

Bachelorthesis
in
Allgemeine Informatik

**Prototypische Entwicklung eines Hybrid-
App-Frameworks für Android mit Ja-
vaScript-Java-Brücke im Umfeld eines
mittelständischen Medienunternehmens**

Referent:	Prof. Dr. Stefan Betermieux
Korreferent:	Armin D. Koppert M.Sc.
Vorgelegt am:	29.02.2020
Vorgelegt von:	Maximilian Rudi Marzeck
	Matrikelnummer: 251990
	Schiltenbachstraße 13
	79771 Klettgau-Erzingen
	maximilian.marzeck@hs-furtwangen.de

Abstract

This thesis focuses on the prototypical development of an android hybrid application framework. The prototypical implementation is carried out in cooperation with a media agency and should simplify the development of smartphone applications, also to reduce costs. The aim is to ensure that medium-sized companies are not left behind by larger companies in the course of digitalization. In order to determine how such an app framework should be structured, existing app development procedures were analyzed and an expert interview was made. Based on this, a requirements analysis was conducted. The prototypical hybrid application framework created in this thesis enables the development of applications with HTML, CSS and JavaScript embedded in native wrapper applications to compensate the disadvantages of pure web applications.

Die vorliegende Thesis beschäftigt sich mit der prototypischen Entwicklung eines Hybridapp-Frameworks für die Android-Plattform. Die prototypische Umsetzung erfolgt in Zusammenarbeit mit einer mittelständischen Medienagentur und soll die Entwicklung von Smartphone-Applikationen vereinfachen, wodurch die Kosten ebenfalls reduziert werden sollen. Ziel ist es, dass mittelständische Unternehmen im Laufe der Digitalisierung nicht von größeren Unternehmen abgehängt werden. Um zu ermitteln, wie ein solches App-Framework aufgebaut sein sollte, wurden bestehende App-Entwicklungsverfahren analysiert und ein Experteninterview durchgeführt. Darauf basierend, wurde eine Anforderungsanalyse durchgeführt. Das in dieser Arbeit prototypisch umgesetzte Hybrid-Applikations-Framework ermöglicht die Entwicklung von Applikationen mit den klassischen Web-Sprachen HTML, CSS und JavaScript. Die Applikationen werden in native Wrapper-Applikationen eingebettet, um die Nachteile reiner Web-Apps zu kompensieren.

Inhaltsverzeichnis

Abstract.....	III
Inhaltsverzeichnis	V
Abbildungsverzeichnis	VII
Tabellenverzeichnis	IX
Abkürzungsverzeichnis	XI
1 Einleitung.....	1
2 Analyse des Ist-Zustands.....	7
2.1 Ist-Analyse: Relevanz des Smartphones für die Wirtschaft	7
2.1.1 Umsatz im Mobile-Market.....	8
2.1.2 Smartphone-Nutzung in Zahlen	8
2.1.3 Relevanz von Apps für Unternehmen.....	10
2.2 Ist-Analyse: Der Alltag von mittelständischen Unternehmen	11
2.3 Ist-Analyse: App-Entwicklung	13
2.3.1 Native Applikation.....	13
2.3.2 Web-Apps	15
2.3.3 Progressive Web-Apps.....	17
3 Grundlagen.....	19
3.1 Anforderungsanalyse.....	19
3.1.1 Wie wird eine Anforderungsanalyse umgesetzt?	20
3.1.2 Anforderungen	21
3.2 Hybrid-Apps	23
3.2.1 Struktur von Hybrid-Apps	24
3.2.2 Apache Cordova und Phonegap	25
3.3 Komponenten des Hybrid-App-Frameworks	27
3.4 Werkzeuge für die Entwicklung.....	29
3.4.1 Betriebssystem.....	29

3.4.2	Entwicklungsumgebungen	32
3.4.3	Drittanbieter-Frameworks	34
3.4.4	Zusammenfassung	37
4	Prototypische Umsetzung.....	39
4.1	Android-Studio-Projekte	39
4.2	Entwicklung der Hauptklasse	41
4.2.1	Der Konstruktor.....	42
4.2.2	Grundlegende Funktionen	44
4.2.3	Schnittstelle: Java- zu JavaScript-Code.....	45
4.2.4	Schnittstelle: JavaScript zu Java-Code.....	47
4.2.5	HTML-Knoten.....	52
4.3	Entwicklung des JavaScript-Frameworks	53
4.4	Prototypische Umsetzung der Benutzeroberfläche	53
4.4.1	Aufbau der Benutzeroberfläche	54
4.4.2	Konfiguration von Applikationen.....	55
5	Ergebnis, Ausblick und Fazit	59
5.1	Zusammenfassung.....	59
5.2	Ergebnis und Bewertung	59
5.3	Bewertung der Werkzeuge	63
5.4	Ausblick und Entwicklungsvorschläge	65
5.4.1	Erweiterung des Funktionsumfangs.....	66
5.4.2	Entwicklung für weitere Systeme	67
5.5	Fazit.....	67
	Literaturverzeichnis	69
	Eidesstattliche Erklärung	75
	Anhang A: Experteninterview	77
	Anhang B: Monatsberichte	81

Abbildungsverzeichnis

Abbildung 1: Verwendete Plattform im Internet (weltweit), Dezember 2019 ..	3
Abbildung 2: Desktop vs. Mobile als Internetendgerät, Dez. 09 – Dez. 19	9
Abbildung 3: Struktur einer Hybrid-Applikation.....	24
Abbildung 4: Geplante Hybrid-App-Framework-Struktur.....	28
Abbildung 5: Marktanteil mobiler Betriebssysteme weltweit (2010-2019)	30
Abbildung 6: Übersicht einiger Bootstrap-Komponenten	36
Abbildung 7: Android-Versionen im Überblick, Dezember 2019	40
Abbildung 8: Ausführung von JavaScript-Code in der Adresszeile	45
Abbildung 9: Konfiguration der Startseite und Farben	56
Abbildung 10: Konfiguration von Icons, Name, Theme und Berechtigungen	57

Tabellenverzeichnis

Tabelle 1: Vor- und Nachteile von nativen Apps	15
Tabelle 2: Vor- und Nachteile einer Web-App.....	17
Tabelle 3: Vor- und Nachteile von Progressive Web-Apps	18
Tabelle 4: Anforderungskatalog an ein Framework für App-Entwicklung.....	22
Tabelle 5: Überprüfung der Entwicklungsmodelle für Apps aus Kapitel 2.3 um deren Übereinstimmung mit den Anforderungen zu analysieren	23

Abkürzungsverzeichnis

APK	Android Package (Paketdateiformat)
App	Applikation / Anwendung
CSS	Cascading Style Sheets
GUI	Grafische Benutzeroberfläche
HTML	Hypertext Markup Language
ID	Identifikationsnummer
IDE	Integrierte Entwicklungsumgebung
IPA	iOS App Store Package (Paketdateiformat)
JS	JavaScript
NFC	Near Field Communication
OS	Operating System (Betriebssystem)
PWA	Progressive Web App
SDK	Software Development Kit
URL	Uniform Resource Locator
XML	Extensible Markup Language

1 Einleitung

Technologisch gesehen, hat die Menschheit in den letzten Jahrzehnten eine große Entwicklung durchgemacht. Als 1949 Edmund Berkeley den ersten Heimrechner *Simon* vorstellte[1], hätte wohl niemand gedacht, dass Computer binnen weniger Jahrzehnte einen so großen Einfluss auf unser Leben haben würden, dass diese Entwicklung als *digitale Revolution* (in Anlehnung an die industrielle Revolution) bezeichnet werden würde [2]. Dank der Erfindung des Mikroprozessors wurden Heimrechner immer kleiner, leistungsfähiger und günstiger [1]. Dies mündete in den 80er-Jahren mit der Erfindung des World Wide Webs, ein weiterer revolutionärer Schritt im Prozess der Digitalisierung [2]. In den 90er Jahren setzte sich der Name Internet allgemein für den Netzwerkdatenaustausch durch [2]. Dr. Oliver Stengel, Alexander van Looy und Stephan Wallaschkowski vertreten in Ihrem Buch *Digitalzeitalter - Digitalgesellschaft* die Ansicht, dass die Erfindung des World Wide Webs von ähnlicher Bedeutung für die Menschheit ist, wie die Erfindung des Buchdrucks [2].

Die Leistungssteigerung von Computern führte dazu, dass diese im Laufe der Zeit Einzug in neue Einsatzgebiete erhielten. Geräte wie Waschmaschinen, Autos aber auch Telefone wurden mit kleinen Computern bestückt [1]. Zu den wichtigsten dieser neuen Mikrocomputer zählen die Smartphones [3]. Sie sind heute kaum noch aus unserer Gesellschaft wegzudenken, denn sie erfüllen neben der Freizeit-Bespaßung durch Apps und Spiele eine Reihe wichtiger Aufgaben in der Wirtschaft [3]. So läuft heute ein großer Teil der Kommunikation über diese Geräte [3]. Außerdem verwenden laut einer Studie von kaufDa aus dem Jahr 2016, 50 Millionen Menschen in Deutschland das Smartphone, um im Internet shoppen zu gehen [3].

Der Durchbruch dieser Geräte wird auf das Jahr 2007 datiert, in dem das erste Apple iPhone erschien [4]. Zwar hatten zu diesem Zeitpunkt Hersteller wie Nokia und Blackberry ebenbürtige Geräte auf dem Markt, jedoch machte Apple mit seinem iOS etwas gänzlich Neues [5]. Das System wurde ein Stück weit für Entwickler geöffnet, sodass Dritte eigene Anwendungen erstellen und diese im Anwendungsmarktplatz *App-Store* anbieten konnten [5]. Zur gleichen Zeit arbeitete der Suchmaschinen-Konzern Google an einem eigenen Gerät,

welches ebenfalls ein Smartphone sein sollte [6]. Unter dem Codenamen *Sooner* entwickelten sie ein Tastatur-Smartphone [6]. Langfristig sollte daraus eine Touchscreen-Plattform Namens *Dream* entstehen [6]. Nach dem Aufkommen des iPhones wurde dieses Projekt eingestellt und mit der Entwicklung von Android begonnen [6].

Mit der Arbeit an Android wurde noch im Jahr 2007 durch die *Open Handset Alliance* begonnen [3]. Die *Open Handset Alliance* wurde von Google ins Leben gerufen, jedoch sind viele andere Gerätehersteller und Netzanbieter Teil dieser Allianz [3]. Android ist größtenteils quelloffen, es gibt kaum proprietäre Komponenten [3]. Dadurch fallen für Gerätehersteller keine Lizenzgebühren für das Betriebssystem an, was sich positiv auf die Preise der Geräte auswirkt [3]. Auch bei Android können Entwickler auf Anwendungsmarktplätzen ihre Applikationen anbieten [7]. Neben dem Google Play Store bestehen auch eine ganze Reihe anderer Anwendungsmarktplätze von Drittanbietern, wodurch Google im Vergleich zu Apple kein Monopol auf das Anbieten von Applikationen hat [7].

Bei einer App handelt es sich um eine Anwendung die ein um nützliche Funktionalitäten erweitert [3]. Im deutschen Sprachraum hat sich das Wort *App* als Bezeichnung für Smartphone-Applikationen durchgesetzt [3]. Unter den Begriff *App* können unterschiedlichste Programme fallen z.B. Spiele, Büroanwendungen, Film-Plattformen oder Online Shops [3]. Im Jahr 2016 wurden weltweit 90 Milliarden Apps heruntergeladen, während ein durchschnittlicher Smartphone-Nutzer am Tag 1,5 Stunden mit Apps verbrachte [3]. Aus wirtschaftlicher Sicht sind Smartphones ebenfalls ein nicht zu vernachlässigender Faktor. So sollen verschiedene Studien nachweisen, dass die Nutzung von Apps die Kundenbindung erhöht und sowohl für einen höheren Umsatz pro Kunde, als auch für eine höhere Kauffrequenz der Kunden sorgt [3]. 54% benutzen laut einer kaufDa-Studie im Jahr 2016 eine App, um lokale Infos für einen Kauf einzusehen [3]. Im Dezember 2019 erfolgten laut Statcounter 53,25% (siehe Abbildung 1) aller Zugriffe auf Partner-Webseiten von Statcounter durch Smartphone-Nutzer [8]. Trotzdem wird das Potenzial von Smartphones im Mittelstand bei mittleren und kleinen Unternehmen komplett

vernachlässigt. Gerade die Hälfte aller Webseiten ist mobil optimiert, bei angebotenen Apps sieht die Lage noch schlechter aus [3].

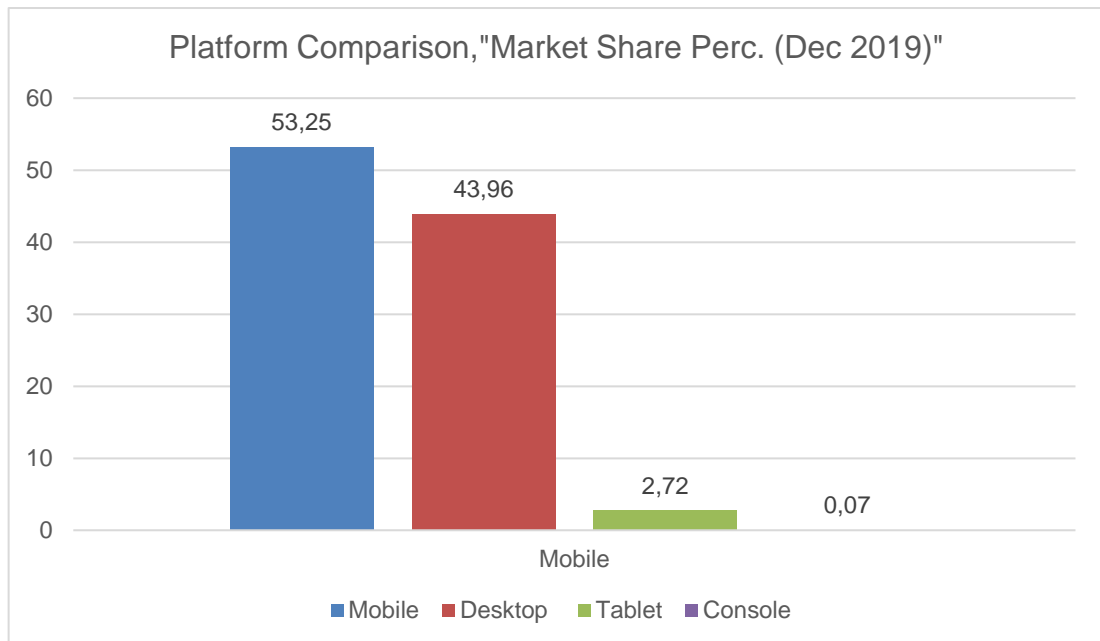


Abbildung 1: Verwendete Plattform im Internet (weltweit), Dezember 2019¹

Um kleineren und mittleren, mittelständischen Unternehmen den Einstieg in den App-Markt zugänglicher zu machen, soll im Rahmen dieser Bachelorarbeit ein App-Framework geschaffen werden, welches die Entwicklung von Apps erleichtert. So soll diesen Unternehmen ermöglicht werden, auch in Zeiten der Digitalisierung gegenüber größeren Unternehmen konkurrenzfähig bleiben zu können. Dafür soll analysiert werden, wo die Gründe liegen, dass mittelständische Unternehmen in diesem Bereich noch keinen Anschluss gefunden haben. Um eine glaubwürdige und praxisorientierte Arbeit zu schaffen, wird daher das Unternehmen Koppert Mikromarketing-Systeme als betreuendes Unternehmen in diese Arbeit mit einbezogen. Das Unternehmen setzt sich mit Mittelstandsberatung auseinander, bietet aber auch zahlreiche Dienste in den Bereichen Medien- und Webdesign an [9]. In Zuge dessen wird seit Jahren mit mittelständischen und Unternehmen zusammengearbeitet [9]. Die Agentur hat bisher über 500 Klienten unterstützt und über 90 Existenzgründungen begleitet [9]. Koppert Mikromarketing-Systeme hat sich

¹ Quelle: <https://gs.statcounter.com/platform-market-share#monthly-201912-201912-bar>

bereit erklärt, diese Thesis zu unterstützen sowie ihre Kenntnisse und Erfahrungen weiterzugeben.

Die Ziele dieser Arbeit sind somit folgendermaßen zusammenzufassen:

- Zunächst wird eine Ist-Analyse durchgeführt, welche die Wichtigkeit von Smartphone für die heutige Wirtschaft herausstellen soll. Teil dieser Analyse sollen auch populäre Entwicklungsmodelle für Smartphones sein, wobei darauf geachtet werden soll, welche Vor- und Nachteile diese bieten. Durch ein Experteninterview im betreuenden Unternehmen soll der Arbeitsalltag von mittelständischen Unternehmen veranschaulicht und eine Expertise eingeholt werden, mit welchen Problemen und Chancen der Mittelstand in der Digitalisierung zu kämpfen hat.
- Basierend auf der Ist-Analyse soll eine Anforderungsanalyse für ein Smartphone-Applikations-Framework durchgeführt werden, welche definiert, wie Apps optimal entwickelt werden können.
- Auf Grundlage der Anforderungsanalyse soll eine Prototypische Umsetzung des App-Frameworks entwickelt werden. Dieses soll mittelständischen Unternehmen als Werkzeug zur einfachen App-Entwicklung dienen und dadurch kleineren Unternehmen ermöglichen, eigene Apps entwickeln zu lassen.

Die Arbeit lässt sich in fünf Kapitel einteilen. Nach der Einleitung folgt im zweiten Kapitel eine Analyse des Ist-Zustands. Hier soll ergründet werden, von welcher Wichtigkeit Smartphones und Smartphone-Applikationen in der heutigen Wirtschaft sind. Daraufhin werden die Erkenntnisse eines Experteninterviews näher behandelt, in welchem die Situation des Mittelstands in Bezug auf die Digitalisierung behandelt werden soll. Anschließend folgt eine Analyse von bekannten App-Entwicklungs-Verfahren.

Im Anschluss danach folgt Kapitel drei. Hier sollen die Erkenntnisse aus der Istzustands-Analyse genutzt werden, um eine Anforderungsanalyse aufzustellen. Hier soll auch behandelt werden, welche Art App-Entwicklungs-Framework im Laufe dieser Thesis entwickelt werden soll, welche Komponenten ein solches Framework beinhaltet und welche Werkzeuge gebraucht werden, um dieses Framework umzusetzen. Daraufhin folgt in Kapitel vier die

prototypische Umsetzung des Frameworks, dabei sollen auf die wichtigsten Details der Durchführung eingegangen werden.

Im letzten Kapitel folgt dann das Ergebnis, dabei soll überprüft werden, ob die Arbeit erfolgreich durchgeführt werden konnte. Dies wird dann an den gestellten Kriterien in der Einleitung und in der Anforderungsanalyse festgemacht. Danach soll noch ein kleiner Ausblick darauf gegeben werden, wie die Entwicklung des Frameworks weitergehen könnte. Die Arbeit soll durch ein kurzes Fazit abgeschlossen werden.

2 Analyse des Ist-Zustands

In der Einleitung konnte bereits herausgestellt werden, dass Smartphones heute nicht mehr aus dem Alltag der Menschen wegzudenken sind. Der Einfluss der intelligenten Telefone beschränkt sich jedoch nicht nur auf die soziale Ebene. Auch große Teile der Wirtschaft, insbesondere der E-Commerce macht einen großen Wandel durch [3]. Einige Unternehmen verwenden Smartphones gezielt, um Informationen zu verbreiten, als Kommunikationsmittel, Werbeplattform oder um direkt eigene Waren und Dienste anzubieten [3]. Für einige Unternehmen sind sogar Apps selbst das vertriebene Produkt [3]. In diesem Kapitel sollen nun mehrere Ist-Analysen durchgeführt werden. Dabei soll herausgestellt werden, wie wichtig Smartphones als Plattform sind. Danach soll auch ermittelt werden, weshalb mittelständische Unternehmen die Chancen des Smartphone-Applikations-Marktes nicht wahrnehmen können. Dafür sollen zum Schluss auch Vorgehensmodelle in der App-Entwicklung analysiert werden.

2.1 Ist-Analyse: Relevanz des Smartphones für die Wirtschaft

Wenn die Relevanz von Smartphones für Unternehmen beschrieben werden soll, so muss man zunächst festlegen, wie diese Wichtigkeit definiert werden kann. Um die Wichtigkeit von Smartphones in der Wirtschaft zu messen, können verschiedene Aspekte in Betracht gezogen werden. In dieser Analyse sollen jedoch nur zwei dieser Aspekte näher beleuchtet werden. Die Relevanz von Smartphones soll in dieser Arbeit so festgelegt werden, dass sie sich aus dem Faktor des Umsatzes und der Anzahl von Smartphone-Nutzern definiert. Um die Aussagekraft des Umsatzes zu erhöhen, soll nicht nur der Wert eines Jahres, sondern auch die Entwicklung des Wertes betrachtet werden. So kann ein Blick darauf gewonnen werden, wie sich diese Entwicklung fortführen wird. Die Betrachtung der Smartphone-Nutzer-Anzahl ist von Bedeutung, wenn betrachtet werden soll, wie viele Menschen potenziell durch den App-Markt erreicht werden könnten. Daher soll auch auf diesen Wert umfassend eingegangen werden.

2.1.1 Umsatz im Mobile-Market

In seinem 2018 erschienenen Buch *Die Neuausrichtung des App- und Smartphone-Shopping* beschreibt Prof. Dr. Gerrit Heinemann, dass die Nutzung des Smartphones im Online-Shopping in Deutschland noch in den Kinderschuhen steckt [3]. Er führt dies auf den schlechten Netzausbau in Deutschland zurück, da Kunden sich immer und überall Zugriff auf das Internet wünschen [3]. Dennoch hat der Online-Handel über das Smartphone einen signifikanten Anteil am Gesamtumsatz. Für das Jahr 2017 wurde der mobile Online-Umsatzanteil mit etwa 24% betitelt, Umsatzprognosen gehen davon aus, dass im Jahr 2020 etwa ein Drittel des Gesamtumsatzes durch das Smartphone generiert wird [3]. Prof. Dr. Heinemann weist in diesem Kontext auch darauf hin, dass die Zubringerrolle, die Smartphones spielen, in diesen Statistiken nicht beachtet würden [3]. Dabei wird das Smartphone nicht nur für den Handel selbst, sondern auch zum Beschaffen von Produktinformationen verwendet, bevor sie sich in den Handel begeben, um Waren zu erwerben [3]. Laut einer Studie von kaufDa aus dem Jahr 2016 verwenden 80% der Smartphone-Nutzer ihr Gerät, um Produktinformationen im Internet zu erhalten [3]. Der Mobile-Commerce-Umsatz belief sich im Jahr 2017 für ganz Europa auf etwa 190 Mrd. Euro, wovon in Deutschland alleine 20 Mrd. Euro umgesetzt werden [3].

Diese Zahlen zeigen zwar bereits eindrucksvoll, dass der mobile Online-Markt nicht vernachlässigt werden sollte. Welches Potenzial auf dem App-Markt jedoch insgesamt ausgeschöpft werden kann, zeigt diese Zahl noch nicht. Dafür soll nun betrachtet werden, wie sich die Zahl der Smartphone-Nutzer in den letzten Jahren insgesamt entwickelt hat.

2.1.2 Smartphone-Nutzung in Zahlen

Um die Zahl aktiver Smartphone-Nutzer zu ermitteln, kann sich wieder auf mehreren Wegen genähert werden. Internetnutzungsstatistiken können, wenn sie über mehrere Jahre betrachtet werden, Aufschluss darüber geben, wie sich das Nutzerverhalten ändert. Steigt das Smartphone in diesen Statistiken wird es häufiger als Internetendgerät verwendet, was bedeutet, dass diese Geräte entweder häufiger genutzt werden oder mehr Endgeräte existieren. Prof. Dr. Heinemann beschreibt, dass sich der Internetverkehr massiv vom Desktop auf das Smartphone verschiebt [3]. Im Jahr 2016 blieb der Internetverkehr auf den

Desktoprechnern nahezu gleich im Vorjahresvergleich, während das Smartphone massiv zugelegt hatte [3]. Auch die Statistiken von Statcounter scheinen diesen Trend aufzuzeichnen (siehe Abbildung 2).

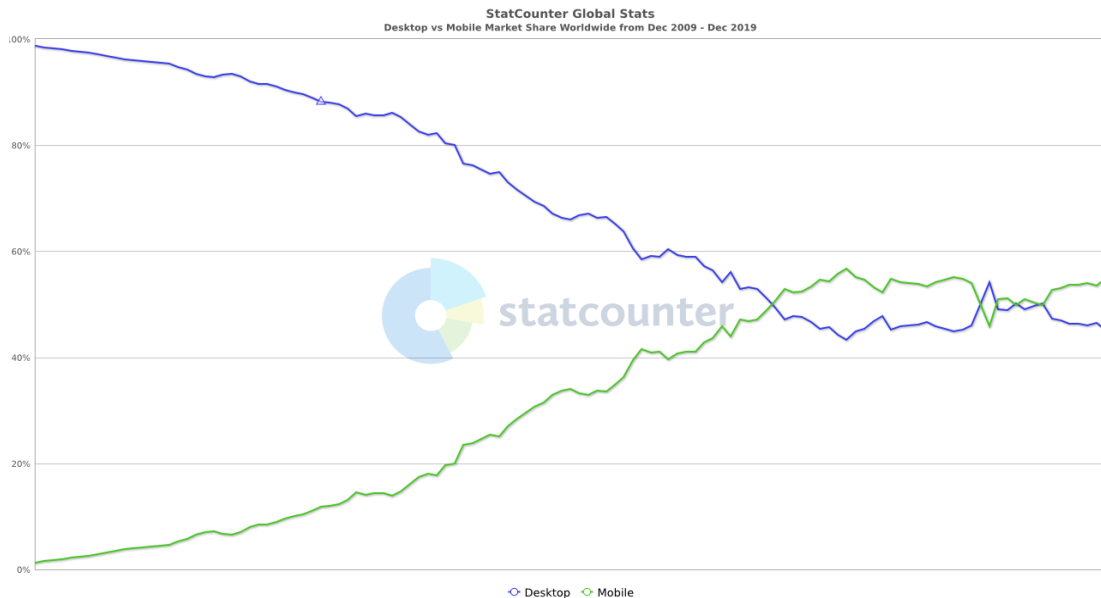


Abbildung 2: Desktop vs. Mobile als Internetendgerät, Dez. 09 – Dez. 19²

Die Statistiken von Statcounter können gut Aufschluss geben, wenn es um Statistiken in der Internetnutzung geht. Das Geschäftsmodell von Statcounter besteht aus einem Analyse-Tool für die Webseiten ihrer Kunden, dabei werden von jedem Nutzer der eine dieser Kundenseiten besucht Daten erfasst. Die Daten können von den Kunden dann eingesehen und analysiert werden. Doch auch Statcounter fertigt aus den Daten ihrer Kunden eigene Statistiken an. Die Interessanteste dieser ist zunächst die Statistik zur Internetnutzung eingeteilt in die Plattformen, welche bereits im ersten Kapitel Erwähnung fand. Dabei lässt sich feststellen, dass das Smartphone einen Gesamtanteil von 53,25% am Internetverkehr hat [8]. Smartphones machen also mehr als die Hälfte des Internetverkehrs aus. Zur gleichen Zeit wurde der Desktop nur noch zu 43,96% verwendet [8]. Welcher Wandel sich auf diesem Gebiet zugetragen hat, kann leicht ermittelt werden, wenn man Smartphone und Desktop im Zeitraum von Dezember 2009 bis Dezember 2019 betrachtet. Hierfür soll ebenfalls

² Quelle: <https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide/#monthly-200912-201912>

Statcounter verwendet werden. In dieser Statistik lässt sich betrachten, dass der Desktop im Jahr 2009 diesen Bereich mit fast 100% fast beherrschte [10]. In den letzten Jahren pendelte sich dieser Wert bei beiden Plattformen zwischen 40% und 60% ein [10]. Das Smartphone führt diese Statistik jedoch nahezu ununterbrochen an [10].

