Förderkreis der Angewandten Informatik

an der
Westfälischen
Wilhelms-Universität Münster e.V.

Working Paper No. 4

Business Apps: Grundlagen und Status quo

Henning Heitkötter, Tim A. Majchrzak,
Ulrich Wolffgang, Herbert Kuchen

Dezember 2012

Westfälische
Wilhelms-Universität
Münster

Working Paper No. 4

Business Apps: Grundlagen und Status quo

Henning Heitkötter, Tim A. Majchrzak,
Ulrich Wolffgang, Herbert Kuchen

Dezember 2012

Henning Heitkötter, Tim A. Majchrzak,
Ulrich Wolffgang, Herbert Kuchen
Institut für Wirtschaftsinformatik
Westfälische Wilhelms-Universität Münster
Leonardo Campus 3
48149 Münster
tima@ercis.de

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http:/dnb.ddb.de abrufbar.

ISSN 1868-0801

Printed in Germany

Vorwort

In den letzten Jahren hat sich die Leistungsfähigkeit mobiler Endgeräte stark erhöht. Neben klassischen Mobiltelefonen werden Smartphones und Tablets angeboten, die einen deutlich erhöhten Funktionsumfang bieten. Um die Geräte optimal zu nutzen und ihre Funktionen zu erweitern, können Applikationen – die sogenannten Apps – installiert werden. Diese rücken zunehmend in den Fokus von Unternehmen.

Viele der Unternehmen im IHK-Bezirk Nord Westfalen zeichnen sich dadurch aus, dass Informationstechnologie (IT) für sie eine hohe Bedeutung hat oder dass sie IT-Leistungen erbringen. Aus diesem Grund ist für sie die Beschäftigung mit dem Hype-Thema *Apps* unerlässlich. Apps erzeugen per se aber keinen Nutzen. Vielmehr ist ein geschickter Einsatz als Business Apps angebracht, um sich ergebende Möglichkeiten optimal und vor allem wirtschaftlich zu nutzen.

Die vorliegende Broschüre "Business Apps" trägt den Status quo der Entwicklung und des Einsatzes von Applikationen für mobile Endgeräte von Unternehmen der Region zusammen. Ferner wird ein Überblick über Ansätze zur Cross-Plattform-Entwicklung gegeben. Darüber hinaus erfolgt die Betrachtung erfolgreicher Strategien, die von den Unternehmen der Region entwickelt wurden, sowie einiger angrenzender Themen, die von ihnen als besonders wichtig erachtet werden.

Die Broschüre ist durch den Förderkreis für Angewandte Informatik an der Westfälischen Wilhelms-Universität Münster angeregt und finanziert worden. Der Förderkreis wird von rund 30 Unternehmen aus der Region und der Industrie- und Handelskammer Nord Westfalen getragen. Hauptziel ist die Förderung der praxisorientierten Forschung und Lehre, sowie der schnelle Wissenstransfer. Besonderer Dank gebührt dem Direktor des Instituts für Angewandte Informatik, Herrn Professor Dr. Herbert Kuchen, und seinen Mitarbeitern, Herrn Henning Heitkötter, Herrn Dr. Tim A. Majchrzak und Herrn Ulrich Wolffgang für die außerordentlich engagierte und praxisnahe Umsetzung des Projektes.

Großer Dank gilt auch den Mitgliedsunternehmen, die sich am Projekt beteiligt haben und deren Mitarbeiter für umfangreiche Interviews zur Verfügung standen. Diese Unterstützung bildet die Basis der Broschüre. Aufgrund der Sensibilität der Informationen werden einzelne Unternehmen in diesem Rahmen nicht benannt.

Martin Kittner
Vorsitzender des Förderkreises
für Angewandte Informatik

Dr. Christoph Asmacher
Industrie- und Handelskammer
Nord Westfalen

Inhaltsverzeichnis

Abkürzungsverzeichnis

Die Nummern hinter Einträgen bezeichnen die Seitenzahlen der Verwendung.

API	Application Programming Interface 20, 23, 24, 58, 60, 63, 64, 66, 73–78, 81, 82, 104–107, 112
BYOD	Bring your own device 36, 42, 105
CSS	Cascading Style Sheets 61, 69–73, 76
CSS3	Weiterentwicklung von CSS 70
DSL	Domain-specific Language 82
GPS	Global Positioning System 15, 48, 59, 109, 113
GUI	Graphical User Interface 44, 64, 67, 69, 72, 77, 78, 80, 82–85, 87
HTML	Hypertext Markup Language 23, 61, 69–73, 76, 78
HTML5	Zukünftige (im Wesentlichen fertiggestellte) Version von HTML und Menge von Web-Standards . . . 37, 50, 61, 64, 66, 70–72, 75, 76, 80, 112
HTTP	Hypertext Transfer Protocol 69
IAI	Institut für Angewandte Informatik 27, 46, 117, 118
IDE	Integrated Development Environment 67
IT	Informationstechnologie i, 2, 3, 16, 28, 29, 32, 39, 50, 51, 87, 91, 102
MDM	Mobile Device Management 19, 42, 104, 114, 115
MDSD	Model-driven Software Development 96
NFC	Near Field Communication 66
PC	Personal Computer . 14, 15, 18, 19, 22, 92, 95, 100, 101, 105, 106, 114
PDA	Personal Digital Assistant 35

PIM	Personal Information Manager 35, 115
REST	Representational State Transfer 87
RIM	Research In Motion 16, 17, 23, 24, 42
SDK	Software Development Kit . 18, 20, 23, 58, 60, 63, 64, 75, 87, 105
SOA	Service-oriented Architecture 90, 91

Executive Summary

Moderne mobile Endgeräte wie Smartphones und Tablets eröffnen neue Nutzungsszenarien. *Apps* sind daher in den Fokus von Unternehmen gerückt. Aktuell wird diskutiert, Apps zur Unterstützung von Geschäftsprozessen und zur Kundenbindung einzusetzen. Darüber hinaus werden Apps für IT-Dienstleister und Softwareentwickler als Produkt interessant.

Apps werden aller Regel nach individuell und nicht als Standardsoftware entwickelt. Problematisch ist die allgemein recht geringe Erfahrung mit der Entwicklung kommerzieller Apps sowie die Fragmentierung des Marktes für Plattformen mobiler Endgeräte. Mit Googles Android, RIMs BlackBerry, Apples iOS, Nokias Symbian und Microsofts Windows Phone gibt es mindestens fünf relevante Plattformen, die weitgehend inkompatibel sind. Insbesondere bei der Entwicklung für Kunden müssen mehrere oder sogar alle dieser Plattformen abgedeckt werden. Für Unternehmen sind daher Ansätze zur Cross-Plattform-Entwicklung relevant, deren Ziel die einmalige Programmierung bei gleichzeitiger Unterstützung mehrerer Plattformen ist.

Vor diesem Hintergrund initiierte das Institut für Angewandte Informatik (IAI) das Projekt *Business Apps*, das der Lehrstuhl für Praktische Informatik durchführte. Ziel war es, den Status quo der App-Entwicklung in Unternehmen der Region darzustellen, Handlungsempfehlungen insbesondere für die Cross-Plattform-Entwicklung zusammenzutragen und weitere Fragestellungen rund um die App-Entwicklung aufzudecken. Die Ergebnisse werden im folgenden stark verdichtet wiedergegeben.

Eine Umfrage sowie elf sehr umfangreiche Interviews mit Mitgliedsunternehmen des IAI-Förderkreises bilden die Grundlage der Projektergebnisse. Alle teilnehmenden Unternehmen erbringen entweder IT-Leistungen für Kunden oder betreiben eine umfangreiche Softwareentwicklung zur Unterstützung eigener Geschäftsprozesse. Ziel der Interviews war es, tiefe Einblicke in die Erfahrungen der Interviewpartner mit der App-Entwicklung zu bekommen und Details zu erfolgreichen Vorgehensweisen und Problemen zu erfahren. Nach Abschluss der Interviews wurden die Ergebnisse ausgewertet und die vorliegende Broschüre erstellt (vgl. Kapitel 1).

Hinsichtlich Smartphones und Tablets lassen sich einige Besonderheiten feststellen (Kapitel 3). Festzuhalten ist, dass zur erfolgreichen Entwicklung von Apps sowohl auf die Eigenschaften der Geräte als auch auf die Möglichkeiten der Plattformen Rücksicht genommen werden muss. Eine spezifische Einarbeitung ist unbedingt anzuraten, auch wenn die Lernkurve selbst für wenig erfahrene Entwickler als steil gelten kann.

Die Interviews lieferten zahlreiche Erkenntnisse (Kapitel 4). Während im Detail sehr heterogene Aussagen getroffen wurden, gab es in fast allen wesentlichen Punkte eine breite Übereinstimmung. Zunächst ist das hohe Interesse der Unternehmen der Region festzustellen. Sie achten darauf, das Thema Business Apps nicht unreflektiert als *Hype* zu verfolgen. Gleichzeitig haben fast alle Teilnehmer konkrete Pläne, Apps umzusetzen. Bisherige Projekte verliefen recht unstrukturiert, insbesondere wurden Mitarbeiter kaum geschult. Das Interesse bei den Mitarbeitern wurde aber als durchgehend hoch beschrieben, vor allem aus Interesse an der Technologie.

Die Cross-Plattform-Entwicklung spielt für die Mehrzahl der Unternehmen eine Rolle, da sie die Unterstützung mehrerer Plattformen für wichtig erachten. Dies gilt besonders bei Apps für Endkunden. Festzuhalten ist ferner, dass Web-basierte Cross-Plattform-Lösungen aufgrund des nicht nativen Look & Feels nur in einigen Fällen in Betracht gezogen werden.

Es gibt verschiedene Ansätze für die Cross-Plattform-Entwicklung von Apps (Kapitel 5). *Web-Apps* bauen auf Web-Technologien wie HTML und JavaScript auf. Sie sind in der Regel einfach zu entwickeln und in den Browsern aller neueren Endgeräte lauffähig, allerdings bezüglich der Leistungsfähigkeit eingeschränkt. Sie bieten *kein* plattformspezifisches Look & Feel. *Hybride Ansätze* bauen ebenfalls auf Web-Technologie auf, schließen allerdings die Lücke zur Nutzung gerätespezifischer Funktionen. *Apache Cordova* hat sich hierbei als sehr gut zu nutzendes Framework herausgestellt. Es kann aber wie alle hybriden Ansätze kein vollständig natives Look & Feel bieten. Einen Schritt weiter gehen Ansätze mit *eigener Laufzeitumgebung*. Sie können einem nativen Look & Feel näher kommen, haben aber einen erhöhten Einarbeitungsaufwand. *Modellgetriebene Ansätze* und *Cross Compiler* erzeugen vollständig native Apps. Bestehende Ansätze kommen jedoch kaum über den Stand von Forschungsprototypen heraus. Zum Vergleich muss schließlich die Möglichkeit erwähnt werden, nativ für alle zu unterstützenden Plattformen zu entwickeln. Der Aufwand steigt hierbei fast linear mit der Zahl der Plattformen.

Aufbauend auf Kriterien zur Auswahl und Bewertung von Ansätzen kann die Empfehlung gegeben werden, Cordova auszuprobieren, sofern ein natives Look & Feel nicht im Vordergrund steht. Auch reine Web-Apps können sinnvoll sein, vor allem zur Ergänzung bestehender Web-Lösungen bzw. deren Optimierung für mobile Endgeräte. Die native Entwicklung sollte dann erfolgen, wenn mit Web-basierten Lösungen nur unbefriedigende Ergebnisse zu erzielen sind, etwa bezüglich Look & Feel oder der Nutzung von Gerätefunktionen.

Cross-Plattform-Lösungen, die nativen Code erzeugen, sollten im Auge behalten werden. In diesem Rahmen wurde vom Lehrstuhl das Framework MD^2 entwickelt, mit dem sich modellgetrieben Apps generieren lassen. Diese werden in einer domänenspezifischen Sprache (DSL) beschrieben, die eine sehr kompakte Darstellung ermöglicht. Sie ist im Allgemeinen für Domänenexperten verständlich, nicht nur für Programmierer. Aus dem Modell der App wird plattformspezifischer Code gene-

riert; derzeit unterstützt MD^2 Android und iOS. Trotz Einschränkungen – im Fokus stehen datengetriebene, formularbasierte Business Apps – verläuft die Erprobung unseres Ansatzes sehr vielversprechend.

Durch die Neuheit des Themas sehen fast alle Unternehmen Forschungsbedarf und die Notwendigkeit, in verschiedenen Bereichen Erfahrungen aufzubauen. Nichts desto trotz konnten bereits erste positive Herangehensweisen identifiziert werden (Kapitel 6). Aus technischer Sicht hervorzuheben ist der Aufbau von Apps aus Komponenten bzw. die Orchestrierung von Web-Services zur Nutzung durch Apps. Betriebswirtschaftlich empfehlenswert ist es, durch Apps die Verfügbarkeit von Informationen im Unternehmen zu erhöhen, um Abläufe effizienter zu gestalten. Sehr zuversichtlich sind die Unternehmen zudem hinsichtlich des Einsatzes mobiler Endgeräte in Außendienst und Vertrieb.

Eine Reihe der besonders häufig erwähnten bzw. besonders herausfordernden Aspekte der App-Entwicklung wurde genauer analysiert. Wo möglich, geben wir auch erste Empfehlungen (vgl. Kapitel 7). In Zukunft werden vor allem die Sicherheit auf mobilen Endgeräten, Offline-Nutzung von Apps, das Testen sowie das Mobile Device Management (MDM) eine Rolle spielen. Die Sicherheit von Apps ist für Unternehmen sehr wichtig. Gleichzeitig bieten die gängigen Plattformen größtenteils wenig „eingebaute“ Sicherheit. Darüber hinaus müssen Strategien entwickelt werden, mit Bedrohungen umzugehen. Unternehmen stehen zudem vor wegbereitenden Entscheidungen, etwa ob die Nutzung mitarbeitereigener Geräte (BYOD – *Bring your own device*) zugelassen wird. Apps müssen in vielen Fällen mit schwankender oder fehlender Netzabdeckung arbeiten können. Ein Verkaufsgespräch durch einen Außendienstmitarbeiter, der ein Tablet benutzt, wäre zum Scheitern verurteilt, wenn plötzlich keine Daten mehr abgerufen werden können. Während bereits eine Reihe von Mechanismen zur Verfügung steht, muss ihr Einsatz im Detail geprüft werden.

Das Testen von Apps wird dadurch erschwert, dass verschiedenste Geräte, Plattformen und Frameworks mit einbezogen werden müssen. Während in diesem Fall Emulatoren und Cloud-basierte Lösungen Hilfestellung bieten, stehen erprobte Strategien zum Testen von *Kontextwechseln* noch aus. Apps sind ständig wechselnden Bedingungen ausgesetzt, etwa hinsichtlich der Qualität des verfügbaren Netzzugangs oder externen Ereignissen wie das „Kippen“ des Bildschirms. Diese können die Zahl notwendiger Testfälle vervielfachen. Geeignete technische Lösungen stehen nur begrenzt bereit; generell ist beim Testen von Apps auf Kontextwechsel zu achten.

