Arbeiten im Backend das UI der Applikation blockiert. Dies macht sich vor allem bei Zugriffen auf externe Informationen, wie beispielsweise dem Webservice oder die Datenbank, bemerkbar. Neben der schlechten Usability liegt ein weiteres Problem darin, dass bei zu lange gesperrten Applikationen das Betriebssystem einen Application Not Responding (ANR)-Dialog erzeugt. (vgl. Android Developer 2014g) Um dieser Problematik vorzubeugen bietet das

SDK verschiedenen Lösungen an. Innerhalb der Beispiel-Applikation wurden `AsyncTask`'s für die einzelnen Aufgaben der Controller (siehe Abschnitt: Usecase-Controller eingesetzt. Dadurch ist es möglich, zeitintensive Aufgaben in Hintergrund-Operationen auszulagern, ohne dabei die Applikation zu blockieren.

4.2 Windows Phone Applikation

In diesem Abschnitt wird die Implementierung der Beispiel-Applikation Event-Doku als Windows Phone (WP)8 App betrachtet. Bei der Entwicklung für die WP-Plattform ist der Einsatz von Visual Studio als IDE zu empfehlen. An dieser Stelle sollte erwähnt werden, dass das Einrichten der Entwicklungsumgebung nicht ganz problemlos von der Hand geht. Die Problematik liegt in dem Ausführen sowie Testen der Applikation. Grundsätzlich kann die Entwicklung im Emulator oder auf einem WP8-Gerät ausgeführt werden. Allerdings gibt es bei beiden Möglichkeiten gewisse Hürden, welche in dem Abschnitt: Windows Phone Entwicklung des 2. Kapitels (State of the Art) genauer betrachtet werden.

Grafische Oberfläche

EntwicklerInnen, die über Erfahrung mit dem Windows Presentation Foundation (WPF)-Framework verfügen, werden sich in der WP-Entwicklung schnell zurecht finden. Durch die Nähe zum `.NET`-Framework kann für die Implementierung einer MVVM-Architektur auch auf bekannte Konzepte, wie zum Beispiel `Databinding`, zurück gegriffen werden. Wie auch in der Android-Entwicklung stehen für die Erzeugung eines neuen WP-Projektes bestehende Templates bereit, die als Grundgerüst einer Applikationen verwendet werden können.

Application Page

Die `PhoneApplicationPage` stellt einen wiederverwendbaren UI-Container dar. Eine WP-Application Page kann als Äquivalent zu den Android-Activities betrachtet werden (vgl.: Microsoft MSDN 2014e). Ähnlich wie bei einer Android-Anwendung besteht die grafische Oberfläche einer WP8-Applikation aus einer Design- sowie einer Code-basierten Datei. Um das Design zu definieren kann die XAML-Beschreibungssprache verwendet werden. Für diesen Zweck ist es möglich, den integrierten Visual Studio-Designer oder Microsoft Blend einzusetzen. Für die Anbindung des UI an die Applikation wird die in `C#`

geschriebene `Codebehind`-Datei verwendet.

In der Beispiel-Applikation wurde für jedes Szenario (siehe Abschnitt: Entwicklungsspezifikationen des 3. Kapitels) eine eigene Application Page erstellt. Parallel dazu wurde für jede Application Page ein eigenes `ViewModel` angelegt, das sich die Daten für Darstellung hält und aufbereitet.

Panorama Items

Durch die Panorama Items wird der Inhalt einer Application Page in vertikal angeordnete Seiten aufgeteilt (siehe Abb.: 5 - Panorama Controll). Dabei ist es möglich, die Panorama Items so zu designen, dass sie als wiederverwertbare Module in den unterschiedlichen Application Page eingesetzt werden können. Panorama Items kann man vom Anwendungsprinzip mit Android Fragments vergleichen.

In der Beispielanwendung wurden, bis auf den Login, alle Application Page (Szenarios) in Panorama Items unterteilt. Ein Beispiel dafür ist der nach dem erfolgreichen Login dargestellte Startbildschirm. Auf diesem werden in den ersten Panorama Items die offenen und in den zweiten Panorama Items die bearbeiteten Veranstaltungen dargestellt (siehe Abb.: 9 - Startbildschirm - Windows Phone (Mock-Ups) im Anhang: Diagramme und Bilder). Für die Darstellung der Daten wurden Listenansichten gewählt, die jeweils an die oben erwähnten `ViewModels` gebunden sind. Ziel dieses Vorgehen war es, das MVVM-Muster in der WP-Applikation zu implementieren.