Im April 2019 wurde festgestellt, dass weltweit vier Milliarden unterschiedliche Mobiltelefon-Endnutzer existieren [11]. Dem gegenüber stehen im ersten Quartal des Jahres 2019 in den beliebtesten Anwendungs-Marktplätzen, 2,1 Millionen Applikationen in Googles Play Store und 1,8 Millionen Applikationen in Apples App-Store [11]. Diese Statistiken und Daten zeigen bereits auf erstaunliche Weise wie relevant es ist, mit den eigenen Produkten auch das Potenzial des Smartphone-Software-Marktes zu erreichen. Klar ist nun auch, dass das Smartphone das wichtigste Endgerät für den Internetzugriff ist. Um jedoch spezifiziert auf die Relevanz von Smartphone-Applikationen einzugehen müssen weitere Statistiken betrachtet werden.

2.1.3 Relevanz von Apps für Unternehmen

Laut Statista verbringen Smartphone-Nutzer 89% ihrer Zeit am Smartphone mit Apps [11]. Prof. Dr. Heinemann weist darauf hin, dass bei Onlinehändlern auf zehn Bestellungen über den Browser, neun Bestellungen aus Apps kommen [3]. Die Verkäufe durch Apps machen somit fast die Hälfte aller Verkäufe aus. Er führt zudem aus, dass Smartphone-Apps als Suchhelfer von Produkten fungieren. Auf dem Smartphone würden Informationen zu Produkten und Diensten meist nicht mehr über Google und andere Suchmaschinen gesucht, sondern direkt in Apps abgefragt [3]. Prof. Dr. Heinemann beschreibt, dass große Anbieter wie Google, Amazon und Apple nach und nach Inhalte anbieten, die lediglich durch Apps abrufbar sind [3]. Nach eigenen Angaben macht Apple alleine mit Services/iTunes einen Umsatz von 24 Milliarden US-Dollar [3]. Dies beschreibt seiner Ansicht nach, dass der Webbrowser nach und nach durch Apps im E-Commerce abgelöst wird [3].

Problematisch ist an dieser Sache jedoch, dass viele Unternehmen dieses Potenzial nicht ausschöpfen und daher einen Wettbewerbsnachteil haben. Während 50% der mittelständischen Unternehmen eine mobil optimierte Webseite

anbieten, so verfügen nur 32% der Online-Händler in Großbritannien Inhalte in Smartphone-Applikationen an [3].

Prof. Dr. Heinemann beschreibt Smartphone-Apps auch als eine Art Kundenkarte, welche den Einkauf für Kunden vereinfacht und auch dem Händler diverse Vorteile verschafft [3]. So beschreibt er, dass Unternehmen die eine eigene App anbieten, einen gewichtig höheren Umsatz im Vergleich zu dem Zeitpunkt vor der eigenen App kreieren [3]. Dies sei darauf zurückzuführen, dass App-Kunden im Allgemeinen die loyaleren Kunden seien, die frequenter einkaufen und dabei auch bereit sind mehr Geld auszugeben [3]. Der Händler kann außerdem diese Daten der Einkäufe und Kunden auswerten und so gezielteres Marketing betreiben [3].

2.2 Ist-Analyse: Der Alltag von mittelständischen Unternehmen

Da nun auf verschiedene Art und Weise aufgezeigt werden konnte, wie relevant das Smartphone und die darauf installierten Anwendungen sind und wie wenig dieses Potenzial gerade in mittelständischen Unternehmen ausgenutzt wird, soll nun ermittelt werden, woran dies liegen könnte. Deshalb wurde im Zuge dieser Thesis ein Experteninterview mit Frau Haberer von Koppert-Mikromarketing durchgeführt. So soll ein Einblick in den Alltag und die Herausforderungen gewonnen werden, mit denen mittelständische Unternehmen im Zuge der Digitalisierung konfrontiert sind. Als Mediendesignerin von Koppert-Mikromarketing arbeitet Frau Haberer seit 6,5 Jahren eng mit mittelständischen Unternehmen zusammen und gestaltet deren Inhalte [12].

Ihre Tätigkeiten erstrecken sich dabei über ein weites Spektrum in den Bereichen Werbung, Marketing und Informationen. So erstellt Frau Haberer für Unternehmen Flyer, Kurzfilme und Webseiten. Auch Suchmaschinenoptimierung betreibt Frau Haberer [12]. Wenn es um die Entwicklung von unternehmensspezifischen Apps geht, ist die Situation jedoch problematisch. Die Entwicklung von Apps ist sehr aufwendig und daher auch sehr teuer [12]. Gerade kleinere Unternehmen könnten da nicht mithalten, da für sie häufig schon die Kosten einer modernen Webseite schwer zu tragen sind [12].

Auch in der Literatur und im Netz werden die Kosten für Smartphone-Applikationen deutlich höher angegeben als die Entwicklungskosten für Webseiten.

Die angegebenen Zahlen unterscheiden sich jedoch je nach Quelle stark. In dieser Thesis wurden daher zwei Quellen herangezogen, deren ausgezeichnete Kosten im Bereich häufig genannter Werte liegen. Für Webseiten werden dort Kosten zwischen 900€ bis 17700€ angegeben [13]. Während die Entwicklung von Smartphone-Anwendungen mit 2828€ bis 175000€ beziffert wird [14]. So muss im besten Fall davon ausgegangen werden, dass die Kosten einer App dreimal so teuer sind, wie die einer Webseite. Im schlechtesten Fall spitzt sich dieser Wert auf das Zehnfache zu.

Wodurch diese Diskrepanz entsteht, ist für Frau Haberer eindeutig. Sie geht davon aus, dass Applikationen deshalb teurer als Webseiten sind, da der Entwicklungsprozess von Applikationen deutlich komplizierter ist:

“Also [...], bis man schon mal so etwas konzipiert hat, dauert dies eine ganze Weile, das weiß ich aus Erfahrung. Es dauert immer ein bisschen, bis die Idee da ist, bis geplant ist, was wo sein soll, damit ein Nutzer auch durchblickt, was jetzt gerade zu tun ist. Dann muss das Ganze gestaltet werden. Die Gestaltung braucht auch Zeit, denn es soll ja alles verständlich und schön sein. Dann muss man wieder Einzelteile rausspeichern und mit einem Programmierer kommunizieren, wohin alles gehört und was programmiert werden soll. Und die Entwicklung mit der Fehlersuche und allem, dauert auch wieder eine Weile. Das sind einfach viele Arbeitsstunden. Könnte man alles so in einem Rutsch machen, also Konzept, Design und Umsetzung, wie bei Webseiten, dann würde dies sehr viel Zeit ersparen.“ [12]

Frau Haberer beschreibt, dass die App-Entwicklung nicht in einem Rutsch erledigt werden kann. Für die Entwicklung einer App werden Designer und Programmierer benötigt [12]. Diese müssen ihre Arbeit miteinander abstimmen und in Einklang mit dem Konzept bringen [12]. Designer haben zwar Erfahrungen in den Web-Sprachen HTML, CSS und teilweise auch in JavaScript, mit welchen sie auch im Laufe ihres Studiums in Berührung kommen [12]. Klassische Programmiersprachen wie z.B. Java werden im Studium nicht gelehrt und sind somit den Designern somit meist fremd [12]. Das Beherrschen klassischer Programmiersprachen ist in der App-Entwicklung jedoch von höchster

Wichtigkeit. So werden z.B. für die Erstellung von Android-Applikationen die Sprachen Java und Kotlin verwendet [15].

Es könnte also versucht werden, eine Methode zur App-Entwicklung zu finden, welche die Kenntnisse der Medien- und Webgestalter anstelle von klassischen Programmiersprachen nutzt. Die Verwendung der Web-Sprachen HTML, CSS und JavaScript könnte die App-Entwicklung vereinfachen und dafür sorgen, dass diese in einem Rutsch ablaufen, wodurch die Kosten reduziert werden könnten [12]. Frau Haberer stellt dabei jedoch auch heraus, dass es in diesem Fall äußerst empfehlenswert ist, dass der Entwickler bei der Wahl seiner Tools nicht eingeschränkt wird [12]. Auch die Verwendung der häufig genutzten Internetbaukästen sollte unterstützt sein, um den Mediendesignern und Webdesignern keine unnötige Arbeit aufzubürden [12].

2.3 Ist-Analyse: App-Entwicklung

Für die Entwicklung einer App gibt es verschiedenste Ansätze. Darunter lassen sich auch Ansätze finden, welche tatsächlich auf eine App-Entwicklung mit den Web-Sprachen HTML, CSS und JavaScript setzen [18]. Im folgenden Abschnitt sollen verschiedene Vorgehensweisen der App-Entwicklung genauer unter die Lupe genommen werden und auf ihre Vor- und Nachteile geprüft werden. Dadurch könnte es möglich sein, Probleme der App-Entwicklung zu erkennen und gegebenenfalls Schlüsse zu ziehen, was getan werden muss um die App-Entwicklung zu erleichtern. Dafür werden drei Vorgehensmodelle analysiert. Die nativen Apps, Web-Apps und progressive Web-Apps (PWAs).

2.3.1 Native Applikation

Bei einer nativen Applikation handelt es sich um Anwendungen, welche konkret für eine Zielplattform wie Android oder iOS mithilfe der standardmäßigen Programmierschnittstellen entwickelt werden [3]. Bei Android werden dafür die Programmiersprachen Java und seit dem Jahr 2017 auch Kotlin verwendet [15]. Bei iOS kommen Objective-C und Swift als Programmiersprachen zum Einsatz [3]. Als Entwicklungsumgebung werden Android-Studio (Android) und Xcode (iOS) eingesetzt [3]. Daraus ergibt sich ein erster Nachteil der nativen Apps. Eine native App muss in einer spezifizierten Entwicklungsumgebung für

eine Plattform entwickelt werden [16]. Programmcode kann für andere Plattformen nicht oder nur in veränderter Form wiederverwendet werden [16, 17].

Dieser Umstand ist jedoch nicht nur ein Nachteil. Da die vorgesehenen Programmierschnittstellen für eine native Applikation verwendet werden, kann sie so den vollen Funktionsumfang nutzen, der für die Zielplattform verfügbar ist [3]. Zusätzlich ist es in der nativen App möglich, die gesamte Hardware in die App-Entwicklung einzubinden [16]. Dies bedeutet, dass eine native App auf die Kamera, das Mikrofon, das GPS und den Speicher in vollem Umfang zugreifen kann [16]. Auch wenn die Performanz einer Applikation von hoher Bedeutung ist, z.B. bei grafikintensiven Inhalten, so ist der native Weg in diesem Fall der beste [18].

Die Benutzbarkeit nativer Anwendungen ist ein weiterer Pluspunkt [19]. So hat eine das für eine native Applikation typische Look-and-Feel [19]. Dabei hilft auch die Möglichkeit alle Touch-Gesten definieren zu können, wodurch sich die Benutzbarkeit deutlich verbessern kann [19]. Zu guter Letzt sollte dabei auch nicht außer Acht gelassen werden, dass auch die Verteilung und Vermarktung nativer Anwendungen sehr leicht ist. Sie können in verschiedenen Anwendungsmarktplätzen angeboten werden, wo sie auch von zufälligen Nutzern gefunden werden können [19, 20].

Die Wahl zur Entwicklung einer nativen Applikation bringt jedoch nicht nur Vorteile mit sich. Eine native App ist mit einem großen Entwicklungsaufwand verbunden, da jede unterstützte Plattform einen eigenen Programmcode benötigt [19]. Da bei der Entwicklung nativer Anwendungen höhere Programmiersprachen verwendet werden, ist die Zusammenarbeit mit einem Software-Ingenieur unerlässlich [3, 12]. Für das Front-End empfiehlt es sich meist jedoch eher, einen ausgebildeten Designer einzusetzen, da diese am ehesten ein Verständnis dafür besitzen, wie eine moderne, einfach-bedienbare Benutzeroberfläche umzusetzen ist [12]. Der Entwicklungsaufwand steigt auch dadurch, dass eine große Menge verschiedener Betriebssysteme und Betriebssystem-Versionen unterstützt werden müssen [19]. Der hohe Entwicklungsaufwand führt auch zu höheren Kosten [12, 20].

Obwohl die Distribution von nativen Applikationen über den App-Store oft von Vorteil ist [19], so hat dies auch seine Haken. Um eine Anwendung in einem der App-Marktplätze anbieten zu können, müssen viele Voraussetzungen erfüllt sein, die durch die Betreiber der App-Stores überprüft werden müssen [20]. Dies führt dazu, dass es bis zu 14-Tage dauern kann, bis eine App in den jeweiligen Stores zugelassen wird [20]. Wird ein Fehler innerhalb einer dieser Applikationen entdeckt, so müsste für jedes Betriebssystem ein eigenes Update entwickelt werden, da sich die Codebasis je nach Plattform unterscheidet [19]. Auch in diesem Fall würde das Update zunächst erst vom App-Store-Betreiber geprüft werden, was wiederum 14-Tage dauern kann [19]. Eine solche Wartezeit kann gerade bei kritischen Fehlern sehr problematisch sein [19].

Die Verteilungsmethode von Anwendungen über den App-Store ist auch von daher diffizil, da ein Teil der Einnahmen an den Betreiber des App-Stores abgeführt werden muss [19]. In der Folge senken diese Gebühren die Einnahmen der Entwickler.

Vorteile	Nachteile
<ul style="list-style-type: none"> • Entwicklung in klassischen Programmiersprachen <ul style="list-style-type: none"> ○ Voller Hardwarezugriff ○ Voller Funktionsumfang • Performant • Gute Benutzbarkeit • Leichte Vermarktung 	<ul style="list-style-type: none"> • Entwicklung in klassischen Programmiersprachen <ul style="list-style-type: none"> ○ Braucht Programmierer ○ Ist aufwendig und somit teuer ○ Jede Plattform hat eigene Codebasis • Langwieriger Veröffentlichungsprozess (auch bei Updates)

Tabelle 1: Vor- und Nachteile von nativen Apps

2.3.2 Web-Apps

Im Gegensatz zu einer nativen App, handelt es sich bei der Web-App nur um eine Webseite, die innerhalb des Webbrowsers aufgerufen wird [17, 20]. Sie stellen somit keine Apps im klassischen Sinne dar [17]. Ein Kennzeichen zur Unterscheidung von normalen Webseiten ist dabei, dass sie dem Nutzer einen Mehrwert bei der Erfüllung einer Aufgabe bieten [21]. Beispielsweise dient Google Maps als Navigationssystem, während Google Mail E-Mail-Postfächer anbietet [21]. Sie können in der Regel über eine URL im Webbrowser erreicht

werden [18]. Aus diesem Grunde kann die Applikation auch ohne aufwendiges und zeitintensives App-Store-genehmigungsverfahren angeboten werden [19]. Erwähnenswert ist in diesem Zusammenhang auch, dass Web-Apps nicht auf dem Endgerät installiert werden müssen [3]. Die Daten sind in der Regel auf entfernten Servern abgelegt, wodurch Updates ohne Umwege durch Veränderungen an den auf den Servern gelagerten Daten durchgeführt werden können [19, 20]. Sobald der Nutzer das nächste Mal die Webseite öffnet, wird diese in aktuellster Form angezeigt, wenn das Offline-Caching dies zulässt [20].

Da es sich bei Web-Apps nur um Webseiten handelt, können diese Apps von einem größeren Personenkreis entwickelt werden. Denn reine Web-Apps benötigen lediglich Kenntnisse in den Web-Sprachen HTML, CSS und gegebenenfalls JavaScript [20]. Diese Sprachen sind nicht nur bei Software-Ingenieuren, sondern auch bei Web-Entwicklern und Mediendesignern verbreitet [12, 20]. Dadurch sind diese einfacher zu entwickeln, wodurch die Entwicklungskosten einer Web-App im Vergleich zu einer nativen App deutlich geringer ausfallen [20]. Ein weiterer Vorteil durch die Verwendung einer Webseite als Anwendung ist, dass eine Anwendung für mehrere Systeme gleichzeitig entwickelt werden kann [20]. Dafür müssen diese Systeme lediglich einen Browser besitzen, der die Features der Anwendung unterstützt [20]. Im Idealfall reicht es aus, geringfügige Anpassungen für die unterschiedlichen Zielsysteme und deren Browser vorzunehmen [20]. Der Code müsste in der Regel jedoch nicht für jedes System komplett neu entwickelt werden [20].

Dennoch sind auch Web-Apps nicht fehlerfrei. Die Entwicklung von Anwendungen dieser Art bringt nicht nur Vorteile mit sich. So haben Web-Apps keinen vollen Zugriff auf die Hardware von Geräten [20]. Sie können beispielsweise nicht auf die Kontakte des Geräts zugreifen [20]. Auch ein Zugriff auf die Kamera wird nicht von allen mobilen Betriebssystemen unterstützt [20]. Ein weiterer großer Nachteil von Web-Apps ergibt sich dadurch, dass sie ihre Inhalte von Servern laden müssen [18]. Gerade das Laden von größeren Dateien kann so dazu führen, dass die Bedienbarkeit der Web-App darunter leidet [3]. Zudem kann dieser Umstand dazu führen, dass bei Nutzern hohe Kosten für die Datennutzung anfallen, besonders, wenn die Applikation im Ausland

im Roaming-Modus verwendet wird [3, 18]. Problematisch ist hier ebenfalls, dass Web-Apps ohne Internetverbindung gar nicht oder nur eingeschränkt verwendet werden können [19].

Zuletzt soll noch auf die Benutzerfreundlichkeit von Web-Applikationen eingegangen werden. So ist es bei geeigneter Benutzeroberfläche kaum möglich, diese von nativen Apps zu unterscheiden [3]. Jedoch werden zum Beispiel nicht alle Touch-Gesten unterstützt, worunter die Bedienbarkeit leiden kann [19].

Vorteile	Nachteile
<ul style="list-style-type: none"> Entwicklung in den Web-Sprachen HTML, CSS und JavaScript <ul style="list-style-type: none"> Einfach zu lernen Weit verbreitet Plattformübergreifend verwendbar Keine Installation nötig Auslagerung von Code 	<ul style="list-style-type: none"> Kein voller Funktionsumfang nutzbar Kein voller Hardwarezugriff möglich Keine Vermarktung über App-Store Benutzerfreundlichkeit heikel Verbraucht teures Datenvolumen

Tabelle 2: Vor- und Nachteile einer Web-App

2.3.3 Progressive Web-Apps

Neben den klassischen Web-Apps existieren inzwischen auch die sogenannten Progressive Web Apps. PWAs sind eine Erweiterung der normalen Web-Apps [22], weshalb hier nur die Unterschiede zu klassischen Web-Anwendungen erläutert werden sollen. Das *Progressive* steht in diesem Kontext für *Progressive Enhancement* [22]. Damit ist gemeint, dass die Features neuer Browser voll genutzt werden sollen, ohne dass dabei andere Browser ausgeschlossen werden [22]. Für die Entwicklung von progressiven Web-Anwendungen steht daher ein größerer Funktionsumfang bereit [22]. So können progressive Web-Applikationen beispielsweise auch ohne Internetverbindung geladen werden, wenn diese zuvor schon einmal geladen wurde [22]. Bei guter Bandbreite lädt Android im Hintergrund bereits die für die PWAs benötigten Dateien herunter und hält diese im Cache möglichst aktuell [23].

Progressive Web-Apps können auf einen weiteren Funktionsumfang zugreifen, so ermöglicht eine Progressive Webanwendung das Senden von

Benachrichtigungen auf das Gerät und kann als Verknüpfung auf dem Startbildschirm hinzugefügt werden [22]. Zudem können PWAs in einem Browserfenster ohne Browsernavigationselemente geladen werden, wodurch sie das Gefühl einer nativen App vermitteln [22]. Unterstützt werden Progressive Web-Apps derzeit von Google und Mozilla [23].

Zwar bringen Progressive Web-Apps einige Verbesserungen im Vergleich zu klassischen Web-Apps, jedoch ist auch die Progressive Web-App auch nicht ohne Fehler. So befinden sich PWAs noch im Entwicklungsstadium, dadurch ist es nicht absehbar, um welche Funktionalitäten die progressiven Web-Applikationen noch erweitert werden [24]. Jedoch kann mit progressiven Web-Apps nicht der volle Funktionsumfang wie bei nativen Apps verwendet werden und es gilt derzeit als unwahrscheinlich, dass progressive Webanwendungen diesen Funktionsumfang je erreichen werden [24]. So sind z.B. Bluetooth-, NFC-, Kontakt- und Kalenderzugriff nicht möglich [24]. Auch das Einfügen von PWAs in die Übersicht installierter Anwendungen (App-Drawer) oder den Anwendungsmarktplatz sind nicht möglich [24].

Vorteile	Nachteile
<ul style="list-style-type: none"> Entwicklung in den Web-Sprachen HTML, CSS und JavaScript <ul style="list-style-type: none"> Einfach zu lernen Weit verbreitet Plattformübergreifend verwendbar Keine Installation nötig, aber möglich Auslagerung von Code Besserer Hardwarezugriff und Funktionszugriff als klassische Webapp 	<ul style="list-style-type: none"> Kein voller Funktionsumfang nutzbar Kein voller Hardwarezugriff möglich Keine Vermarktung über App-Store Benutzerfreundlichkeit manchmal heikel Verbraucht teures Datenvolumen, wenn App zum ersten Mal gestartet wird Zukunft ungewiss

Tabelle 3: Vor- und Nachteile von Progressive Web-Apps

3 Grundlagen

In diesem Kapitel sollen die Grundlagen zur Entwicklung des App-Frameworks geklärt werden, das in dieser Thesis prototypisch umgesetzt werden soll. Dabei ist das Kapitel in mehrere, unterschiedliche Schritte unterteilt. Nachdem im vorhergehenden Kapitel die Ist-Situation analysiert und auf verschiedene Aspekte der App-Entwicklung eingegangen wurde, sollen nun die Daten dieser Analysen ausgewertet werden. Auf Grundlage dieser Auswertung sollen zunächst Anforderungen an ein Framework für die App-Entwicklung definiert werden. Danach sollen bereits bestehende, zu den Anforderungen passende Frameworks darauf überprüft werden, wie sie aufgebaut sind und welche Ansätze diese verfolgen. Im Anschluss soll auf Grundlage der Anforderungsanalyse und der Analyse der bereits bestehenden Frameworks besprochen werden, welche Komponenten ein solches App-Framework benötigt und welche Werkzeuge nötig sind, um dieses umzusetzen.

3.1 Anforderungsanalyse

Um eine Anforderungsanalyse durchzuführen, muss zunächst geklärt werden, was eine Anforderungsanalyse ist und wie diese funktioniert. Die Antwort auf die erste Frage, kann folgendermaßen zusammengefasst werden: Eine Anforderungsanalyse beschreibt, was mit einem Projekt erreicht werden soll, also was für Ziele das Projekt verfolgt [25]. Den Weg zu diesem Ziel, also das *wie* wird mit der Anforderungsanalyse nicht beschrieben [25]. Dabei sollen auch Ansprüche ausgearbeitet werden, die nicht explizit ausgesprochen werden, aber von dem Ergebnis des Projekts erwartet werden [25]. Bernhard Hobel und Silke Schütte beschreiben dies in ihrem Buch *Business-Wissen Projektmanagement von A - Z: Kompetent entscheiden. Richtig handeln*. Wie folgt:

„Wenn man Sie um eine Tasse Kaffee bittet, wird implizit erwartet, dass diese Tasse Kaffee auch heiß ist — explizit wurde dies nicht ausgesprochen.“ [25]

In der Anforderungsanalyse wird ein Anforderungskatalog angefertigt, der alle Anforderungen an ein Projekt in einer Liste aufzählt [25]. Diese werden in fachliche, technische, methodische und organisatorische Anforderungen eingeteilt

[25]. Die Anforderungsanalyse erfolgt während oder unmittelbar nach der Zielsetzung des Projekts [25].

3.1.1 Wie wird eine Anforderungsanalyse umgesetzt?

Zunächst müssen ein Personenkreis und eine Methode für die Anforderungsanalyse festgelegt werden [25]. Im Falle dieser Thesis ist die Anforderungsanalyse jedoch ein Sonderfall. Ein Personenkreis existiert nicht wirklich, da für die Anforderungsspezifikation eine Analyse aus der Ist-Situation erstellt werden soll, die auf den Erkenntnissen aus dem zweiten Kapitel dieser Arbeit aufbaut. Daher kann lediglich eine Methode für die Anforderungsanalyse gewählt werden.

Im zweiten Schritt müssen die Anforderungen als Spezifikation erfasst werden [25]. Die Anforderungen müssen also in Schriftform umgesetzt werden. Dafür müssen bestimmte Regeln eingehalten werden [25]:

- Anforderungen müssen in einer möglichst einfachen, verständlichen Sprache beschrieben werden. Sowohl der Auftraggeber als auch der spätere Anwender muss verstehen, was damit gemeint wird.
- Anforderungen müssen eindeutig definiert sein. Es darf kein Platz für Spekulationen bleiben. Dennoch muss die Einfachheit der Anforderungen gewahrt werden.
- Anforderungen sollen begründet sein. Damit ist gemeint, dass der Nutzen einer Anforderung klar sein muss.
- Anforderungen müssen machbar sein.
- Es muss prüfbar sein, ob eine Anforderung erfüllt wurde.
- Anforderungen sollten identifizierbar sein, dafür kann man diese beispielsweise nummerieren [25].

Im dritten und letzten Schritt müssen Anforderungen bewertet und priorisiert werden [25]. Dabei können Anforderungen in grob drei Kategorien eingeordnet werden:

- Muss-Anforderung: Die Umsetzung dieser Anforderung ist zwingend. Erfolgt die Umsetzung einer solchen Anforderung nicht, wird das Endprodukt eventuell nicht abgenommen.