Sollen mobile Geräte in Unternehmen genutzt werden, ist eine zentrale Verwaltung inklusive Softwareverteilung erstrebenswert. Für PCs und Laptops ist dies selbstverständlich. Der Markt für entsprechende MDM-Produkte wächst schnell, ist aber vergleichsweise überschaubar. Auch bieten viele Produkte noch keinen Funktionsumfang, der für typische Szenarien ausreichend erscheint. Die Beschäftigung mit MDM-Lösungen ist für alle Unternehmen, die eine größere Zahl mobiler Endgeräte für ihre Mitarbeiter zur Verfügung stellen (wollen), empfehlenswert.

Kapitel 1

Einführung

Mobile Endgeräte bieten zunehmend mehr Funktionen als klassische *Handys*. Das Telefonieren ist nur noch eine Funktion unter vielen; Rechengeschwindigkeit und Speicherplatz heutiger *Smartphones* überschreiten Werte, die erst vor wenigen Jahren nur bei leistungsfähigen Workstations zu beobachten waren. Darüber hinaus bieten sich diese Geräte aufgrund ihrer handlichen Größe und ihres Bedienkonzepts – die Tastatur wird durch *Touchscreens* ersetzt – an, um neue Einsatzszenarien zu erschließen. Smartphones finden verstärkt Verbreitung unter Endverbrauchern.

Die Möglichkeiten von Smartphones und Geräten der neu hinzugekommenen Kategorie der *Tablets* lassen sich allerdings nur durch den Einsatz entsprechender Software voll erschließen. Während frühere Handys mit einer kaum oder nur unwesentlich erweiterbaren Palette von Funktionen und – als solche in der Regel kaum zu erkennenden – Programmen kamen, ist die Installation neuer Anwendungen auf modernen Geräten nicht die Ausnahme, sondern die Regel. Ähnlich wie dies im Rahmen der Entwicklung von Computern für geschäftliche Szenarien bereits in den 60er Jahren des letzten Jahrhunderts zu beobachten war, nimmt die Bedeutung der Hardware gegenüber der Software stark zu. Es ist zu erwarten, dass trotz des Bestrebens der Gerätehersteller, durch entsprechende Produkte zu überzeugen, der Fokus der Kunden zukünftig stärker auf die Software gerichtet sein wird.

Für Anwendungen für mobile Endgeräte hat sich der Begriff *Apps* eingebürgert. Obgleich Apps in technischer Hinsicht schlicht Computerprogramme sind, transportiert der Begriff App viele Charaktergereigenschaften. Apps werden als leichtgewichtig empfunden und sind im Allgemeinen schnell herunterzuladen und zu installieren. Über entsprechende Vertriebskanäle (*app stores*) sind sie in der Regel leicht aufzufinden. So entsteht vor allem für Nutzer mit wenig Interesse an technischen Grundlagen der Eindruck, es gäbe für jeden möglichen Bedarf eine App. Von der Industrie wird hiermit sogar explizit geworben, obwohl zum jetzigen Zeitpunkt verfügbare Programme für Microsoft Windows- und Linux-basierte Systeme (noch) in deutlich größerer Zahl und mit dramatisch größerem Funktionsumfang verfügbar sein dürften. Ein wesentlicher Vorzug von Apps ist demnach die Zugänglichkeit sowie die empfundene Einfachheit der Nutzung.

Die oben genannten Möglichkeiten ergeben drei wesentliche Zielgruppen für Apps aus Sicht eines Unternehmens. Als erstes bieten sie sich für die Unterstützung der Geschäftsprozesse des eigenen Unternehmens an. Dazu richten sie sich an Mitarbeiter oder binden Geschäftspartnern mit ein. Als zweite Kategorie lässt sich die Nutzung durch Endkunden, etwa im Rahmen von Vertriebs-, Support- oder Marketingaktivitäten nennen. Die dritte Kategorie ist Entwicklung und gegebenenfalls Betrieb *für*, bzw. *im Auftrag von* Kunden. Dies ist nur für solche Unternehmen interessant, die primär Software entwickeln.

Die ersten beiden Kategorien sind für fast jede Art von Unternehmen interessant. IT-gestützte Prozesse sind seit vielen Jahren die Regel. Dementsprechend kann geprüft werden, ob die Integration von mobilen Endgeräten zur Prozessverbesserung beitragen kann. Denkbar ist sowohl, Informationen mobil zur Verfügung zu stellen, als auch Daten mobil zu erfassen. Mobile E-Mail-Dienste sind spätestens mit der Verbreitung von *BlackBerry*-Geräten vor einigen Jahren populär geworden; der mobile Datenabruf bietet sich z. B. für Mitarbeiter im Außendienst an. Die mobile Datenerfassung kann soweit gehen, dass ein klassisches Klemmbrett mit einer Checkliste durch ein Smartphone ersetzt wird. Apps für den Kunden können ebenfalls zur Dateneingabe und -bereitstellung genutzt werden – etwa als passendes Gegenstück zu einer App, die von einem Versicherungsvertreter im Außendienst genutzt wird. Daneben bietet sich die Nutzung als Marketingwerkzeug an. Wird Kunden ein Mehrwert geboten – häufig in Form von unterhaltsamen Inhalten – kann dies etwa zur Werbung genutzt werden.

Zu bedenken ist, dass Apps im Allgemeinen keine Standardsoftware sind. Apps bieten sich gerade für den Einsatz in Gebieten an, in denen die Integration von IT zur Erreichung fachlicher Ziele erst kürzlich in den Fokus gerückt ist und individuelle Lösungen benötigt werden. Hier sollten sie in enger Absprache mit Kunden entwickelt werden. Ein Beispiel hierfür ist die Nutzung von Apps für patientenorientierte medizinische Informationssysteme. Auch wenn IT zum medizinischen Alltag gehört, so ist erst mit dem Aufkommen von Apps zu beobachten, dass Systeme entwickelt werden, derer *primärer* Nutzerkreis die Patienten sind. Die Erfahrung zeigt dabei, dass solche Systeme auch von Menschen nutzbar sind, die vorher keine Erfahrung mit PCs hatten. Es ist zu vermuten, dass sich dieser Zusammenhang auf weitere Einsatzbereiche übertragen lässt.

Ein Problem bei der Entwicklung von Apps ist die starke Fragmentierung des Markts für mobile Endgeräte. Mit Googles *Android*, RIMs *BlackBerry*, Apples *iOS*, Nokias *Symbian* und Microsofts *Windows Phone* gibt es mindestens fünf relevante Plattformen, die weitgehend inkompatibel sind. Darüber hinaus gibt es weitere Plattformen, die z. T. Nischenmärkte besetzen. Die Marktanteile verschieben sich aufgrund des schnell wachsenden Absatzes von mobilen Endgeräten mit einer hohen Dynamik. Dazu kommt, dass es mitunter erhebliche Unterschiede zwischen mobilen Endgeräten derselben Plattform gibt. Die Leistungsfähigkeit kann sehr heterogen

ausfallen; durch die Unterstützung von Smartphones *und* Tablets werden zudem in der Form sehr unterschiedliche Geräte abgedeckt. Schließlich sorgt die schnelle Versionsfolge der Plattformen selbst für eine Fragmentierung innerhalb einer Plattform (intra-Plattform), bei der unterschiedliche Versionen sich erheblich im Funktionsumfang unterscheiden. Dies gilt vor allem für Android.

Die Fragmentierung stellt aus Entwicklersicht eine große Herausforderung dar, die sich aber mit entsprechendem Aufwand meistern lässt. Viel größer sind die betriebswirtschaftlichen Konsequenzen. Um einen hinreichend großen Anteil der Zielgruppe zu erreichen, müssen mehrere Plattformen unterstützt werden. Unklar ist, wie sich diese geschickt auswählen lassen. Die Entwicklung nativer Applikationen (und dem vorausgehend z. B. die Ausbildung der Entwickler) für einzelne Plattformen ist sehr kostenintensiv. Ad-hoc-Lösungen, etwa das Ausweichen auf ausschließlich Webbasierte-Lösungen, sind je nach Einsatzszenario nur ein sehr unbefriedigender Kompromiss. Nicht komplett durchdachte Lösungen können gar kontraproduktiv sein, etwa wenn Kunden Apps als unvollkommen empfinden oder fehlende Unterstützung für "ihre" Plattform monieren. Aber auch für den unternehmensinternen Einsatz ergeben sich Probleme. So stellt sich die Frage, ob eine unternehmensweit einheitliche Plattform sinnvoll ist, oder ob im Gegenteil das populäre *bring your own device* unterstützt werden soll. Selbst im ersteren Fall ist die intra-Plattform-Fragmentierung problematisch. Es lässt sich zusammenfassen, dass die effektive und effiziente Cross-Plattform-Entwicklung eine der größten Herausforderungen der App-Entwicklung ist – wenn nicht die bedeutsamste.

Aufgrund der stark gestiegenen und weiter zunehmenden Bedeutung des Themas beschäftigen sich Unternehmen aus dem Münsterland und der Emscher-Lippe-Region intensiv mit Apps. Eine Reihe der dem IHK-Bezirk Nord Westfalen angehörenden Unternehmen produzieren Software für ihre Kunden; einige bieten die App-Entwicklung bereits an, für fast alle wird die Entwicklung von Apps zumindest eine Option sein. Darüber hinaus beheimatet der Bezirk eine große Anzahl Unternehmen verschiedenster Branchen, deren Prozesse ohnehin stark durch IT geprägt sind oder für die eine sorgfältige Integration von IT in ihre Aktivitäten bedeutsam ist. Viele dieser Unternehmen wirken im Förderkreis der Angewandten Informatik an der Westfälischen Wilhelms-Universität Münster [1] mit. Vor diesem Hintergrund wurde das Projekt *Business Apps* diskutiert und schließlich beschlossen.

Im Rahmen des Projekts soll der Status quo der App-Entwicklung in der Region festgehalten werden. Darüber hinaus soll eine allgemeine Einführung in das Thema erfolgen. Ein besonderer Schwerpunkt im Projekt soll der Cross-Plattform-Entwicklung zukommen. Ziel ist es, den Unternehmen konkrete Ansätze zu empfehlen oder zur Verfügung zu stellen, mit denen sie einfacher, kostengünstiger und mit größerem Erfolg Apps für verschiedene Endgeräte bzw. Plattformen entwickeln können. Darüber hinaus sollen weitere für die Unternehmen relevante Themen erörtert werden. Als Ergebnisse soll zum einen eine Broschüre entstehen: sie liegt hiermit vor.

Zum anderen soll den Unternehmen ein Werkzeug an die Hand gegeben werden, mit dem kontextabhängig eine optimale Lösung für die Cross-Plattform-Entwicklung gefunden werden kann. Zum Zeitpunkt des Projektstarts wurden hierfür sowohl ein detaillierter Test am Markt befindlicher Werkzeuge als auch die Entwicklung eines neuartigen Rahmenwerks diskutiert. Als Möglichkeit bot sich auch eine Kombination aus beidem an. Aufgrund der Anforderung der Unternehmen wurde im Projektverlauf entschieden, eine Entscheidungshilfe im Rahmen dieser Broschüre zu geben.

Die Ausrichtung des Projekts bedingte bewusst einen inhaltlichen Spagat. Zwar weisen alle beteiligten Unternehmen einen engen Bezug zu Informationstechnologie auf, aber Interviewpartner und auch Adressaten sind sowohl Manager als auch Anwendungsentwickler. Darüber hinaus ist es einerseits Ziel der Broschüre, einen umfassenden Überblick der Thematik zu bieten, andererseits aber auch dann ein hilfreiches Werk zu sein, wenn eine ausgiebige Lektüre unerwünscht ist. Aus diesem Grund kann die Broschüre als Gesamtwerk, in Auszügen, oder auch als Nachschlagewerk genutzt werden.

Um für einzelne Leser relevante Teil der Broschüre zu identifizieren, führen wir zu Beginn jedes Abschnitts kurz in die zu erwartenden Inhalte ein. Falls zutreffend, grenzen wir überdies ab, an wen sich das Kapitel wendet, ob Vorwissen aus anderen Kapiteln erforderlich ist, und ob das Kapitel übersprungen werden kann. Des Weiteren gehen wir davon aus, dass neben allgemein interessierten Lesern vor allem zwei weitere Gruppen von Mitarbeitern die Broschüre zu Rate ziehen werden: solche mit in erster Linie betriebswirtschaftlichen Aufgaben (Management, Marketing, Vertrieb, etc.) sowie primär technisch orientierte Mitarbeiter (etwa Anwendungsentwickler, Softwarearchitekten, usw.).

Textabschnitte, die sich vorwiegend an eine bestimmte der beiden Gruppen richten, sind mit einem Symbol gekennzeichnet. Der Manager neben diesem Abschnitt steht für Textpassagen, die wenig technisch sind und betriebswirtschaftlichen Fragestellungen nachgehen.

Abschnitte, die hingegen technische Aspekte beleuchten, sind mit zwei symbolisierten Zahnrädern gekennzeichnet. Trotz dieser Unterscheidung hat die Broschüre den Anspruch, allgemeinverständlich zu sein. Wir vermeiden daher Manager-Floskeln ebenso wie “Techie-Talk”.

Die Broschüre ist wie folgt aufgebaut. Das Kapitel im direkten Anschluss stellt die Grundlagen des Projekts vor. In diesem Rahmen wird insbesondere das Vorgehen beleuchtet. In Kapitel 3 wird der Hintergrund von Business Apps erläutert. Kapitel 4 stellt den Status quo der Entwicklung von Apps durch Unternehmen der Region dar.

In Kapitel 5 werden die Möglichkeiten zur Entwicklung von Apps für mehrere Plattformen vorgestellt. In diesem Rahmen wird insbesondere auch eine Auswahl bereits bestehender Lösungen vorgestellt. Einige Handlungsempfehlungen zur Entwicklung von Apps sind in Kapitel 6 zusammengestellt. Sie basieren auf positiven Erfahrungen der teilnehmenden Unternehmen mit von ihnen gewählten Vorgehens-

weisen. In Kapitel 7 wird eine Reihe verschiedener Aspekte der App-Entwicklung thematisiert, die zwar nicht unmittelbar zur Cross-Plattform-Entwicklung gehören, für Unternehmen der Region aber eine besondere Bedeutung haben. Kapitel 8 bildet den Schluss der Broschüre. Es folgen das Literaturverzeichnis sowie eine Liste relevanter Web-Links. Um die Broschüre sowohl als Projektbericht als auch als Nachschlagewerk nutzen zu können, kann die folgende zielgruppenspezifische Lesehilfe genutzt werden (Tabelle 1.1).

Ausrichtung	**Kapitel**	**Zielgruppe**
Einführung	1	
Grundlagen	2	am Projekthintergrund Interessierte
	3	an der Einführung in Apps Interessierte
Projektergebnis	4	an direkten Projektergebnissen Interessierte
Kernbeitrag	5	
Weitere Beiträge	6	an indirekten Projektergebnissen Interessierte
	7	an den jeweiligen Aspekten Interessierte
Schluss	8	

Tabelle 1.1: Zielgruppenspezifische Lesehilfe

Die Autoren der Broschüre danken an dieser Stelle den beteiligten Unternehmen für die freundliche Bereitschaft, ausführliche Interviews führen zu können. Ohne diese Gespräche, die dafür notwendige Offenheit und das Engagement der Unternehmen und ihrer Mitarbeiter wäre diese Broschüre nicht denkbar gewesen.