Login - Windows Phone Applikation

An dieser Stelle wird die Implementierung des Login-Szenarios (siehe Abschnitt: Entwicklungsspezifikationen des 3. Kapitels) für die WP-Applikation demonstriert. Bei der Implementierung wurde versucht, einen Großteil der Backend-Logik aus der Android-Applikation für die WP-App zu übernehmen, beziehungsweise diese zu portieren. Durch dieses Vorgehen wurde ein einheitliches Backend erzeugt mit dem Ziel, die Wartbarkeit zu erhöhen und die Fehleranfälligkeit in der Logik zu minimieren.

Service

Der hier verwendete Service ist nicht, wie im Android-Beispiel, ein Bestandteil des WP-SDK, sondern stellt eine implementierte Abstraktionsschicht der

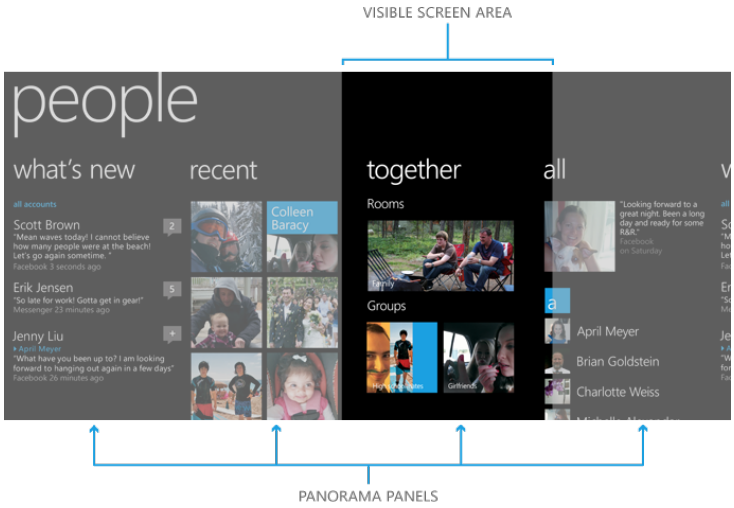

Abbildung 5: Ansicht der Panorama Control mit ihren Panorama-Panels. (Quelle: Microsoft MSDN 2014c)

Backend-Architektur da. Dabei besitzt die `Codebehind`-Datei eine Referenz auf den Service. Mithilfe des `Singleton`-Musters wurde sichergestellt, dass maximal nur eine Initialisierung des Services existiert.

Das Verhalten sowie die Implementierung des Service wurde größtenteils von der Android-Applikation (siehe Abschnitt: Login Szenario - Android Applikation) in die WP-App übernommen. Der Ablauf beziehungsweise die Logik des Login-Szenarios wurde in dem Punkt der Service-Portierung nicht verändert. Das gleiche zählt auch für die `Usecase-Controller`, weshalb sie an dieser Stelle nicht wiederholt behandelt werden. Weitere Information bezüglich der `Usecase-Controller` befinden sich im Abschnitt: Login Szenario - Android Applikation.

Asynchrone Tasks

Die Asynchrone Entwicklung wird für die gleichen Zwecke wie in der Android-Applikation benötigt. Allerdings werden dafür die Bibliotheksfunktionen des `.NET`-Framework 4.5 (`async/await`) eingesetzt (vgl.: Microsoft MSDN 2014b).

Kapitel 5

Reflexion

Im letzten Kapitel dieser Arbeit sollen die gesammelten Erfahrungen und Eindrücke, die durch die Recherche, Entwicklung und Implementierung an der Beispiel-Applikation erworben wurden, möglichst kritisch besprochen werden. Für diesen Zweck wurde das Kapitel in die zwei Abschnitte: `Native-Entwicklung` und `Auswahl des Entwicklungsansatzes` unterteilt. In dem Abschnitt Native-Entwicklung werden Anmerkungen zu den eingesetzten Technologien sowie Vorgehensweisen betrachtet. Anschließend werden weitere Kriterien für die Auswahl des Entwicklungsansatzes behandelt. Darin werden weiterführende Themen angesprochen, die zwar außerhalb der Spezifikation der Beispiel-Anwendung liegen, allerdings für die Entwicklung von professionellen Applikationen durchaus von Interesse sind.

5.1 Native-Entwicklung

Den Anfang bildet der direkte Vergleich zwischen den zwei eingesetzten Mobilen Entwicklungsplattformen. Dabei basieren die gesammelten Erfahrungen ausschließlich auf der durchgeführten Entwicklung der Beispiel-Anwendung. Bis auf die verwendeten Programmiersprachen, Java und C#, waren vor dem Projekt keine Vorkenntnisse in einer der verwendeten Plattformen vorhanden.