- **Soll-Anforderung:** Die Umsetzung dieser Anforderung ist nicht zwingend, aber wünschenswert. Die Umsetzung ist Teil des Projekts, aber für die Abnahme nicht entscheidend.
- **Kann-Anforderung:** Umsetzung ist nicht Teil des Projekts und somit nicht zwingend für eine Abnahme, eine Umsetzung wäre jedoch von Vorteil [25].

3.1.2 Anforderungen

Mit den in Kapitel 3.1.1 gewonnenen Erkenntnissen, kann nun eine Anforderungsanalyse im Rahmen dieser Bachelorthesis durchgeführt werden. Die Anforderungen, die hierbei gestellt werden, richten sich an ein Framework zur App-Entwicklung. Quelle dieser Anforderungsanalyse sind die Kapitel 2.2 und Kapitel 2.3 dieser Bachelorthesis. Die Anforderungen sollen hier in Tabellenform umgesetzt werden. Jede Anforderung erhält dabei eine eindeutige ID zur Identifikation der Anforderung, einen Titel, eine Beschreibung und eine Klassifikation. In der Klassifikation wird eine Einschätzung gegeben, ob die Anforderung für das Projekt eine Muss-, Soll- oder Kann-Anforderung ist.

Die Anforderungen definieren sich wie folgt (Tabelle 4):

ID	Titel	Beschreibung	Klasse
01	Entwicklung durch Web-Sprachen	Die Entwicklung von Apps soll hauptsächlich durch HTML, CSS und JavaScript geschehen.	Muss
02	Wiederverwendung	Quellcode von Anwendungen sollte für andere Betriebssysteme wiederverwendbar sein.	Soll
03	Voller Hardware-zugriff	Applikationen müssen im Rahmen der Möglichkeiten vollen Zugriff auf die Gerätehardware haben, wie native Applikationen (unter Berücksichtigung des Sicherheitsaspekts)	Muss
04	Voller Funktions-zugriff	Applikationen müssen im Rahmen der Möglichkeiten den gleichen Funktionsrahmen nutzen können, wie native Applikationen	Muss
05	Bedienbarkeit	Die mit diesem Framework erstellten Applikationen sollten sich wie eine native App anfühlen und auch so verhalten.	Soll
06	Einfache Verteilung	Applikationen die mit dem Framework erstellt werden, sollten in Anwendungsmarktplätzen angeboten werden können.	Soll

07	Auslagerung	Die Auslagerung von Web-Code sollte möglich sein, um leistungsschwache Geräte unterstützen zu können	Soll
08	Performanz	Aufgaben sollten parallelisiert werden können, um eine gute Performanz der App zu erzielen.	Soll
09	Flexibilität (App-Entwicklung)	Die App-Entwicklung mit dem Framework flexibel sein. Es sollte möglich sein, jeden HTML-Code (online/offline) als Applikation zu nutzen.	Soll
10	Flexibilität (Erweiterbarkeit)	Das Framework sollte leicht um (native) Komponenten erweitert werden können, auf die im Web-Code zugegriffen werden kann.	Soll
11	Einfachheit	Die Einbindung des Frameworks in bestehende Projekte sollte möglichst einfach sein. Auch die Arbeit mit dem Framework sollte möglichst einfach sein, ohne dabei den Funktionsumfang einzuschränken.	Soll
12	Funktionsumfang	Das Framework wichtige immer wiederkehrende Methoden implementiert haben, wodurch redundante Programmierung vermieden werden kann.	Kann

Tabelle 4: Anforderungskatalog an ein Framework für App-Entwicklung

Die erstellten Anforderungen wurden mit dem Chef-Entwickler von Koppert Mikromarketing-Systeme überprüft und validiert. Die Anforderungen 01 bis 08 ergeben sich dabei aus der Analyse der Entwicklungsmodelle (Kapitel 2.3). Die Anforderungen 09 bis 12 sind die Anforderungen, die aus dem Experteninterview stammen. Die Vorgehensweisen zur App-Entwicklung aus Kapitel 2.3 sollen nun mithilfe des Anforderungskatalogs erneut analysiert werden. Wobei die letzten Anforderungen zunächst vernachlässigt werden sollen, da diese explizit die Implementierung des prototypischen Frameworks betreffen und somit nicht auf die Entwicklungsmethoden anwendbar sind. Hierbei soll noch einmal überprüft werden, inwieweit die entsprechenden Entwicklungsmethoden sich mit den Anforderungen decken. So soll eine schnelle Übersicht darüber gegeben werden, in welcher Methode die Vorteile die Nachteile am ehesten überwiegen. So kann ermittelt werden, welche dieser Methoden eine geeignete Basis für die Entwicklung eines Frameworks bietet.

Die Anforderungen aus Tabelle 4 werden von den in Kapitel 2.3 genannten Entwicklungsmodellen in folgenderweise abgedeckt:

Anforderung	Native App	Web-App	Progressive Web-App
Entwicklung durch Web-Sprachen			
Wiederverwendung			
Voller Hardwarezugriff			
Voller Funktionszugriff			
Bedienbarkeit			
Einfache Verteilung			
Auslagerung			
Performanz			
Legende: ■ = ganz erfüllt, ■ = teilweise erfüllt, ■ = nicht/kaum erfüllt			

Tabelle 5: Überprüfung der Entwicklungsmodelle für Apps aus Kapitel 2.3 um deren Übereinstimmung mit den Anforderungen zu analysieren

Wie in Tabelle 2 gesehen werden kann, ist die Situation nicht ganz so einfach. Native Apps erfüllen fünf Anforderungen komplett und drei überhaupt nicht. Klassische Web-Apps erfüllen hingegen drei Anforderungen, fünf Anforderungen jedoch nicht. Progressive Web-Apps erfüllen jeweils drei Anforderungen ganz und teilweise, während sie nur zwei nicht unterstützen. Je nach Prioritäten des Entwicklers und Projekts eignet sich also immer eine andere Herangehensweise in der Entwicklung von Smartphone-Anwendungen. Was sich jedoch auch zeigt ist, dass alle Anforderungen erfüllt werden könnten, wenn eine native Applikation mit einer Web-App verschmolzen werden würde. Hier kommen sogenannte Hybrid-Applikationen ins Spiel.

3.2 Hybrid-Apps

Hybrid Apps sind eine Sonderform von Applikationen. In ihr werden die Vorteile von nativen Apps und Web-Apps vereint [3]. Dadurch erfüllen hybride Apps einen Großteil der in Tabelle 4 aufgestellten Anforderungen. Hybrid-Applikationen werden primär in den Web-Sprachen HTML (HTML5), CSS (CSS3) und JavaScript geschrieben [18]. Der Web-Code wird dann in einem für den Nutzer nicht sichtbaren Webbrowser-Container ausgeführt [3]. Bei diesem Container handelt es sich um eine sogenannte WebView. Die WebView dient in nativen Apps dazu, Code für webbasierte Sprachen auszuführen, ohne dass dabei die typischen Webbrowser-Elemente, wie eine Navigations-, Tool- oder Scroll-Leisten angezeigt werden [18].

3.2.1 Struktur von Hybrid-Apps

Die WebView wird in eine native Applikation eingebettet [18]. Sie führt HTML-Code aus, der die Sichten (Views) und Funktionalitäten beinhaltet [18]. Als Programmiersprache kann im WebView-Code die Scriptsprache JavaScript verwendet werden [18]. Die Datenspeicherung findet entweder auf Servern oder in dem Speicher der nativen App bzw. des Geräts selbst statt [18].

Da es sich bei dieser WebView um ein natives Element handelt, ist es über eine JavaScript-Java-Bridge (JavaScript-Java-Schnittstelle) theoretisch möglich, zwischen nativem und Web-Code zu kommunizieren und Daten auszutauschen (siehe Abbildung 3) [18].

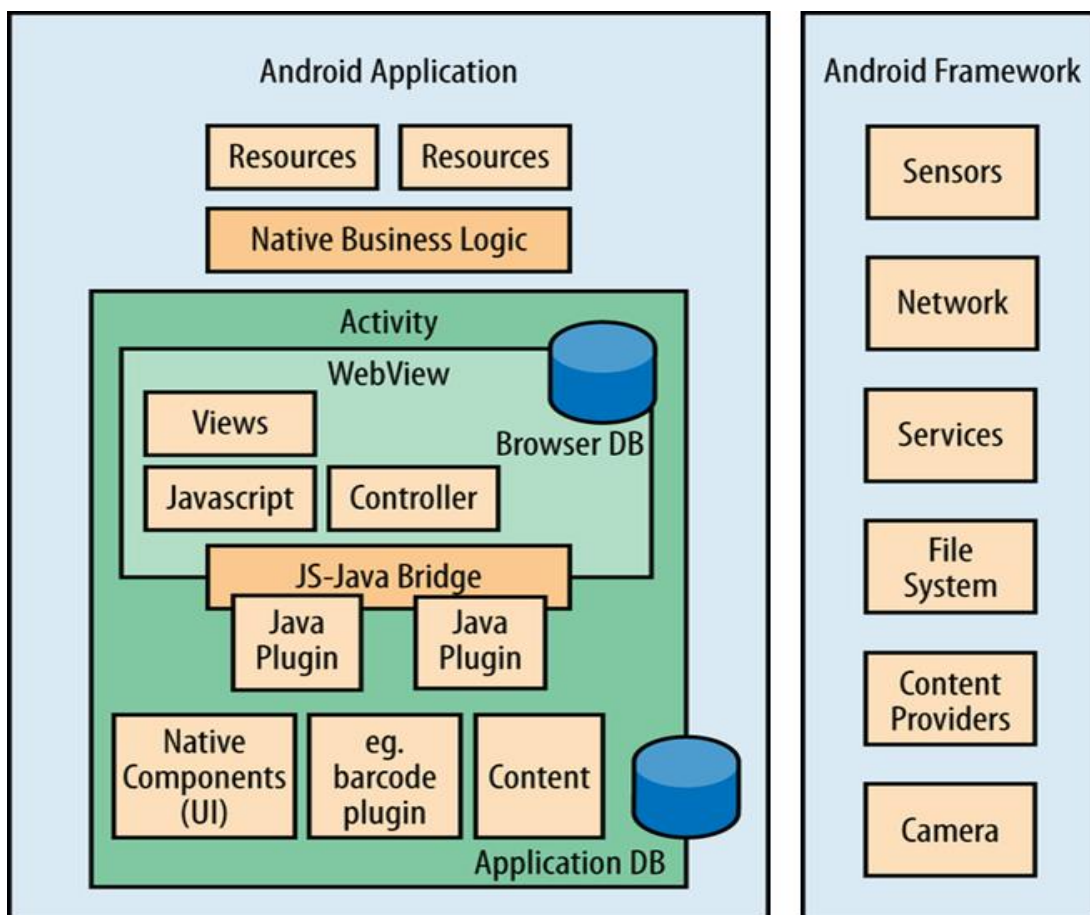


Abbildung 3: Struktur einer Hybrid-Applikation³

³ Quelle: Nizamettin Gok, Nitin Khanna: *Building Hybrid Android Apps with Java and JavaScript*, Newton (Massachusetts/USA): O'Reilly Media, Inc. (2013), ISBN: 1449361897

Dies erweitert die eingebettete Web-App um Funktionen die normalerweise nativen Apps vorbehalten sind. Auch ein voller Hardware-Zugriff ist so möglich [3, 18]. So ist es auch stets möglich, Neuerungen in neuen SDK-Versionen in die Web-App zu übernehmen [18]. Die native Wrapper-Applikation ermöglicht außerdem, dass die Applikation über Anwendungsmarktplätze verteilt werden können, wodurch Nutzer diese dort suchen und finden können [18].

Durch die Verwendung der Web-Sprachen als primäre Entwicklungssprache, können Komponenten wiederverwendet werden [18]. So muss nicht für jede Plattform eine vollständig eigene Applikation angewendet werden, da Code auch in anderen Versionen angewendet werden kann [18]. Aufgrund des geringeren Wartungs- und Entwicklungsaufwands ist die Erstellung von Hybrid Apps kostengünstiger [18]. Auch die problematischen Prüfungszeiten von Updates in den App-Stores können so umgangen werden, wenn der zu erneuernde Code auf dem Server liegt [18]. Auch ist es hier möglich, HTML-Code ohne Internetverbindung lokal auszuführen [18]. Die native App kann so direkt geladen werden und lokalen Inhalt bereits darstellen, ohne dabei auf die Internetverbindung warten zu müssen [18].

Ein weiterer negativer Punkt von Web-Apps kann in Hybrid Apps ausgehebelt werden. Die Ausführung von JavaScript erfolgt in der Regel auf einem einzigen Thread [18], dies bedeutet, dass stets nur eine Operation gleichzeitig ausgeführt werden kann. Durch die Verwendung von Threads im nativen Code können Operationen parallel ausgeführt werden, deren Ergebnisse, falls nötig, nach Beendigung an den JavaScript-Code übergeben werden können [18].

Da Hybrid-Applikationen die Vorteile von nativer und Web-App in sich vereinen können und größtenteils mittels HTML, CSS und JavaScript erstellt werden können [18], soll das in dieser Thesis angefertigte Framework ein Hybrid-App-Framework sein. Um ein besseres Verständnis für diese zu erhalten wie Hybrid-App-Frameworks sollen zwei dieser Frameworks untersucht werden.

3.2.2 Apache Cordova und Phonegap

Apache Cordova und Adobe Phonegap sind Code-Basen und Bridges (Schnittstellen) für die Entwicklung von Hybrid-Applikationen [40], wobei Phonegap einen Fork von Cordova darstellt [41]. Sie ermöglichen eingebetteten

Web-Apps, Zugriff auf native Funktionalitäten zu erhalten [40]. Sie bieten Schnittstellen für z.B. die Verwendung von Benachrichtigungen, Kamera oder Zugriff auf das Dateisystem an [40]. Zusammengefasst kann gesagt werden, dass Cordova und Phonegap eine Hybrid-App-Entwicklung über mehrere Plattformen hinweg ermöglichen [40]. Dabei setzen die Frameworks hauptsächlich auf eine Umsetzung von App-Konzepten mit den Web-Technologien HTML, CSS und JavaScript [40]. Die Frameworks ermöglichen einen grundlegenden Zugriff von JavaScript auf native Funktionen und Sensoren [40].

Daneben sind Phonegap und Cordova außerdem für die finale Erzeugung einer App als APK für Android oder IPA für iOS verantwortlich [40]. PhoneGap bietet unter anderem eine Unterstützung für die Entwicklung von Apps in Android, iOS, Windows Phone 7/8, Blackberry, Symbian OS und Tizen an [42]. Cordova unterstützt die Entwicklung für Android, iOS und Windows [41]. Cordova und Phonegap bieten eine plattformübergreifende Entwicklung mit Web-Technologien an [40]. Sie stellen die dafür nötigen Brücken zum nativen Code bereit und übernehmen später das Einbetten von Web-Apps in native Wrapper-Applikationen [40].

Apache Cordova und Adobe Phonegap bieten nur grundlegende Zugriffe auf native Funktionen und Sensoren an, jedoch können Plugins heruntergeladen und installiert werden, welche diese Funktionen erweitern [40]. Sollten spezielle Plugins nicht existieren, so können über die systemspezifischen Programmiersprachen (z.B. Java in Android) eigene Plugins entwickelt und zur Verteilung hochgeladen werden [40]. Leider kommt es immer wieder vor, dass angebotene Plugins fehlerbehaftet sind oder nicht hinreichend unterstützt werden, in diesen Fällen ist eine Anpassung vonnöten [41, 42]. Eine weitere Kritik an Cordova und Phonegap ist, dass häufig sehr viel JavaScript-Programmierung nötig ist, um die entsprechenden Sensoren anzusprechen [18].

Dennoch eignen sich Phonegap und Cordova für die Entwicklung eigener Hybrid-App-Frameworks. Als ausgereifte Frameworks kann die grobe Struktur als Leitfaden zur eigenen Entwicklung verwendet werden. Auch das Hybrid-App-Framework, welches im Laufe dieser Thesis entwickelt werden soll, wird im Wesentlichen aus einer Web-App eingebettet in eine native Applikation

bestehen. Auch sollen ebenfalls Schnittstellen zwischen nativem und Web-Code implementiert werden. Hier unterscheidet sich der Ansatz jedoch ein wenig. Statt Plugins zu schreiben, sollen grundlegend Brücken in beide Richtungen gebaut werden. Das in dieser Thesis umzusetzende Framework soll in der Lage sein von Web-Code nativem Code auszuführen, aber auch die Kommunikation von nativem Code zu Web-Code soll ermöglicht werden.

Dabei soll versucht werden, die Schnittstellen so zu gestalten, dass die Implementierung dieser Brücke für App-Entwickler kaum eine Rolle spielt. Die JavaScript-Java-Bridge soll eine Möglichkeit erhalten, flexibel zusätzlichen Code bereitzustellen. Dies könnte zu zwei potentiellen Vorteilen für die App-Entwickler führen. Das Framework richtet sich in allererster Linie an Mediendesigner und Webentwickler. Da diese meist keine traditionellen Programmiersprachen beherrschen [12], ist die Einbindung und Entwicklung von Plugins womöglich eine Aufgabe, die ohne Programmierkenntnisse nur schwer zu bewältigen ist. Sollte dennoch zusätzlicher nativer Code als Komponente gebraucht werden, so könnte ein Programmierer diesen unabhängig von der Implementierung des Front-Ends und somit ohne große Absprache entwickeln.

3.3 Komponenten des Hybrid-App-Frameworks

Auf Abbildung 3 in Kapitel 3.2 kann nachvollzogen werden, wie eine hybride Applikation aufgebaut ist. Auch der Aufbau bereits bestehender Hybrid-App-Frameworks, wie Cordova oder Phonegap geben Aufschluss darüber, wie ein Hybrid-App-Framework aufgebaut ist. Durch eine Analyse können Rückschlüsse daraus gezogen, welche Komponenten ein Hybrid-App-Framework benötigt. Die geplante Struktur des Hybrid-App-Frameworks, dessen prototypische Umsetzung in Kapitel 4 folgt, kann auf Abbildung 4 eingesehen werden.

Die Kernkomponente eines solchen Frameworks ist die JS-Java-Bridge [18]. Sie ermöglicht die Kommunikation und den Datenaustausch zwischen JavaScript und in diesem Fall nativem Java-Code [18]. Nur über die JS-Java-Bridge ist es möglich, die Web-App-Funktionalitäten um native zu erweitern [18]. In ihr sollten die Methoden und Funktionen bereitstehen, die ein Hinzufügen, Verändern oder Löschen von nativem Code aus dem JavaScript-Code erlauben. Das in dieser Thesis erstellte Framework soll jedoch auch Zugriff auf

den Web-Code aus dem nativen Code haben, deshalb sollen auch hier entsprechende Methoden in der JavaScript-Java-Bridge implementiert werden.

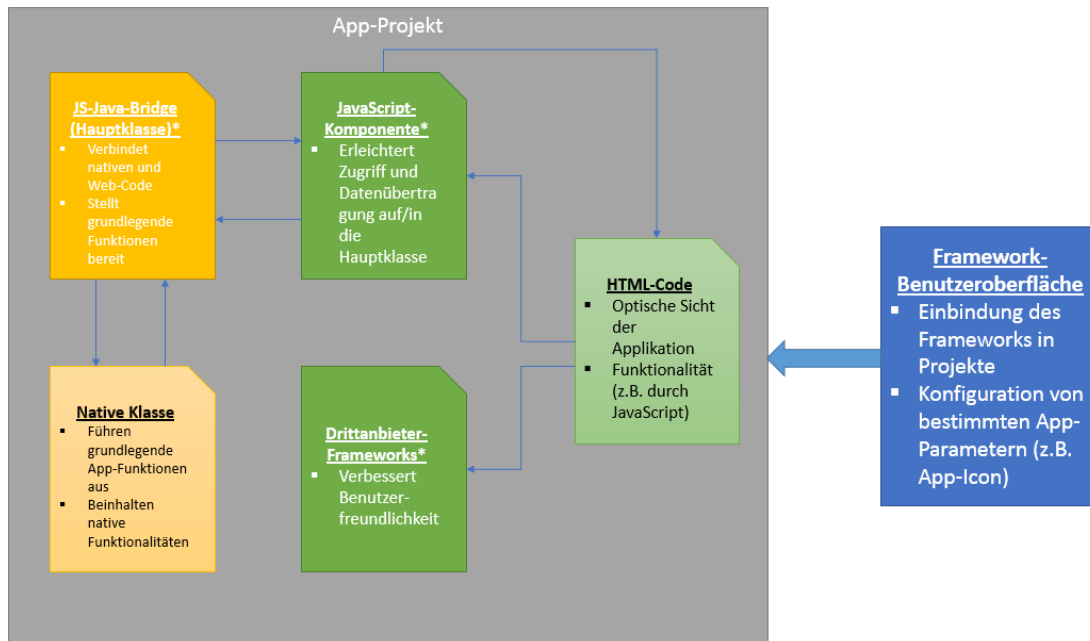


Abbildung 4: Geplante Hybrid-App-Framework-Struktur

Gleichzeitig empfiehlt es sich, hier auch eine JavaScript-Datei vorzusehen, durch welche die Arbeit mit dem Framework im Allgemeinen erleichtert werden könnte. Ist das Aufrufen von nativen Methoden und Funktionen zu kompliziert, so stört diese eventuell den Arbeitsfluss des Entwicklers. In einem solchen Fall helfen JavaScript-Wrapper-Methoden (siehe Programmcode 1), welche im inneren den Code für den Aufruf nativer Komponenten und Methoden enthalten, selbst aber über einen simplen Methodenaufruf gestartet werden.

```
function exampleFunction(){
    NativeInterface.nativeFunction();
}
```

Programmcode 1: Pseudocode einer JavaScript-Wrapper-Methode

Die Framework-Hauptklasse (JS-Java-Brücke) und die JavaScript-Komponente sind jedoch nicht die einzigen wichtigen Komponenten eines solchen Frameworks. Da dieses Framework auch an Mediendesigner und Webdesigner gerichtet sein soll, ist es von Vorteil auch andere kompliziertere Prozesse zu vereinfachen. Das Festlegen des App-Icons, des App-Titels und des App-Themes geschehen innerhalb der Android-Manifest-Datei [18]. Diese ist in

XML geschrieben und erfordert neben XML-Kenntnissen natürlich auch das Wissen, welcher Wert wie verändert werden muss. Auch die Erteilung von Berechtigungen oder das Setzen des WebView-Inhalts, könnte so manchen Entwickler ohne Programmiererfahrung in Java herausfordern. Um dies zu verhindern, könnte eine Benutzeroberfläche erstellt werden, in welcher die Grundkonfiguration und die Einbindung des Frameworks in ein App-Projekt automatisiert werden könnten.

Zu guter Letzt sollen auch Drittanbieter-Komponenten durch das Hybrid-App-Framework mitgeliefert werden. Die Einbindung weiterer CSS und JavaScript-Frameworks in das Hybrid-App-Framework kann dem Entwickler einige Arbeit abnehmen und weitere Funktionalitäten hinzufügen [16]. Beispielsweise können durch das JavaScript-Framework Hammer.js typische Smartphone-Steuerungs-Gesten wie Swipe, Pan, Pinch oder Zoom hinzugefügt werden [16]. Das Fehlen solcher Gesten wurde im Kontext der Web-Apps in Kapitel 2.3.2 als Nachteil für die Usability herausgestellt und sollte daher nicht fehlen.

3.4 Werkzeuge für die Entwicklung

Nachdem nun die Entwicklung eines Hybrid-App-Frameworks und die gewünschte Entwicklungsstruktur beschlossen wurden, muss nun darauf eingegangen werden, mit welchen Mitteln die prototypische Umsetzung gelingen kann. Dafür müssen verschiedene Werkzeuge ausgewählt werden, die bei der Umsetzung hilfreich sein können oder gar benötigt werden. Dabei muss auch festgelegt werden, auf welchem mobilen Betriebssystem die Apps funktionieren sollen, die mit dem prototypischen Framework erstellt werden sollen. Da Betriebssysteme oft eine eigene Codebasis und Projektstruktur haben, müssen gewisse Komponenten stets gezielt für das Zielsystem entwickelt werden, in diesem Fall muss die Schnittstelle zwischen nativem und Web-Code spezifisch für eine Zielplattform entwickelt werden. [18, 19].

3.4.1 Betriebssystem

Zunächst muss festgelegt werden, für welches Betriebssystem die prototypische Umsetzung erfolgen soll. Dafür soll zunächst herausgefunden werden, welche die wichtigsten mobilen Betriebssysteme sind und wie stark diese vertreten sind. Um diese Analyse durchzuführen wird eine weitere Statistik von

Statcounter zu Rate gezogen. In dieser Statistik werden die mobilen Betriebssysteme aufgelistet, welche auf den Geräten installiert sind, die auf Partner-Webseiten von Statcounter zugreifen.

Das Diagramm in Abbildung 5 zeigt die Entwicklung der Verteilung der einzelnen mobilen Betriebssysteme vom Januar 2010 bis Oktober 2019. Zu Beginn dieses Zeitraumes waren noch mobile Betriebssysteme von Bedeutung, welche heute kaum noch benutzt werden, wie etwa Symbian OS mit 34,16% oder BlackBerry OS mit 20,28% [26]. Im Jahr 2019 sind nur noch zwei mobile Betriebssysteme mit nennenswerten Marktanteilen übriggeblieben. Dabei handelt es sich um Googles Android mit 76,67% und Apples iOS 22,09% [26]. Weit abgeschlagen lässt sich noch KaiOS (0,42%) finden.