Kapitel 2

Grundlagen des Projekts

Inhaltsübersicht

Um eine allgemeine Einführung in die Broschüre zu bieten und um das zugrundeliegende Projekt zu beleuchten, werden in diesem Kapitel der Teilnehmerkreis vorgestellt und der Verlauf des Projektes skizziert. Insbesondere wird erläutert, wie in den Interviews und bei der Auswertung vorgegangen wurde. Schließlich finden sich Hinweise zur Verwertung der Ergebnisse.

2.1 Wichtige Begriffe

Der Begriff *App* ist innerhalb kürzester Zeit in den Sprachgebrauch eingegangen und wird weithin als eigenständige Anwendung auf einem mobilen Endgerät verstanden. Allerdings existieren zahlreiche weitere Begriffe, die nicht notwendigerweise geläufig sind, aber in dieser Broschüre Verwendung finden. Dies gilt insbesondere für Fachtermini, die in den Teilen der Broschüre zu eher technischen Themen verwendet werden.

Grundsätzlich wurde die Broschüre so geschrieben, dass entsprechende Begriffe nur in den für die jeweilige Zielgruppe relevanten Teilen verwenden werden. Technische Fachbegriffe finden sich also vorwiegend in den mit dem entsprechenden Symbol gekennzeichneten Teilen. Um *allen* Lesern die Lektüre der gesamten Broschüre zu ermöglichen und um eine gemeinsame Begriffsbasis zu schaffen, werden unvermeidbare Fachbegriffe an der Stelle ihrer ersten Verwendung erklärt.

2.2 Teilnehmer

Grundsätzlich werden alle Ergebnisse anonymisiert dargestellt. Im Interesse der teilnehmenden Unternehmen findet keine direkt Nennung statt. Daher soll ein Überblick über das Teilnehmerfeld bezüglich Unternehmensgröße und Branche gegeben werden.

Umfangreiche Interviews wurden mit elf Unternehmen geführt. Von diesen Unternehmen beschäftigen jeweils zwei Unternehmen 51-250 bzw. 251-500 Mitarbeiter. Ein Unternehmen fällt in die Kategorie mit 501-1000 Angestellten. Sechs Unternehmen beschäftigen mehr als 1 000 Mitarbeiter, vier von diesen sogar mehr als 5 000.

Die dominierenden Branchen der teilnehmenden Unternehmen sind Dienstleistung und Beratung, häufig mit IT-Bezug. Zwei Unternehmen sind dem Bereich Infrastruktur zuzuordnen und jeweils eins der Industrie bzw. den Finanzdienstleistungen. Von den elf Teilnehmern ist für sechs die Entwicklung von Software – und damit potentiell auch Apps – für Kunden eine wesentliche Aufgabe. Bei den restlichen Teilnehmern wird Software aller Regel nach für die eigene Nutzung entwickelt.

2.3 Projektablauf

Das Projekt gliedert sind in mehrere Phasen. Das Thema Business Apps wurde bereits im Frühjahr 2011 im Rahmen des Förderkreises der Angewandten Informatik diskutiert. Im Sommer 2011 folgte dann der Beschluss, das Projekt durchzuführen. Als akademischer Partner wurde hierzu der Lehrstuhl für Praktische Informatik des Instituts für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster ausgewählt.

Im Vorfeld des eigentlichen Projektstarts wurde das Vorgehen diskutiert. Um Erkenntnisse darüber zu gewinnen, ob und in welchem Umfang bereits Apps bei Unternehmens der Region entwickelt werden und welche Erfahrungen damit gemacht wurden, wurde eine Interviewphase eingeplant. Das weitere Vorgehen sollte sich nach dieser Phase richten.

Von August bis November 2011 wurden in der ersten Phase des Projekt die Interviews geführt. Dabei kam ein Interviewleitfaden zum Einsatz, der in Anhang B zu finden ist. Die Gesprächsführung war semi-strukturiert. Zwar wurden einige grundlegende Fragen gestellt, Ziel war es allerdings, die Gesprächspartner aus den Unternehmen möglichst frei berichten zu lassen. Auf diese Weise konnte – auch durch die Bereitschaft der Gesprächspartner, sich bis zu drei Stunden Zeit zu nehmen – ein sehr umfangreicher Einblick gewonnen werden. Die ersten Erkenntnisse konnten direkt für die später geführten Interviews dieser Phase genutzt werden, sodass noch gezielter zukünftige Herausforderungen bestimmt werden konnten.

In der zweiten Phase, die im Dezember 2011 begann, wurden die Ergebnisse ausgewertet. Dabei wurde schnell klar, dass der ursprüngliche Ansatz, im Rahmen des

Projekts ein Rahmenwerk für die Cross-Plattform-Entwicklung zu realisieren, gar nicht das Kerninteresse deckt. Vielmehr erscheint eine Markübersicht inklusive einer projektspezifischen Auswahlhilfe sinnvoll. Parallel zur Auswertung wurde mit der Anfertigung dieser Broschüre begonnen.

Darüber hinaus wurde am Lehrstuhl für Praktische Informatik ein Projektseminar gestartet, in dem ein Rahmenwerk für die modellgetriebene Entwicklung von Apps für mehrere Plattformen entwickelt wird. Die ersten Ergebnisse dieser proof-of-concept-Implementierung unter dem Namen MD^2 werden in Kapitel 5.6 dargestellt.

Durch die Interviews wurde bekannt, dass es neben dem Kernthema des Projekts zahlreiche weitere Fragestellungen zu Business Apps gibt, die zum Teil einige, zum Teil aber auch fast alle teilnehmenden Unternehmen beschäftigen. Diese Aspekte der App-Entwicklung und -Nutzung werden in Kapitel 7 vorgestellt. Wenn möglich, werden gleichzeitig auch Lösungsansätze skizziert und Hinweise auf die mögliche zukünftige Entwicklung gegeben. Einzelne Aspekte könnten auch für die detaillierte Untersuchung in Folgeprojekten geeignet sein.

2.4 Vorgehen in den Interviews

Dem Interviewleitfaden (siehe Anhang B) folgend wurde zunächst eine gemeinsame Gesprächsbasis geschaffen, indem beteiligte Personen vorgestellt, Begrifflichkeiten und das Vorgehen geklärt und Wünsche in Bezug auf das Interview aufgenommen wurden.

Anschließend wurde der Status-Quo der App-Entwicklung im Unternehmen erörtert. Falls bereits Apps im Unternehmen entwickelt wurden bzw. entwickelt werden, wurde dies umfangreich thematisiert. Für den Fall, dass bisher keine Entwicklung stattfindet, wurden die Gründe hierfür erfragt. Die Fragen zur App-Entwicklung waren sowohl betrieblicher als auch technischer Natur. Im Allgemeinen wurden sie nicht Punkt für Punkt durchgegangen; durch das Gespräch mit den Interviewpartner klärten sich viele Fragen, ohne dass sie explizit angesprochen werden mussten.

Im zweiten Frageblock wurden Anforderungen an die App-Entwicklung diskutiert. Auch hier ging es sowohl um organisatorische als auch um technische Aspekte. Der dritte Frageblock schließlich diente dem Ausblick. Ziel hierbei war es, ein Bild der zukünftigen Bedeutung von Apps für Unternehmen der Region zeichnen zu können.

Zwar ist der Bezug zur Cross-Plattform-Entwicklung nur in einigen Fragen direkt erkennbar, aber im Verlauf des Interviews wurde immer wieder in diese Richtung gehend nachgehakt. So entstand in jedem Interview ein Eindruck davon, welche Bedeutung der Cross-Plattform-Entwicklung beigemessen wird, welche Erfahrungen bereits damit gemacht wurden, welche Anstrengungen ein Unternehmen in Kauf zu nehmen bereit ist, und welche Wünsche diesbezüglich bestehen.

2.5 Vorgehen bei der Auswertung

Schon während der Interviewphase zeichneten sich zwei Tendenzen ab:

- Insbesondere aufgrund der zeitgleichen Weiterentwicklung bestehender Cross-Plattform-Rahmenwerke wurde klar, dass es bereits leistungsfähige Lösungen gibt. Wichtiger als die Entwicklung eines weiteren Frameworks erschien zunehmend die Bereitstellung einer Auswahlhilfe.
- Neben den Schwierigkeiten der Cross-Plattform-Entwicklung existiert eine Reihe weiterer Problemstellungen, die zumindest in einigen Unternehmen bzw. in einigen Kontexten einer effektiven und effizienten App-Entwicklung im Wege stehen.

Diese Beobachtung wurde durch die Auswertung bestätigt. Für diese wurden die Protokolle aller Interviews zusammengetragen und analysiert. Dabei wurden Informationen in vier Kategorien extrahiert:

- Qualitative und quantitative Daten zum Status quo,
- Herausforderungen und Wünsche bezüglich der Cross-Plattform-Entwicklung,
- Hinweise auf Handlungsempfehlungen (also etwa positive und negative Erfahrungen bei der App-Entwicklung), sowie
- Aspekte der App-Entwicklung, die weiterer Aufmerksamkeit bedürfen.

Der Status quo wurde so aufbereitet, dass er nicht nur ein allgemeines Bild zeichnet, sondern auch zur Selbsteinschätzung dienen kann. Die Herausforderungen und Wünsche bezüglich der Cross-Plattform-Entwicklung werden im Rahmen des Status quo skizziert. Handlungsempfehlungen fallen zu diesem Zeitpunkt noch recht rudimentär aus, da viele Unternehmen erst anfängliche Erfahrungen mit der App-Entwicklung gemacht haben.

2.6 Empfehlungen zur Verwendung der Ergebnisse

Die Verwendungsmöglichkeiten der in dieser Broschüre zusammengetragenen Ergebnisse richten sich nach Kapitel und Interesse. Kapitel 3 wurde unabhängig von den Interviews zusammengestellt und bietet allen interessierten Lesern eine allgemeine Einführung in das Thema *Business Apps*. In Kapitel 4 findet sich die Darstellung des Status quo. Das Wissen bezüglich der Cross-Plattform-Entwicklung kann durch Lektüre von Kapitel 5 vertieft werden. Insbesondere werden hier technische Fragestellungen erörtert.

Die in Kapitel 6 vorgestellten Handlungsempfehlungen haben einen rudimentären Charakter. Sie adressieren verschiedene Aspekte der App-Entwicklung. Wir hoffen, dass Sie auch als Anregung dienen und wir sie in der zweiten Auflage der Broschüre bereits ergänzen können.

Die ausgewählten Aspekte der App-Entwicklung (Kapitel 7) schließlich dienen sowohl der Sensibilisierung für entsprechende Themenbereiche als auch der Bereitstellung erster Ansätze. Da diese Aspekte nicht im Fokus der Projekts waren, können keine abschließenden Antworten bereitgestellt werden. Vielmehr werden Denkanstöße gegeben und, falls möglich, Lösungsansätze skizziert. Wir hoffen, mit diesem Kapitel Interesse wecken zu können und hoffen auf Rückmeldungen und Ideen.

Kapitel 3

Grundlagen von Business Apps

Inhaltsübersicht

Um die Grundlage für die folgenden Kapitel zu schaffen, wird zunächst der Kontext vorgestellt, in dem Business Apps ausgeführt und entwickelt werden. Danach werden die derzeit relevanten Plattformen für Apps vorgestellt.

3.1 Einführung

Im Folgenden werden unter *Business Apps* solche mobilen Applikationen bzw. Apps verstanden, die zur Unterstützung betrieblicher Aufgabenstellungen oder für geschäftliche Zwecke auf Smartphones oder Tablets zur Ausführung gebracht werden. Die Definition schließt sowohl native Apps ein, als auch solche Anwendungen, die auf Web-Technologien basieren. Optional kann als Charakteristikum die Distribution und Monetarisierung über Appstores hinzukommen, sowie eine Anbindung an betriebliche Backend-Systeme erfolgen.

In der überwiegenden Zahl der Fälle bestehen Business Apps aus Formular- und Ausgabefeldern. Damit ähneln sie Webseiten, auch wenn die Darstellung über Oberflächenelemente erfolgt, die zu einer spezifischen Plattform gehören. Ein Spiel, das gerenderte Grafiken verwendet, kann durchaus einen geschäftlichen Zweck erfüllen, etwa wenn es im Rahmen einer Marketing-Aktion zum Einsatz kommt. Als Business App wird es indes üblicherweise *nicht* verstanden.

Apps und Business Apps im Speziellen sind durch die Fähigkeiten der am Markt verfügbaren bzw. bereits bei Endkunden vorhandenen Endgeräte restringiert. Einschränkungen betreffen die Bildschirmauflösung und -größe, die Akkulaufzeit, sowie weitere Ressourcen wie z. B. den Arbeitsspeicher, den persistenten Speicher, die Prozessorgeschwindigkeit und die Datenanbindung. Sämtliche der Restriktionen sind

aufgrund der dynamischen Weiterentwicklungen der Endgeräte im Fluss, allerdings nicht mit denen klassischer Computersysteme gleichzusetzen. Das folgende Kapitel betrachtet die Besonderheiten mobiler Geräte und deren Implikationen auf Apps genauer.

3.2 Besonderheiten

Die neuartigen mobilen Geräte, d. h. Smartphones und Tablets, weisen einige Besonderheiten auf, die sie von althergebrachten Personal Computern (PCs) auf der einen und klassischen Mobiltelefonen auf der anderen Seite abgrenzen. Beispiele sind ihr Formfaktor (Größe und Seitenverhältnis) und ihre Benutzerschnittstelle. Da diese Geräteeigenheiten Auswirkungen auf Apps und die Entwicklung von Apps haben, stellt dieses Kapitel die wesentlichen Besonderheiten und ihre Implikationen für Business Apps im Folgenden kurz vor. Teilweise handelt es sich bei den Besonderheiten um Einschränkungen im Vergleich zu Personal Computers (PCs), teilweise eröffnen sie neue Möglichkeiten.

Nicht jede der folgenden Eigenschaften beeinflusst den Entwicklungsprozess in allen Fällen. Ebenso muss nicht jedes neuartige mobile Gerät zwangsläufig alle genannten Eigenschaften aufweisen. Einzelne Besonderheiten sind auch bei anderen Geräteklassen, z. B. bei Laptops oder klassischen Handys, anzutreffen. Die Kombination vieler Eigenschaften macht die besondere Stellung mobiler Geräte und – in Folge dessen – von Apps aus. Die Kombination ist der Grund, weshalb ein Projekt wie das vorliegende notwendig und sinnvoll ist. Klassische Entwicklungsprozesse sind nicht ohne Weiteres übertragbar und neue Ansätze zur Ergänzung ratsam.