Vorbereitung der Entwicklungsumgebung

Das Einrichten der Android-Umgebung geht einfach und unkompliziert von statten. Von der `Android Developer`-Seite kann ein Paket herunter geladen werden, in dem alle für die Entwicklung benötigten Werkzeuge enthalten

sind. Im Gegensatz dazu muss die WP-Plattform installiert und gegebenenfalls noch aktualisiert werden. Ärgerlich dabei sind die hohen Hard- und Software-Anforderungen für den mitgelieferten Emulator (siehe Abschnitt: Windows Phone Entwicklung der State of the Art-Analyse). Des Weiteren gibt es Unterschiede in der Preispolitik für Entwicklerlizenzen. Während für eine Android-Lizenz eine einmalige Gebühr von 25$ anfällt, entstehen für die Windows-Entwickler-Lizenz jährliche Gebühren von 14 EUR. (vgl.: Android Developer 2014d; sowie: Microsoft MSDN 2014a)

Android- versus WP8-Entwicklung

Durch die Nähe zum `.NET`-Framework bietet die WP-Entwicklung ein gewisses Maß an Komfort. Das WP-SDK unterstützt die Entwicklung durch Optionen wie Beispielsweise die bidirektionale Datenbindung.

Ein Vorteil der Android-Plattform liegt in der weitläufigen Unterstützung von (älteren) Endgeräten. Dabei kann das Entwickler-Team selbst definieren, welche Android-Versionen von dem eigenen Projekt unterstützt werden. Mithilfe der Android-`Support Library` ist auch der aktuelle Funktionsumfang des SDK für ältere Android-Versionen nutzbar. (vgl.: Android Developer 2014i)

Im Gegensatz dazu setzt Microsoft auf restriktiven Anforderungen bezüglich der WP8-Endgeräte (vgl.: engadget.com 2013, ab Abschnitt: Hardware). Durch die von Microsoft vorgeschriebenen Hardwareanforderungen an die WP-Geräte muss bei der Entwicklung auf deutlich weniger Parameter, wie beispielsweise die Auflösung oder Größe des Displays, geachtet werden.

Portierung des Backends

Das Ziel der Portierung bestand darin, die Logik und Abläufe des Backends von den Plattform-Abhängigen Mechanismen zu lösen. Dadurch sollte sichergestellt werden, dass die Abläufe ab dem Aufruf des Service relativ identisch von statten gehen, um somit die Wartbarkeit des Projektes zu maximieren und einen gewissen Grad der Unabhängigkeit gegenüber den mobilen Plattformen zu erreichen[1]. Dies wurde allerdings, wie beispielsweise in den Asyn-

[1]Der Programmablauf sowie die Beschreibung des Backends befinden sich in dem Abschnitt: Login Szenario - Android Applikation sowie Login - Windows Phone Applikation des 4. Kapitels - Implementierung.

chronen Bereichen, nicht konsequent durchgezogen. Hierzu wurden unter dem Aspekten der Stabilität sowie der Zuverlässigkeit die jeweiligen Techniken der Zielplattformen eingesetzt. Im Nachhinein betrachtet, war die Portierung des `Backends` - der Beispiel-Applikation - nicht immer zielführend und verursachte mehr Quellcode, als es für die jeweiligen Applikationen nötig gewesen wäre. Ob sich der Mehraufwand rechtfertigt, wird sich schlussendlich bei der Erweiterung und Wartung des Projektes zeigen. Durch die Wahl des nativen Ansatzes sind beide Applikationen eigenständige Implementierungen. Dies bringt unter anderem Nachteile bei den Aktualisierungsvorgängen sowie der Fehlerbeseitigung mit sich. Angenommen, es wurde in der WP-Applikation ein Fehler gemeldet und beseitigt. Im Zuge des kontinuierlichen Qualitätsmanagements muss nun evaluiert werden, ob der dokumentierte Fehler auch in der Android-Applikation reproduzierbar ist. Sollte dies der Fall sein, muss möglicherweise für diesen eine andere Lösung erarbeitet werden. Durch den Einsatz eines Multiplattform-Frameworks würde der Arbeitsaufwand für diesen Wartungsvorgang deutlich geringer ausfallen. Der Fehler müsste nur einmalig korrigiert werden und kann anschließend über eine aktualisierte Version an alle Plattformen verteilt werden.

5.2 Zusätzliche Faktoren zur Auswahl des Entwicklungsansatzes

Als Grundlage für die Auswahl des Entwicklungsansatzes diente die State of the Art-Analyse. Zum damaligen Erfahrungs- und Wissens-Stand wirkte ein Nativer Ansatz unter der Berücksichtigung der durchgeführten Auswertung als Sinnvoll (siehe Abschnitt: Auswertung des 2. Kapitels - State of the Art). Durch die in der Entwicklungsphase gesammelten Erfahrungen hat sich zwischenzeitlich der Blickwinkel etwas verschoben beziehungsweise wurde erkannt, dass es weitere Kriterien gibt, welche in den Entscheidungsprozess einfließen sollten. Dazu zählen zum einen Wirtschaftliche Aspekte und zum anderen die Erweiterbarkeit der Zielplattformen, welche in den folgenden Abschnitten genauer betrachtet werden.