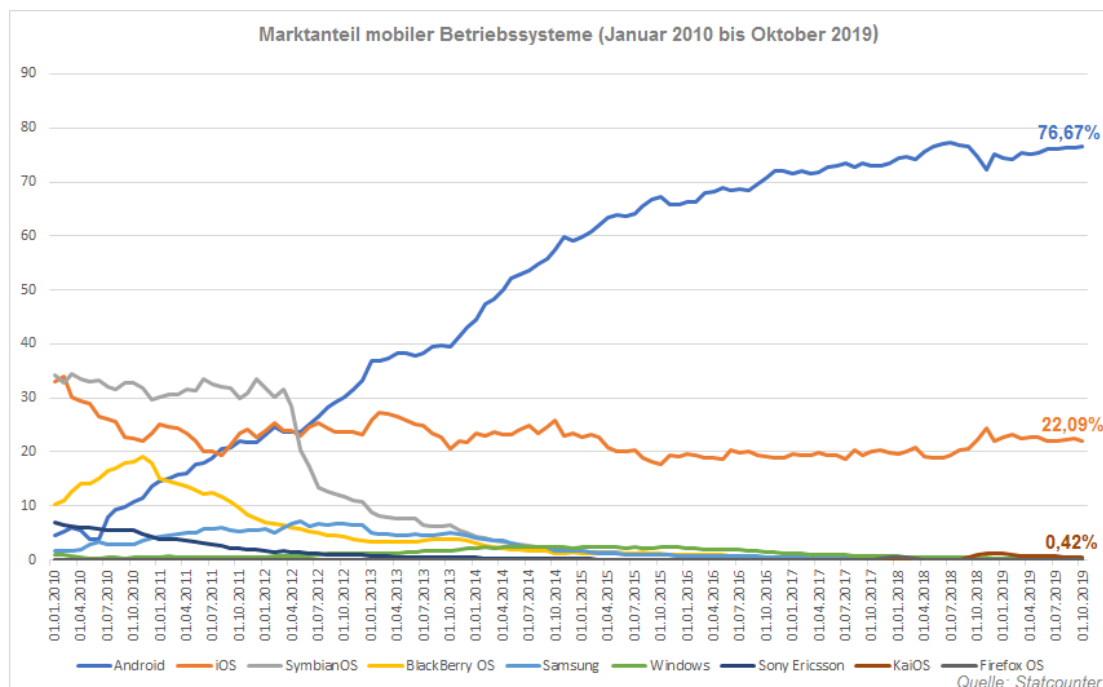


Abbildung 5: Marktanteil mobiler Betriebssysteme weltweit (2010-2019)⁴

3.4.1.1 KaiOS

Bei KaiOS handelt es sich um ein hauptsächlich auf Feature-Phones funktionierendes, mobiles Betriebssystem [27]. KaiOS basiert auf FirefoxOS, einem Open-Source-Betriebssystem der Mozilla Foundation aus dem Jahr 2015 [27].

⁴ Quelle: <https://gs.statcounter.com/platform-marketshare/desktop-mobile/worldwide/#monthly-200912-201912>

Nach eigener Aussage ist es das Ziel von KaiOS den vier Billionen Menschen die keinen Internetzugriff haben, durch günstige Feature-Phones Zugriff auf das Internet zu verschaffen [28]. KaiOS bietet Funktionen wie LTE-, GPS und WIFI-Zugriff [29]. Die Applikationen werden nicht in nativem Code, sondern in HTML5 geschrieben und sind somit Web-Apps [28, 30]. Im Rahmen dieser Thesis ist die Entwicklung für KaiOS jedoch nicht sinnvoll, zum einen werden Applikationen bei KaiOS ohnehin in HTML5 geschrieben, wodurch ein Hybrid-App-Framework nicht zielführend wäre, zum anderen ist die Verbreitung mit 0,42% [26] im Oktober 2019 äußerst gering.

3.4.1.2 iOS

iOS ist die Abkürzung für iPhone OS und kommt auf iPhones, iPads und iPods zum Einsatz [3]. Die Entwicklung erfolgte durch den Technologiehersteller Apple [3]. Das Betriebssystem basiert auf dem Computer-Betriebssystem MacOS(X), dieses basiert wiederum auf dem UNIX-Betriebssystem [3]. Die Marktanteile von Apples iOS liegen in Deutschland weit hinter denen von Android [3] und auch weltweit kam iOS im Oktober 2019 nur auf einen Marktanteil von 22,09% [26]. Dennoch nimmt iOS im E-Commerce eine besondere Stellung ein, da die Mobile-Commerce-Umsätze von iOS vor denen von Android liegen [3]. Ein Vorteil von iOS ist, dass Geräte und Betriebssystem vom gleichen Hersteller kommen, wodurch eine bessere Performanz erreicht werden kann [3]. Ein wesentlicher Nachteil des Betriebssystems ist die Geschlossenheit des Systems, was zur Sicherheit beitragen soll, jedoch enorme Einschränkungen nach sich zieht [3]. Aufgrund der Geschlossenheit und der im Vergleich zur Android geringen Verbreitung von iOS ist eine Entwicklung des Hybrid-App-Frameworks für Android zu bevorzugen.

3.4.1.3 Android

Bei Android handelt es sich um ein kostenloses, mobiles Open-Source-Betriebssystem welches von Google entwickelt und vorangetrieben wird [18]. Die Entwicklung erfolgte im Jahr 2007 durch die Open Handset Alliance [3]. Es basiert auf einer mobilzentrierten Version des Linux Betriebssystems [18]. Android bietet Entwicklern umfassende Features und Bibliotheken zur Umsetzung von Applikationen [18]. Android steht Geräteherstellern kostenlos zur

Verfügung [3], was in zahlreichen Android-Distributionen von z.B. Samsung, Amazon oder Motorola mündete [18].

Der überwiegende Teil von Android ist Open-Source, es gibt jedoch auch proprietäre herstellerspezifische Komponenten in manchen Distributionen [3]. Durch die Offenheit des Systems ist ein großer Teil des Android-Codes für Anwendungsentwickler geöffnet, was die Anwendungsentwicklung für Android unterstützt [3]. Native Android-Applikationen können in Java, Kotlin oder C/C++ geschrieben werden [30].

Mit 76,67% ist Android nicht nur das weitverbreitetste Betriebssystem auf Smartphones, sondern auch noch etwas mehr als dreimal häufiger vertreten als sein größter Konkurrent [26]. Schon aus diesem Grund scheint ein Framework für Android als logischste Entscheidung, da eine Vielzahl der Endgeräte auf dieses System setzt. Auch einige Tablet-Hersteller setzen auf Android [3], wodurch mit Android bereits Teile der Plattformen Smartphones und Tablets abgedeckt würden. Die Offenheit des Systems könnte sich ebenfalls als Vorteil herausstellen.

3.4.2 Entwicklungsumgebungen

Da nun entschieden wurde, dass die Umsetzung für Android am sinnvollsten ist, muss nun geklärt werden, welche Entwicklungsumgebungen für die einzelnen Komponenten verwendet werden sollen. Dabei geht es um das die Hauptklasse (JS-Java-Bridge), das JavaScript-Framework und die grafische Benutzeroberfläche, die in dieser Thesis umgesetzt werden sollen.

3.4.2.1 Android Studio

Bei Android Studio handelt es sich um die offizielle, von Google selbst vorgestellte Entwicklungsumgebung zur Entwicklung von Android-Apps [31]. Zuvor wurde die Entwicklungsumgebung Eclipse mit den Android Development Tools verwendet [31]. Android Studio ist wie Android selbst ein Open-Source-Projekt [18,32] und basiert auf der IntelliJ IDEA [31]. Als offizielle Entwicklungsumgebung für Android Applikationen eignet sie sich bestens für die Umsetzung der Hauptklasse. Die Umsetzung der Hauptklasse soll innerhalb eines Android-Studio-Projekts geschehen, da dies einiges erleichtert. So können schon während der Implementierung Tests mit der Hauptklasse durchgeführt

werden. Ein Vorteil von Android-Studio ist außerdem, dass es Windows, MacOS, Linux und Google Chrome OS, also Plattformübergreifend installiert und verwendet werden kann [30].

3.4.2.2 Eclipse

Bei Eclipse handelt es sich um eine Java-Entwicklungsumgebung, die ursprünglich von IBM entwickelt wurde [32]. Seit dem Jahr 2004 wird die Entwicklungsumgebung jedoch durch die Non-Profit-Organisation *Eclipse Foundation* weiterentwickelt [32]. Auf Basis von Eclipse sind inzwischen einige weitere Entwicklungsumgebung, auch für andere Programmiersprachen, entstanden [32]. Ein Vorteil von Eclipse ist, dass auch diese Entwicklungsumgebung auf zahlreichen Betriebssystemen wie Windows, Linux und MacOS läuft und Java-Applikationen ebenfalls auf diesen Plattformen ausgeführt werden können [32]. Ein weiterer Vorteil von Eclipse ist, dass die Plattform Open-Source und somit kostenlos ist [32]. Mit Java und Eclipse soll die grafische Benutzeroberfläche des Hybrid-App-Frameworks entwickelt werden.

Um eine grafische Benutzeroberfläche umzusetzen, wurde zunächst eine Arbeit mit der Klassen-Bibliothek Swing angestrebt. Swing wurde jedoch bereits durch das neuere Java-GUI-Framework JavaFX abgelöst, welches die Erstellung von modernen plattformübergreifenden Benutzeroberflächen erlaubt [33]. Die Erstellung der Oberflächen erfolgt JavaFX über die leichte, deklarative Sprache FXML [33]. Diese erlaubt eine Definition von Benutzeroberflächen in XML, wobei dieser Code durch CSS und JavaScript-Code erweitert werden kann [33]. XML (Extensible Markup Language) ist, wie HTML auch, eine Auszeichnungssprache [48]. Dies bedeutet, dass Objekte durch Tags beschrieben werden und komplexere Objekte durch Schachtelung mehrerer Tags oder durch die Verwendung von Tag-Attributen erzeugt werden [48]. Um die Arbeit noch weiter zu vereinfachen kann auf den *Scene Builder* zurückgegriffen werden, der die Entwicklung von grafischen Benutzeroberflächen per Drag & Drop erlaubt [33]. Die Entwicklung der Logik erfolgt dann in Java [33].

3.4.2.3 Visual Studio Code

Bei Visual Studio Code handelt es sich um einen leichtgewichtigen Editor für Websprachen, der für die drei großen Betriebssysteme Windows, Linux und

MacOS angeboten wird [34, 35]. Visual Studio Code ist kostenlos, Open-Source, Code-zentriert und bietet eine Syntax-Erkennung (mit Kolorierung) unter anderem für HTML, CSS und JavaScript, aber auch für die CSS-Präprozessoren LESS und SASS an [34]. Damit eignet sich Visual Studio Code gut für das JavaScript-Framework, welches Teil des Hybrid-App-Frameworks werden soll. Außerdem kann Visual Studio Code verwendet werden, um mit HTML-Testdokumente für das Hybrid-Framework zu schreiben.

3.4.3 Drittanbieter-Frameworks

Um das Hybrid-App-Framework zu vervollständigen sollen auch Drittanbieter-Frameworks standardmäßig mit dem Hybrid-App-Framework mitgeliefert werden. Damit sollen dem Entwickler Tools an die Hand gegeben werden, welche die Entwicklung der Applikationen erleichtern sollen und die Benutzbarkeit für den Nutzer erhöhen. In dieser Angelegenheit wurden drei Frameworks näher betrachtet, wobei jedoch mit Hammer.js und Bootstrap nur zwei Frameworks den Weg in das Hybrid-App-Framework gefunden haben.

3.4.3.1 Hammer.js

Bei Hammer.js handelt es sich um eine JavaScript-Bibliothek, die verwendet werden kann, um Touch, Maus und Pointer-Events in HTML-Dokumente einzubinden [36]. Mit einer Größe von gerade einmal 7,34kB [36] eignet es sich gut, um Touch-Events für die HTML-Dokumente nachzuladen, welche als Front-End für die Hybrid-Applikationen dienen sollen. Um die Bibliothek benutzen zu können, muss lediglich der JavaScript-Code in die HTML-Datei eingebunden werden (siehe Programmcode 2) [36]. Bei Hammer.js bestehen keine Abhängigkeiten zu anderen Bibliotheken und Frameworks [36].

```
var hammertime = new Hammer(myElement, myOptions);
hammertime.on('pan', function(ev) {
    console.log(ev);
});
```

Programmcode 2: Erstellen eines Hammer.js-Events

Die Bibliothek unterstützt die Touch-Events Pan, Pinch, Press, Rotate, Swipe und Tap [36]. Bei der Benutzung muss eine Instanz von Hammer erstellt und das Element, dem das Event zugewiesen werden soll als Parameter

übergeben werden. Danach kann ein Listener mit einer bestimmten Aktion definiert werden (siehe Programmcode 2).

3.4.3.2 Ratchet.js

Bei Ratchet.js handelt es sich um ein App-Framework, durch welches Smartphone-Applikationen auf Basis von HTML, CSS und JavaScript erstellt werden können [37]. Das Framework beinhaltet kompilierte JavaScript und CSS-Dateien sowie ein Icon-Pack [37]. Das Framework beinhaltet zahlreiche Komponenten für Android und iOS [38].

Während der Implementierung des Hybrid-App-Frameworks wurden mehrere Test-Dokumente mit Ratchet.js und Bootstrap erstellt. Dabei wurde die Arbeit mit Bootstrap als einfacher empfunden, weshalb Ratchet.js schlussendlich nicht im Hybrid-App-Framework als Komponente mitgeliefert werden soll.

3.4.3.3 Bootstrap

Bei Twitter Bootstrap handelt es sich um ein CSS-Framework, welches zahlreiche Komponenten zur Erstellung von responsiven Webseiten bereitstellt [39]. Bootstrap ist ein mobile-first System, somit hat die Entwicklung für mobile Geräte Vorrang, jedoch kann das Framework auch bei Webseiten auf Tablets, Desktop-PCs oder sogar Fernseher angewendet werden [39]. Um Webseiten zu designen wurde das Konzept des Grid-Systems aus den Print-Medien übernommen [39]. Bootstrap ermöglicht eine Aufteilung des Viewports und von Container-Inhalten in bis zu 12 Spalten [39].

CSS-Frameworks erleichtern Entwicklern die Arbeit, da vorgefertigte Komponenten für den Bau der HTML-Dokumente verwendet werden können [39]. Zusätzlich enthält Bootstrap auch ein CSS-Grid-System, welches das responsive Design, also die Anpassung an unterschiedliche Bildschirmgrößen erleichtern soll [39]. Der Entwickler kann sich so auf das Design konzentrieren und muss keine eigenen Komponenten schreiben, welche eine Menge Entwicklungszeit in Anspruch nehmen können. [39]. In Abbildung 6 ist eine Reihe dieser Komponenten stellvertretend dargestellt [39]. Bootstrap umfasst jedoch deutlich mehr Komponenten.

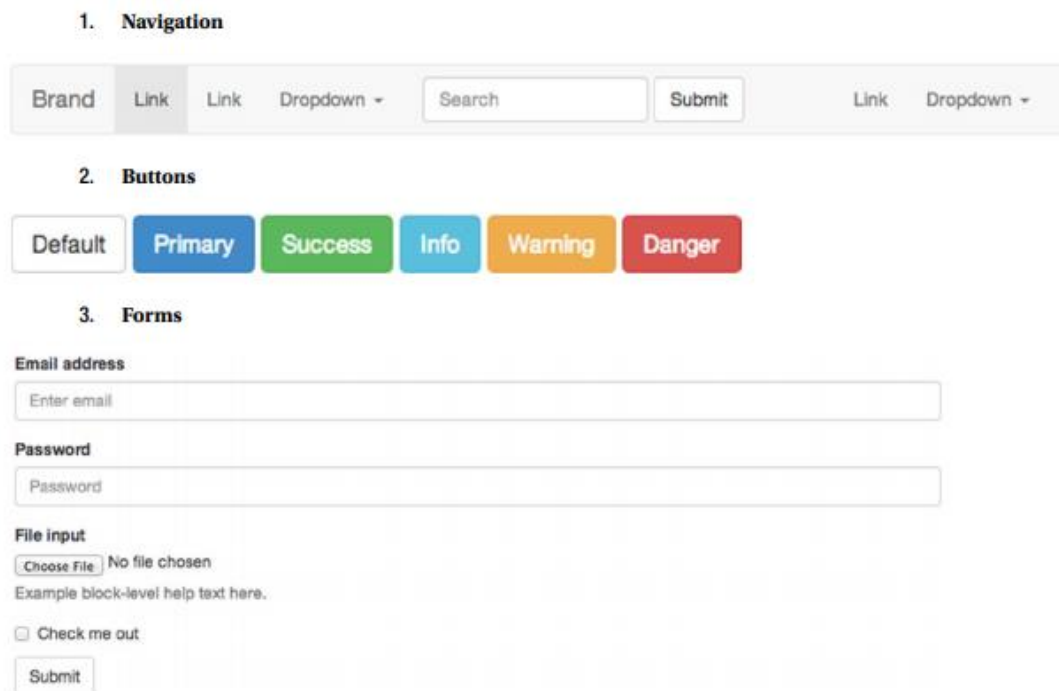


Abbildung 6: Übersicht einiger Bootstrap-Komponenten⁵

Um mit Bootstrap arbeiten zu können, muss Bootstrap zunächst heruntergeladen und in das HTML-Dokument eingebunden werden [39]. Für die Arbeit mit Bootstrap wird *jquery* benötigt, da es obwohl es ein CSS-Framework ist, auch JavaScript-Code enthält [39]. Dann muss die CSS-Datei im Head der HTML-Datei eingebunden werden, was folgendermaßen aussehen kann [39]:

```
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
```

Programmcode 3: Einbindung des CSS-Codes von Bootstrap

Die JavaScript-Dateien müssen in den Body der HTML-Datei eingebunden werden [39]:

```
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script src="js/bootstrap.min.js"></script>
```

Programmcode 4: Einbindung des JavaScript-Codes von Bootstrap

In der Theorie arbeitet der Entwickler dann ausschließlich mit HTML, da mit Bootstrap gearbeitet werden kann, ohne auch nur eine Zeile CSS oder JavaScript zu schreiben [39]. Um Beispielsweise einen Button zu erzeugen, wird

⁵ Quelle: Jonathan Fielding: *Beginning Responsive Web Design with HTML5 and CSS3*, Berkeley, Kalifornien (USA): Apress (2014), ISBN: 978-1-4302-6694-5

einem HTML-Element, z.B. dem Anker (a), einfach die Klasse „btn“ zugewiesen werden [39]:

```
<a href="#" class="btn btn-primary btn-lg" role="button">Learn more</a>
```

Programmcode 5: Einbindung eines Buttons mit Bootstrap

3.4.4 Zusammenfassung

Um geeignete Werkzeuge für die Umsetzung des Hybrid-App-Frameworks zu finden, wurden verschiedene Werkzeuge vorgestellt und einige Vor- und Nachteile dieser erläutert. Als Entwicklungs-Plattform wurde Android ausgewählt. Für die Umsetzung der Hauptklasse, der JavaScript-Java-Bridge, wurde Android-Studio ausgewählt. Die grafische Benutzeroberfläche soll mithilfe von Eclipse und JavaFX erstellt werden. Visual Studio Code soll dann eingesetzt werden, um das JavaScript-Framework zu implementieren und Test-HTML-Dokumente für das Framework zu entwerfen. Bei der automatischen Einbindung des zu erstellenden Android-Hybrid-App-Frameworks sollen dann auch Hammer.js für die Gestensteuerung der Applikation, sowie Bootstrap als CSS-Framework mit integriertem Grid-System eingebunden werden.

4 Prototypische Umsetzung

Nachdem entschieden wurde, eine prototypische Version eines Hybrid-App-Frameworks zu entwickeln, die Anforderungen an das Framework definiert wurden und nützliche Werkzeuge gewählt wurde, soll nun mit der Durchführung begonnen werden. Zunächst soll mit der Programmierung der Hauptklasse angefangen werden. Darin werden die Schnittstellenfunktionen aus dem nativen Code in den WebView-Code, aber auch die Schnittstellen für den Zugriff auf den nativen Code aus dem WebView-Code enthalten sein. Die Hauptklasse selbst soll zunächst als Teil eines Android-Studio-Projekts entwickelt werden. So ist es möglich den Code ohne Umwege zu testen. Bevor jedoch mit der Implementierung der Hauptklasse begonnen wird, soll zunächst eine kleine Abhandlung beschreiben, wie Android-Studio-Projekte aufgebaut sind, um diese besser begreifen zu können.

4.1 Android-Studio-Projekte

Für die Erstellung von Android-Studio-Projekten müssen zunächst drei Komponenten installiert werden, dabei handelt es sich um Android-Studio selbst, das Java Development Kit und das Android Software Development Kit [43]. Sind diese installiert, so kann mit der Erstellung eines Projekts begonnen werden [43]. Mit der Erstellung eines neuen Projekts werden eine ganze Reihe neuer Dateien und Ordner erstellt, von denen jede eine andere Aufgabe hat [43]. Ein Android-Studio-Projekt entspricht also einem Ordner mit vorgefertigten Dateien [43]. Jene Dateien, die für dieses Projekt wichtig sind, sollen hier also zunächst erklärt werden. Dabei ist zu beachten, dass folgend die Ordnerstruktur auf dem Dateisystem behandelt wird, welche sich von der Projektstruktur unterscheidet, die in Android-Studio angezeigt wird.

Die für dieses Projekt wichtigen Dateien und Ordner befinden sich allesamt im Unterordner *main* ([Projektname]\app\src\main), des Projektordners. Die erste wichtige Datei ist die Android-Manifest-Datei, dabei handelt es sich um eine XML-Datei [18]. Hier werden die Grundeinstellungen einer App festgelegt [18], auf diese soll im weiteren Verlauf des Kapitels jedoch gesondert eingegangen werden. Zusätzlich zu der App-Konfiguration dient die Android-Manifest zur Abfrage von Berechtigungen bei älteren Android-Versionen [18, 45]. Ab

Android 6.0 können Berechtigungen zur Laufzeit im Code abgefragt und gesetzt werden [44]. Nach der Statistik von Statcounter waren im Dezember 2019 noch immer 6,04% der Android Nutzer mit der Version 5.1 unterwegs und immerhin noch 2,16% mit Android 4.4 (siehe Abbildung 7) [45]. Die Definition der Berechtigungen in der Manifest-Datei ist also noch immer nötig. Sollte die App später über einen App-Store angeboten werden, so muss auch bei neueren Versionen in der Android-Manifest-Datei beschrieben werden, welche Berechtigungen die Applikation verwendet.

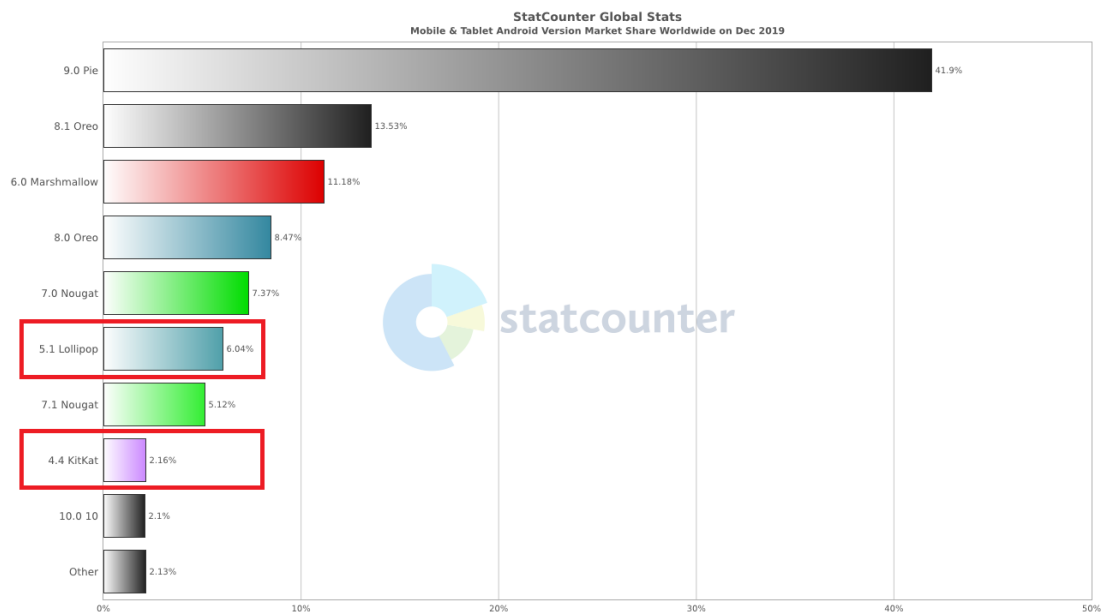


Abbildung 7: Android-Versionen im Überblick, Dezember 2019⁶

In diesem Projekt werden weitere XML-Dateien eine Rolle bei der Umsetzung des Projekts spielen. Dazu zählen die Dateien *colors*, *strings* und *styles*, welche sich im Ordner *values* (res/values) im Main-Ordner befinden [44]. In der Datei *colors* werden die Farben der App definiert [43]. Üblicherweise befinden sich hier die Werte *colorPrimary*, *colorPrimaryDark* und *colorAccent* [46]. Der Wert *colorPrimary* legt dabei die Farbe für die Titelleiste der App und andere primäre UI-Elemente fest [46]. Der Wert *colorPrimaryDark* wird seit Android 5 dazu verwendet, die Statusleiste des Geräts zu färben [46]. Der letzte Wert *colorAccent* wird verwendet, um sekundäre Elemente wie Checkboxes und

⁶ Quelle: <https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide#monthly-201912-201912-bar>

Textfelder in einer bestimmten Farbe darzustellen [46]. In der Datei *strings* werden nützliche Strings für Applikation wie z.B. der App-Name definiert [7, 18]. Die Datei *styles* legt das Thema der App fest, die Farbwerte hier sind zu vernachlässigen, da diese lediglich eine Referenz auf die Farben in der Datei *colors* enthalten [46]. Diese Werte sind in Form von Verweisen in der Android-Manifest eingebunden.

Der Ordner *res* ist dem Ordner *values* übergeordnet und enthält noch weitere wichtige XML-Dateien. Die Ordner *mipmap-hdpi*, *mipmap-mdpi*, *mipmap-xdpi*, *mipmap-xxdpi* und *mipmap-xxxdpi* enthalten die Bilddateien für die Icons, die in der Android-Manifest definiert werden [44, 45]. Üblicherweise tragen sie den Namen *ic_launcher* und *ic_launcher_round* [44]. Will man das App-Icon also ändern, so kann dies durch eine Veränderung der Bilddateien erreicht werden. Der letzte für dieses Projekt wichtige Ordner in *res* ist der Ordner *layout*. Hierbei handelt es sich um den Ordner, in denen die XML-Dateien der *Activities* liegen [43]. In diesen Dateien werden die Bildschirmseiten definiert, sie legen fest, was in den jeweiligen Seiten der App angezeigt werden soll [43]. Da das Augenmerk dieser Thesis auf einem Hybrid-Framework liegt, in dem die Sichten (Views) durch HTML-Code definiert werden, ist in diesem Fall nur die Datei *activity_main* (Standardname) wichtig, da in dieser später die WebView liegen wird, welche das Front-End beinhaltet [18, 44]. Dies ist die Seite, die dem Nutzer der App zunächst angezeigt wird [43].

Damit fehlt jetzt nur noch eine für das Projekt wichtige Datei. Diese befindet sich in einem Unterordner des Ordners *java*, welcher sich wiederum im Ordner *main* ([Projektname]\app\src\main) befindet. Der Name dieses Unterordners ist jedoch je nach Projektbenennung unterschiedlich. Bei der Datei handelt es sich um die Datei *MainActivity.java*. Die *MainActivity.java* enthält den Java-Code, der definiert, was beim Starten der App und bei der Interaktion mit den Elementen der Hauptaktivität passieren soll [47].