Im Einzelnen weisen Smartphones und Tablets Besonderheiten in den folgenden Gebieten auf:

Ausgabemedium Der Bildschirm eines Smartphones und, in geringerem Maße, eines Tablets ist deutlich kleiner als Bildschirme von PCs. Typischerweise ist der Bildschirm eines Smartphones (Tablets) in der Diagonale 3 bis 5 Zoll (7 bis 10 Zoll) groß[1]. Demgegenüber weisen die meisten Notebooks zumindest 13 Zoll auf; Desktop-PCs werden üblicherweise mit noch deutlich größeren Bildschirmen betrieben. Der kleinere Platz bedingt zumindest auf Smartphones geringere Auflösungen, z. B. 1280×720 Pixel beim Samsung Galaxy S III oder 1136×640 beim iPhone 5. Gleichzeitig ist die Auflösung in Bezug auf die genutzte Fläche sehr hoch, was bezüglich der Darstellung von Inhalten herausfordernd sein kann, die für ähnliche Auflösungen auf deutlich größeren Bildschirmen entworfen wurden. Smartphones werden zudem oftmals im Hochformat verwendet, wodurch der in der Breite verfügbare Platz weiter sinkt.

[1] Metrische Angaben sind hinsichtlich der Bildschirmgröße nach wie vor unüblich. Die Angaben entsprechen bei Smartphones 7,62 cm bis 12,7 cm bzw. bei Tablets 17,78 cm bis 25,4 cm.

Der im Mittel stark begrenzte Platz erfordert eine andere Herangehensweise an das Design der Benutzeroberfläche von Anwendungen. Zusätzlich zum Bildschirm haben Mobilgeräte weitere Ausgabemedien wie Vibration und Ton, letzteres teilweise verbunden mit Unterstützung für Sprachausgabe seitens des Betriebssystems.

Eingabemedium Die meisten aktuellen Smartphones und Tablets verwenden einen Touchscreen als primäres Eingabemedium. Für die Texteingabe ist eine Bildschirmtastatur vorgesehen, die bei umfangreichen Eingaben mühsam zu bedienen ist. Die Touchbedienung eröffnet im Gegenzug aber auch neuartige Interaktionsmöglichkeiten wie intuitive Gesten. Ergänzt wird der Touchscreen oft durch eine begrenzte Zahl von Hardware-Buttons. Apps sollten die Besonderheiten berücksichtigen, indem sie langwierige Benutzereingaben minimieren und die neuen Optionen, wo sinnvoll, innovativ nutzen. Als alternatives Eingabemedium ist insbesondere die Spracheingabe über das eingebaute Mikrofon zu erwähnen, in besonderen Fällen kann auch die Kamera zur Eingabe genutzt werden. Gerätesensoren können als indirekte, nicht benutzergesteuerte Eingabemedien genutzt werden (siehe unten).

Datenübertragung Mobilgeräte kommunizieren immer über drahtlose Netzwerkverbindungen. Diese sind häufig instabil und weisen eine eingeschränkte Bandbreite auf. Hinzu kommen in vielen Fällen eine gegenüber Kabelverbindungen höhere Latenz sowie gegebenenfalls Jitter (eine Varianz in der Laufzeit von Paketen). Insbesondere Smartphones unterstützen unterschiedliche Verbindungsstandards wie WLAN oder Mobilfunk. Im mobilen Einsatz kann das Netzwerk im Betrieb wechseln. Apps müssen temporäre Verbindungsabbrüche, langsame Übertragungsraten und langfristige Nichtverfügbarkeit einkalkulieren.

Begrenzte Hardwareressourcen Aufgrund des i. d. R. kompakten Formfaktors mobiler Geräte sind deren Hardwareressourcen begrenzt. Trotz der enormen Leistungssteigerungen in den zurückliegenden Jahren erreichen mobile Geräte hinsichtlich Prozessorleistung, Grafikkarte und Speicherplatz nicht die Leistung typischer PCs.

Begrenzte Laufzeit Ein kritischer Punkt ist die Laufzeit mobiler Geräte angesichts begrenzter Akkukapazitäten. Apps müssen besondere Vorkehrungen treffen, um schonend mit den Ressourcen umzugehen. Nutzer sehen eine App, die den Akku übermäßig beansprucht, oftmals kritisch und deinstallieren sie gegebenenfalls.

Sensoren Neben den vom Nutzer gesteuerten Eingabemedien (siehe oben) stellen Mobilgeräte den Apps eine Vielzahl weiterer Hardwaresensoren zur Verfügung. Insbesondere der Global Positioning System (GPS)-Sensor zur Ermittlung der

Geolokation kann für Business Apps von Interesse sein. In Einzelfällen kann auch der Beschleunigungssensor oder der Kompass zum Einsatz kommen.

Betriebssystem Mobilgeräte betreiben spezialisierte Betriebssysteme wie Android oder iOS. Anwendungen können im Allgemeinen nicht vom Desktop oder zwischen unterschiedlichen Mobilbetriebssystemen portiert werden. Die Betriebssysteme bieten Apps über standardisierte Schnittstellen Zugriff auf Funktionen und Daten. Sie ermöglichen so eine tiefgehende Integration der Apps in das Betriebssystem. Zugleich sind Entwickler aber bezüglich der Schnittstelle sowie bei der Auswahl an Entwicklungswerkzeugen und Programmiersprachen auf die Vorgaben des Plattformherstellers beschränkt. Die Entwicklungsumgebung und die Freiheit des Entwicklers sind stärker als auf dem Desktop begrenzt: Anwendungen für Microsoft Windows können z. B. in vielen verschiedenen Programmiersprachen und mit unterschiedlichen Tools erstellt werden.

Heterogenität Mobile Plattformen sind auf Hardware- und Softwareebene sehr heterogen. Auch Bedienmuster unterscheiden sich von Plattform zu Plattform. Professionelle Apps müssen diese Heterogenität berücksichtigen, um in allen Plattformumgebungen optimal zu funktionieren.

Mobiler Einsatz Mobile Geräte werden ihrer eigentlichen Bestimmung folgend mobil eingesetzt. Eine solche mobile Verwendung ist gekennzeichnet durch häufige Kontextwechsel. Sowohl die vom Gerät wahrgenommene Umgebung – z. B. das Netzwerk oder der Standort – als auch die den Nutzer beeinflussende Umgebung – Lichtverhältnisse, andere Personen oder seine Haltung – ändern sich. Nutzer verwenden ihre mobile Geräte und deren Anwendungen oftmals beiläufig. Häufige Nutzungsunterbrechungen und Multitasking von Seiten des Nutzers sind die sichtbarsten Erscheinungsformen einer beiläufigen Verwendung. Apps müssen auf Kontextwechsel sowie Nutzungsunterbrechungen durch den Nutzer selbst oder durch andere Anwendungen, z. B. einen Telefonanruf auf dem Gerät, vorbereitet sein und einen raschen Wiedereinstieg ermöglichen.

3.3 Betriebssysteme für mobile Endgeräte

In der heterogenen Landschaft mobiler Endgeräte existieren diverse Betriebssysteme, welche im Unterschied zu älteren Mobiltelefonen häufig nicht durch Telekommunikationsunternehmen, sondern durch solche aus der IT-Branche entwickelt werden. Zu den Betriebssystemen zählen als wichtigste iOS von Apple, Android von Google, Windows Phone und Windows Mobile von Microsoft, BlackBerry OS von Research In Motion (RIM) und Symbian von Nokia. Sie werden jeweils in Verbindung mit dem Endgerät vertrieben. Aufgrund mangelnder Standardisierung sind die Betriebssysteme jeweils nur auf einer Klasse von ähnlichen Geräten ausführbar. So

werden beispielsweise im Fall von iOS und BlackBerry die Betriebssysteme exklusiv und spezifisch durch Apple bzw. RIM für deren Endgeräte entwickelt, oder im Fall von Android und Windows Phone durch den Hersteller des Endgeräts auf dieses angepasst.

3.3.1 Grundlagen und Marktüberblick

Die Markteinführung des iPhone bzw. des iOS im Jahr 2007 ist insofern als Zäsur zu werten, als dass die bis dato übliche Bedienung der Endgeräte mit Tastatur und Stift durch touchbasierte Lösungen ergänzt und mit der Zeit zu einem großen Teil abgelöst wurde. Die Absatzzahlen von Smartphones steigen seitdem stetig. Aus der neuen Form der Bedienung resultierte ein Bedarf nach einer grundlegenden Überarbeitung bzw. einer Neukonzeption der graphischen Benutzerschnittstellen von Smartphones. Dieser Trend führte als disruptive Innovation zu einer Neuverteilung auf dem Markt für Mobilgeräte. 2011 – dem letzten Jahr, für das vollständige Zahlen verfügbar sind – war im globalen Markt für Smartphones Android mit 47 % Marktanteil dominierend, gefolgt von iOS und Symbian mit jeweils 19 % sowie BlackBerry mit 11 % Marktanteil [2] (vgl. Abbildung 3.1). Die Zahlen für die ersten drei Quartale von 2012 zeigen eine stärkere Konzentration: Android vereinigt etwa zwei Drittel des Marktes und iOS ein Fünftel. Die Verteilung der im Einsatz befindlichen Geräte ist allerdings noch deutlich heterogener.

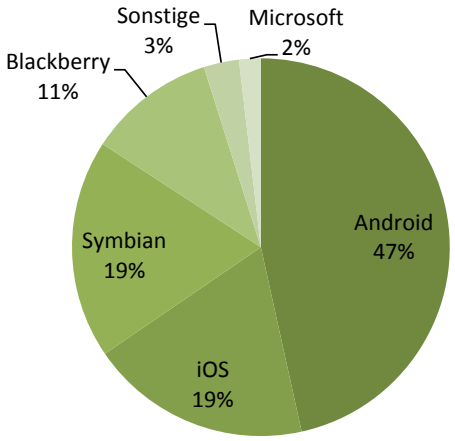

Abbildung 3.1: Marktanteile mobiler Plattformen auf Smartphones 2011 [2]

Eine Konsolidierung der heterogenen Betriebssystemlandschaft ist aufgrund der dynamischen Entwicklung aktuell nicht absehbar. So führen sowohl bei Smartphones als auch bei Tablets Innovationen in Soft- und Hardware jährlich zu neuen Gerätegenerationen. Da die Geräte zunehmend in Internet-basierte Ökosysteme eingebettet werden, die (Cloud-)Dienstleistungen für die Kommunikation, die Versorgung mit

Applikationen, die Mediennutzung, etc. bereitstellen, ist grundsätzlich aber von einer Konzentration auf größere und breit aufgestellte Internet-Unternehmen auszugehen. Solche sind in vorderer Reihe Google, Apple, Microsoft, Amazon und Facebook.

Das Ziel aus Sicht der Unternehmen ist die langfristige Bindung des Kunden an deren jeweiliges Ökosystem (*Walled Garden*) und auf dieser Basis die Etablierung eines dauerhaften Distributionskanals. Im Vergleich zum klassischen PC schränken einige Hersteller und Telekommunikationsdienstleister die freie Nutzung der Geräte zum Teil erheblich ein. Sie verhindern z. B. die Installation von Software außerhalb von Appstores oder die freie Nutzung von Datentarifen auch an angeschlossenen Computern (*Tethering*). Sie werden dem Kunden als Komfort- und Sicherheitsgewinn kommuniziert und können lediglich unter Bedingungen umgangen werden (*Jailbreak*, *Rooten*), die für Unternehmen aufgrund von Garantieverlust und Sicherheitsproblemen nicht akzeptabel sind.

Im Folgenden werden die aktuell wichtigsten mobilen Betriebssysteme vorgestellt und im Hinblick auf Fähigkeiten zur Ausführung und Entwicklung von Business Apps beleuchtet.

3.3.2 iOS

Apple Inc. gab die erste Version von iOS Mitte 2007 als Betriebssystem des ersten iPhones heraus. Seitdem kommt es im mobilen Umfeld auf den Geräten der iPhone-, iPod- und iPad-Serie zum Einsatz. Es handelt sich um proprietäre Software. iOS ist abgeleitet vom stationären Betriebssystem Mac OS X und somit Unix-basiert. Als separates Betriebssystem adressiert es die Eigenarten mobiler Endgeräte, z. B. bezüglich der Multi-Touch-Bedienung, der kompakten Bildschirmfläche und der begrenzten Prozessorleistung.

Charakteristisch für die Entwicklung von Apps für iOS sind die obligatorische Programmiersprache Objective-C und die Frameworks. Objective-C ist ein Derivat der Programmiersprache C und erweitert diese in Anlehnung an die Programmiersprache Smalltalk um sprachliche Mittel zur objektorientierten Programmierung. Es werden übliche Konzepte objektorientierter Programmierung unterstützt, wie z. B. Objekte mit Instanzvariablen und Methoden. Im Allgemeinen kann von einer recht leichten Erlernbarkeit der Sprache ausgegangen werden, wenn man sich an die an Smalltalk angelehnte Syntax gewöhnt hat.

Die Entwicklung nativer iOS-Apps baut auf dem Software Development Kit (SDK) auf, das Apple für iOS bereitstellt. Die Architektur von iOS besteht aus vier Schichten: Cocoa Touch, Media, Core Services und Core OS. Für jede Schicht stellt das SDK eine Reihe von Frameworks zur Verfügung, mit denen Apps auf bestimmte Funktionalität zugreifen können. Apps werden zumeist auf Frameworks der obersten Schicht, Cocoa Touch, zugreifen. Zur Implementierung der Benutzeroberfläche bietet *Cocoa Touch* das UIKit-Framework. Es wird nicht nur zur Gestaltung der Oberflä-

che genutzt, sondern auch für Ereignisbehandlung und Gerätefunktionen wie die Kamera. Weitere Frameworks unterstützen Entwickler beim Zugriff auf plattformspezifische Funktionen wie Adressbuch, E-Mail, SMS oder Karten.

Die *Media*-Schicht kümmert sich um Multimediaaspekte wie fortgeschrittene Grafikdarstellung, Video und Audio. In der *Core-Services*-Schicht sind Frameworks zusammengefasst, die grundlegende Systemdienste wie Datenhaltung oder Geolokation zur Verfügung stellen. Systemnahe Funktionen werden von der *Core-OS*-Schicht behandelt. Deren Frameworks z. B. für Bewegungssensorik oder Dateisystemzugriffe sind für Entwickler von direktem Interesse, wenn eine App spezielle Anforderungen hat. Ansonsten abstrahieren die oberen Schichten oftmals von der niedrigen Abstraktionsebene der Core-OS-Schicht.

Um iOS-Apps zu entwickeln, ist exklusiv die Entwicklungsumgebung Xcode zu verwenden. Xcode selbst steht kostenfrei zur Verfügung, läuft allerdings nur auf Mac OS X. Für die iOS-Programmierung sind deshalb zwingend Computer mit einem Mac-Betriebsystem notwendig. Apps können nur mit einer kostenpflichtigen Mitgliedschaft im iOS Developer Program [3] auf physischen Geräten getestet und ausgeführt werden. Dafür fallen jährliche Kosten in Höhe von 99$ an.