Wirtschaftliche Aspekte

Grundsätzlich sollte ein Nativer Ansatz aus wirtschaftlicher Sicht gut begründet sein, da für jede Plattform eine eigene Entwicklung und Implementierung von Nöten ist und sich der Aufwand deutlich auf die Entwicklungsdauer und somit Entwicklungskosten (siehe Abb.: 6 - Kosten: Native Entwicklung) niederschlägt. Im Sinne der Kostenoptimierung liegt es daher nahe, einen anderen Entwicklungsansatz zu wählen.

	Cost of single application	Number of applications needed	Total cost
Single-platform app	$20,000 - $150,000	1	**$20,000 - $150,000**
Cross-platform smartphone apps	$20,000 - $150,000	4	**$80,000 - $600,000**
Cross-platform smartphone & tablet apps	$20,000 - $150,000	8	**$160,000 - $1,200,000**

Abbildung 6: Kostenbeispiel der Nativen Entwicklung (Quelle: michaels, ross and cole, ltd. (mrc) 2014)

Der Erfolg von Frameworks wie beispielsweise Adobe PhoneGap oder Appcelerator Titanium basiert unter anderem darauf, dass die Entwicklung plattformunabhängig von statten geht wodurch die Entwicklungsdauer deutlich minimiert wird. Nichts desto trotz stellen alle plattformübergreifende Frameworks eine Abstraktion zu den nativen SDK's da. Dies bringt zwar auf der einen Seite eine komfortable und zügig Entwicklung mit sich, allerdings erhöht sich auf der anderen Seite auch die Abhängigkeit gegenüber dem eingesetzten Frameworks. Solche Abhängigkeiten können sich unter anderem in der Update-Politik niederschlagen. Vor der Auswahl des Frameworks sollte unbedingt eine Recherche über Update-Intervalle sowie die Stabilität des aktualisierten Frameworks durchgeführt werden. Diese Punkte sollten, speziell beim erstmaligen Einsatz des Frameworks, besonders berücksichtigt werden, da sie sich deutlich auf die Qualität des Produktes auswirken können.

Erweiterbarkeit der Zielplattformen

Der Markt für mobile Betriebssystem ist hoch dynamisch. In den letzten fünf Jahren wurde der Marktführer Symbian fast zur Gänze vom Markt verdrängt, während Android vom unbekannten- zum meist genutzten Betriebssystem aufgestiegen ist (siehe Abb.: 7 - Marktanteile der führenden mobilen Betriebssysteme).

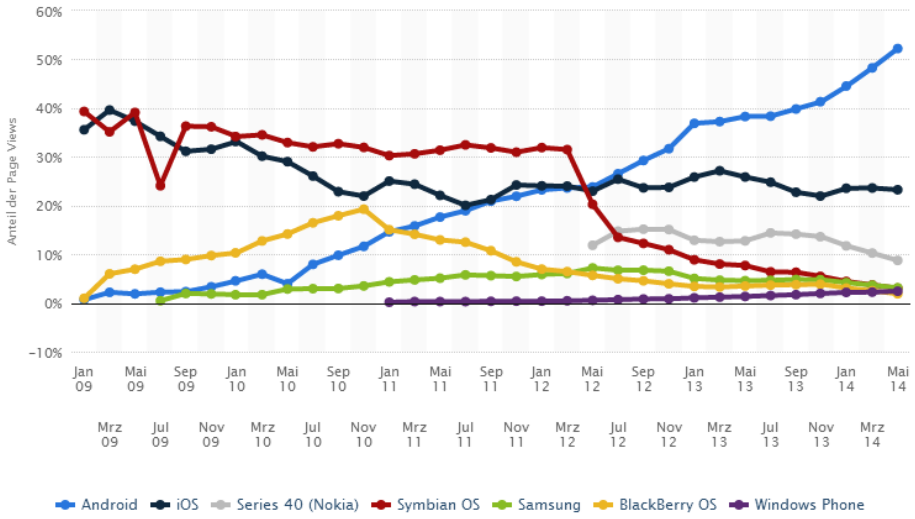

Abbildung 7: Marktanteile der führenden mobilen Betriebssysteme an der Internetnutzung mit Mobilgeräten weltweit von Januar 2009 bis Mai 2014 (Quelle: Statista 2014b)