4.2 Entwicklung der Hauptklasse

Die Hauptklasse, welche die Schnittstellen zwischen dem nativen und dem Web-Code bereitstellt wird in Java geschrieben. Diese Klasse wird jedoch nicht einfach in die *MainActivity.java*, sondern in eigene Datei implementiert.

So soll der projekteigene, native Code vom Framework-Code getrennt werden, wodurch besser zwischen eigenem und Framework-Code unterschieden werden kann. Möglicherweise kann so auch das Testen vereinfacht werden. Zusätzlich soll so gewährleistet werden, dass die JavaScript-Java-Brücke später leichter in eine Vielzahl von unterschiedlichen Projekten integriert werden kann.

Diese Klasse erhält den Namen *JavaScriptJavaConnector*, da die Klasse als Brücke zwischen dem nativen Java-Code und dem WebView-JavaScript-Code bilden soll. Wichtig für die Erzeugung von Instanzen der Klasse ist der Konstruktor, dessen Implementierung nun genauer erläutert werden soll [49]. Dabei ist zu beachten, dass in den nachfolgenden Abschnitten nur die wichtigsten Methoden und Funktionen der Framework-Hauptklasse erläutert werden sollen. Der gesamte Umfang des Programmcodes kann auf der zur Bachelorarbeit beiliegenden CD eingesehen werden.

4.2.1 Der Konstruktor

Bei der Erzeugung einer neuen Instanz werden jeweils eine WebView und ein Context übertragen. Bei der WebView handelt es sich um das Java-Objekt der WebView. Dieses ermöglicht den Zugriff auf die Konfiguration und verschiedene Variablen der WebView. Bei dem Context handelt es sich um ein Objekt der Activity, in der sich die WebView befindet und in der eine Instanz des *JavaScriptJavaConnectors* erzeugt werden soll. Der Context wird später für zahlreiche native Funktionen, wie das Einblenden von Fehlermeldungen oder den Zugriff auf Hardware benötigt.

Innerhalb des Konstruktors werden auch zwei Methoden des *WebViewClient* überschrieben (siehe Programmcode 6). Dabei handelt es sich um die *onPageFinished*-Methode, welche immer dann ausgelöst wird, wenn eine Seite innerhalb der WebView vollständig geladen wurde [50]. Diese ist später für zwei Funktionen wichtig. Zum einen soll diese verhindern, dass JavaScript-Aufrufe vor dem vollständigen Laden der Webseite und dadurch meist nicht korrekt ausgeführt werden. Zum anderen sollen hier die aktuellen URLs extrahiert werden, welche in Kapitel 4.2.4.3 eine wichtige Rolle spielen. Die URLs sollen für ein Konzept zum dynamischen Einbinden nativer Funktionen aus

dem JavaScript verwendet werden. Bei der zweiten überschriebenen Methode handelt es sich um die *onReceivedError*-Methode. Damit soll später eine Methode implementiert werden, die es erlaubt eigene Fehler- oder Ersatzseiten zu laden, falls z.B. keine Internetverbindung besteht. Dies soll dem Nutzer eine bessere Usability bieten.

```
this.webView.setWebViewClient(new WebViewClient() {  
    @Override  
    public void onPageFinished(WebView view, String url) {  
        currentUrl = url;  
        if (url.contains("#")) splitUrlToCallMethod();  
    }  
    @Override  
    public void onReceivedError(WebView view, int errorCode,  
        String description, String failingUrl) {  
        super.onReceivedError(view, errorCode, description, failingUrl);  
        if (errorCode == -2)  
            if (!pageNotFoundUrl.equals("")) view.loadUrl(pageNotFoundUrl);  
    }  
});
```

Programmcode 6: Die *onPageFinished*- und *onReceivedError*-Methoden

Daneben enthält der Konstruktor noch weitere kleinere Funktionen. So werden einige *CallableFunctions* definiert, wobei deren Rolle im Laufe des Kapitels noch genauer erläutert werden soll. Die Ausführung von JavaScript-Code in einer Android-WebView ist standardmäßig deaktiviert [18]. JavaScript kann also erst ausgeführt werden, wenn die entsprechende Funktion dafür aktiviert wurde. Dieser Schritt wird innerhalb des Konstruktors ebenfalls automatisiert, da die Ausführung von JavaScript für das in dieser Thesis entstehende prototypische Framework unerlässlich ist. Sollte nachträglich dennoch gewünscht sein JavaScript zu deaktivieren, so kann diese Funktion einfach mittels der Methode *setJavaScriptEnabled* ausgeschaltet werden.

Zuletzt wird der WebView ein *JavaScript-Interface* zugewiesen. Dieses ermöglicht später den Zugriff auf Funktionen im nativen Code aus dem JavaScript-Code innerhalb der WebView [18]. Diese Funktionen werden mittels eigener Klasse innerhalb der Hauptklasse definiert.

4.2.2 Grundlegende Funktionen

Das Framework benötigt grundlegende Methoden, die es ermöglichen sollen, die wichtigsten Konfigurationen an der WebView vorzunehmen. Darunter fallen die Methoden *loadUrl* und *loadData*. Die Methode *loadUrl* erhält als Parameter einen String in dem eine Internetadresse steht. Diese Adresse wird dann in der WebView geladen. Dabei können sowohl lokale als auch Webseiten von entfernten Servern geladen werden [18]. Wird also die URL *https://google.de/* an die Methode übergeben, so wird die Webseite der Suchmaschine Google in der WebView geöffnet. Sollen lokale HTML-Dokumente geladen werden, so bietet es sich an, diese in den Android-Assets zu speichern.

Bei dem Android-Asset-Ordner handelt es sich um einen Unterordner im Ordner *main* ([Projektname]\app\src\main). In diesem können Dateien abgelegt werden, die später intern mit der App ausgeliefert und abgerufen werden sollen. Dieser bietet sich für die Speicherung von lokalen HTML-Dokumenten an, da so keine weiteren Berechtigungen z.B. Speicherzugriff zum Öffnen der HTML-Dokumente benötigt werden. Außerdem können so die HTML-Dokumente bei der Installation mitgeliefert werden [18], statt dass diese ins offene Dateisystem nachgeladen werden müssen. Um eine Seite aus den Assets zu laden muss in der URL lediglich ein Muster beibehalten werden *file:///android_asset/[filename]*, wobei *android_asset* für den Pfad des Assets-Ordners steht [18].

Die zweite der genannten Methoden ist *loadData* diese ermöglicht, HTML-Code direkt als String in die WebView zu übertragen und dort darzustellen. Diese Methode kann verwendet werden um Daten aus einem String im Java-Code als WebView-Inhalt zu laden. Um solchen Code vereinfacht zu erstellen, wurde die Klasse *HtmlNode* implementiert, welche in Kapitel 4.2.5 genauer besprochen werden soll. Hier können in Java ganze HTML-Dokumente erstellt und verschachtelt werden und diese als die Methode *HtmlNode.get()* in einen String konvertiert werden. Die Implementierung dieser Klasse wird in diesem Kapitel noch genauer erläutert.

Neben diesen grundlegenden Methoden existieren noch weitere Methoden. Diese Methoden dienen zur Konfiguration verschiedener WebView-

Einstellungen. Diese Methoden enthalten in ihren Namen stets eine Option der *WebView* und nehmen als Parameter einen Boolean an, der bestimmt ob die gleichnamige Option der *WebViewSettings* ein- oder ausgeschaltet sein soll.

4.2.3 Schnittstelle: Java- zu JavaScript-Code

Um aus dem nativen Code den Inhalt der *WebView*, welcher aus HTML, CSS und JavaScript besteht, zu erweitern, zu verändern oder zu löschen, sollen JavaScript-Injektionen verwendet werden. Um mittels JavaScript den gesamten *WebView*-Inhalt manipulieren zu können, wird eine Methode benötigt, welche die Ausführung von JavaScript-Code in der *WebView*, durch Injektionen aus dem nativen Java-Code zu ermöglicht. Dafür soll die Methode *loadUrl* verwendet werden. Wird statt einer Internet-Adresse z.B. folgendes eingegeben: *javascript: var a = "Hallo";* so wird weder eine neue Seite geladen, noch wird eine Fehlermeldung ausgegeben. Stattdessen wird eine JavaScript-Variable *a* mit dem Inhalt *Hallo* erstellt. Diese kann aus der JavaScript-Konsole des Browsers aufgerufen werden (siehe Abbildung 8).

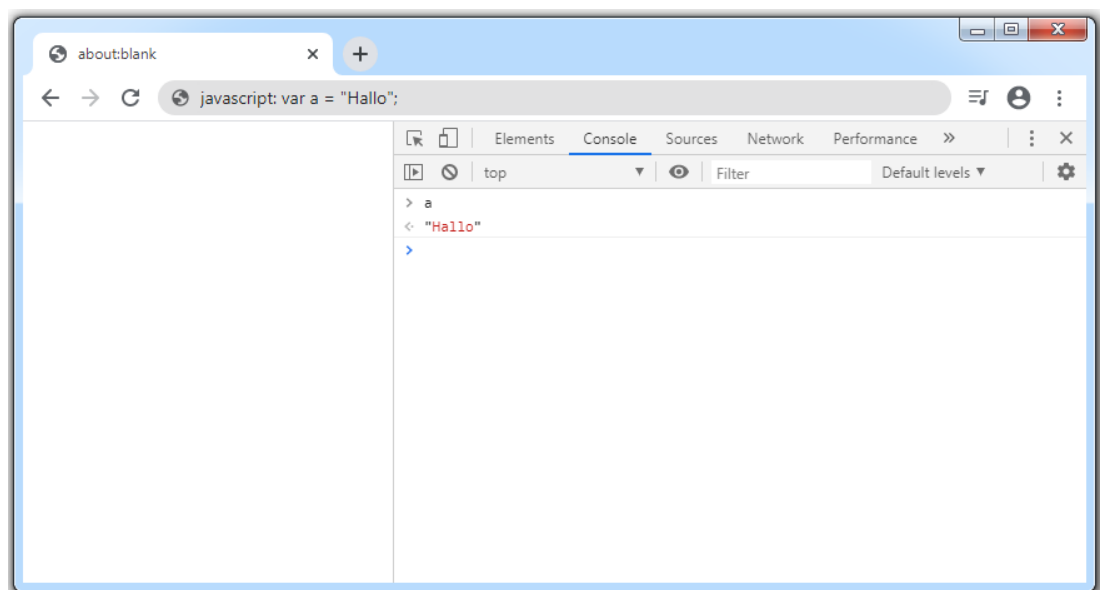


Abbildung 8: Ausführung von JavaScript-Code in der Adresszeile

Bei der *WebView*-Programmierung ersetzt *loadUrl* die Adresszeile des Browsers, da Webseiten mithilfe dieser Methode geladen werden können [18]. Um also aus dem nativen Java-Code, Zugriff auf den *WebView*-Code zu erhalten, wurde in der JavaScript-Java-Brücke eine Methode geschrieben, die auf die Methode *loadUrl* zugreift. Diese nennt sich *executeJavaScript* und erhält über

einen String einen JavaScript-Befehl. Innerhalb der Methode wird zunächst die Funktion *onUrlLoad* in einem neuen *WebViewClient* überschrieben. Der *WebViewClient* wird der *WebView* zugewiesen. So soll sichergestellt werden, dass erst nach dem vollständigen Laden der Webseite der JavaScript-Code ausgeführt wird, da ohne diese Methode JavaScript-Code oft nicht ausgeführt wurde.

Diese Probleme kamen auf, da die *WebView* in einem eigenen Thread (dt. Abhandlungsstrang) ausgeführt wird [18]. Die Benutzeroberfläche und Logik der Applikationen werden in einem Mainthread (dt. Hauptstrang) bearbeitet, der nicht blockiert werden sollte [18]. Wird also unmittelbar nach der Initialisierung der *WebView* ein JavaScript-Befehl übermittelt, so kann passieren, dass der Hauptthread diesen Befehl in der *WebView* ausführen lässt, bevor der *WebView*-Thread die Seite vollständig geladen hat.

An den JavaScript-Befehl wird noch ein *void(0);* angehängen (siehe Programmcode 7). Dies wurde deshalb getan, da die *WebView* während der Implementierung immer wieder den Rückgabewert einer JavaScript-Anfrage genutzt hat um diesen als Webseiten Quellcode zu verwenden. Um bei dem Beispiel aus diesem Kapitel zu bleiben: Als der Aufruf *javascript var a = "Hallo";* ausgeführt wurde, wurde der Wert *Hallo* als Inhalt des *WebView*-Bodys geladen. Die *WebView* lud also eine leere auf der das Wort *Hallo* angezeigt wurde.

```
loadUrl("javascript:" + cmd + "; void(0);");
```

Programmcode 7: Ausführung von JavaScript durch *loadUrl*

Als zweite Methode wird *evaluateJavaScript* verwendet. Diese ist eine standardmäßig implementierte Methode zum Ausführen von JavaScript und kann auch Rückgabewerte in sogenannten *ValueCallbacks* entgegennehmen. Die Methode existiert jedoch erst seit API 19 (Android 4.4) und kann daher erst ab dieser Version verwendet werden [51]. Aus diesem Grund prüft die Framework-Methode *executeJavaScript* zunächst, welche Android-Version auf einem Gerät installiert ist. Dann führt sie für Geräte unter Android API 19 die *loadUrl*-Methode aus, für Geräte darüber die *evaluateJavaScript*-Methode

4.2.4 Schnittstelle: JavaScript zu Java-Code

Wie nun gezeigt wurde, kann JavaScript-Code in der WebView sehr leicht aus dem nativen Code ausgeführt werden. Soll jedoch andersherum Java-Code aus dem WebView-Code ausgeführt werden, so ist dies deutlich schwerer. Grund dafür ist, dass die WebView, wie im vorhergehenden Abschnitt (4.2.3) erklärt, in einem eigenen Thread ausgeführt wird [18]. Die größte Schwierigkeit dadurch ist, Variablen und Werte aus dem Web-Code zu verwenden um native Methoden aufzurufen, mit diesen dort Threadsicher zu arbeiten und die WebView danach zu manipulieren.

Um herauszufinden, wie Methodenaufrufe und Datenübertragung aus dem JavaScript-Code in den nativen Java-Code Threadsicher durchgeführt werden können, mussten verschiedene Ansätze ausprobiert werden. Wichtig war, dass später flexibel eigene native Methoden dem Framework hinzugefügt werden können und diese Methoden threadsicher und mit vollem Funktionsumfang arbeiten können.

4.2.4.1 Ansatz 1: Variablen über JavaScript-Injektion übertragen

Dieser Ansatz beschäftigt sich zunächst nicht mit dem Aufruf nativer Funktionen aus dem WebView-Code, sondern lediglich mit der Übertragung von Daten aus dem WebView-Code in den nativen Code. Da bereits erklärt wurde, wie JavaScript aus dem Java-Code ausgeführt werden kann (siehe Kapitel 4.2.3), wurde zunächst der Ansatz verfolgt, Werte direkt durch JavaScript-Injektionen erhalten zu können. Benötigt eine native Methode oder Funktionen einen Wert aus dem WebView-Inhalt, so könnte man versuchen diesen als Rückgabewert eines JavaScript-Befehls zu erhalten. Seit Android KitKat (4.4) besitzen WebView-Objekte es mit *evaluateJavaScript* dafür eine Methode [51].

Jedoch ist die Verwendung des *ValueCallback*-Wertes problematisch. Um den Wert zu erhalten, wurde innerhalb des *ValueCallback*-Elements die Methode *onReceiveValue* überschrieben. Hier wird der Wert aus dem WebView-Inhalt als String übergeben. Da es sich aber um eine Methode innerhalb eines Elements handelt, welches wiederum in eine weitere Methode verpackt ist (siehe Programmcode 8), kann der Wert nicht als Rückgabewert für die Framework-Methode verwendet werden. Ein Aufruf von *return* würde als Rückgabe-Wert

von *onReceiveValue* interpretiert werden. Um dies zu umzugehen, wurde versucht vor dem JavaScript-Aufruf ein String-Element in der Framework-Methode zu erzeugen, welches durch den String in *onReceiveValue* initialisiert wird. Dieser Wert sollte dann schlussendlich als Rückgabewert für die Framework-Methode verwendet werden. Jedoch wurde der String nie initialisiert, was darauf zurückzuführen ist, dass die WebView in einem eigenen Thread ausgeführt wird.

```
public void executeJavaScript(final String cmd){
    this.webView.setWebViewClient(new WebViewClient() {
        @Override
        public void onPageFinished(WebView view, String url) {
            if (Build.VERSION.SDK_INT >= 19) {
                view.evaluateJavascript(cmd, new ValueCallback<String>() {
                    @Override
                    public void onReceiveValue(String s){}
                });
            }
        }
    });
}
```

Programmcode 8: Die executeJavaScript-Methode

Somit ist die Ausführung des JavaScript-Befehls möglicherweise noch nicht abgeschlossen, wenn der *return*-Aufruf kommt und der String bleibt leer. Zuletzt wurde versucht den Hauptthread der Applikation solange zu blockieren, bis der JavaScript-Code ausgeführt und der Rückgabewert bereitsteht. Das Blockieren des Threads führte jedoch zum Absturz der Applikation. Aus diesem Grunde wurden weitere Ansätze zur Übertragung von Daten aus dem JavaScript in den nativen Code untersucht.

4.2.4.2 Ansatz 2: Variablen über JavaScript-Interface übertragen

Der zweite Ansatz erlaubt sowohl die Übertragung von Daten, als auch das Aufrufen von nativen Funktionen aus dem WebView-Code. Dafür soll ein *JavaScript-Interface* verwendet werden, welches Entwicklern ermöglicht, Methoden des nativen Codes aus dem JavaScript-Code der WebView aufzurufen [18]. Um eine solche Methode jedoch aufrufen zu können, muss sich die Methode innerhalb eines *JavaScript-Interfaces* befinden und mit der Annotation *@JavascriptInterface* versehen sein [18]. Um ein *JavaScript-Interface* zu

erstellen, muss eine eigene Klasse erstellt werden, in welcher die Methoden definiert sind, die aufgerufen können werden sollen [18]. Dann muss diese Klasse als *JavaScript-Interface* gemeinsam mit einem Keyword der WebView zugewiesen werden [18].

Damit ist es nun möglich, native Methoden aus dem JavaScript-Code aufzurufen. Dazu kann der Methodenname der nativen Methode verwendet werden, jedoch muss das Keyword (in diesem Fall *Android*) mit einem Punkt getrennt vorangestellt werden (siehe Programmcode 9).

```
Android.nativeMethode(Parameter1,...);
```

Programmcode 9: Aufrufen einer nativen Methode durch *JavaScript-Interface*

So wäre nun zumindest die parallele Ausführung der WebView, in einem eigenen Thread, kein Problem mehr. Da nun die Methodenaufrufe direkt aus dem JavaScript-Code kommen, können benötigte Variablen zusammen mit dem Methoden-Aufruf übertragen werden. Die Funktion kann so in einem Rutsch abgearbeitet werden. Jedoch trat nun ein neues Problem auf, denn der Code aus dem *JavaScript-Interface* wird mit dem WebView-Thread bearbeitet [18]. Soll nun in einer Methode aus dem *JavaScript-Interface* eine Veränderung an der WebView oder dem darin ausgeführten Code durchgeführt werden, so führte dies zu Fehlermeldungen. Die Fehlermeldung gab an, dass lediglich Hauptthreads berechtigt sind, auf die WebView zuzugreifen.

Da es jedoch manchmal nötig ist, dass aufgerufene Methoden Veränderungen an der WebView oder dem darin enthaltenen Code durchführen, müssen weitere Ansätze ausprobiert werden. Problematisch ist auch, dass native Funktionen nicht flexibel hinzugefügt werden können, da sie erfordern, dass entweder eigene *JavaScript-Interfaces* implementiert werden müssen oder zumindest das bestehende *JavaScript-Interface* bearbeitet werden muss. Jedoch eignet sich das *JavaScript-Interface* für einige Methoden. Methoden die einen Rückgabewert liefern und keinen Zugriff auf die WebView brauchen können mit dem *JavaScript-interface* umgesetzt werden. Aus diesem Grunde wurden einige Funktionen der Hauptklasse im *JavaScript-Interface* implementiert.

So können durch das in der Hauptklasse definierte *JavaScript-Interface* Methoden wie die Ausgabe von Toasts, Fehlermeldungen und Warnungen, aber

auch für Entwickler sinnvolle Methoden wie *System.out.print* oder *System.err.println* verwirklicht werden. Auch Sensorzugriffe wie die Aktuelle Position, Kamerazugriffe oder Benachrichtigungen sind Teil des *JavaScript-Interfaces*.

4.2.4.3 Ansatz 3: Reflexion

Nun fehlt lediglich eine Methode zur Datenübertragung und dem Aufruf von Methoden mit Zugriff auf die *WebView*. Um dies zu erreichen, sollen die Methoden per Reflexion aufgerufen werden. Dafür muss lediglich ein Weg gefunden werden, Text aus der *WebView* in den nativen Code zu übertragen. Dieser Text könnte dann den Methodennamen und die benötigten Parameter besitzen. Um Text aus dem *WebView*-Code in den nativen Code zu übertragen, soll die bereits erwähnte *onPageFinished*-Methode verwendet werden. Die enthält nach jedem vollständigen Laden einer Webseite die aktuelle URL der *WebView*. Werden nun der Methodennamen und die Parameter der Methode in die URL eingegeben, so werden diese Werte ebenfalls übertragen. Das HTTP-URL-Schema bietet hierfür eine gute Möglichkeit, solche Übertragungen durchzuführen. So können sich innerhalb einer URL sogenannte *Fragmente* und *Queries* (engl. Abfragen) befinden.

Fragmente werden durch ein „#“ markiert und üblicherweise bei URLs verwendet, um auf einen Teilbereich eines Dokuments zu verweisen [52]. Eine Query wird verwendet um z.B. Webformulare an einen Server zu übertragen [52]. Eine Query wird durch ein Fragezeichen symbolisiert, mehrere Werte können durch „&“ voneinander getrennt werden. Durch Testen konnte ermittelt werden, dass die Android-*WebView* das Öffnen von Fragmenten interpretiert, als würde eine neue Seite geladen werden. Somit wird die *onPageFinished*-Methode aufgerufen und die URL übertragen. Von großem Nutzen ist hier, dass die *WebView* die aufgerufene Seite nach dem Aufruf eines Fragments nicht wirklich neu lädt. Dies hätte andernfalls zu Informationsverlusten z.B. bei ausgefüllten Formularen führen können. Wird eine parameterlose Methode aufgerufen, so kann auf eine Query verzichtet werden und stattdessen lediglich ein Fragment verwendet werden. Die Werte können dann aus dem JavaScript-Code mittels JavaScript in die URL übertragen werden. Hilfreich ist dafür der folgende Aufruf:

```
window.location.href = "#" + keyword + "?" + value1 + "&" + value 2;
```

Programmcode 10: Aufrufen nativer Methoden durch *CallableFunction* (1)

Der Aufruf verändert die aktuelle URL eines Webbrowsers entsprechend dem String. Müssen keine Parameter oder nur statische Strings an die aufzurufende Methode übergeben werden, so ist es sogar ausreichend, ein HTML-Ankerelement einzubinden und dessen *href*-Attribut entsprechend einem Keyword einer *CallableFunction* zu setzen (siehe Programmcode 11). In diesen Fällen könnte sogar auf JavaScript verzichtet werden, was für die Performance der App sicher von Vorteil ist [18].

```
<a href="#Keyword">Klick mich</a>
```

Programmcode 11: Aufrufen nativer Methoden durch *CallableFunction* (2)

Der native Code erhält nach einem Aufruf das Keyword und die Parameter als URL in der *onPageFinished*-Methode im WebView-Client. Dort wird der Wert in die Klassenvariable *currentUrl* des *JavaScriptJavaConnectors* geschrieben. Damit die Werte in der URL überhaupt von Nutzen sind, wurde die Klasse der *CallableFunctions* gebaut. Die Hauptklasse besitzt eine *HashMap* in welcher die Methoden und Funktionen mittels Instanz (einer Klasse), Methodennamen, Keyword und Parameter hinzugefügt werden können (siehe Programmcode 12). Das Keyword wird verwendet um die gesuchte Methode in der *HashMap* zu finden und ist der eigentliche Inhalt des Fragments.

```
this.addCallableFunction(this, "loadUrl", "loadUrl", new Object[] {""});
```

Programmcode 12: Hinzufügen einer Methode zu den *CallableFunctions*

Wird eine neue URL empfangen und abgespeichert, so löst dies eine Reihe weiterer Methoden aus. Zunächst wird überprüft, ob ein Fragment in der URL enthalten ist. Ist dies der Fall, wird die Funktion *splitUrlToCallMethod* aufgerufen. Diese liest das Keyword aus der URL und prüft ob ebenfalls Parameter übertragen wurden. Falls Parameter übertragen wurden, werden diese in einem Objekt-Array gespeichert. Schlussendlich wird die Funktion *proofCallability* aufgerufen, welche überprüft, ob eine Methode innerhalb der aufrufbaren Funktionen dieses Schlüsselwort besitzt.

Wird eine Funktion mit gleichem Schlüsselwort gefunden, so wird versucht, diese Methode auszuführen, dafür werden die übergebenen Parameter dieser

Funktion weitergeleitet. Das Aufwecken geschieht in der Methode *invokeMethod* der Klasse *CallableFunction*. Hierbei wird der Name der übertragenen Klasse der Methode extrahiert und nach dem Methodennamen und der Parameteranzahl und dem Parametertyp gesucht. Danach werden diese Werte verwendet, um die Methode aufzuwecken. Dabei ist zu beachten, dass bisher nur die komplexen Datentypen String, Boolean, Float und Integer als Parameter übertragen werden können. Eine Verwendung der primitiven Datentypen int, bool und float konnte im Laufe dieser Thesis nicht umgesetzt werden.

Die Vorgehensweise mit Reflektionen hat gleich mehrere Vorteile. So können dynamisch aus jeder beliebigen Klasse Methoden zu den aufrufbaren Methoden, zur Laufzeit hinzugefügt oder verändert werden. Da die Methoden nicht wie in einem *JavaScript-interface* mit dem WebView-Thread, sondern mit dem Hauptthread aufgeweckt werden, ist auch eine Manipulation der WebView aus den aufgeweckten Methoden möglich. So können in den so aufgeweckten Methoden wiederum Werte mittels JavaScript in den WebView-Code übertragen werden. Der einzige Nachteil ist, dass die Typen nicht dynamisch erkannt werden und die Verwendung von „?“, „#“ und „&“ innerhalb eines Strings nicht möglich sind, da diese zur Markierung von Fragmenten und Queries dienen.