Neben üblichen Features einer Entwicklungsumgebung wie Editoren für Objective-C und Debuggern unterstützt Xcode die iOS-Programmierung im Speziellen mit einem iOS-Simulator, auf dem Apps während der Entwicklungsphase ausgeführt werden können. Außerdem vereinfacht Xcode die Erstellung grafischer Benutzeroberflächen, unter anderem mit sogenannten Storyboards. Storyboards beschreiben nicht nur den Aufbau der Oberfläche, sondern auch die Navigationspfade innerhalb einer App.

Entwickler können native iOS-Apps auf drei Wegen auf Endgeräte ausliefern:

- Der primäre Distributionsweg ist der Apple-eigene App Store (*App Store Distribution*). Die Veröffentlichung einer App im App Store setzt eine kostenpflichtige Registrierung im iOS Developer Program voraus. Vor der Veröffentlichung wird jede App durch Apple einem Review-Prozess unterworfen und auf Konformität zu den Richtlinien des App Store geprüft [4]. Die Veröffentlichung kann auf bestimmte nationale Märkte beschränkt werden.

- Zweitens kann eine App unternehmensweit ausgeliefert werden (*In-House/ Enterprise Distribution*). Dazu ist eine kostenpflichtige Registrierung im iOS Developer Enterprise Program notwendig. Für die Auslieferung der zu signierenden App existieren verschiedene Varianten. So ist es möglich, die App auf die Unternehmens-PCs und von diesen über iTunes auf die Endgeräte zu verteilen. Des Weiteren kann per Mobile Device Management (MDM) die initiale Installation und Aktualisierung automatisiert werden [5]. Kapitel 7.10 gibt einen Überblick über mögliche MDM-Lösungen. Schließlich erlaubt die Applikation *Apple Configurator* die Auslieferung von Apps auf Endgeräte, ist aber

auf 30 gleichzeitige Auslieferungsvorgänge begrenzt [6, Kap. 5], sodass Sie sich tendenziell für kleinere und mittlere Unternehmen eignet.

- Drittens kann eine App direkt an einen begrenzten Kreis von Endnutzern verteilt werden, welche sich diese mit iTunes auf dem Endgerät installieren (*Ad Hoc-Distribution*). Dieses Vorgehen setzt eine Registrierung im iOS Developer Program und eine Angabe der sogenannten *UDIDs* der Endgeräte im App Bundle voraus. Letzteres begrenzt den Kreis der Endgeräte, auf denen die App ausgeführt werden kann. Die Menge der UDIDs ist beschränkt und zielt auf einen Einsatz für Testzwecke ab.

3.3.3 Android

Das Betriebssystem *Android* wird als freie Software von der Open Handset Alliance [7] entwickelt, die im Jahr 2007 von Google gegründet wurde. Es basiert auf dem Linux-Kernel und umfasst Funktionalität, welche speziell auf mobile Endgeräte zugeschnitten ist. Aufgrund der offenen Lizensierung dient es als Softwareplattform für mobile Endgeräte einer Vielzahl verschiedener Hersteller von Smartphones und Tablets.

Apps für Android werden primär in der Programmiersprache Java unter Rückgriff auf bereitgestellte Frameworks entwickelt. In Java entwickelte Apps werden auf den Endgeräten nicht in einer standardkonformen Java Virtual Machine ausgeführt, sondern jeweils in einer eigenen Instanz der *Dalvik Virtual Machine*. Diese ist für Android in Hinblick auf einen schonenden Umgang mit Ressourcen konzipiert, und setzt im Zuge der Kompilierung einer App eine Konvertierung herkömmlichen Java-Bytecodes in das *Dalvik Executable Format* (dex) voraus. Damit entspricht sie nicht dem Standard für die Java Virtual Machine [LY99].

Android stellt nur Teile der Bibliotheken standardisierter Java-Plattformen wie Java SE oder Java ME zur Verfügung. Somit erfolgt die Softwareentwicklung zwar in Java, für bestimmte, insbesondere Hardware-nahe Funktionen müssen Entwickler aber anstelle der Standard-Java-Bibliotheken das Android-eigene Application Programming Interface (API) nutzen. Auch Android-spezifische Funktionen werden von der im sogenannten Application Framework zusammengefassten API bereitgestellt. Sie umfasst Funktionalität z. B. für die Entwicklung der Benutzerschnittstelle, den Zugriff auf das Adressbuch sowie den Kalender, und die Interaktion mit dem Benachrichtigungsdienst.

Für die Entwicklung von Java-basierten Apps für Android steht das Android SDK bereit. Jede App wird durch sogenannte *Application Components* aus verschiedenen Perspektiven implementiert. Diese sind kategorisiert als *Activity*, *Service*, *Content provider* und *Broadcast receiver* [8]. Eine *Activity* implementiert eine Teilkomponente einer App, die auf der graphischen Benutzerschnittstelle dargestellt wird. Da verschiedene Ansichten der Benutzerschnittstelle und verschiedene Funktionalitäten

jeweils in separaten Activities realisiert werden sollen, besteht eine App im Allgemeinen aus mehreren Activities. Im Beispiel einer Adressverwaltung könnte eine Activity für die listenartige Darstellung von Adressdaten zuständig sein, während eine weitere Activity es ermöglicht, einzelne Adressdaten zu editieren. Ein *Service* läuft als Hintergrundprozess ohne Anbindung an die Benutzerschnittstelle, und führt längere Operationen wie etwa die Wiedergabe von Musik oder den Transfer von Daten aus. Ein *Content provider* stellt App-übergreifend oder App-intern Daten bereit, welche Datei- oder Datenbank-basiert auf dem Endgeräte gespeichert oder über die Datenanbindung des Endgeräts geladen werden können. Ein *Broadcast receiver* dient der Ereignisbehandlung, indem er als *Listener* auf vordefinierte Ereignisse reagiert, wie z. B. einen niedrigen Ladezustand der Batterie.

Der Fokus liegt bei einfachen Apps somit auf der Entwicklung von Activities, welche im Quelltext als Unterklassen der Klasse *Activity* implementiert werden [9]. Aus der Menge der Activities ist eine als Haupt-Activity zu deklarieren. Sie wird beim Starten der App dargestellt. Weitere Activities können anschließend aus dieser aufgerufen werden, wobei jeweils immer nur eine einzelne Activity im Vordergrund dargestellt wird. Die restlichen Activities werden von Android im Hintergrund auf einem Stack verwaltet. Das Schließen einer Activity, z.B. durch den Zurück-Button, resultiert in der Wiederherstellung der vorherigen Activity von diesem sogenannten Back-Stack. Falls der Back-Stack keine Activity mehr enthält, kehrt der Benutzer zum Hauptbildschirm oder zur vorherigen App zurück.

Jede Activity durchläuft einen Lebenszyklus, der durch Android verwaltet wird und verschiedene Phasen zur Erstellung im Speicher, zum Start, zur Ausführung, zum Pausieren, zum Stoppen, und zur Entfernung aus dem Speicher umfasst. Korrespondierend zu den Phasen sind *Callback*-Methoden definiert, welche von Android bei Übergängen zwischen den Phasen aufgerufen werden, und zur Reaktion auf diese Ereignisse genutzt werden können. Diese sind beispielsweise die Methoden `onStart` und `onResume` [9].

Die Darstellung einer Activity auf der graphischen Benutzerschnittstelle erfolgt durch ein *Window*, das eine Hierarchie von *ViewGroups* und *Views* umfasst [10]. Das Framework bietet mit *Widgets* einige vordefinierte Darstellungs- und Bedienelemente für die Benutzerschnittstelle, z. B. Textfelder und Knöpfe. Diese können durch *Layouts* als Spezialisierung von *ViewGroup* angeordnet werden. Die Definition der Benutzerschnittstelle erfolgt standardmäßig separat vom Java-Quelltext durch Layout-Dateien in Form von XML-Dokumenten.

Diese und weitere Framework-Klassen sowie eine Vielzahl von Beispielapps sind in der offiziellen Entwicklerreferenz zu Android [11] detailliert erläutert, und können als Ausgangspunkt zur Realisierung eigener Apps dienen.

Prinzipiell ist es auch möglich, Teile einer App oder eine vollständige App in C bzw. C++ zu entwickeln. Ein solche Entwicklung in nativen Sprachen ist von nativen Apps im Allgemeinen zu unterscheiden, die zumeist nicht in nativen Sprachen entwi-

ckelt werden, sondern z. B. im Fall von Android in Java. Zur Einbindung nativen Codes kann man zum einen auf die üblichen Java-Mechanismen zur Einbindung nativsprachlicher Elemente zurückgreifen, indem man mit JNI [12] auf nativ-sprachliche Funktionen zugreift. Zum anderen kann man mit einer nativen Aktivität (*NativeActivity*) den gesamten Lebenszyklus einer Android-Aktivität nativ-sprachlich implementieren. Generell geht die Verwendung nativen C/C++-Codes mit einer erhöhten Komplexität bei der Entwicklung einher, ohne dass daraus im Allgemeinen Performancegewinne zu erwarten sind. Die Android-Entwicklerreferenz empfiehlt deshalb soweit möglich vollständig Java-basierte Lösungen [13]. Für die nativ-sprachliche Entwicklung sind einige der nativen Bibliotheken von Android als stabil deklariert, sodass diese auch in zukünftigen Versionen des Betriebssystems als verfügbar gelten. Zu diesen zählen korrespondierend zum Zweck der nativ-sprachlichen Entwicklung Bibliotheken für die Programmierung von Computergrafik und -audio mit OpenGL bzw. OpenSL ES.

Bevor sie ausgeliefert werden, müssen alle nativen Android-Apps, unabhängig davon, ob in Java oder C++ entwickelt, signiert werden. Das Zertifikat kann vom Entwickler selbst erstellt werden und muss nicht durch eine dritte vertrauensvolle Instanz bzw. Zertifizierungsstelle signiert werden. Für die Auslieferung nativer Android-Apps auf Endgeräte existieren verschiedene Möglichkeiten:

- Der primäre Distributionsweg führt über einen der Märkte für Android-Apps. Der größte Markt ist dabei Google Play [14] (ehemals Android Market), auf dem Apps und Medien angeboten werden. Im Unterschied zu iOS ist Android *offener* konzipiert, sodass weitere Märkte existieren, die aber den Charakter von Nischenmärkten aufweisen. Solche sind beispielsweise AndroidPIT [15] und SlideMe [16]. Aber auch größere Anbieter wie Amazon [17] betreiben eigene Android-Appstores.

- Die unternehmensweite Auslieferung geschieht über Lösungen für das Mobile Device Management (siehe Kapitel 7.10). Einen einfach gehaltenen Ansatz bietet der Dienstleister Push-Link, der auf die Distribution einzelner Apps fokussiert und für diese keine weitere Infrastruktur im Unternehmen erfordert.

- Des Weiteren ist es möglich, Apps manuell zu installieren, z. B. von einem PC aus über eine USB-Verbindung oder über den Webbrowser des Endgeräts.

3.3.4 Windows Phone

Windows Phone, Microsofts Betriebssystem für mobile Geräte, löste 2010 den Vorgänger Windows Mobile ab, dem kein Erfolg beschieden gewesen war. Die aktuelle Verbreitung von Windows Phone ist ebenfalls noch begrenzt – verschiedene Marktforschungsunternehmen prognostizieren der Plattform aber Potenzial für die Zu-

kunft, auch vor dem Hintergrund der engen Verbindung zwischen Windows Phone 8 und dem Desktop- und Tabletbetriebssystem Microsoft Windows 8.

Das SDK, mit dem Apps für Windows Phone entwickelt werden können, stellt Microsoft kostenfrei zur Verfügung. Als Entwicklungsumgebung kommt Visual Studio zum Einsatz, zum Beispiel in der kostenfreien Express-Version für Windows Phone. Apps für Windows Phone 8 können nur auf Windows 8 entwickelt werden. Das SDK enthält einen Emulator zum Testen von Apps. Zum Test auf tatsächlichen Geräten ist ein Dev-Center-Konto erforderlich, das jährlich 99$ kostet.

Die API von Windows-Phone besteht zum einen aus einer *.NET-API*, die allgemeine .NET-Funktionalität und für Mobilgeräte spezifische Funktionalität bereitstellt. Zum anderen gibt die *Windows Phone Runtime API* Zugriff auf spezifische Funktionen von Windows Phone. Sie überschneidet sich mit der API von Windows 8. Windows-Phone-Apps können dementsprechend in C# oder Visual Basic entwickelt werden. Die Benutzeroberfläche wird typischerweise im XAML-Format beschrieben, wie es auch für .NET-Anwendungen auf dem Desktop zum Einsatz kommt. Falls entsprechende Anforderungen es nötig machen, können Apps auch in nativem C++-Code entwickelt werden.

Native Apps für Windows Phone können über den Microsoft-eigenen Windows Phone Store vertrieben und installiert werden. Für Unternehmensgeräte besteht zudem die Möglichkeit, Apps nicht-öffentlich unternehmensweit zu verteilen [18]. In jedem Fall ist eine Dev-Center-Mitgliedschaft erforderlich.

3.3.5 BlackBerry OS

Das Betriebssystem *BlackBerry OS* wird von dem Unternehmen RIM in der aktuellen Version 7 exklusiv zur Ausführung auf dessen Endgeräten angeboten. Es ist in C++ entwickelt und bietet eine Java-basierte Laufzeitumgebung für native Apps sowie mit WebWorks eine HTML-basierte Ausführungsumgebung für Web-basierte Apps. Letztere bietet über eine JavaScript-basierte Programmierschnittstelle den Zugriff auf die native Funktionalität des Endgeräts. Diese Web-Option ähnelt somit Ansätzen im Bereich der Cross-Plattform-Entwicklung wie z.B. PhoneGap (siehe Kapitel 5).

BlackBerry steht seit 2011 unter zunehmenden Druck von Seiten der konkurrierenden mobilen Plattformen und verliert Marktanteile. Für das Anfang 2013 erscheinende BlackBerry 10 ist deshalb eine grundlegende Umstellung der technologischen Plattform angekündigt. Die Unterstützung von Java zur App-Entwicklung wird eingestellt, und für native Apps ein Wechsel zu C++ vorgenommen, sodass die Lauffähigkeit für ältere Java-basierte Apps nicht mehr gegeben ist. Neben der Fortführung von WebWorks sollen ergänzend Apps basierend auf Adobe Air sowie mit technischen Einschränkungen Android-Apps zur Ausführung gebracht werden können [19].

Im Folgenden liegt der Fokus auf der aktuellen Version 7 des BlackBerry OS und dessen Programmierschnittstellen für die Entwicklung Java-basierter Apps [20]. Java-basierte BlackBerry-Apps werden auf dem Endgerät in einer herstellereigenen Java Virtual Machine ausgeführt, welche konform zum Java ME-MIDP-Standard [21] ist. Die Programmierschnittstelle ist darüber hinaus konform zu diversen weiteren Java Specification Requests (JSR) [22] und bietet zudem BlackBerry-eigene APIs. Die Softwareentwicklung erfolgt entsprechend basierend auf einer Mischung von standardkonformen und proprietären Frameworks.