Diese Dynamik bringt für die Applikation-Entwicklung sowohl Vor- als auch Nachteile mit sich. Durch den harten Wettkampf zwischen den einzelnen Plattformen sind die Hersteller darum bemüht, die bestehende Basis der Applikations-EntwicklerInnen zu halten beziehungsweise zu erweitern. Auf der anderen Seite kann die Fluktuation der Marktanteile zu einem Risiko für die eigenen Projekt werden, wenn diese nicht in der Lage sind, sich an die aktuellen Trends anzupassen. Der Einsatz von plattformunabhängig Frameworks in der Entwicklung kann dabei deutlich zur Flexibilitäts-Steigerungen beitragen. Eine Möglichkeit besteht darin, die Implementierung des Projektes auf der Basis des Adobe PhoneGap Framework's durchzuführen. Wie die State of the Art-Analyse ergeben hat, unterstützt das Adobe PhoneGap-Framework die meisten Plattformen. Zur Erzeugung einer nativen Container-Applikation für die unterstützte Zielplattform muss neben der PhoneGap-Infrastruktur auch das jeweilige SDK vorhanden sein. Neben der Unterstützung von diversen Mobilen Betriebssystemen kann Adobe PhoneGap auch Desktop-Applikationen für Microsoft Windows8 sowie Canonical Ubuntu erzeugen. (vgl.: *Platform Guides* 2014)

5.3 Schlusswort

Die zügigen Trend-Wechsel sowie die kurzen Entwicklungszyklen von Plattformen fordern der Planung und Entwicklung von mobilen Applikationen ein hohes Maß an Flexibilität und Umsicht ab. Nur durch kontinuierliche Recherchen, Fort- und Weiterbildung sowie Konsultation von Fachzeitpublikationen lässt sich der Überblick über aktuelle Trends, UI-Designs und unterstützende Frameworks nicht verlieren. Allerdings reicht das Wissen über die vorhandenen Technologien nicht aus. Wie so oft werden die Vor- und Nachteile erst durch ein gewisses Maß an Erfahrung ersichtlich. Abschließend kann gesagt werden, dass es im Moment keine allumfassende Lösung für die Multiplattform-Entwicklung gibt. Somit sollte die Auswahl schlussendlich auf der Basis der Prioritäten und Bedürfnissen des Projektes und dessen Infrastruktur gewählt werden.

Abkürzungsverzeichnis

ADT	Android Developer Tools
ANR	Application Not Responding
API	Application Programming Interface
CSS	Cascading Style Sheets
GB	Gigabyte
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JSON	Javascript Object Notation
MVA	Microsoft Virtual Academy
MVVM	Model View ViewModel
NDK	Native Development Kit
OJAD	Offene Jugendarbeit Dornbirn
RAM	Random-Access Memory
REST	Representational State Transfer
SDK	Software Development Kit
SLAT	Second Level Address Translation
SSL	Secure Sockets Layer
UI	User Interface
URL	Uniform Resource Locator

WP	Windows Phone
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

Literatur

Adobe PhoneGap Build (2013). Adobe. URL: https://build.phonegap.com/ (besucht am 2014).

Android Developer (2014a). *API Guides*. Open Handset Alliance. URL: http://developer.android.com/guide/index.html (besucht am 2014).

— (2014b). *API Reference*. Open Handset Alliance. URL: http://developer.android.com/reference/packages.html (besucht am 2014).

— (2014c). *Building a Dynamic UI with Fragments*. Open Handset Alliance. URL: https://developer.android.com/training/basics/fragments/index.html (besucht am 2014).

— (2014d). *Entwicklerregistrierung*. Google Play. URL: https://support.google.com/googleplay/android-developer/answer/113468?hl=de (besucht am 2014).

— (2014e). *Get the Android SDK*. Open Handset Alliance. URL: http://developer.android.com/sdk/index.html (besucht am 2014).

— (2014f). *Getting Started*. Open Handset Alliance. URL: http://developer.android.com/training/index.html (besucht am 2014).

— (2014g). *Keeping Your App Responsive*. Open Handset Alliance. URL: http://developer.android.com/training/articles/perf-anr.html (besucht am 2014).

— (2014h). *Services*. Open Handset Alliance. URL: http://developer.android.com/guide/components/services.html (besucht am 2014).

— (2014i). *Support Library*. Open Handset Alliance. URL: http://developer.android.com/tools/support-library/index.html (besucht am 2014).

Basic Steps In PhoneGap app Development (2013). URL: http://phonegap4u.blogspot.co.at/2013/07/how-phonegap-works.html (besucht am 2014).

Debugging in PhoneGap (2014). Adobe. URL: https://github.com/phonegap/phonegap/wiki/Debugging-in-PhoneGap (besucht am 2014).

Gifford, Matt (2012). *PhoneGap Mobile Application Development Cookbook*. Packt Publishing.

Meier, Reto (2012). *Professional Android 4 Application Development*. John Wiley und Sons.

Microsoft MSDN (2014a). *Account types, locations, and fees*. Microsoft MSDN. URL: http://msdn.microsoft.com/en-us/library/windows/apps/jj863494.aspx (besucht am 2014).