4.2.5 HTML-Knoten

Als letzte Komponente der Framework-Hauptklasse soll die Klasse *HtmlNode* behandelt werden. Diese wurde implementiert, um leichter HTML-Code an die WebView übertragen zu können. Diese Klasse ermöglicht die Erstellung von HTML-Knoten. Dabei muss kein HTML geschrieben werden. Stattdessen können der Tagname, Attribute des HTML-Elements, CSS-Code, Text und Child-Nodes per Java übertragen, manipuliert und entfernt werden. Diese HTML-Knoten können mittels der Methode *get* inklusive ihrer Child-Nodes problemlos und einfach in einen String umgewandelt werden. Dieser String kann verwendet werden um ganze HTML-Seiten zu erstellen und diese in der WebView anzeigen zu lassen. Er kann aber auch benutzt werden, um ein neues HTML-Element an bestehende Elemente anzuhängen.

4.3 Entwicklung des JavaScript-Frameworks

Mit der Umsetzung der Hauptklasse ist der wichtigste Teil des Frameworks geschaffen. Um die Arbeit mit dem Framework jedoch weiter zu vereinfachen, wurde auch ein JavaScript-Framework (als Teil des Hybrid-App-Frameworks) geschrieben. In diesem können die Standard-Methoden des *JavaScript-Interfaces* und der *CallableFunctions* durch simple Methodenaufrufe angesteuert werden. Standard-Methoden des *JavaScript-Interfaces* können beispielsweise nur mithilfe des vordefinierten Präfixes *Android* aufgerufen werden (siehe Programmcode 13).

```
function showToast(text){  
    Android.showToast(text);  
}
```

Programmcod 13: JavaScript-Wrapper-Methode

Ein solcher Funktionsaufruf ist aus zwei Gründen problematisch. Es könnte beispielsweise als lästig aufgenommen werden, wenn vor jedem Methodenaufruf, zunächst das Schlüsselwort des *JavaScript-Interfaces* geschrieben werden muss. Zum anderen ist es für einen Entwickler dadurch wichtig zu wissen, ob eine Methode oder Funktion durch ein *JavaScript-Interface* oder durch eine *CallableFunction* aufgerufen wird. Das Aufrufen der beiden Funktionstypen unterscheidet sich. Letztere brauchen das Präfix nicht und können auch über die URL aufgerufen werden. Um Entwicklern die Arbeit mit dem Framework zu erleichtern, wurden daher JavaScript-Wrapper-Funktionen geschrieben, die Methodenaufrufe erleichtern (siehe Programmcode 13).

4.4 Prototypische Umsetzung der Benutzeroberfläche

Im Laufe des Projekts hat sich herausgestellt, dass dem Framework eine geeignete Benutzeroberfläche, bei der es sich um ein unterstützendes Programm handelt, hinzugefügt werden sollte. Dafür gibt es vor allem zwei Hauptgründe. Zunächst müssen die benötigten Dateien und der benötigte Code des in dieser Thesis entwickelten Hybrid-App-Frameworks in bestehende Projekte eingebunden werden. Zu den benötigten Dateien zählen unter anderem die JavaScript-Java-Brücke und das JavaScript-Framework. Problematisch ist jedoch auch, dass bestimmte Dinge einer Applikation nicht zur Laufzeit eingestellt werden können, da Sie Teil der Android-Manifest-Datei sind. Das

Android-Manifest wird bei der Installation und dem Starten einer App abgerufen [18]. Es umfasst Eigenschaften wie den App-Namen und das App-Icon [18]. Um Entwicklern eine aufwändige Konfiguration in den XML-Dateien ersparen zu können, soll daher eine Konfiguration durch die Benutzeroberfläche ermöglicht werden.

4.4.1 Aufbau der Benutzeroberfläche

Die Benutzeroberfläche wurde programmatisch in drei Schichten unterteilt, deren Übergang teilweise fließend ist. Die unterste Schicht stellt dabei die Datenschicht dar. Diese ist primär für Datenzugriff und Datenspeicherung zuständig. Die Datenschicht hat keinen Zugriff auf andere Schichten des Modells. Auf der Datenschicht (Data Layer) wird die Logikschicht (Logic Layer) aufgebaut. Diese nimmt zum einen Daten aus der Anwendungsschicht entgegen, verarbeitet und prüft die Daten und leitet diese an die Datenschicht weiter. Zum anderen ruft diese auch Daten aus der Datenschicht ab und leitet diese an die Anwendungsschicht weiter, wenn die Anwendungsschicht diese benötigt. Die Anwendungsschicht besteht aus verschiedenen Controller-Klassen, welche Daten aus der Oberfläche an die Logikschicht weiterleitet oder von der Logikschicht Daten anfordert, um diese an der Benutzeroberfläche darzustellen. Jedoch ist festzuhalten, dass die Anwendungsschicht in diesem Fall als Controller eher als eine Mischung aus Logikschicht und Anwendungsschicht gesehen werden muss, also nicht nur Daten lädt oder weitergibt, sondern auch eine gewisse, einfache Logik enthält.

Für die Views (dt. Sichten/Ansichten) wurde JavaFX verwendet. Die Sichten selbst wurden in vier FXML-Dateien unterteilt. Die erste Sicht (View) wird verwendet, um ein Android-Studio-Projekt in die Benutzeroberfläche zu laden. Hier können der Projektpfad und der Pfad der *MainActivity.java* (Java-Hauptaktivität) festgelegt werden. Wird der Projektpfad festgelegt, überprüft die View bereits selbst, ob die Java-Klasse der Hauptaktivität gefunden werden kann. Dies klappt dann, wenn die Projektstruktur noch nicht zu stark verändert wurde. Sollte die Hauptaktivität nicht automatisch gefunden werden, so kann der Pfad noch händisch nachgetragen werden.

Die zweite View beschäftigt sich mit der Einbindung benötigter Dateien und Daten. Ohne diese Daten könnte mit dem Framework nicht gearbeitet werden. Wird die Option zum automatisierten hinzufügen des Frameworks in ein Android-Studio-Projekt ausgewählt, so werden zunächst einige Dateien eingebunden. Dabei handelt es sich um die Framework-Hauptklasse, die JavaScript-Framework-Datei und das Drittanbieter-CSS-Framework Bootstrap. Sind diese Dateien eingebunden, so werden noch verschiedene Daten in die Projektdateien geladen. Dabei handelt es sich z.B. um die WebView in der XML- und Java-Datei der Hauptaktivität. Außerdem wird die Methode *goBack* (dt. gehe zurück) automatisch implementiert.

4.4.2 Konfiguration von Applikationen

Zu Beginn dieses Kapitels wurde bereits beschrieben, welche Dateien für die Konfiguration einer Applikation wichtig sind. Durch Anpassung dieser Werte kann eine Konfiguration der Applikation vorgenommen werden. Die für dieses Projekt wichtigsten dieser Dateien sind die Android-Manifest im Ordner *main* ([Projektname]\app\src\main) und die Dateien Colors (Farben), Strings (Zeichenketten) und Styles (Stile) im Ordner *values* ([Projektname]\app\src\main\res\values). Bei all diesen Dateien handelt es sich um XML-Dateien.

Um nun die wichtige Konfiguration der Applikationen zu verändern, werden zunächst die wichtigsten XML-Dateien im Java-Code serialisiert. Die grafische Benutzeroberfläche liest die Daten der Android-Manifest-Datei, der Value-Dateien (Colors, Strings und Styles) aus und speichert diese in Variablen innerhalb der Datenschicht des Programms. Die ausgelesenen Werte wie der gewählte Stil, Farben, Strings und Berechtigungen werden an die Benutzeroberfläche der GUI weitergeleitet. Dort kann der Nutzer diese sehen und verändern (siehe Abbildung 9 und 10). Findet eine Veränderung statt, so werden die veränderten Werte aus der Darstellungsschicht an die Datenschicht weitergeleitet und anschließend in den entsprechenden XML-Dateien abgespeichert. Für die URL, die bei der Öffnung der Applikation in die WebView geladen werden soll, wurde eine Variable in der String.xml angelegt. Dabei handelt es sich um die Variable *myurl*. Auch diese kann in der grafischen Benutzeroberfläche festgelegt werden (siehe Abbildung 9). Hierbei kann ausgewählt werden, ob eine

lokale Seite aus den Assets geladen werden soll oder ob die Seite über eine externe URL erreichbar ist.

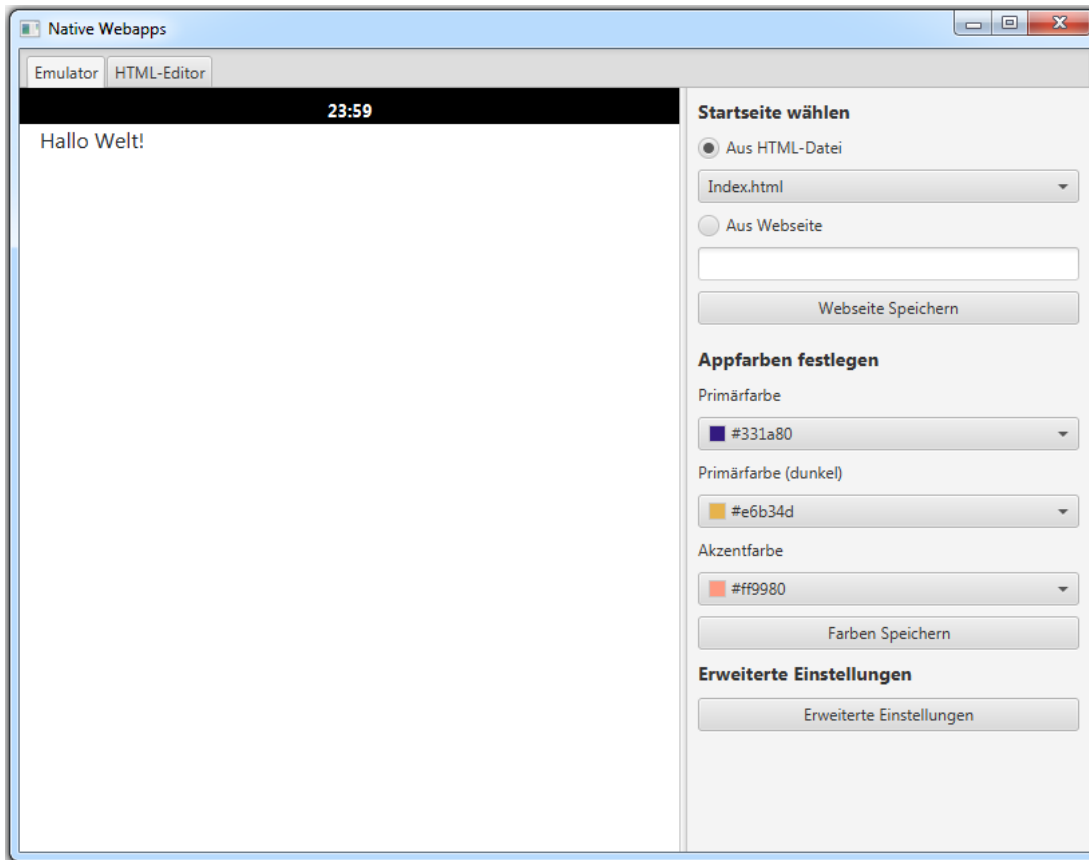


Abbildung 9: Konfiguration der Startseite und Farben

Das App-Icon und der App-Name werden ebenfalls ausgelesen. In der grafischen Benutzeroberfläche des Hybrid-App-Frameworks lassen sie sich ebenfalls verändern (siehe Abbildung 10).

Es gibt jedoch auch Aktionen, die nur bei dem ersten Öffnen eines Projekts durchgeführt werden müssen. Beispielsweise muss ein Projekt für die Arbeit mit dem Hybrid-Framework vorbereitet werden. Benötigte Dateien, Daten und Einstellungen müssen vorhanden sein, um überhaupt mit der Hybrid-App-Entwicklung beginnen zu können. Aus diesem Grund fügt die Benutzeroberfläche bei dem ersten Öffnen eine Konfigurationsdatei in das zu öffnende Android-Studio-Projekt. So erkennt das Programm der grafischen Benutzeroberfläche, ob das Projekt noch für die Arbeit mit dem Framework vorbereitet werden muss oder nicht. Wird das Projekt zum ersten Mal in der grafischen Benutzeroberfläche des Hybrid-App-Frameworks geöffnet, so kann der Entwickler

seine Zustimmung geben, dass benötigter Code und benötigte Dateien automatisch in das Projekt eingebunden werden sollen. Ist dies nicht gewünscht, hat der Entwickler die Möglichkeit dies abzulehnen.

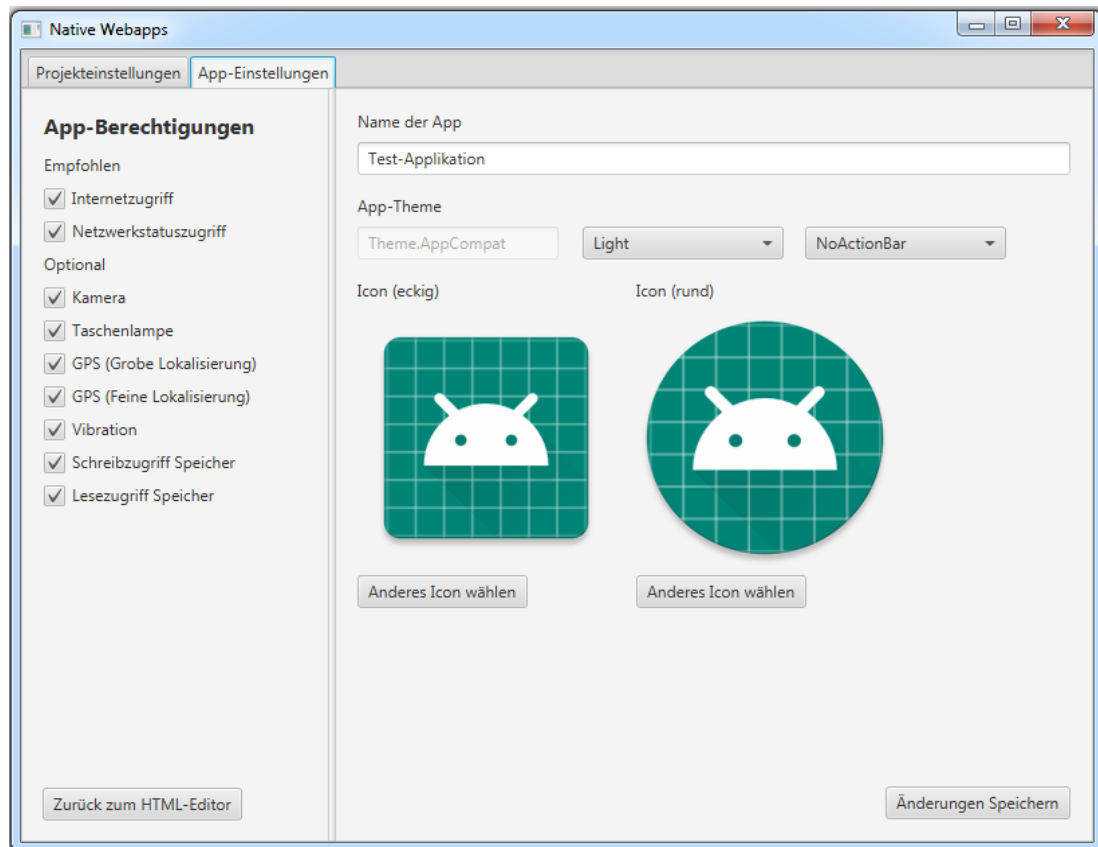


Abbildung 10: Konfiguration von Icons, Name, Theme und Berechtigungen

Bei Zustimmung werden dem Ordner *Assets* die Dateien für Bootstrap, Hammer.JS und das JavaScript-Framework des in dieser Thesis entwickelten Hybrid-App-Frameworks automatisch hinzugefügt. In den Ordner der Java-Main wird die Framework Hauptklasse, die JavaScript-Java-Brücke eingebunden. Zuletzt werden der XML- und der Java-Datei der Hauptaktivität die WebView und weiterer benötigter Code hinzugefügt. Dazu zählt auch, dass als Startseite der WebView, der Inhalt der Variable *myurl* aus der *Strings.xml* definiert wird.

5 Ergebnis, Ausblick und Fazit

Nachdem nun ein großer Einblick in die Durchführung der Implementierung des Hybrid-App-Frameworks gewährt wurde, sollen im letzten Kapitel nun noch die Zusammenfassung, die Ergebnisse, ein Fazit und ein Ausblick folgen.

5.1 Zusammenfassung

Am Anfang dieser Bachelorarbeit stand ein Problem. Trotz der wachsenden Relevanz von Smartphones, ist ein großer Teil des Mittelstands, noch immer von diesem Markt ausgeschlossen [3]. Um zu ergründen, was die Gründe dafür sein könnten, wurde ein Experteninterview durchgeführt. Mithilfe dieses Experteninterviews und einer Ist-Analyse unterschiedlicher Vorgehensmodelle in der App-Entwicklung, konnten die Probleme und Chancen der App-Entwicklung ermittelt werden.

Auf Basis dieser Ist-Analyse wurde beschlossen, ein Hybrid-Applikation-Framework umzusetzen, dessen Verwendung auch ohne Kenntnisse in den klassischen Programmiersprachen weitestgehend möglich sein sollte. Darauf folgte eine Anforderungsanalyse, welche die Grundlage für eine prototypische Umsetzung eines Hybrid-App-Frameworks ist. Dafür wurden auch existierende Hybrid-App-Frameworks betrachtet. Zuletzt folgte die prototypische Entwicklung des Frameworks.

5.2 Ergebnis und Bewertung

Das Ergebnis dieser Bachelorarbeit ist die prototypische Umsetzung eines Hybrid-App-Frameworks für Android. Dieses enthält mehrere Komponenten, die eine Reihe von Aufgaben erfüllen und somit einen großen Funktionsumfang beinhalten. Die Funktionen der einzelnen Komponenten sollen nachfolgend in Listenform aufgeführt werden.

- Grafische Benutzeroberfläche
 - Einbindung der Hauptklasse in bestehende Projekte
 - Einbindung des JavaScript-Frameworks in bestehende Projekte
 - Einbindung von Drittanbieter-Frameworks
 - Bootstrap
 - Hammer.js

- Automatisiertes Vorbereiten Hauptaktivitäts-Klasse (Java)
- Automatische Einbindung der WebView in Hauptaktivität
 - Festlegen der Startseite in der WebView
- Grundkonfiguration von Android-Applikationen
 - App-Name
 - App-Icon
 - App-Farbschema
 - App-Theme
 - Berechtigungen
- Erstellen, Bearbeiten und Löschen von HTML-Dokumenten
- JavaScript-Java-Bridge (Hauptklasse)
 - Schnittstellen: JavaScript zu Java
 - Schnittstellen: Java zu JavaScript
 - Ausführen von JavaScript-Code
 - Hinzufügen, verändern und entfernen von HTML-Code
 - Hinzufügen, verändern und entfernen von CSS-Code
 - Zugriff auf native Funktionen aus dem Web-Code
 - URLs laden
 - Zurückfunktion
 - Prüfen von Berechtigungen
 - Abfragen von Berechtigungen
 - Textausgabe in Java-Konsole
 - Native Fehlermeldungen
 - Native Warnungen
 - Native Toasts
 - Native Benachrichtigungen
 - Text intern speichern und auslesen
 - CallableFunctions
 - Flexibles hinzufügen von eigenen nativen Funktionen, welche durch URLs aufgerufen werden können
 - Zugriff auf Hardware
 - Kamera
 - GPS

- Drehungssensor
- Kamera-Blitzlicht
- Vibrationsmotor
- JavaScript-Framework
 - Erleichtert Zugriff auf native Funktionen

Um das Ergebnis der Thesis bewerten zu können, muss in Erinnerung gerufen werden, was überhaupt die Ziele dieser Thesis waren. Von größter Wichtigkeit sind dafür die Forschungsziele aus der Einleitung. Dort wurde zunächst definiert, dass erforscht werden sollte, warum mittelständische Unternehmen eher selten am App-Markt partizipieren. Dafür sollte ein Experteninterview und eine Ist-Analyse verschiedener Vorgehensmodelle in der App-durchgeführt werden. Diese Aufgaben wurden bereits in Kapitel zwei der Ist-Analyse erfolgreich durchgeführt. So konnte ermittelt werden, dass mittelständische Unternehmen vor allem finanzielle Mittel fehlen, da die App-Entwicklung aufwändig und kompliziert ist.

In Kapitel drei konnte dann das zweite Ziel erreicht werden. Hier wurde ermittelt, wie die App-Entwicklung erleichtert werden kann. Dafür wurde auch eine Anforderungsanalyse durchgeführt, welche auf der Ist-Situation von Kapitel zwei basiert. Anhand der Anforderungsanalyse wurde die Entwicklung eines Hybrid-Applikations-Frameworks beschlossen. Das darauffolgende vierte Kapitel umfasste dann die Umsetzung des Projekts. Ob die prototypische Umsetzung von Erfolg war, muss anhand der Anforderungsanalyse aus Kapitel 3.1.2 bewertet werden.

Im Zuge der Anforderungsanalyse wurden drei Anforderungen als essenziell für den erfolgreichen Abschluss des Projekts bestimmt. So sollte das Framework ermöglichen Applikationen hauptsächlich mit den Web-Sprachen HTML, CSS und JavaScript umzusetzen. Diese Anforderung konnte definitiv umgesetzt werden. Das Framework ermöglicht die Einbindung von lokalen sowie auf entfernten Servern gelagerte Webseiten. Das Ausführen und Injizieren von HTML, CSS und JavaScript-Code aus dem nativen Code ist ebenfalls möglich. Mithilfe des Frameworks können diese Daten in native Wrapper-Applikationen eingebunden werden.

Als zweite und dritte essenzielle Anforderung an das Hybrid-App-Framework wurde definiert, dass Zugriff auf jegliche Hardware und native Funktion möglich sein muss. Auch dieser Punkt ist erfüllt. Zwar werden noch nicht alle Funktionalitäten nativer Funktion bereits standardmäßig durch das prototypische Hybrid-App-Framework unterstützt, jedoch ist eine Erweiterung dank der *CallableFunctions* äußerst einfach. Das Framework bietet sowohl Schnittstellen für den Web-Code zum nativen Code, als auch Schnittstellen aus dem nativen in den Web-Code.

Neben den unabdingbaren Anforderungen wurden auch wünschenswerte Anforderungen, sogenannte Soll-Anforderungen, an das Hybrid-App-Framework gestellt. Dabei geht vor allem um Anforderungen, welche die Qualität des Projekts verbessern sollen. So sollte es möglich sein, Komponenten in Applikationen für verschiedene Systeme wiederzuverwenden. Durch die Verwendung von HTML, CSS und JavaScript als Primär-Entwicklungssprachen konnte das Ergebnis auch dieser Anforderung gerecht werden, da diese Sprachen plattformübergreifend verwendbar sind.

Eine Soll-Anforderung war auch, dass die Applikationen, die mithilfe des prototypischen Frameworks entwickelt werden, sich wie native Anwendungen anfühlen, theoretisch über App-Stores vermarktet werden können und möglichst performant sind. Diesen Anforderungen konnte nachgekommen werden, indem die in den Web-Sprachen geschriebenen Komponenten in eine native Wrapper-App mit WebView eingesetzt werden. So können diese als APK in einem App-Store angeboten werden. Auch das Native-App-Feeling ist gegeben, da es sich einerseits um eine native App handelt und andererseits Drittanbieter-Frameworks wie z.B. Hammer.js eingebunden wurden, welches die Websprachen um Smartphone-typische Gesten erweitern. Eine gute Performanz kann durch Auslagerung und Parallelisierung im nativen Code erreicht werden. Womit auch die nächste Soll-Anforderung erfüllt wäre. Code sollte ausgelagert werden können, um die Performanz zu verbessern. Mithilfe dieses App-Frameworks kann Code gleich in zwei Richtungen ausgelagert werden. Zum einen kann Code in den nativen Code ausgelagert werden, zum anderen kann auch Rechenleistung und Speicher von Online-Servern benutzt werden, um Code und Prozesse auszulagern.

Die letzten Anforderungen sind schwer zu bewerten. Dabei handelt es sich um die Soll-Anforderungen, dass das Framework möglichst einfach verwendet und erweitert werden soll und dabei möglichst flexibel sein soll. Tendenziell wurde versucht, dem Entwickler alle Wege offen zu halten. So können jegliche Webseiten und HTML-Dokumente online und offline verwendet werden. Es gibt keine Einschränkungen darin, welche JavaScript-Frameworks oder Backend-Frameworks bei Servern verwendet werden können. Auch wurde versucht möglichst viele unterschiedliche Wege zu implementieren, die einen Zugriff auf den nativen Code aus dem Web-Code ermöglichen. Zusätzlich wurde versucht, die Arbeit mit dem Framework möglichst einfach zu halten. Dazu wurde eine Benutzeroberfläche entwickelt, welche die App-Konfiguration und die Einbindung des Frameworks möglichst erleichtern soll. Durch die Implementierung des Konzepts der *CallableFunctions* wird Java-Programmierern die Möglichkeit gegeben das Framework flexibel, um eigenen nativen Code zu erweitern.

Da sowohl die Forschungsziele als auch die Anforderungen der Anforderungsanalyse erfüllt wurden, ist die Thesis daher als erfolgreich anzusehen. Das Framework konnte prototypisch umgesetzt werden und eine grundlegende App-Entwicklung ist bereits möglich.

5.3 Bewertung der Werkzeuge

Die Hauptklasse des Frameworks, welche den Datenaustausch und die Kommunikation zwischen Java- und JavaScript-Code in der WebView ermöglicht, wurde mithilfe von Android-Studio entwickelt. Dabei müssen also direkt zwei Punkte betrachtet werden. Erwies sich die Entscheidung, auf der Android-Plattform zu arbeiten als hilfreich und war wiederum in Zuge dessen die Arbeit mit Android-Studio eine gute Entscheidung.

Die Arbeit auf der Android-Plattform war in diesem Fall eine gute Entscheidung, da das quelloffene System nur so gering wie nötig beschränkt wurde, was sich in vielerlei Hinsicht bemerkbar machte. Die Kosten bei der Arbeit mit Android beschränken sich auf die Anschaffungskosten eines PCs für die Entwicklung sowie Geräte für das Testen. Da diese im Falle dieser Thesis bereits

vorhanden waren, kamen hier also keine Anschaffungskosten auf. Die App-Entwicklung mit Android-Studio ist ebenfalls kostenlos.

Da Android quelloffen ist, kann also auch der Code des Systems eingesehen werden. Dadurch existieren neben der umfangreichen, von Google veröffentlichten Dokumentation, viele weitere Codebeispiele und Hilfestellungen in Büchern und im Internet. Die Informationsdichte zu diesem Thema half bei der Umsetzung des Projekts ungemein. Grundsätzlich ist die Arbeit mit Android-Studio und der Android-Plattform als sehr gut zu bewerten. Wobei jedoch angemerkt werden muss, die Arbeit nicht zunächst wie geplant auf Windows 7 umgesetzt wurde. Stattdessen wurde die Linux-Distribution *Deepin* verwendet. Der Grund dafür war, dass der Android-Emulator wegen eines Intel HAXM-Kompatibilitätsproblems zu Windows 7 nicht gestartet werden konnte.