Jede Java-basierte BlackBerry-App wird im Quelltext als eine Unterklasse der Klasse `UIApplication` oder im Fall von Apps ohne graphische Benutzerschnittstelle als Unterklasse der Klasse `Application` implementiert. Beide Framework-Klassen sind Beispiele für BlackBerry-spezifische Java-Klassen. Jede derartige BlackBerry-App durchläuft einen Lebenszyklus, welcher durch den Benutzer über den *Homescreen*, das System oder eine andere App initiiert wird. Der Lebenszyklus wird durch den ApplicationManager verwaltet, mit dem über dessen gleichnamige Klasse interagiert wird.

Als Entwicklungsumgebung stellt RIM ein Eclipse-Plugin bereit, welches die bisherige BlackBerry Java Development Environment (BlackBerry JDE) abgelöst hat.

Abhängig von der genutzten Funktionalität des BlackBerry OS bzw. dessen Programmierschnittstellen ist eine Signierung nativer BlackBerry-Apps vor der Auslieferung obligatorisch. So sind in der Schnittstellenreferenz einige der Klassen und Methoden als sicherheitsrelevant gekennzeichnet, deren Inanspruchnahme eine Signierung voraussetzt. Die benötigten Schlüsselpaare werden durch RIM ausgestellt und dem Entwickler zugesandt, der mit diesen die App signiert [23].

Im Fall von BlackBerry OS existieren mehrere Möglichkeiten für die Auslieferung:

- RIM betreibt einen Appstore namens *BlackBerry App World*, in dem Apps öffentlich zugänglich gemacht werden können. Die Veröffentlichung setzt eine Registrierung voraus und führt durch einen Review-Prozess.

- Der BlackBerry Enterprise Server umfasst Funktionalität für das Mobile Device Management, mit dem Administratoren Apps unternehmensweit auf Endgeräte verteilen können.

- Über den BlackBerry Desktop Manager ist eine manuelle und lokale Installation per USB möglich. Alternativ kann per Kommandozeile mit dem Werkzeug JavaLoader [24] über USB mit dem Endgerät interagiert und Apps installiert werden. JavaLoader richtet sich dabei primär an Softwareentwickler.

- Der als *Over The Air* bezeichnete Distributionsweg sieht vor, dass der Benutzer mit dem Webbrowser des Endgeräts die App manuell von einem Webserver lädt und installiert.

- Mit dem Application Web Loader können Webseiten erstellt werden, deren manueller Aufruf durch den Benutzer z. B. auf einem Desktoprechner ein ActiveX-Control ausführt. Dieses kontaktiert lokal aus dem Webbrowser heraus per USB das Endgerät und installiert die App, die vorab zusammen mit der Webseite auf einem Webserver zu veröffentlichen ist.

- Eine weitere Möglichkeit ist die Distribution von Apps durch Netzbetreiber auf drei Wegen. Erstens kann der Netzbetreiber App-Icons für Apps auf den Endgeräten einblenden (*Virtual Preload*). Eine solche App wird im Hintergrund installiert, sobald der Benutzer sie erstmals aufruft. Zweitens kann der Netzbetreiber die Apps bei der Aktivierung des Endgeräts im Netz ausliefern, sodass der initiale Ladevorgang bei Aufruf der App entfällt (*Instant Load*). Drittens kann der Benutzer über eine Software namens *Application Center* auf Apps zugreifen, die ähnlich einem Appstore durch den Netzbetreiber veröffentlicht werden.

Kapitel 4

Status quo

Inhaltsübersicht

Zur Einordnung sind die Ergebnisse zum Status quo der App-Entwicklung in diesem Kapitel in vier Abschnitte unterteilt. Der nächste Abschnitt führt zunächst in die allgemeinen Ergebnisse ein. In der Folge werden dann quantitative (Fragebögen) und qualitative (Interviews) Ergebnisse vorgestellt und kurz diskutiert. Schließlich wird ein Zwischenfazit gezogen.

4.1 Allgemeines

Die quantitative Datenbasis ist mit 13 Fragebögen zu dünn, um eine umfangreiche statistische Auswertung vorzunehmen. Dennoch lassen sich erste Schlüsse aus den Umfragedaten ziehen. Die umfangreichen Interviews erlauben sogar, eine detaillierte Analyse vorzunehmen.

Auffällig – und im Sinne des zugrundeliegenden Projekts auch positiv – ist, dass fast alle Unternehmen sich bereits mit der App-Entwicklung auseinandersetzen und erste Erfahrungen gesammelt haben. Hierbei ist wiederum bemerkenswert, dass bei einer groben Betrachtung viele Unternehmen auf einem ähnlichen Stand sind: Apps werden erst seit Kurzem entwickelt und es liegen noch keine umfangreichen Erfahrungen vor. Im Detail zeigen sich dann aber zahlreiche Unterschiede, sowohl in der Herangehensweise, als auch in der Zielsetzung und Bedeutung. Vorab lässt sich festhalten, dass die Unternehmen der Region besonnen an das neue Thema herangehen und nicht Gefahr laufen, einem *Hype* zu erliegen. Auch die Initiierung des Projekts über das Institut für Angewandte Informatik (IAI) unterstreicht diese Herangehensweise. Während die Risiken der Nutzung minimiert werden sollen, stellen sich die

Unternehmen so auf, dass sie die neuen Möglichkeiten nicht nur nicht verschlafen, sondern idealerweise bereits vor Wettbewerbern zum Einsatz bringen.

4.2 Ergebnisse der Umfrage

Zur besseren Übersicht ist die Darstellung der Ergebnisse in eine Einführung sowie mehrere Abschnitte gegliedert, die Ergebnisse nach Kategorien zusammenstellen. Dabei folgt die Darstellung sequentiell dem verwendeten Fragebogen, der in Anhang A zu finden ist.

4.2.1 Allgemeines

Die Fragebögen wurden von uns mit zwei Zielen verwendet. Zum einen sollte eine Datenbasis geschaffen werden, die eine quantitative Auswertung erlaubt. Zum anderen sollten sie eine gezieltere Vorbereitung auf die Interviews ermöglichen.

Insgesamt liegen 13 ausgefüllte Fragebögen vor. Sieben davon wurden im Vorfeld der Interviews durch Unternehmen hereingereicht; die restlichen sechs wurden im Rahmen der Auswertung aus den Daten erstellt, die in den Interviews gewonnen wurden. Da zwei Unternehmen zwar Fragebögen einreichten, aber für Interviews nicht zur Verfügung standen, ist die Zahl der Unternehmen für die qualitative Auswertung (Kapitel 4.3) geringer.

Da nicht alle Fragen von allen Unternehmen zu beantworten waren, wird für jede Frage die Basis der Auswertung explizit angegeben. Dieser Umstand ist der Neuigkeit des Themas bzw. der organisatorischen Zuordnung der Ansprechpartner geschuldet: es lagen schlicht nicht für alle Fragen ausreichende Daten vor bzw. einzelne Entscheidungen waren zum Zeitpunkt der Bearbeitung noch nicht getroffen.

4.2.2 Struktur der Unternehmen und Art der Entwicklung

Die teilnehmenden Unternehmen fallen in unterschiedliche Branchen; deutliche Unterschiede gibt es bezüglich der beschäftigten Mitarbeiter. Drei Unternehmen fallen in die Kategorie 51–250 Beschäftigter. Jeweils zwei weitere Unternehmen haben 251–500 bzw. 501–1 000 Mitarbeiter. Die übrigen Unternehmen beschäftigen mehr als 1 000 Mitarbeiter; von diesen haben vier über 5 000, zwei sogar über 10 000 Angestellte.

Maßgebliche Unterschiede ergeben sich auch bezüglich der Zahl der Mitarbeiter in der Softwareentwicklung. Neun Unternehmen machten hierzu Angaben. Die Zahl der Entwickler liegt dabei zwischen 3 und 1 500. In einem extremen Fall wurde zudem 0 genannt, da IT-Projekte zwar intern koordiniert werden, reine Entwicklung aber stets als Auftrag nach außen vergeben wird. Der Anteil von Entwicklern an der Gesamtzahl der Beschäftigen bewegt sich zwischen 75% und geringen einstelligen

Werten; da keines der Unternehmen ein reines Softwarehaus ist, sondern es sich um IT-Dienstleister sowie IT-intensive Unternehmen anderer Branchen handelt, nimmt die Entwicklung jeweils nur eine Teil- oder Randfunktion ein.

Um ein besseres Verständnis der Art der Entwicklung zu erhalten, wurde der Auftraggeber von Entwicklungsaufträgen abgefragt. 46% der Unternehmen entwickeln Software im Auftrag ihrer Kunden, 77% für die interne Nutzung und 31%, um Software am Markt anzubieten. Ferner wurde der Nutzerkreis abgefragt. Für 77% der Unternehmen sind dies die eigenen Mitarbeiter, für 70% die Mitarbeiter des Kunden und für 38% Privatanwender bzw. Endkunden (also Kunden von Kunden). Bei beiden Fragen waren Mehrfachnennungen möglich.

10 der Unternehmen haben bereits Erfahrungen in der App-Entwicklung gemacht. Drei haben hierauf bisher verzichtet.

4.2.3 Verzicht auf App-Entwicklung

Für den Fall, dass auf die Entwicklung von Apps bisher verzichtet wurde, wurden die Gründe hierfür erfragt. Zur Auswahl standen *Kein Bedarf*, *Nutzen fraglich*, *Fehlende Erfahrung*, *Kein Budget* sowie ein Freitextfeld. Übereinstimmend wurde auf die fehlende Erfahrung verwiesen. In Kombination mit der Beobachtung von zehn Unternehmen, die Apps bereits entwickelt haben, lässt sich folgern, dass grundsätzlich Bedarf identifiziert wurde und auch Mittel zur Verfügung stehen. Deren Höhe ist Gegenstand der Auswertung in Kapitel 4.3.

Ein Unternehmen gab zusätzlich an, die Entwicklung von Apps erfolge durch Partner. Dies geht einher damit, dass von fast 500 Mitarbeitern weniger als fünf auf Entwicklerstellen entfallen. Zwei der drei Unternehmen, die bisher auf die App-Entwicklung verzichten, planen sich dieser in Zukunft zuzuwenden.

4.2.4 Bisherige Entwicklung von Apps

Die zehn Unternehmen, die bereits Apps entwickelt haben, wurden um nähere Angaben zu den Entwicklungsprojekten gebeten. 70% gaben an, dass sie bisher zwischen einer und vier Apps entwickelt hatten. 30% konnten keine genaue Angabe machen, allerdings liegt auch bei ihnen die Gesamtzahl bei wenigen Apps.

Neun Unternehmen konnten Angaben dazu machen, seit wann sie sich mit der App-Entwicklung beschäftigen. Sechs von ihnen haben 2009 oder 2010 erste Erfahrungen gemacht. Ein weiteres Unternehmen gab an, bereits seit “einigen“ Jahren Erfahrungen mit Apps zu sammeln, während das Thema für ein Unternehmen bereits seit 2002 von Bedeutung ist. Es ist davon auszugehen, dass die beiden Unternehmen mit mehrjähriger Erfahrung bereits Technologien wie Java ME erprobt haben, die zum Einsatz kamen, bevor überhaupt von Apps gesprochen wurde und Smartphones sich verbreitet hatten. Die übrigen Unternehmen sind der aktuellen Entwicklung gefolgt, haben sie aber offenbar nicht antizipiert.

80% der bisherigen Apps waren kompatibel zu Apple iOS, 70% zu Google Android, je 10% zu Windows Phone bzw. zu Symbian und 20% zu BlackBerry. Die App eines Unternehmens unterstützte zudem Samsung TV. Ein Teil der Apps wurde also für mehrere Plattformen entwickelt.

Die Einschätzung, welcher Anteil der Entwickler Erfahrungen mit der Implementierung von Apps hat, fiel offenbar schwer. Sie wurde nur von sechs Unternehmen beantwortet. In einem Fall lag sie bei 100% (allerdings bei nur drei Entwicklern absolut), in allen anderen Fällen im einstelligen Prozentbereich. Nur von einem noch kleineren Teil konnten bereits Angaben darüber gemacht werden, wie die App-Entwicklung organisatorisch aufgestellt ist. Ein Unternehmen hat Entwickler speziell für die App-Entwicklung abgestellt; bei drei Unternehmen werden Apps im Rahmen der üblichen Entwicklung erstellt. Keines der Unternehmen verfügt über eigene eigene Abteilung, die sich speziell mit Apps beschäftigt. Sechs Unternehmen konnten keine Angaben hierzu machen.

Ebenso wenig möglich war es den meisten Unternehmen, den Aufwand der App-Entwicklung im Verhältnis zur klassischen Softwareentwicklung zu bestimmen. Zwei der drei Antworten beschrieben den Aufwand als identisch. Ein Unternehmen schätzt den Entwicklungsaufwand als etwas erhöht ein.

4.2.5 Bedeutung einzelner Aspekte

Unabhängig von den Erfahrungen mit der Realisation von Apps wurden alle Unternehmen gebeten, einige Aspekte der Entwicklung zu bewerten. Dabei sollte auf einer fünfstufigen Skala von *sehr wichtig* (1) bis *gar nicht wichtig* (5) gewichtet werden. Die Ergebnisse sind in Abbildung 4.1 zusammengetragen.

Die Unterstützung mehrerer Plattformen wurde mit einem Durchschnittswert von 1,81 als besonders wichtig angesehen. Mit einem Durchschnittswert von 2,08 wurde ein natives Look & Feel von Apps ebenfalls als wichtig eingeschätzt. Auffällig hierbei ist, dass die Bewertung *etwas wichtig* überwiegt.

Differenzierter ist das Bild für die Einschätzung der Wichtigkeit des Kamerazugriffs sowie der Nutzung weiterer Gerätefunktionen. Die Durchschnittswerte von 2,8 geben Neutralität wieder, die konkreten Antworten zeigen aber Uneinigkeit. Nur ein (Kamera) bzw. zwei (Gerätespezifika) Unternehmen positionieren sich tatsächlich neutral. Je fünf schätzen den Zugriff als wichtig ein, für vier bzw. drei ist er weniger bedeutsam. Auffällig dabei ist aber, dass nur jeweils ein Unternehmen den Zugriff als *sehr wichtig* bewertet. Hingegen hält kein Unternehmen den Zugriff auf die Kamera für *gar nicht wichtig*.