— (2014b). *Asynchrone Programmierung mit Async und Await*. Microsoft MSDN. URL: http://msdn.microsoft.com/de-de/library/hh191443.aspx (besucht am 2014).

— (2014c). *Panorama control for Windows Phone 8*. Microsoft MSDN. URL: http://msdn.microsoft.com/library/windows/apps/ff941104(v=vs.105).aspx (besucht am 2014).

— (2014d). *System requirements for the emulator for Windows Phone 8*. Microsoft MSDN. URL: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff626524(v=vs.105).aspx (besucht am 2014).

— (2014e). *User interface programming overview (Android to Windows)*. Microsoft MSDN. URL: http://msdn.microsoft.com/en-US/library/windows/apps/dn263251.aspx (besucht am 2014).

PhoneGap Build is Launched (2012). Adobe. URL: http://phonegap.com/blog/2012/09/24/phonegap-build-is-launched/ (besucht am 2014).

Platform Guides (2014). Adobe. URL: http://docs.phonegap.com/en/edge/guide_platforms_index.md.html (besucht am 2014).

Ross, Marcus (2013). *Cross-Plattform-Apps mit PhoneGap entwickeln*. heise Developer. URL: http://www.heise.de/developer/artikel/Cross-Plattform-Apps-mit-PhoneGap-entwickeln-1934535.html?artikelseite=6 (besucht am 2014).

Satya Komatineni, Dave MacLean (2012). *Pro Android 4*. Professional Apress.

Statista (2014a). *Anzahl der angebotenen Apps in den Top App-Stores im Juli 2013*. Statista. URL: http://de.statista.com/statistik/daten/studie/208599/umfrage/anzahl-der-apps-in-den-top-app-stores/ (besucht am 2014).

— (2014b). *Marktanteile der führenden mobilen Betriebssysteme an der Internetnutzung mit Mobilgeräten weltweit von Januar 2009 bis Mai 2014*. Statista. URL: http://de.statista.com/statistik/daten/

studie/184335/umfrage/marktanteil-der-mobilen-betriebssysteme-weltweit-seit-2009/ (besucht am 2014).

The Apache ANT Project (2012). *ANT-Homepage*. Apache Software Foundation. URL: http://ant.apache.org/.

Titanium Platform Overview (2014). Appcelerator. URL: http://docs.appcelerator.com/titanium/3.0/#!/guide/Titanium_Platform_Overview (besucht am 2014).

Titanium SDK (2014). Appcelerator. URL: http://www.appcelerator.com/titanium/titanium-sdk/ (besucht am 2014).

Titanium Support Plans for Windows 8 (2013). Appcelerator. URL: http://www.appcelerator.com/blog/2013/01/titanium-support-plans-for-windows-8/ (besucht am 2014).

Xamarin Inc. (2014a). *Key Strategies for Mobile Excellence*.

— (2014b). *pricing*. Xamarin Inc. URL: https://store.xamarin.com/.

— (2014c). *pricing*. Xamarin Inc. URL: http://xamarin.com/faq#pricing.

engadget.com (2013). *Windows Phone 8 review*. Aol Tech. URL: http://www.engadget.com/2012/10/29/windows-phone-8-review/ (besucht am 2014).

michaels, ross and cole, ltd. (mrc) (2014). *Native mobile apps: The wrong choice for business?* michaels, ross and cole, ltd. (mrc). URL: http://www.mrc-productivity.com/Research/whitepapers/NativeAppsWrongChoice.pdf (besucht am 2014).

Anhang A

Diagramme und Bilder

A.1 Übersicht

- Mock-Ups
  - Startbildschirm - Android (Mock-Ups)
  - Startbildschirm - Windows Phone (Mock-Ups)
  - Auswahl der Android-Mock-Ups
  - Auswahl der Windows Phone-Mock-Ups
- Diagramme
  - Service Lifecycle