Das JavaScript-Framework wurde in Microsofts Visual Studio Code entwickelt. Dies war nötig, da Android-Studio bei der Auslieferung nur bedingt die Arbeit HTML, CSS und JavaScript unterstützt. Um den Code durch Hervorhebungen lesbarer zu gestalten und eine Code-Vervollständigung zu erhalten, musste infolge dessen ein Ersatz gefunden werden. Auch die Arbeit mit Visual Studio Code ist sehr positiv zu bewerten. Der Editor ist gut strukturiert, übersichtlich und sehr leichtgewichtig, ergo auch auf leistungsschwächeren Systemen gut auszuführen. Zudem ist auch die Entwicklung in Visual Studio Code ebenfalls ohne Kosten verbunden.

Für die Entwicklung des grafischen Einstellungs- und Einbindungstools wurde entschieden, Java zu verwenden, da in Java geschriebene Anwendungen Plattformübergreifend ausgeführt werden können. Für eine moderne Benutzeroberfläche bot sich das JavaFX Framework an. Ursprünglich sollte die Umsetzung der Benutzeroberfläche auf der Codebasis von Java 13 geschehen. Jedoch gestaltete sich die Arbeit mit OpenJFX (Gratis-Version von JavaFX), als schwieriger als zunächst angenommen wurde. Da diese Probleme lange nicht gelöst werden konnten, wurde die Entwicklung schlussendlich auf Java 8 durchgeführt, wodurch JavaFX nicht nachinstalliert werden musste. Auch die Verwendung des SceneBuilder für die FXML-Dateien bereitete immer wieder Probleme, weshalb die Arbeit an einzelnen Fenstern mehrfach neu begonnen

werden musste. Die Arbeit mit JavaFX hinterließ daher gemischte Gefühle, wobei erwähnt werden muss, dass mit JavaFX benutzerfreundliche und moderne Benutzeroberflächen gestaltet werden können.

Als IDE wurde Eclipse verwendet, da dieses ebenfalls kostenlos ist und durch seine weite Verbreitung viel Hilfestellung und Information gefunden werden kann. Die Umsetzung der Benutzeroberflächen-Logik mithilfe von Eclipse erfolgte problemlos. Die Arbeit mit Eclipse ist daher als positiv zu bewerten.

Um den Arbeitsfortschritt einerseits nicht zu verlieren, aber andererseits auch noch die Möglichkeit zu offen zu halten fehlerhafte, neue Versionen durch funktionstüchtige Alte ersetzen zu können, wurde GitHub verwendet. Hier wurden jeweils ein Repository für das Android-Studio-Projekt, für die Hauptklassenentwicklung und das JavaScript-Framework und ein Repository für die Entwicklung der grafischen Benutzeroberfläche erstellt. Auch diese Entscheidung ist nachträglich als sehr gut zu bewerten, da sie die Sicherheit gab, dass nach fehlgeschlagenen Programmierexperimenten nicht der ganze Code unbrauchbar wurde und neu entwickelt werden musste. Da für die Entwicklung der einzelnen Komponenten unterschiedliche Texteditoren und IDEs verwendet wurden, gaben die Repositories auch eine bessere Übersicht darüber, auf welchem Stand die Entwicklung ist.

5.4 Ausblick und Entwicklungsvorschläge

Zum Abschluss dieser Thesis soll ein kleiner Ausblick darauf gegeben werden, was mit dem Framework nach dem Ende der Thesis geschieht und welche Möglichkeiten zur Weiterentwicklung bestehen. Zuerst wird die prototypische Umsetzung des Frameworks und der dazugehörige Quellcode dem betreuenden Unternehmen übergeben. Der Quellcode des Frameworks kann auf der CD eingesehen werden, welche der Thesis beiliegt. Wie und ob die prototypische Umsetzung des Frameworks im betreuenden Unternehmen weiterentwickelt wird oder zum Einsatz kommt, ist zum Zeitpunkt des Thesis-Abschlusses leider noch nicht bekannt. Die bisherige Umsetzung bietet jedoch noch eine Menge Möglichkeiten zur Weiterentwicklung. Diese lassen sich grob in die Kategorien *Erweiterung des Funktionsumfangs* und *Entwicklung für andere Plattformen* einteilen und sollen folgend erläutert werden.

5.4.1 Erweiterung des Funktionsumfangs

Da das Framework bisher nur prototypisch umgesetzt wurde, kann der Funktionsumfang aber auch die Qualität des Quellcodes noch gesteigert werden. Zwar wurden bereits einige Methoden in das Framework integriert, die einen Zugriff auf native Funktionen und die Geräte-Hardware erlauben, jedoch sind die Möglichkeiten dessen noch lange nicht ausgereizt. In Sachen Hardware wurden zwar bisher unter anderem Funktionen zum Schießen von Fotos, zur Abfrage des Standorts, zum Auslösen des Vibrationssensors und zum Ein- und Ausschalten des Bewegungssensors innerhalb der App integriert, jedoch bieten Sensoren wie das GPS oder die Kamera noch deutlich mehr Optionen an, als bisher implementiert wurden. Deshalb sollte die Unterstützung weiterer Hardware-Komponenten durch das Framework in Betracht gezogen werden. Auch die Software bietet noch weitere Perspektiven, so kann die Methoden-Bibliothek zur Überbrückung von JavaScript und Java erweitert und die Qualität des Codes in Bezug auf die Sicherheit und Performanz verbessert werden. Zusätzlich kann versucht werden, eine verbesserte Parallelisierung zu ermöglichen. Problematisch ist noch immer, dass die WebView in einem einzelnen Thread arbeitet. Werden mehrere Anfragen an den WebView-Code nacheinander getätigt, so gehen manche Aufrufe verloren, da der WebView-Thread häufig noch mit dem vorherigen Befehl beschäftigt ist. Dadurch müssen unmittelbar aufeinanderfolgende Befehle aus dem nativen Code, die auf den WebView-Code Einfluss nehmen, derzeit kanalisiert und zusammengefasst werden, um Datenverlust und Fehler zu vermeiden. Auch die Entwicklung der Benutzeroberfläche bietet einige Möglichkeiten. Um eine fehlerfreie Einbindung des Frameworks in bestehende Projekte zu garantieren, wird eine strikte Form und Struktur des Android-Studio-Projekts gefordert. Man könnte die Einbindung des Frameworks in Projekte und die Arbeit mit der Benutzeroberfläche flexibler gestalten, um so die Entwicklungsarbeit weiter zu vereinfachen. Ebenfalls sollte analysiert werden, welche Funktionalitäten die Benutzeroberfläche bereitstellen könnte, die für die App-Entwicklung noch von Bedeutung sind. So könnte der Umfang der Benutzeroberfläche um nützliche Funktionen für Entwickler erweitert werden, die eine Menge repetitive Arbeit abnimmt und so Zeit und Geld sparen könnten.

5.4.2 Entwicklung für weitere Systeme

Da die prototypische Umsetzung zunächst nur für Android erfolgte, ist es auch ratsam, die Entwicklung für andere Plattformen in Betracht zu ziehen. So könnten mithilfe des Frameworks mehrere Systeme gleichzeitig abgedeckt werden. Dabei ist es zunächst ratsam eine Entwicklung für das zweitgrößte, mobile Betriebssystem iOS anzustoßen.

Bei einer solchen Entwicklung sollte darauf geachtet werden, dass der Code in der Hauptklasse, der JavaScript-Native-Brücke, die gleichen Schnittstellen bietet. Diese müssen zwar nicht identisch implementiert werden, wie die Schnittstellen der Android-Version, die Aufrufe der Schnittstellen aus dem nativen und Web-Code sollten jedoch in gleicher Form erfolgen. So kann sichergestellt werden, dass die in HTML geschriebenen Anwendungen auf den Systemen ohne Veränderung des Web-Quellcodes funktionieren. Somit könnte eine aufwändige Refaktorisierung der Applikationen im Zuge der Portierung auf andere Systeme wegfallen.

Auch eine Entwicklung des Frameworks für andere mobile Plattformen wie Windows Phone, SamsungOS oder KaiOS ist denkbar. Zwar machen diese nur einen geringen Anteil am Smartphone-Markt aus [26], wenn jedoch möglichst alle Plattformen eingebunden werden sollen, ist es auch empfehlenswert, diese Plattformen für die Entwicklung in Betracht zu ziehen.

Zuletzt ist natürlich auch eine Ausweitung des Frameworks durch Versionen für andere Plattform-Systeme wie Notebooks oder Desktop-Rechner denkbar. Hier ist eine Entwicklung für die Systeme Windows, Linux und MacOS sicher am sinnvollsten.

5.5 Fazit

Diese Thesis konnte darstellen, dass die Entwicklung und das Besitzen von Smartphone-Applikationen für die moderne Wirtschaft von großer Bedeutung sind. Auch mittelständische Unternehmen müssen mit der Zeit auf diesen Zug aufspringen, wenn sie den Anschluss an den Markt und die Kundenbasis nicht verlieren wollen.

Durch die Ist-Analyse konnte herauskristallisiert werden, welche Hürden noch bewältigt werden müssen, um auch kleineren Unternehmen eine Chance auf dem mobilen Markt zu gewähren. Das Ziel dieser Arbeit war ein prototypisches Hybrid-App-Framework, welches besonders Mediendesigner und Webentwickler ansprechen sollte, um Smartphone-Applikationen in ähnlicher Weise zu entwickeln, wie Webseiten. Dies würde die Kosten und den Entwicklungsaufwand erheblich senken.

Das Ergebnis dieser Thesis war ein Hybrid-App-Framework, welches als prototypische Umsetzung zwar noch nicht das volle, mögliche Potential ausschöpft, jedoch bereits wichtige Werkzeuge und Möglichkeiten zur Hybrid-App-Entwicklung durch Mediendesigner bereitstellt. Das Framework ist durch die mitgelieferte Benutzeroberfläche leicht zu verwenden. Applikationen können in den Web-Sprachen HTML, CSS und JavaScript geschrieben werden und dennoch auf einen gewissen Rahmen nativer Methoden und Hardware zugreifen. Der im Rahmen dieser Thesis erarbeitete Prototyp kann somit die Arbeit von Agenturen erleichtern, die mit mittelständischen Unternehmen zusammenarbeiten, und diese unterstützen.

Literaturverzeichnis

Bücher

- [1] Axel Bruns: *Die Geschichte des Computers: Wie es bis zur Form des heutigen 'PC' kam*, Berlin: neobooks (2015), ISBN: 978-3-7380-2145-5
- [2] Dr. Oliver Stengel, Alexander van Looy und Stephan Wallaschkowski: *Digitalzeitalter - Digitalgesellschaft: Das Ende des Industriezeitalters und der Beginn einer neuen Epoche*, Luxemburg (Luxemburg): Springer Verlag (2017), ISBN: 978-3-658-16508-6
- [3] Prof. Dr. Gerrit Heinemann: *Die Neuausrichtung des App- und Smartphone-Shopping: Mobile Commerce, Mobile Payment, LBS, Social Apps und Chatbots im Handel*, Luxemburg (Luxemburg): Springer Verlag (2017), ISBN: 978-3-658-19134-4
- [4] Fred Vogelstein, *Dogfight: How Apple and Google Went to War and Started a Revolution*, New York (USA): Farrar, Straus and Giroux (2013), ISBN: 978-0-374-71100-9
- [6] Steven Levy, *Google Inside: Wie Google denkt, arbeitet und unser Leben verändert*, Köln: MITP-Verlags GmbH & Co. (2012), ISBN: 978-3-826-69243-7
- [7] Marko Gargenta: *Einführung in die Android-Entwicklung*, Heidelberg: O'Reilly Germany (2011), ISBN: 978-3-868-99114-7
- [15] Reto Meier: *Professionelle Android App-Entwicklung*, Hoboken, New Jersey (USA): John Wiley & Sons (2019), ISBN: 978-3-527-68681-0
- [16] Florian Franke, Johannes Ippen: *Apps mit HTML5, CSS3 und JavaScript: Für Android, iPhone und iPad*, Bonn: Rheinwerk (2015), ISBN: 978-3-8362-3486-3
- [17] Alexander Sollberger, Pascal Müller: *Mobile Marketing und Mobile Apps: Auflage April*, Vaduz (Lichtenstein): wifimaku (2015)

- [18] Nizamettin Gok, Nitin Khanna: *Building Hybrid Android Apps with Java and JavaScript*, Newton, Massachusetts (USA): O'Reilly Media, Inc. (2013), ISBN: 978-1-449-36191-4
- [19] Daniel Knott: *Mobile App Testing: Praxisleitfaden für Softwaretester und Entwickler mobiler Anwendungen*, Heidelberg: dpunkt.verlag (2016), ISBN: 978-3-864-90379-3
- [20] Janosch Skuplik: *Web-Apps erstellen mit CMS-Daten: Web-App-Erstellung am Beispiel von WordPress und Contao*, Haar: Franzis Verlag (2013), ISBN: 978-3-645-60214-3
- [21] Brian Messenlehner, Jason Coleman: *Building Apps with WordPress: WordPress as an Application Framework*, Newton, Massachusetts (USA): O'Reilly Media, Inc. (2019), ISBN: 978-1-4919-9008-7
- [22] Herbert Braun: Progressive Web-Apps Entwickeln, erschienen in c't Webdesign: Entwicklung - Performance - SEO - Content S. 31-34, Hannover: Heise Medien GmbH & Co. KG (2017), ISBN: 978-3-957-88169-4
- [25] Bernhard Hobel, Silke Schütte: *Business-Wissen Projektmanagement von A - Z: Kompetent entscheiden. Richtig handeln.*, Luxemburg (Luxemburg): Springer Verlag (2011), ISBN: 978-3-409-12547-5
- [31] Eugen Richter: *Android-Apps programmieren: Praxiseinstieg mit Android Studio*, Frechen: MITP-Verlags GmbH & Co. KG (2019), ISBN: 978-3-958-45890-1
- [32] Ed Burnette, Joerg Staudemeyer: *Eclipse IDE kurz & gut*, Heidelberg: O'Reilly Germany (2013), ISBN: 978-3-955-61153-8
- [33] Ralph Steyer: *Einführung in JavaFX - Moderne GUIs für RIAs und Java-Applikationen*, Wiesbaden: Springer Verlag (2014), ISBN: 978-3-658-02835-0

- [34] Bruce Johnson: *Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers*, Hoboken, New Jersey (USA): John Wiley & Sons, ISBN: 978-1-119-58818-4
- [35] Alessandro Del Sole: *Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux*, Berkeley, Kalifornien (USA): Apress (2019), ISBN: 978-1-4842-4223-0
- [39] Jonathan Fielding: *Beginning Responsive Web Design with HTML5 and CSS3*, Berkeley, Kalifornien (USA): Apress (2014), ISBN: 978-1-4302-6694-5
- [40] Bengt Weiße: *AngularJS & Ionic Framework: Hybride App-Entwicklung mit JavaScript und HTML5*, München: Carl Hanser Verlag GmbH Co KG (2016), ISBN: 978-3-446-44671-7
- [43] Dirk Louis, Peter Müller: *Android: Der schnelle und einfache Einstieg in die Programmierung und Entwicklungsumgebung*, München: Carl Hanser Verlag GmbH Co KG (2016), ISBN: 978-3-446-43823-1
- [47] Donn Felker, Michael Burton: *Android App Entwicklung für Dummies*, Hoboken, New Jersey (USA): John Wiley & Sons (2015), ISBN: 978-3-527-71149-9
- [48] Günter Born: *Jetzt lerne ich XML: der einfache Einstieg in den führenden Dokumenten- und Web-Standard*, Hallbergmoos: Pearson Deutschland GmbH (2005), ISBN: 3-8272-6854-0
- [49] David Flanagan: *Java in a nutshell: deutsche Ausgabe für Java 1.4*, Heidelberg: O'Reilly Germany (2003), ISBN: 978-3-897-21190-2
- [52] Thomas Hoffmann-Walbeck et al.: *Standards in der Medienproduktion*, Wiesbaden: Wiesbaden: Springer Verlag (2013), ISBN: 978-3-642-15042-5

Zeitschriften

- [5] Dr. Marcus Raitner: Eine kurze Geschichte der Digitalisierung, erschienen in *Digitale Welt* 1/2019 S.86, Luxemburg: Springer Verlag (2019)

Internetquellen

- [8] Statcounter (2019). Desktop vs Mobile vs Tablet vs Console Market Share Worldwide. <https://gs.statcounter.com/platform-market-share#monthly-201912-201912-bar> zuletzt aufgerufen am 31.01.2020
- [9] Koppert. <https://www.koppert.de/unternehmen> zuletzt abgerufen am 24.11.2019
- [10] Statcounter (2020). Desktop vs Mobile Market Share Worldwide: Dec 2009 – Dec 2019. <https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide/#monthly-200912-201912> zuletzt abgerufen am 15.02.2020
- [11] Statista (2019). Mobile internet usage worldwide - Statistics & Facts. <https://www.statista.com/topics/779/mobile-internet/> zuletzt abgerufen am 15.02.2020
- [13] Gutado (2014) Was kostet dich eine gute Webseite? <http://gutado.de/was-kostet-dich-eine-gute-webseite/> zuletzt abgerufen 23.01.2020
- [14] Mobilbranche (2015). Fritz Ramisch: Studie: Die Entwicklung einer App kostet im Schnitt 30.000 Euro. <https://mobilbranche.de/2015/07/ibusiness-studie-die> zuletzt abgerufen am 23.01.2020,
- [23] t3n (2018). Andres Dickehut: Progressive-Web-Apps: Das App-Modell der Zukunft. <https://t3n.de/news/progressive-web-apps-app-modell-1078161/> zuletzt abgerufen am 17.02.2020
- [24] IONOS (2019). Progressive Web-Apps: Was versprechen die progressiven Apps? <https://www.ionos.de/digitalguide/websites/web-entwicklung/progressive-web-apps-welche-vorteile-bieten-sie/> zuletzt abgerufen 25.11.2019
- [26] Statcounter (2019). Mobile Operating System Market Share Worldwide – Jan 2010 – Oct 2019. <https://gs.statcounter.com/os-market->

share/mobile/worldwide/#monthly-201001-201910 zuletzt abgerufen am 20.01.2020

- [27] WindowsUnited (2018). Tomás Freres: Was ist eigentlich KaiOS (Nokia 8110)? *<https://windowsunited.de/was-ist-eigentlich-kaios-nokia-8110/>* zuletzt abgerufen am 19.02.2020
- [28] KaiOS. Our mission - Advance digital inclusion and close the digital divide. *<https://www.kaiotech.com/company/>* zuletzt abgerufen am 19.02.2020
- [29] KaiOS. KaiOS, the emerging OS. *<https://developer.kaiotech.com/>* zuletzt abgerufen am 19.02.2020
- [30] Developer, Android. *<https://developer.android.com/studio>* zuletzt abgerufen am 02.02.2020
- [36] HammerJS. *<https://hammerjs.github.io/getting-started/>* zuletzt abgerufen am 13.01.2020
- [37] Github. Ratchet. *<https://github.com/twbs/ratchet>* zuletzt abgerufen am 21.12.2019
- [38] Ratchet. *<http://goratchet.com/components/>* zuletzt abgerufen am 21.12.2019
- [41] hippopunk (2017). Roberto De Simone: Native Apps mit HTML5 und Apache Cordova entwickeln. *<https://www.hippopunk.com/de/native-apps-mit-html5-entwickeln/>* zuletzt abgerufen am 08.12.2019
- [42] YUHIRO (2017). Sascha Thattil: App Entwicklung mit PhoneGap: Unsere Erfahrung. *<https://www.yuhiro.de/app-entwicklung-mit-phonegap-unsere-erfahrung/>* zuletzt abgerufen am 08.12.2019
- [44] Developer, Android. App Manifest Overview. *<https://developer.android.com/guide/topics/manifest/manifest-intro.html>* zuletzt abgerufen am 21.02.2020

- [45] Statcounter (2020). Mobile & Tablet Android Version Market Share Worldwide – Dec 2019. <https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide#monthly-201912-201912-bar> zuletzt abgerufen am 21.02.2020
- [46] Developer, Android. Styles and Themes. <https://developer.android.com/guide/topics/ui/look-and-feel/themes> zuletzt abgerufen am 07.02.2020
- [50] Developer, Android. android.webkit.WebViewClient. <https://developer.android.com/reference/android/webkit/WebViewClient> zuletzt abgerufen am 22.02.2020
- [51] Developer, Android. android.webkit.WebView. <https://developer.android.com/reference/android/webkit/WebView> zuletzt abgerufen 28.12.2019

Andere Quellen

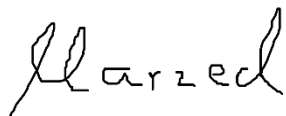
- [12] N. Haberer: *Gesprächsprotokoll Experteninterview*, Albert (bei Albruck), 19. Dezember 2019

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.



Erzingen, der 27.02.2020 Maximilian Marzeck

Anhang A: Experteninterview

Gesprächsprotokoll Experteninterview

Thema: Die aktuelle Lage in Medienagenturen und mittelständischen Unternehmen in der Digitalisierung

Befragte: Natalie Haberer B. Arts
Mediendesignerin bei Koppert Mikromarketing-Systeme

Interviewer: Maximilian Marzeck

Wie lautet Ihre Berufsbezeichnung und wie lange arbeiten Sie in Ihrem Beruf?

Mein Beruf nennt sich Mediendesigner, dies liegt daran, dass wir in verschiedenen Bereichen arbeiten, wir uns also nicht nur auf eine Sache beschränken, wie Print oder Ähnliches, sondern auch digital arbeiten, mit Webseiten, Fotos, Videos und Corporate Designs. Ich arbeite jetzt seit 6,5 Jahren in diesem Beruf.

Wo wurden Sie ausgebildet und wie lange hat ihre Ausbildung gedauert?

Ausgebildet wurde ich insofern, dass ich ein duales Studium gemacht habe, das war einerseits hier bei Koppert Mikromarketing-Systeme und andererseits war ich auch an einer dualen Hochschule. Das Studium hat drei Jahre gedauert. Im Wechsel von drei Monaten in Ravensburg (Studium) und dann wieder drei Monate arbeiten. Und das für 6 Semester.

Was sind Ihre täglichen Aufgaben bei Koppert Mikromarketing-Systeme?

Meine täglichen Aufgaben sind eigentlich ganz unterschiedlich, von Auftrag zu Auftrag. Es kann sein, dass ich allgemeine Printsachen mache, wie Flyer, Broschüren oder komplette Corporate Designs mit Logo usw. mache. Ich kümmere mich jedoch um die Erstellung und Wartung von Webseiten, dazu gehören auch Aufgaben wie SEO (Suchmaschinenoptimierung) für Google etc. Wir schneiden und bearbeiten jedoch auch Fotos und Videos oder begleiten Events und machen Animationen. Die Arbeit ist also sehr vielfältig.

Denken Sie, dass das Smartphone heute das wichtigste, oder zumindest eines der wichtigsten Endgeräte ist, wenn es um die Beschaffung von Informationen oder Software geht, bzw. Und die Verteilung von Software geht?

Das kommt ganz darauf an, wenn man die alltäglichen Aufgaben wie etwas zu Googeln geht, wird das Smartphone eindeutig am häufigsten genutzt. Dies sieht man, wenn man sich Statistiken in diese Richtung ansieht. Auch wenn es im Allgemeinen um die Beschaffung von Verteilung von Informationen geht. Was Applikationen (Software) betrifft, ist es etwas schwieriger, denn der Computer ist noch immer hoch im Kurs. Was daran liegt, dass man dort oft immer noch den besseren Durchblick bekommt, da heute noch immer nicht alles für das Smartphone optimiert ist. Und selbst wenn Software offiziell optimiert ist, gibt es immer noch Probleme, dass man zum Beispiel nicht alles so hinbekommt (wie bei den Desktopversionen), weil das in den verwendeten Baukästen und Frameworks nicht so funktioniert.

Es scheint also so zu sein, dass ein gewisser Nachbesserungsbedarf besteht, wenn es um den Umfang von Smartphone-Applikationen geht?

Das auf jeden Fall, da bisher nicht alles von einem Designer durchkonzipiert ist. Das heißt: Ein Mediendesigner schaut ja schon immer darauf, dass die Leute tatsächlich alles finden, was sie klicken müssen und es dann optimiert ist, auf diese Fläche die zur Verfügung steht. Gerade Internetbaukästen sind noch immer nicht auf alle Formate optimiert. Häufig braucht man dann einen Web-Programmierer. Bei Apps kommen Programmierer jedoch deutlich öfter vor. Bei Webseiten werden heute jedoch tatsächlich meist Internetbaukästen genutzt, weil die Zeitersparnis doch sehr hoch ist.

Apps werden also eher selten von Mediendesignern entwickelt, weil man einen Programmierer braucht. Sollte die Entwicklung also vereinfacht werden?

Ja, auf jeden Fall. Die ersten Schritte in diese Richtung wurden ja auch schon gemacht, wenn man so in Richtung User Experience schaut. Es ist gerade ein sehr großes Thema. Es wird immer beliebter. In vielen großen Unternehmen kümmern sich mittlerweile größere Abteilungen um dieses Thema. Und viele Programme werden jetzt auch mehr in Richtung Prototyping entwickelt, also dass man zumindest mal den Prototypen einfach klickbar machen kann. Bei der Umsetzung hapert es bisher immer noch. Die Entwicklung selbst wird meines Kenntnisstands nach immer noch durch Programmierer durchgeführt. Die Umsetzung ist noch immer nicht so einfach, auch wenn ständig daran weiterentwickelt wird. Momentan gibt es jedoch noch Luft nach oben, was dieses Thema angeht.

Haben Sie den Eindruck, dass Apps auch für kleine Unternehmen immer wichtiger werden, wenn es darum geht, eigene Dienste anzubieten oder Werbung für das Unternehmen zu machen oder auch Arbeitsabläufe innerhalb des Unternehmens zu vereinfachen?

Das Problem ist, wenn es um unternehmensspezifische Apps geht, wird es für das Unternehmen meistens zu teuer. Gerade kleine Unternehmen können sich eine solche Entwicklung nicht leisten. Viele bekommen schon eine Krise, wenn sie erfahren, dass die Webseite ein bisschen teurer sein könnte, als sie gedacht haben. Was interne Arbeitsabläufe betrifft, weiß ich nicht, ob die Unternehmen diese dann auch immer wirklich nutzen. Ich weiß aber von mittelständischen Unternehmen, dass sie gerade im Servicebereich gerne Apps verwenden, um die Mitarbeiter zu kontaktieren, welche Termine anfallen. Gerade wenn Sie unterwegs sind, zum Beispiel bei einem Kunden, ist Telefonieren auch nicht immer das Beste. Also in dieser Hinsicht auf jeden Fall. Ganz kleine Unternehmen, glaube ich, ziehen da jedoch nicht ganz so mit.