4.2.6 Zukünftige App-Entwicklung

Schließlich wurden die Unternehmen gefragt, für welche Plattformen die Entwicklung von Apps zukünftig geplant ist. Mehrfachantworten waren hierbei wiederum

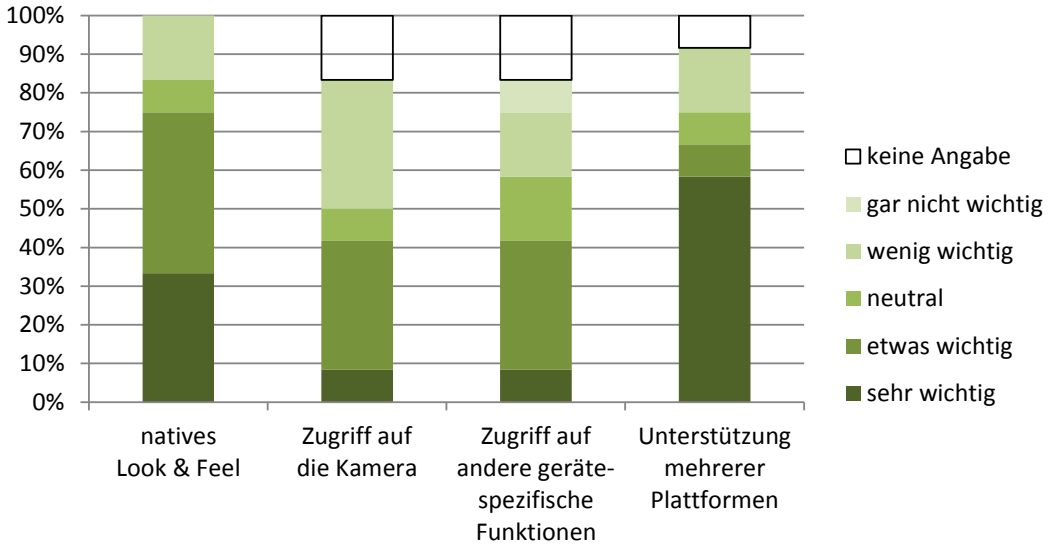

Abbildung 4.1: Auswertung der Fragen 14-17

möglich. Die Frage konnte von zwölf Unternehmen beantwortet werden. Bis auf ein Unternehmen planen alle die Unterstützung von iOS. Drei dieser Unternehmen wollen Android nicht unterstützen, die restlichen (73%) schon. Je vier Unternehmen (36%) werden Apps für Windows Phone bzw. für BlackBerry entwickeln. Symbian oder andere Plattformen wurden nicht genannt; ein Unternehmen betonte allerdings explizit, eine für alle Plattformen kompatible Web-App als Alternative zur nativen Unterstützung anzustreben.

4.3 Ergebnisse der Interviews

Während die Fragebögen einen groben Einblick in die Erfahrungen, Bedürfnisse und Einschätzungen der teilnehmenden Unternehmen geben, konnte mit den Interviews eine umfangreiche Basis für die Analyse geschaffen werden. Im Folgenden werden die Erkenntnisse aus den Interviews in aggregierter Form vorgestellt. Dabei werden an geeigneten Stellen aber auch Hinweise auf individuelle Einschätzungen, besondere Anforderungen und weitere bemerkenswerte Aspekte gegeben. Die Aufteilung der Unterkapitel folgt dabei grob dem Interviewleitfaden (siehe Anhang B).

4.3.1 Grundlagen

Jedes Interview begann mit einer kurzen Vorstellungsrunde, in der auch einige grundsätzliche Fragen geklärt wurden. Die Interviews wurden jeweils von zwei Mitgliedern des Projektteams geführt (Dr. Tim A. Majchrzak und Henning Heitkötter oder Ul-

rich Wolffgang); die Unternehmen waren durch einen oder zwei Ansprechpartner vertreten. Einige Interviews fanden in den Räumlichkeiten der Unternehmen, andere im Institut für Wirtschaftsinformatik[1] statt. Ein Interview wurde im Rahmen einer Telefonkonferenz durchgeführt.

Alle Interviewpartner stimmten der vorgeschlagenen Arbeitsdefinition für Business Apps zu. Dementsprechend werden *Anwendungen für mobile Endgeräte* behandelt, *die in erster Linie einem Geschäftszweck dienen.* In der Regel werden Apps mit schlichter Oberfläche betrachtet, wie die Unternehmensvertreter auch bestätigten. Zum Teil wurden wir noch auf das interne Verständnis hingewiesen, etwa einen Fokus auf Individualsoftware, bzw. auf ursprüngliche Begriffe wie *Enterprise Apps* als Bezeichnung für mobile Anwendungen, die eigene Prozesse unterstützen.

Auch wenn der Fokus des Projekts auf der Cross-Plattform-Entwicklung liegt, soll es möglichst gut auf die Bedürfnisse der beteiligten Firmen im Hinblick auf *alle* Aspekte der App-Entwicklung eingehen. Daher wurden die Interviewpartner zu Beginn des Gesprächs nach Wünschen und Erwartungen für den weiteren Verlauf gefragt. Nur drei betonten dabei besondere Anforderungen:

- Es würden Möglichkeiten zur Verbesserung von Web-Oberflächen gesucht. Native Entwicklung sei eher unbedeutend.

- Besonders auf Framework-Empfehlungen, Tipps zu Werkzeugen und allgemeine Empfehlungen zur Entwicklung werde Wert gelegt.

- „Techniklastige“ Ergebnisse seien besonders gewünscht.

Schließlich wurde der Unternehmenshintergrund erörtert. Die Unternehmen stammen vor allem aus der IT-Beratung sowie den IT- und den IT-Infrastruktur-Dienstleistungen. Zum Teil werden auch Rechenzentrumsleistungen, Softwareentwicklung und IT-Prüfung angeboten. Das Produktgeschäft ist bei den meisten Teilnehmern aber keine Kernfunktion oder nicht relevant.

Generell besteht bei den meisten Teilnehmern das Interesse an universitärer Kooperation, unter anderem auch mit der Absicht, neue Techniken mit Hilfe von Abschlussarbeiten zu erschließen.

Große Unterschiede ließen sich in der Größe der Entwicklungsabteilungen (siehe auch Kapitel 4.2.2) und dem Ausbildungsstand der Entwickler feststellen. Eine abschließende Beobachtung bezüglich der Hintergründe der Unternehmen ist die hohe Bedeutung von SAP-basierten Lösungen in einem maßgeblichen Teil der Unternehmen.

[1] Leonardo-Campus 3, 48149 Münster

4.3.2 Aktueller Status der App-Entwicklung

Da die Erhebung des Standes der App-Entwicklung einen wesentlichen Teil der Interviews ausmachte, sind die Ergebnisse im Folgenden thematisch untergliedert wiedergegeben.

4.3.2.1 Art und Kategorie bisheriger Apps

Alle interviewten Unternehmen haben bereits erste Apps entwickelt. Allerdings wurde maximal eine kleine Anzahl von Apps realisiert; in den meisten Fällen handelt es sich um *proof-of-concept* Implementierungen.

Aufgrund der Heterogenität der Angaben ist eine allgemeine Kategorisierung der bereits entwickelten Apps nicht möglich. Es lässt sich zwar wie im Fragebogen eine Einordnung darüber vornehmen, ob Apps für die interne Nutzung, für den Einsatz beim Kunden oder für den Gebrauch durch Kunden von Kunden entwickelt werden, doch lässt diese Einteilung im Detail keine weiteren Schlüsse zu. Zahlreiche der beschriebenen Apps wurden für die interne Nutzung konzipiert und dienen vor allem der Erprobung des Möglichen. Mitunter wurden daher auch Apps, die sich prinzipiell an Kunden richten würden, nur intern erprobt. Gegebenenfalls wird zukünftig eine Erweiterung dieser Einteilung sinnvoll sein; alternativ bietet sich eine Matrix an, die z. B. noch Organisationseinheiten oder Funktionen im Unternehmen beinhaltet. Ein Unternehmen unterschied etwa zwischen solchen Apps, die Vertriebsmitarbeiter im Außendienst unterstützen, und solchen, die rein intern genutzt werden. Erstere werden auch nur durch eigene Mitarbeiter genutzt, seien allerdings aufgrund des Kundenkontakts der Vertriebsmitarbeiter anders zu betrachten.

Im Allgemeinen haben alle beschriebenen Apps einen sehr geringen Funktionsumfang. Eine Kategorisierung nach Mächtigkeit, enthaltenen Funktionen, Codezeilen oder ähnlichen Kriterien ist daher zum jetzigen Zeitpunkt noch nicht sinnvoll. Anbieten könnte sich zukünftig eine Aufteilung nach Neuentwicklung und Funktionsübernahme. Die meisten beschriebenen Apps implementieren neue Ideen. Insbesondere die Apps, die tatsächlich schon bei Kunden eingesetzt werden, bilden hingegen bestehende Funktionen speziell für mobile Endgeräte ab. So wurden uns mehrere Apps beschrieben, die Funktionalität der von Kunden genutzten Webseiten auf Smartphones zugänglich machen. Die Idee bei diesen Apps ist, komfortabler zu bedienen zu sein als für mobile Endgeräte optimierte Versionen der Webseiten. Darüber hinaus wird die Wahrnehmung als App geschätzt, also etwa die Platzierung im obersten Menü des Smartphones im Gegensatz zum Aufruf einer Webseite über den internen Browser.

Aufgrund des prototypischen Charakters vieler beschriebener Apps erscheint uns eine beispielhafte Beschreibung an dieser Stelle am sinnvollsten. Ziel ist dabei weniger, direkt zur App-Entwicklung nutzbare Informationen bereitzustellen, als vielmehr Unternehmen die Möglichkeit zu geben, ihre eigene Arbeit mit der bereits

erkennbaren Entwicklung zu vergleichen. Besteht hieran kein Interesse, kann direkt zum Unterkapitel 4.3.2.2 gesprungen werden.

Viele der Apps können als mobile Webanwendung bezeichnet werden. Beispiele sind Apps, mit denen sich ein Intranet-Portal eines Unternehmens nutzen lässt. Technisch ähnliche Apps wurden auch als Angebot für Endkunden beschrieben, etwa um Zugriff auf Online-Shops oder Selbstverwaltungsfunktionen zu bieten. Im Wesentlichen werden Funktionen von Webseiten in Form einer App angeboten, die eine gegenüber einer Webseite beschleunigte und eventuell einfachere Bedienung bietet. Üblicherweise bilden solche Apps noch nicht den vollen Funktionsumfang der entsprechenden Webseite ab, sondern nur die als am wichtigsten empfundenen Möglichkeiten. Erprobt wurde auch die Nutzung von Hybrid-Apps (siehe Kapitel 5.2). In der Regel wurde darunter verstanden, dass eine App zwar Browser-Funktionalität nutzt, um eine – eventuell aufbereitete – Webseite anzuzeigen, aber auf dem Endgerät als App erscheint.

Mehrfach beschrieben wurden sehr einfache Apps. Sie bieten nur einen sehr schmalen Funktionsumfang, der aber in dieser Form noch nicht für mobile Endgeräte verfügbar war. Damit erleichtern sie eine bestimmte Arbeit oder stellen Informationen schneller zur Verfügung. Ein Beispiel ist ein Kalender für Müllabholungstermine.

Typisch sind auch Anwendungen, die kontextabhängig Hilfe anbieten, indem sie etwa zu Suchbegriffen passende Treffer aus fachspezifischen Katalogen anbieten. Diese Apps sind mitunter so konzipiert, dass sie zwar als proof-of-concept entstanden, aber ausbaufähig sind.

Im Allgemeinen konzentriert sich die Funktionalität vieler Apps auf die Informationsbereitstellung, die Eingabe bzw. Verwaltung von Informationen spielte bei vielen der beschriebenen Apps eine Randrolle. Dennoch kommen solche Apps in Einzelfällen bereits zum Einsatz. So wurde etwa eine Anwendung beschrieben, die als virtuelles Klemmbrett Mitarbeiter bei Wartungsaufgaben unterstützt. Sie bietet eine Checkliste mit Datenübermittlung an ein Backend, das auch Plausibilitätskontrollen vornimmt. Auch wurden einfachste Apps beschrieben, die mit „Ja“/„Nein“-Antwortdialogen durch Überprüfungsvorgänge führen.

Auch wenn bisher die Darstellungsfunktion überwiegt, nannten unsere Interviewpartner die Mobilisierung von Geschäftsprozessen als ein prinzipielles Ziel von Apps. Die Geschäftslogik verbleibt dabei allerdings auf den Servern; die Apps werden vor allem bzw. ausschließlich zur Darstellung und Eingabe verwendet.

Ebenfalls mehrfach beschrieben wurden Apps für Mitarbeiter in Vertriebs- und Außendienstfunktionen. Hierfür wurden insbesondere auch Tablets geprüft, die aufgrund ihres im Vergleich zu Smartphones deutlich größeren Bildschirms als vorteilhaft angesehen werden. Unterschiede gab es im implementierten Funktionsumfang. Während auch im Vertriebskontext einige Apps rein der Informationsbereitstellung dienen, können andere auch Kalkulationen durchführen oder zur Aufnahme von Kundendaten genutzt werden. In einigen Fällen integrieren sie Funktionen, die dem

Kunden im Gespräch eine Visualisierung bieten. Zum Teil ist auch eine komplexere Anbindung an das Backend vorgesehen, etwa für eine Bestellfunktion. Unterschiede lassen sich in der Zielsetzung solcher Apps erkennen. So wurde sowohl die reine Erprobung als auch die Suche nach einer „einheitlichen Smartphone-Umgebung für den Außendienst“ beschrieben, die bestehende Systeme (etwa mit Notebooks) ablösen könnte.

In wenigen Fällen handelt es sich bei Apps um Weiter- bzw. Neuentwicklungen bereits länger genutzter Applikationen. So wurde von einem Unternehmen beschrieben, bereits für Palm OS[2] eine Anwendung erstellt zu haben. Einige Unternehmen hatten überdies Erfahrungen mit J2ME-Anwendungen; deren Weiterentwicklung wurde allerdings nicht in Erwägung gezogen.

Schließlich wurden Apps entwickelt, die reinen Test- und Evaluationszwecken dienten und nicht notwendigerweise einen tatsächlich *nützlichen* Funktionsumfang hatten. Sie dienten beispielsweise der Begutachtung von Rahmenwerken wie *PhoneGap* [25] (siehe Kapitel 5.4.2) oder der *Sybase Unwired*-Plattform [26] oder erprobten Offline-Funktionen. Zu dieser Art von Apps zählen auch solche für reine Demonstrationszwecke, etwa Management-Cockpits, deren Anbindung an ein datenlieferndes Backend nur rudimentär oder sogar simuliert ist.

Neben den bereits entwickelten Apps wurden auch Projekte genannt, in denen möglicherweise Apps entstehen werden, die sich allerdings noch in einer frühen Phase befinden. Das noch keine Umsetzung erfolgt, könnte am Umfang liegen. So wurden kundenzentrierte Apps beschrieben, die auch positionierungsgestützte Kartenfunktionen nutzen und eine Datenerfassung über gerätespezifische Funktionen wie die Kamera bieten sollen. Auffällig ist, dass die Konzepte deutlich umfangreicher sind als bisher realisierte Apps. Der Nutzen insbesondere für Kunden wäre bei vollständiger Umsetzung der Konzepte gewaltig, allerdings ist der Aufwand der Realisierung in diesen Fällen noch kaum abzuschätzen.

Neben der Entwicklung eigener Apps wird in einigen Unternehmen auch der Einsatz von Apps von Drittanbietern genutzt, um Erfahrungen zu sammeln. So verwenden mehrere Unternehmen sogenannte Personal Information Manager (PIM)-Funktionalität. Projekte, in denen Mitarbeiter über Smartphones Termine, Kontaktdaten u. Ä. verwalten, wurden in diesen Fällen explizit zum Erfahrungsaufbau in der App-Nutzung hinzugezogen und könnten der Vorbereitung zukünftiger Eigenentwicklungen dienen.