Mock-Ups

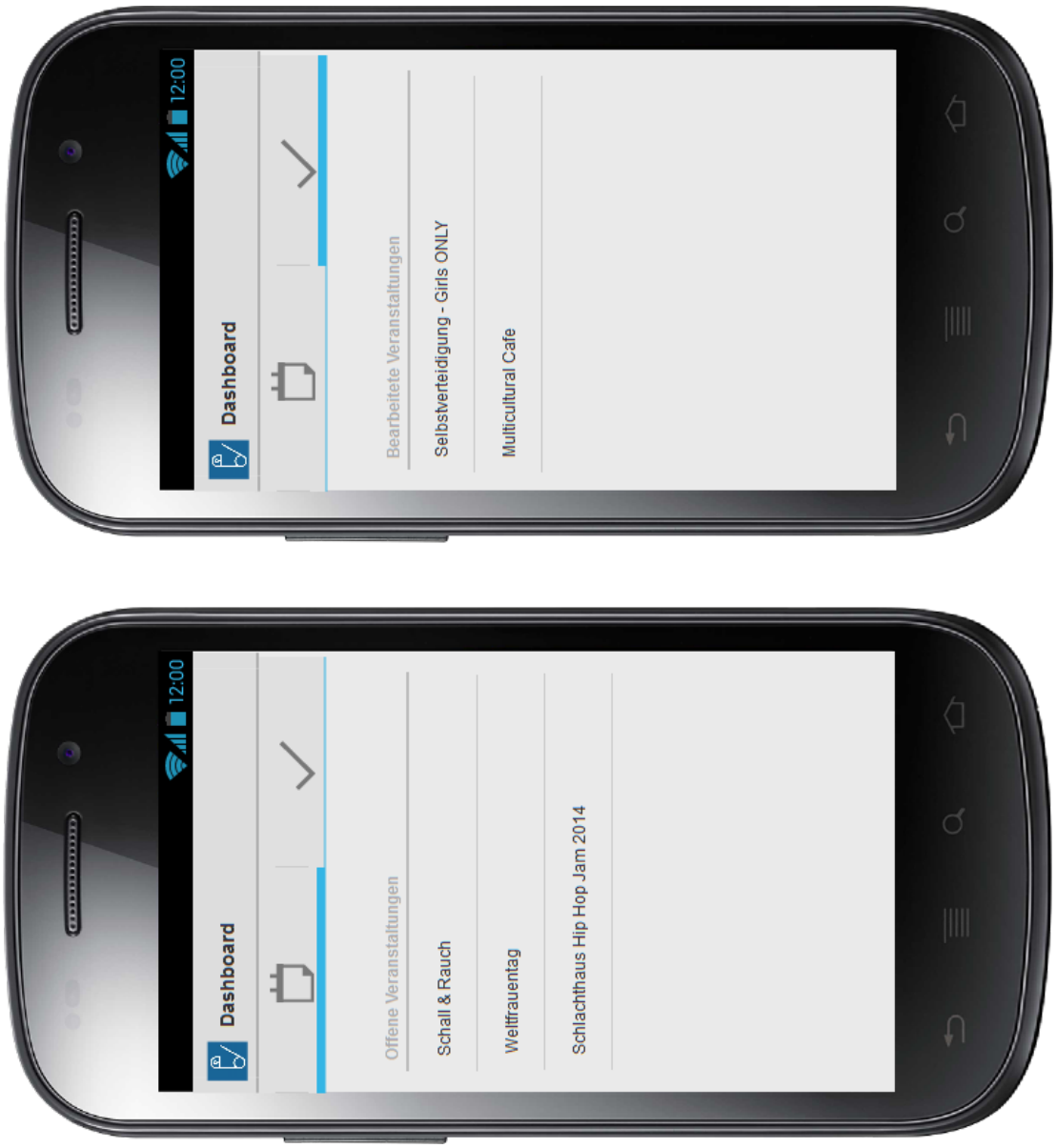

Abbildung 8: Einsatz von Fragments im Startbildschirm (Mock-Ups)

Abbildung 9: Einsatz von Panorama Items im Startbildschirm (Mock-Ups)