Sie meinen, dass Smartphone-Anwendungen sehr teuer sind, gerade in der Entwicklung? Haben Sie auch eine Idee, woran das liegen könnte?

Also zum einen, bis man schon mal so etwas konzipiert hat, dauert dies eine ganze Weile, das weiß ich aus Erfahrung. Es dauert immer ein bisschen, bis die Idee da ist, bis geplant ist, was wo sein soll, damit ein Nutzer auch durchblickt, was jetzt gerade zu tun ist. Dann muss das Ganze gestaltet werden. Die Gestaltung braucht auch Zeit, denn es soll ja alles

verständlich und schön sein. Dann muss man wieder Einzelteile rausspeichern und mit einem Programmierer kommunizieren, wohin alles gehört und was programmiert werden soll. Und die Entwicklung mit der Fehlersuche und allem, dauert auch wieder eine Weile. Das sind einfach viele Arbeitsstunden. Könnte man alles so in einem Rutsch machen, also Konzept, Design und Umsetzung, wie bei Webseiten, dann würde dies sehr viel Zeit ersparen.

Was mich hier interessieren würde, wäre, Sie haben ja Mediendesign studiert. Wie sehr ist in Ihrem Studium auf Computersprachen wie HTML, CSS und JavaScript eingegangen worden?

Man muss sagen, wir hatten eher ein Grundlagenstudium. Wir sind in diese Richtung gar nicht so in die Tiefe gegangen. Wir haben ja im Endeffekt nur eineinhalb Jahre Studium. Aber wir hatten einen Intensivkurs mit ca. 40h, in denen wir uns beschäftigt haben mit diesem Thema. Wir haben uns dann bei HTML und CSS eingearbeitet und einfache Webseiten entwickelt. JavaScript ist eher am Rande behandelt worden. Es wurde eher darauf eingegangen, wo wir Code dafür finden, wenn wir ihn benötigen und wie man ihn einsetzen kann. Aber wir sind hauptsächlich bei HTML und CSS geblieben. Gerade CSS ist für uns Designer ja eigentlich am interessantesten. Es wurde uns quasi gezeigt, welchen Aufwand später die Programmierer haben, wenn sie für uns etwas entwickeln. Denn letztendlich sind wir immer noch Designer und keine Programmierer.

In dem Fall wurden Themen wie klassische Programmiersprachen, also Java usw. im Studium nicht besprochen?

Nein, gar nicht. In der Regel war es bei uns eher so: Mach das Layout für eine Website oder Ähnliches und wenn wir es wirklich umsetzen wollten, haben wir uns dann anschauen müssen, wie es sich leicht umsetzen lässt. Wenn wir Glück hatten, hatten wir einen Programmierer bei uns im Designkurs, der die Aufgabe dann übernommen hat. Oft wurde jedoch auch auf Baukastentools zurückgegriffen.

Denken Sie also, dass Ihnen ein Framework, welches es Ihnen ermöglicht, Webseiten zu Smartphone-Applikationen umzuwandeln, die Entwicklung von Apps vereinfachen würde?

Also dass man quasi eine Webseite hat und diese dann als App verwendet werden kann?

Ja genau.

Ok also im Endeffekt wäre es ähnlich zur Smartphone-Optimierung. Das kann auf jeden Fall hilfreich sein, zum Beispiel für Shop-Systeme oder ähnliche Sachen. Es gibt ja auch viele Baukästen, welche eine gewisse Smartphone-Optimierung anbieten. Wenn es jetzt zum Beispiel um Shops geht, wäre es auf jeden Fall eine tolle Sache, weil dies doch schon viele Sachen vereinfachen würde.

Mit dem Framework sollte es möglich sein, jede Webseite auch als App zu benutzen. Sie muss einfach nur so optimiert sein, dass sie responsive ist. Dabei soll es dem Designer möglich sein auch auf native Elemente und Sensoren, wie eine Kamera oder Drehungssensoren zuzugreifen.

Also, das kann auf jeden Fall nützlich sein. Es kommt aber auf den Kunden an, was er will. Will er einfach nur eine ganz einfache Webseite oder eher aufwendigere Sachen. Dort kann man auf jeden Fall eine ganze Menge Geld einsparen. Da kommt es aber auch dann darauf an, wie flexibel der Designer oder der Programmierer ist, denn sie sind letztendlich die, die das Ganze umsetzen.

Man könnte im Endeffekt die Webseite erstellen, wie man möchte, dabei gäbe es keine Begrenzung.

Das wäre natürlich super. Adobe hat ja auch schon Schritte in diese Richtung mit Prototyping umgesetzt, mit User Experience Design, und wenn man jedoch hinkriegen würde, tatsächlich ganze Apps wie eine Webseite zu bauen, dann würde das sehr viel Arbeit abnehmen.

Vielen Dank, dass Sie sich die Zeit genommen haben, meine Fragen zu beantworten.
Gerne. Gar kein Problem.

Anhang B: Monatsberichte

Monatsbericht: September

Fakultät Informatik

Thesis-Seminar im Bachelor-Studium – monatliche Berichterstattung

Berichtszeitraum: 02.09.2019 bis 27.09.2019

Thema der Thesis:	Entwicklung eines Frameworks für die Kommunikation und den Datenaustausch zwischen Webview- und nativem Code bei Smartphone-Applikationen
Bearbeiter/in:	Maximilian Marzeck (251990)
Kontakt:	Sie sind unter <maximilian.marzeck@hs-furtwangen.de> erreichbar.

Durchgeführte Arbeiten (13-15 Zeilen)

Hier sind die im Berichtszeitraum durchgeführten Arbeiten kurz und übersichtlich anzuführen.

- Die bereits vor Beginn erstellte, grobe Gliederung, welche angefertigt wurde, um dem Betreuer dieser Arbeit (Prof. Dr. Betermieux) einen Überblick über die Thematik zu verschaffen, wurde detaillierter ausgearbeitet.
- Ein Fragebogen an die Mediengestalterin des betreuenden Unternehmens wurde erstellt, dieser muss jedoch noch ausbessert werden (Siehe Abweichungen/Probleme)
- Es wurden zitierfähige Stellen und Quellen für diese Arbeit gesucht. Dabei wurden Zeitschriften, Webseiten und Bücher analysiert. Auch die Hochschulbibliothek der Hochschule Furtwangen wurde besucht, um weiteres Material zu finden.
- Das System zur Entwicklung, sowie die integrierte Entwicklungsumgebung, welche für die Umsetzung dieser Arbeit von Nöten ist, wurde eingerichtet und ein Testprojekt angelegt, mithilfe dessen Versuche durchgeführt werden können
- Es wurde recherchiert welche Frameworks und Bibliotheken ähnliche Aufgaben erfüllen und welche Stärken und Schwächen diese haben.
- Es wurde aufgelistet, welche Anforderungen das Framework erfüllen muss. So konnte auch festgestellt werden, dass die prototypische Umsetzung zunächst auf Basis von Android erfolgen sollte.

Erzielte Ergebnisse (8-10 Zeilen)

Hier sind die im Berichtszeitraum erzielten Ergebnisse kurz und übersichtlich anzuführen.

- Es konnte eine Gliederung angefertigt werden, die geeignet ist, als grober Leitfaden für diese Ausarbeitung zu dienen.
- Es konnte eine Menge Quellen und Material gesammelt werden
- Die Entwicklungsumgebung Android-Studio wurde eingerichtet und ein erstes Testprojekt angelegt. In diesem Testprojekt konnte testweise Code ausgeführt werden, welcher später möglicherweise im richtigen Projekt gebraucht wird.
- Es konnte herausgefunden werden, welche Frameworks und Werkzeuge bereits auf dem Markt sind und welche Vor- und Nachteile diese liefern.
- Für die prototypische Umsetzung des Projekts wird zunächst für Android entwickelt
- Ein großer Teil des „Ist-Analyse“ und Grundlagen-Kapitels konnte geschrieben werden

Abweichungen / Probleme (6-8 Zeilen)

Hier sind die im Berichtszeitraum bestehenden Abweichungen vom Plan oder entstandene / noch bestehende Probleme kurz und prägnant anzugeben. Zudem sind die getroffenen Maßnahmen zu nennen.

- Der für die Mediengestalterin des betreuenden Unternehmens erstellte Fragebogen wurde dem Unternehmen zugesendet, um eine Rückmeldung darüber zu bekommen, ob Unklarheiten in diesem Fragebogen vorhanden sind. Dabei wurde festgestellt, dass der Fragebogen noch einmal überarbeitet werden sollte, da er einige Unklarheiten enthält. Der Fragebogen wird noch einmal überarbeitet und dem betreuenden Unternehmen Anfang des nächsten Monats zugesandt.
- Die große Menge an Material, die bereits gefunden wurde, muss noch einmal überprüft und sortiert werden, da eine Verwendung aller Quellen den Rahmen der Arbeit sprengen würde

Ausblick über die geplanten Tätigkeiten und Ergebnisse des nächsten Berichtszeitraums (4-6 Zeilen)

- Der Fragebogen muss überarbeitet und dem Unternehmen nochmal zugesandt werden
- Es sollte ein Repository für das Projekt selbst eingerichtet werden (ein Vergleich von Repository-Anbietern sollte durchgeführt werden)
- Das Ist-Analyse und Grundlagen-Kapitel sollten vollendet werden
- Mit der Durchführung und dem dazu gehörenden Kapitel sollte begonnen werden

Monatsbericht: Oktober

Fakultät Informatik

Thesis-Seminar im Bachelor-Studium – monatliche Berichterstattung

Berichtszeitraum: 30.09.2019 bis 31.10.2019

Thema der Thesis:	Entwicklung eines Frameworks für die Kommunikation und den Datenaustausch zwischen Webview- und nativem Code bei Smartphone-Applikationen
Bearbeiter/in:	Maximilian Marzeck (251990)
Kontakt:	<code>maximilian.marzeck@hs-furtwangen.de</code>

Durchgeführte Arbeiten (13-15 Zeilen)

Hier sind die im Berichtszeitraum durchgeführten Arbeiten kurz und übersichtlich anzuführen.

- Das richtige Android-Studio-Projekt, in welchem das Framework entwickelt und getestet werden soll, wurde erstellt
- Im Projekt wurden bereits grundlegende Funktionen eingebunden. So kann beispielsweise CSS-„Code“ von Objekten (per ID und Klasse) verändert oder hinzugefügt werden. Außerdem kann aus dem Java-Code bereits sehr einfach Javascript-Code in die WebView injiziert werden.
- Es wurden bereits verschiedene Funktionen eingebunden, welche es erlauben, Java-Funktionen aus dem Javascript-Code der WebView auszuführen. Dabei ist es allerdings
- Nach einer Rücksprache mit der Mediengestalterin des betreuenden Unternehmens wurde festgelegt, dass eine geeignete GUI (grafische Benutzeroberfläche) von Vorteil wäre, die grundlegenden Funktionen und die Einbindung des Frameworks in ein bestehendes und neues Projekt zu erleichtern. Um das System auf allen mit Android-Studio kompatiblen System zugänglich zu machen, wird die GUI aktuell in Java entwickelt, für die grafische Oberfläche wird JavaFX verwendet. Diese GUI erhält nun grundlegende Funktionen (Datalayer und Logiclayer).
- Die Umfrage wurde mehrfach verbessert und ausgearbeitet, diese wird nun von Mediengestaltern und Webdesignern/Webentwicklern durchgeführt

Erzielte Ergebnisse (8-10 Zeilen)

Hier sind die im Berichtszeitraum erzielten Ergebnisse kurz und übersichtlich anzuführen.

- Das Projekt, in welchem die Entwicklung stattfinden soll, wurde eingerichtet und auf Github ein Repository dazu eingerichtet, um im Notfall stets ein Backup zu haben, zu dem zurückgekehrt werden kann.
- Das Framework enthält bereits Funktionen, welche das einfache Einbinden einer WebView, sowie die einfache Ausführung von Java-Code (aus dem WebView-Javascript-Code) und die Ausführung von Javascript-Code (in der WebView) aus dem Java-Code ermöglichen.
- Ein Java-Projekt wurde eingerichtet, in welchem die Benutzeroberfläche entwickelt werden soll. Grundlegende Funktionen in der Datenhaltungsebene und Logikebene sind bereits vorhanden, müssen jedoch noch erweitert werden.
- Die Umfrage befindet sich nun im finalen Stadium

Abweichungen / Probleme (6-8 Zeilen)

Hier sind die im Berichtszeitraum bestehenden Abweichungen vom Plan oder entstandene / noch bestehende Probleme kurz und prägnant anzugeben. Zudem sind die getroffenen Maßnahmen zu nennen.

- Die Entwicklung des Frameworks wurde zunächst unter Windows 7 als Betriebssystem begonnen. Da sich der Support dem Ende neigt und zu Beginn des Jahres 2020 keine neuen Updates mehr kommen sollen, nimmt auch die Softwareunterstützung für Windows 7 ab. Fehler in Intel Haxm führten dazu, dass der Emulator auf dem Windows 7 Rechner nicht mehr ausgeführt werden konnte, weshalb auf Linux Deepin als Entwicklungssystem umgestiegen wurde.
- Aufgrund dessen, dass die WebView unter Android in einem eigenen Thread ausgeführt wird, auf den lediglich vom Hauptthread zugegriffen werden kann, gibt es Probleme, wenn auf den Wert einer Javascript-Variable aus dem Java-Code zugegriffen wird, der Hauptthread jedoch weiterläuft und so mit „null“ statt dem entsprechenden Wert arbeitet.

Ausblick über die geplanten Tätigkeiten und Ergebnisse des nächsten Berichtszeitraums (4-6 Zeilen)

- Die Umfrage wird im nächsten Monat ausgewertet werden und die Ergebnisse protokolliert. Zusätzlich ist ein Interview mit der Mediengestalterin des betreuenden Unternehmens geplant.
- Das Framework soll weiter ausgebaut werden und die Fehler einiger Funktionen beseitigt werden. Des Weiteren werden alle Funktionen weiterhin getestet.
- Die zu dem Framework gehörige GUI soll erweitert werden und die Einbindung des Frameworks in Projekt erleichtern.

Monatsbericht: November

Fakultät Informatik

Thesis-Seminar im Bachelor-Studium – monatliche Berichterstattung

Berichtszeitraum: 01.11.2019 bis 30.11.2019

Thema der Thesis:	Entwicklung eines Frameworks für die Kommunikation und den Datenaustausch zwischen Webview- und nativem Code bei Smartphone-Applikationen
Bearbeiter/in:	Maximilian Marzeck (251990)
Kontakt:	maximilian.marzeck@hs-furtwangen.de

Durchgeführte Arbeiten (13-15 Zeilen)

Hier sind die im Berichtszeitraum durchgeführten Arbeiten kurz und übersichtlich anzuführen.

- Die Arbeit an der Hauptklasse des Frameworks wurde fortgeführt. Es wurden Fehler gesucht und diese verbessert. Außerdem wurde nach einem Weg gesucht Variablen aus dem JavaScript-Code der WebView in den Java-Code zu übertragen. Außerdem wurde ein Weg gesucht Funktionen und Methoden im nativen Code durch JavaScript-Code aus der WebView abzurufen. Dabei wurden verschiedene Methoden implementiert, wobei sich einige als nicht zielführend herausstellten.
- Es wurde eine JavaScript und CSS-Dateien erstellt und daran gearbeitet, welche ebenfalls Teil des Frameworks werden sollen, um leichter native Elemente durch HTML-Elemente ersetzen zu können und um den Prozess des Aufrufens von nativen Funktionen zu erleichtern.
- An der für das Framework erstellten grafischen Benutzeroberfläche wurde gearbeitet. Das Front-End wurde eingebunden und bedeutende Funktionen wurden in die grafische Oberfläche eingebunden.
- Das Ist-Situations-Analyse-Kapitel sowie das Grundlagen-Kapitel wurden überarbeitet. Weitere Quellen wurden der Bachelorarbeit hinzugefügt. Außerdem wurde eine Einleitung und ein Teil der Durchführung der Bachelorarbeit geschrieben.

Erzielte Ergebnisse (8-10 Zeilen)

Hier sind die im Berichtszeitraum erzielten Ergebnisse kurz und übersichtlich anzuführen.

- Es konnte ein Weg gefunden werden, der es ermöglicht JavaScript-Variablen aus der WebView in den nativen Java-Code zu übertragen, ohne dabei Probleme durch Asynchronität verschiedener Threads zu erhalten.
- Fehler der Hauptklasse des Frameworks konnten gefunden und beseitigt werden. Zusätzlich konnten einige weitere Funktionen eingebaut werden.
- Das Framework wurde durch CSS-Klassen und JavaScript-Code erweitert, um die Arbeit mit der Hauptklasse zu erweitern.
- Die GUI des Frameworks wurde erweitert, dadurch lassen sich jetzt Aufgaben, welche normalerweise stets durch Java- oder XML-Code eingebunden werden müssen (z.B. einbinden der WebView in die MainActivity) grafisch gelöst werden.

Abweichungen / Probleme (6-8 Zeilen)

Hier sind die im Berichtszeitraum bestehenden Abweichungen vom Plan oder entstandene / noch bestehende Probleme kurz und prägnant anzugeben. Zudem sind die getroffenen Maßnahmen zu nennen.

- Der Fragebogen konnte noch nicht ausgewertet werden, da zum jetzigen Zeitpunkt keine aussagekräftige Anzahl Teilnehmer gefunden werden konnte. Sollten sich in den nächsten Wochen keine Teilnehmer finden, so wird statt dem Fragebogen ein Interview mit der Medien- und Webgestalterin des betreuenden Unternehmens durchgeführt, um die Arbeitsweise von Webgestaltern besser durchleuchten zu können.
- Das Finden eines Weges zur Übertragung von Daten und Informationen aus dem JavaScript-Code (aus der WebView) in den nativen Code nahm deutlich mehr Zeit in Anspruch, als erwartet, daher muss noch Einiges nachgeholt werden.

Ausblick über die geplanten Tätigkeiten und Ergebnisse des nächsten Berichtszeitraums (4-6 Zeilen)

- Die Hauptklasse des Frameworks soll fertig gestellt werden und dem Informatiker des betreuenden Unternehmens präsentiert und mit ihm besprochen werden.
- Der JavaScript- und CSS-Code des Frameworks muss erweitert werden und diese ebenfalls mit dem Informatiker des betreuenden Unternehmens besprochen werden
- Die grafische Benutzeroberfläche zur Einbindung des Frameworks soll fertiggestellt werden
- Das Kapitel „Einleitung“ und die Durchführung sollen überarbeitet und fertiggestellt werden

Monatsbericht: Dezember

Fakultät Informatik

Thesis-Seminar im Bachelor-Studium - monatliche Berichterstattung

Berichtszeitraum: 01.12.2019 bis 31.12.2019

Thema der Thesis:	Entwicklung eines Frameworks für die Kommunikation und den Datenaustausch zwischen Webview- und nativem Code bei Smartphone-Applikationen
Bearbeiter/in:	Maximilian Marzeck (251990)
Kontakt:	maximilian.marzeck@hs-furtwangen.de

Durchgeführte Arbeiten (13-15 Zeilen)

Hier sind die im Berichtszeitraum durchgeführten Arbeiten kurz und übersichtlich anzuführen.

- Eine Funktion wurde in die Hauptklasse eingebaut, welche ermöglicht, die Drehung von Apps aus dem Javascript-Code der WebView ein- und auszuschalten
- An einer Implementierung zur Speicherung von Online-Webseiten im Webcache wurde gearbeitet. Diese soll ermöglichen, dass Web-Apps auch bei fehlender Internetverbindung geladen werden können, wenn diese zuvor bereits mit funktionierender Internetverbindung abgerufen wurden.
- Das "Ist-Situations"-Kapitel wurde überarbeitet. Außerdem wurde begonnen, bisher erzielte Ergebnisse zu dokumentieren
- Ein Experteninterview wurde im begleitenden Unternehmen durchgeführt, welches genauere Einblicke in die Arbeit von Agenturen geben soll und aufzeigt, warum App-Entwicklung auch für kleine Unternehmen wichtig ist, es jedoch meist die Umsetzung ein Problem ist.
- Die bisherigen Ergebnisse und das weitere Vorgehen wurden mit dem Firmenbetreuer besprochen
- Es wurden weitere nützliche Quellen für die Thesis gesucht.

Erzielte Ergebnisse (8-10 Zeilen)

Hier sind die im Berichtszeitraum erzielten Ergebnisse kurz und übersichtlich anzuführen.

- Der Code wurde auf Fehler und Probleme untersucht. Die entdeckten Fehler wurden beseitigt. Zusätzliche Funktionen wurden dem Code hinzugefügt.
- Der Code wurde umgestaltet, um ihn performanter zu machen.
- Durch das durchgeführte Experteninterview wurde ein Einblick in die Arbeit von Agenturen gewonnen und erkannt, warum Smartphone-Applikationen, trotz ihrer wichtigen Stellung als Informations- und Softwareverbreitungsmethode, immer noch nur großen Unternehmen vorbehalten ist.
- Ein Teil der Thesis konnte überarbeitet werden. Die Arbeit konnte zudem um Text und Quellen ergänzt werden.

Abweichungen / Probleme (6-8 Zeilen)

Hier sind die im Berichtszeitraum bestehenden Abweichungen vom Plan oder entstandene / noch bestehende Probleme kurz und prägnant anzugeben. Zudem sind die getroffenen Maßnahmen zu nennen.

- Ein Teil des Codes stellte sich als nicht sehr performant heraus. Der betroffene Code ist bereits teilweise verbessert worden.
- Die geplante Umfrage hat leider noch immer nicht genügend Teilnahmen erhalten, um Aussagekräftig genug zu sein. Daher wurde stattdessen ein Experteninterview durchgeführt.
- Die Hauptklasse des Frameworks ist noch nicht voll ausgebaut, auch muss ein Teil des Codes der Grafischen Benutzeroberfläche noch ergänzt und verbessert werden.

Ausblick über die geplanten Tätigkeiten und Ergebnisse des nächsten Berichtszeitraums (4-6 Zeilen)

- Die Performanz des geschriebenen Codes soll nochmal überprüft werden und gegebenenfalls problematische Stellen durch performantere ausgetauscht werden.
- Die Hauptklasse soll weitere Sensorzugriffsmöglichkeiten (z.B. Kamera, GPS-Position usw.) hinzugefügt werden.
- Die Hauptklasse, die grafische Benutzeroberfläche und die wichtigsten Teile der Thesis sollen fertiggestellt werden, um diese bei der Hochschulpräsentation vorzustellen.

Monatsbericht: Januar

Fakultät Informatik

Thesis-Seminar im Bachelor-Studium – monatliche Berichterstattung

Berichtszeitraum: 01.01.2020 bis 31.01.2020

Thema der Thesis:	Entwicklung eines Frameworks für die Kommunikation und den Datenaustausch zwischen Webview- und nativem Code bei Smartphone-Applikationen
Bearbeiter/in:	Maximilian Marzeck (251990)
Kontakt:	maximilian.marzeck@hs-furtwangen.de

Durchgeführte Arbeiten (13-15 Zeilen)

Hier sind die im Berichtszeitraum durchgeführten Arbeiten kurz und übersichtlich anzuführen.

- Für die Präsentation der Thesis in Furtwangen wurde eine Power-Point-Präsentation erstellt, welche den Inhalt der Thesis behandelt und die bisherigen Ergebnisse aufzeigt.
- Ebenfalls für die Präsentation, wurde eine Mockup-App entwickelt, welche einen Teil des Funktionsumfangs des Frameworks aufzeigen soll.
- Es wurde an der Einbindung weiterer Sensoren in das Framework gearbeitet. So sollen Kamerazugriff, Standortzugriff und Vibrationsmotorzugriff aus dem HTML/JavaScript-Code möglich sein.
- Die Arbeit an weiteren problematischen Codestellen wurde aufgenommen, um das Framework als Ganzes performanter zu gestalten. Statt eines eigenen Caching-Verfahrens, wurden Standardfunktionen des WebView-Cache in das Framework aufgenommen.
- Das Einleitungskapitel, die Ist-Situation und das Grundlagenkapitel der schriftlichen Thesis wurden überarbeitet.
- Die Einbindung des Frameworks in bestehende Projekte wurde in die grafische Benutzeroberfläche eingebaut.

Erzielte Ergebnisse (8-10 Zeilen)

Hier sind die im Berichtszeitraum erzielten Ergebnisse kurz und übersichtlich anzuführen.

- Das Framework ist nun fast in seinem finalen Zustand. Die prototypische Umsetzung beinhaltet alle wichtigen Sensoren und Funktionen. Gefundene Fehler und Probleme wurden beseitigt.
- Die Präsentation und die Präsentations-App wurden fertiggestellt und die Präsentation in Furtwangen vorgetragen.
- Weite Teile der Thesis sind nun überarbeitet.
- Die grafische Benutzeroberfläche bindet nun das Framework selbstständig in bestehende Android-Studio-Projekte ein und erlaubt die einfache Einstellung wichtiger Parameter.

Abweichungen / Probleme (6-8 Zeilen)

Hier sind die im Berichtszeitraum bestehenden Abweichungen vom Plan oder entstandene / noch bestehende Probleme kurz und prägnant anzugeben. Zudem sind die getroffenen Maßnahmen zu nennen.

- Während der Entwicklung der Präsentations-App wurden kleinere Fehler entdeckt, da das Framework kurz zuvor verändert wurde, jedoch bestimmte Codestellen bei der Überarbeitung vergessen wurden. Die Fehler wurden gesucht und entfernt, so liefen das Framework und die grafische Benutzeroberfläche einwandfrei.
- Durch die breite Splitterung des Android-Betriebssystems in verschiedenste Versionen, kam es zu Problemen bei der Abfrage der Berechtigungen, bei älteren Versionen werden diese direkt bei der Installation angefordert, bei neueren erst im Code. Vor der Verwendung z.B. der Kamera wird daher die Berechtigung im Code nochmals abgefragt, falls es sich um ein neues Android-System handelt.

Ausblick über die geplanten Tätigkeiten und Ergebnisse des nächsten Berichtszeitraums (4-6 Zeilen)

- Das Framework wird nochmals auf Fehler überprüft und diese gegebenenfalls aus dem Code entfernt/verbessert. Zudem wird der Code mit einigen Kommentaren versehen und Sinnvoll strukturiert um die Lesbarkeit zu verbessern.
- Der schriftliche Teil der Thesis wird zu Ende überarbeitet und dieser dann nochmals auf Fehler überprüft. Danach wird diese gedruckt und abgegeben.