Abbildung 10: Auswahl der Android-Mock-Ups

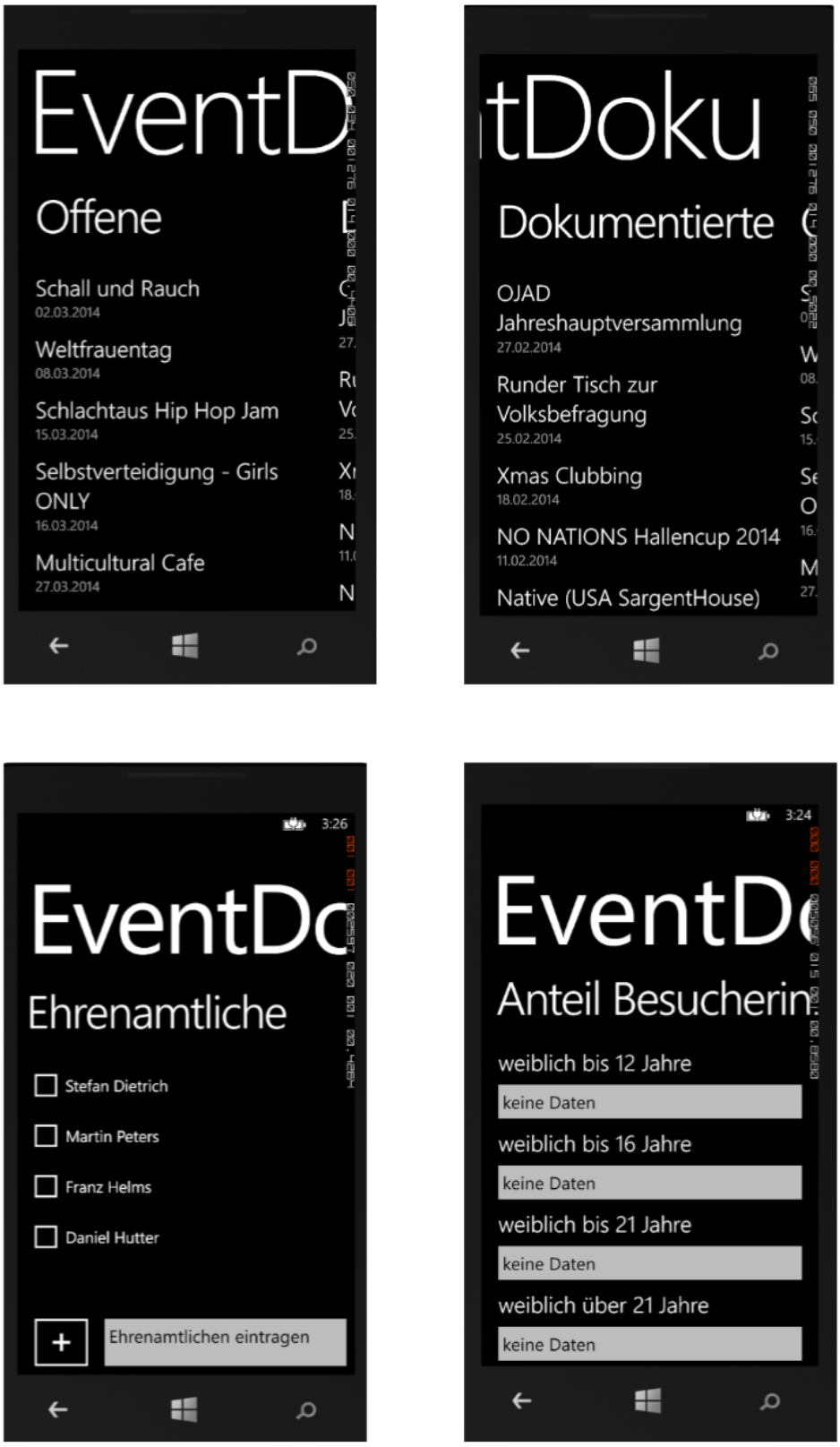

Abbildung 11: Auswahl der Windows Phone-Mock-Ups

Diagramme

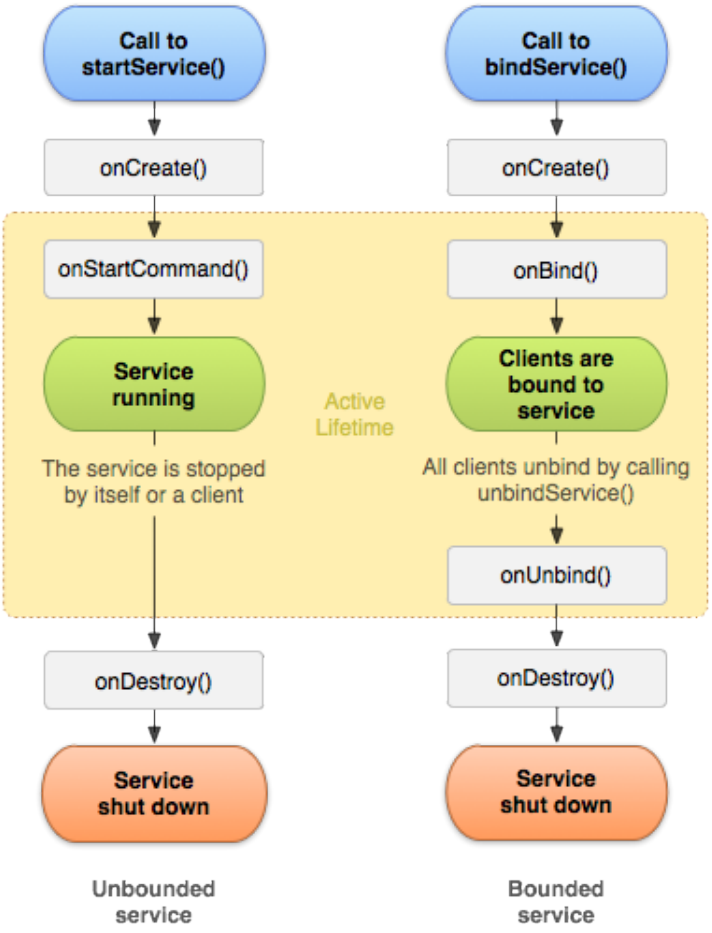

Abbildung 12: Service Lifecycle - (Quelle: Android Developer 2014h)