Festhaltenswert ist abschließend, dass die Initiative für die ersten Projekte zur App-Entwicklungs von unterschiedlichen Quellen ausging. Anstelle eines zukünftigen gezielten Vorgehens (vgl. auch die folgenden Unterkapitel), entstanden prototypische Apps mitunter in eigener Initiative einzelner Mitarbeiter und bauten auf in der Freizeit erworbenen Kenntnissen auf. Einige Unternehmen förderten dieses infor-

[2] Palm OS kam als Betriebssystem auf Personal Digital Assistants (PDAs) zum Einsatz, die als Vorgänger der Smartphones bezeichnet werden können.

melle Vorgehen, indem sie Mitarbeiter für interne Workshops freistellten. Allerdings wurde bei einem Teil der Unternehmen auch ganz gezielt der App-Einsatz erprobt, indem Mitarbeiter mit der Einarbeitung beauftragt bzw. interne Entwicklungsprozesse angestoßen wurden. Dass es in einigen Unternehmen überhaupt informelle Entwicklungen gab, die letztlich gefördert wurden, ist bemerkenswert.

4.3.2.2 Unterstützte Plattformen

Bezüglich der unterstützten Plattformen kann unterschieden werden, zu welchen Plattformen bisher entwickelte Apps kompatibel sind und welche Bedeutung die Unternehmen einer breiten Kompatibilität generell zumessen.

Bei bereits bestehenden Apps ließen sich zwei Ansätze beobachten. Zum einen wurde exemplarisch für eine Plattform implementiert; dies war in der Regel iOS oder Android, in einigen Fällen auch BlackBerry. Zum anderen wurden iOS *und* Android unterstützt. In den Fällen, in denen mehr als eine Plattform unterstützt wird, handelt es sich bei der überwiegenden Zahl der beschriebenen Apps um diese beide Plattformen. Selbst wenn noch weitere Plattformen unterstützt werden, so sind diese beiden immer enthalten.

In vielen Fällen wurde allerdings darauf verzichtet, eine App für mehr als eine Plattform zu entwickeln. Die Gründe hierfür liegen entweder in der Beschränkung auf Geräte einer Plattform bei der internen Nutzung oder im prototypischen Charakter der implementierten Anwendung. Ersterer Fall gilt vor allem in Unternehmen, in denen das *Bring your own device (BYOD)*-Paradigma abgelehnt oder sogar verboten wird[3], bzw. bei denen einheitliche Geräte eingeführt wurden. Bei den teilnehmenden Unternehmen handelte es sich dabei um BlackBerrys, seltener um iPhones oder Android-basierte Geräte. Im prototypischen Fall waren die Apps nicht auf die dauerhafte Nutzung ausgelegt und die Erprobung der Funktionalität stand im Vordergrund.

Anzumerken ist, dass insbesondere die Unternehmen mit längerer Erfahrung in der App-Entwicklung die Gefahr von Image-Schäden sehen, wenn Apps für Kunden angeboten, deren „Lieblingsplattformen“ aber nicht unterstützt würden. Dies gelte vor allem dann, wenn nur eine Plattform bedient würde. So würden etwa Nutzer von iOS oder Android die Wahl der Plattform häufig emotional sehen und die Entscheidung für eine Plattform vor allem als Entscheidung *gegen* eine andere wahrnehmen. Dementsprechend könnten sie das Fehlen einer Version für „ihre“ Plattform als Affront werten. Dazu passend wurde auch angeführt, die Plattformwahl müsse der „Gewinnung bzw. Sicherstellung von Kundenvertrauen“ dienen.

Im Allgemeinen wurden iOS und Android als die unbedingt zu unterstützenden Plattformen genannt, sobald eine App auf mehr als einer Plattform zur Verfügung

[3] Gegen die Erlaubnis, mitgebrachte Geräte für betriebliche Funktionen zu nutzen, sprechen vor allem Sicherheitsbedenken. So fanden sich explizite Richtlinien gegen BYOD vor allem bei Finanzdienstleistern.

stehen sollte. Als Grund hierfür wurde der Marktanteil angeführt. Mitunter wurde sogar argumentiert, es sei sinnvoll weitere native Apps genau dann „nachzuschießen“, wenn sich steigende Marktanteile einer Plattform abzeichneten. Windows Mobile und Windows Phone werden derzeit von fast allen Teilnehmern als vernachlässigbar angesehen. Des Weiteren gehen die Teilnehmer durchgängig davon aus, dass Symbian in Zukunft keine Rolle mehr spielen wird.

Nur in sehr wenigen Fällen wurde neben der Softwareplattform auch auf die Hardware geachtet. Eine Unterscheidung etwa zwischen iPhone und iPad war bisher die Ausnahme, vermutlich aufgrund der geringen Zahl bisheriger Entwicklungsprojekte. Generell ist zu empfehlen, die Unterschiede zwischen Tablets und Smartphones während der Entwicklung zu berücksichtigen, wenn eine App für beide Geräteformen relevant ist.

Zum Teil wurde auf die Entwicklung von Apps für mehrere Plattformen verzichtet, obwohl Vorteile gesehen wurden. Schwierigkeiten wie ein hoher Aufwand, Einschränkungen hinsichtlich Performance und nativer Benutzerführung (etwa unterschiedliche Realisierung von *Gesten*) und mehrfacher Administrationsaufwand wurden als zu schwerwiegend angesehen.

Alle Unternehmen betonten ihr Interesse an Ansätzen zur Cross-Plattform-Entwicklung. Von einigen wird vor allem die Nutzung von Apps angestrebt, die auf Webtechnologien basieren. Dementsprechend wurden als Kompatibilitätsprobleme vor allem die fehlende Unterstützung aller Plattformen durch häufig genutzte Bibliotheken, etwa im JavaScript-Umfeld, beschrieben. Große Erwartungen werden in diesem Bezug HTML5 entgegengebracht.

Die Antwort auf die Frage, ob zukünftig stärker auf die Kompatibilität zu mehreren Plattformen geachtet werden soll, fiel zwar grundsätzlich eindeutig bejahend aus, im Detail zeigen sich aber deutliche Unterschiede in Meinungen und Vorgehensweisen:

1. Die pragmatische Sichtweise ist, auf die Kompatibilität je nach Kundenwunsch zu achten bzw. die Bereitstellung von Apps für einen möglichst großen Kundenkreis anzustreben, „welche Konsequenzen auch immer“ dies für die Entwicklung habe.

2. Die Priorisierung von Usability bzw. Softwareergonomie sieht vor, zwischen Web Apps und nativen Apps abzuwägen. Minimaler Entwicklungsaufwand und möglichst vollständige Plattformabdeckung sind nachrangige Ziele im Vergleich zu einer optimalen Nutzbarkeit der App.

3. Die Fokussierung auf Web-Technologie sieht vor, native Apps zu vernachlässigen und Plattformkompatibilität zu erreichen, indem der von jedem System bereitgestellte Browser genutzt wird. Der Verzicht auf eine native Benutzeroberfläche und native Funktionen wird dabei – zumindest für die nähere Zukunft

– in Kauf genommen. Generell wird eine Standardisierung in absehbarer Zeit erwartet. Allerdings ist die geräteübergreifende Nutzung einzelner gerätespezifischer Funktionen, etwa der Geolokation, dennoch wünschenswert.

4. Der duale Weg sieht vor, die *wichtigsten* Plattformen nativ zu unterstützen und alle weiteren Plattformen mit eingeschränkter Benutzerfreundlichkeit über Web-Apps abzudecken.

5. Die aufwandsscheue Sichtweise strebt zwar die Unterstützung *wichtiger* Plattformen an, sieht einen erhöhten Aufwand für die mehrfache Implementierung und vor allem für die parallele Unterstützung mehrerer Plattformen aber als äußerst kritisch.

6. Bei der abwartenden Strategie wird eine Plattform bevorzugt, parallel aber Wissen zu anderen Plattformen aufgebaut.

7. Für die interne Nutzung wird in einigen Fällen die Unterstützung nur einer Plattform als ausreichend angesehen, da bereits in entsprechende Infrastruktur investiert wurde. Diese Sicht deckt sich dann *nicht* mit der Strategie für die externe Nutzung.

4.3.2.3 Entscheidungsprozesse

Um die Entscheidungsprozesse bezüglich der Entwicklung von Apps und vor allem die Entscheidung für (oder gegen) die Unterstützung mehrerer Plattformen besser zu verstehen, wurde in den Interviews explizit darauf eingegangen, wie und von wem Entscheidungen zur mobilen Entwicklung getroffen werden. Zunächst stellt sich die grundlegende Frage, wer im Unternehmen die Entwicklung von Apps anstößt. Hierbei ließen sich mehrere Ansätze beobachten.

Die Entwicklung von Apps wird von Kunden in Auftrag gegeben. Dies ist die einfachste Art der Entscheidungsfindung (Auftrag annehmen oder ablehnen), allerdings nur für wenige der teilnehmenden Unternehmen relevant. Die Unterstützung mehrerer Plattformen wird durch das Budget und die Anforderungen des Kunden bestimmt.

In mehreren Fällen werden proaktiv Apps für erkannte Bedarfe entwickelt und dem Kundenstamm angeboten. Gegebenenfalls gibt es auch gemeinsame Vorträge oder Workshops, die zur Initiierung von Projekten führen können. Diese Variante ist vor allem bei solchen Unternehmen anzutreffen, die sehr enge Bindungen zu einem Kreis von Kunden haben oder deren Kunden sogar Anteilseigner sind.

Die Entscheidung für die Entwicklung intern genutzter Apps geht in der Regel von den Fachabteilungen aus, in einigen Fällen nachdem das Vorhaben mit der Entwicklungsabteilung diskutiert wurde. Dabei werden Funktionalitäten problemgetrieben definiert. Für extern angebotene Apps ist dieser Weg ebenfalls möglich, aber nicht

die Regel. Wird er gewählt, wurden im Vorfeld Kundenbedürfnisse festgestellt und ein Entwicklungsprojekt zu deren Befriedigung aufgesetzt.

Gerade für Software – und damit auch Apps – wurden von einigen Unternehmen „IT-getriebene“ Entscheidungsprozesse geschildert. Diese folgen allerdings mitunter unkonkreten Anforderungen aus Fachabteilungen oder von Außendienst- bzw. Vertriebsmitarbeitern. Ein aktives Management von Innovationen und zukünftigen Entwicklungen ist dabei die Ausnahme, wurde uns aber durch ein teilnehmendes Unternehmen geschildert. Die Maxime ist, sowohl proaktiv als auch reaktiv zu agieren und dementsprechend keinen wichtigen Trend zu verpassen, idealerweise sogar Innovationsführer zu sein.

Dass die Entwicklung von Apps direkt durch das Management oder sogar die Geschäftsführung angestoßen wird, wurde nur selten berichtet. Allerdings werden zum Teil generelle „Stoßrichtungen“ vorgegeben, zu denen dann auch eine Strategie für mobile Endgeräte gehören kann (vgl. auch Kapitel 4.3.4). Zum Teil ist die Geschäftsführung auch in einer prüfenden Rolle involviert. In diesem Fall werden ihr Vorschläge für Entwicklungsprojekte vorgelegt.

Im Detail ist darüber hinaus von Interesse, warum bestimmte Plattformen bei den bisherigen Apps unterstützt wurden, andere aber nicht. Auch hierbei ergibt sich kein einheitliches Bild:

- Viele der teilnehmenden Unternehmen haben diesbezüglich (noch) keine einheitliche Strategie. Die Wahl der Plattformen findet projektabhängig statt, bei Projekten für den Markt oder im Kundenkontakt auch in Abstimmung mit den potentiellen Nutzern.

- Für die interne Nutzung wurde in vielen Fällen eine verbindliche Entscheidung getroffen, die für mehrere Jahre Bestand hat. Wurden etwa BlackBerrys angeschafft, gibt dies auch die Plattform für sämtliche Entwicklungsprojekte intern genutzter Apps vor.

- Zum Teil wurde eine mittelfristige Strategie festgelegt. Wie bereits vorher festgestellt, umfasst diese typischerweise die Unterstützung von iOS und Android. Als dritte Alternative wird in der Regel BlackBerry genannt. Die Einschätzung dessen zukünftiger Bedeutung und daraus abgeleitet die Entscheidung über die Unterstützung fällt aber unterschiedlich aus.

Einige Unternehmensvertreter nutzen die Diskussion auch dazu, Kritik an den zur Verfügung stehenden Plattformen zu äußern. Dies ist besonders hilfreich, da es Gründe dafür dokumentiert, Entscheidungen nicht rein nach Marktanteilen zu treffen. Mehrfach kritisiert wurden die Restriktionen, denen iPhones und iPads unterliegen. Dabei bereitet weniger die Notwendigkeit von Entwicklerlizenzen Probleme als vielmehr der Eindruck, mit der Plattform nicht nach eigenem Belieben arbeiten

zu können. Trotz der Unterschiede in der Entwicklung und Verwaltung im Vergleich zu den meisten anderen Plattformen könne aber „kaum auf Apple verzichtet werden“. Ein Teilnehmer wies auf potentielle rechtliche Probleme hin. Kann eine Sicherheitsaktualisierung nur über den Appstore des Herstellers bereitgestellt werden und räumt dieser sich eine Prüfzeit ein, muss geklärt werden, wer für in der Zwischenzeit ausgenutzte Sicherheitslücken haftet.

Als problematisch angesehen wurde die Investitionssicherheit. Unternehmen stehen vor der Entscheidung, abzuwarten oder in Personal, Ausbildung und Hardware (etwa für die Ausstattung der Mitarbeiter mit Smartphones) zu investieren. In diesem Zusammenhang kam der Begriff „Plattformkrieg“ zur Sprache.

Kritisch gesehen wird häufig auch die Sicherheit. Letztlich wird hier nur BlackBerry positiv eingeschätzt. Sowohl iOS als auch Android werden für Szenarien, in denen kritische Daten auf Smartphones verarbeitet werden sollen, als problematisch angesehen. Allerdings wurden wir darauf hingewiesen, dass im betrieblichen Umfeld nicht nur die Risiken unterstellt werden dürfen, die auf dem Consumer-Markt angetroffen werden. Neben speziellen Smartphone-Viren, wie sie *alle* Nutzer betreffen, ist zum Beispiel auch Industriespionage zu bedenken. Es läge am Unternehmen, die Privatnutzung dienstlicher Geräte adäquat zu regeln.

Die Interviewpartner waren nicht mit allen technischen Aspekten der Plattformen zufrieden. So würden einige Unternehmen gerne auf die systeminternen Browser zurückgreifen und ihre Apps vor allem als Web-Apps oder hybride Apps gestalten. Zugriffsverhalten und Benutzerführung seien dann aber – unnötig – eingeschränkt.

Neben der Kritik wurden aber auch die neuen Möglichkeiten erörtert, die Innovationen im mobilen Umfeld eröffnen. So schilderten mehrere Teilnehmer, dass der Einsatz von Tablets als Ersatz für Notebooks diskutiert oder sogar geprüft werde. Diese werden dort als sinnvoll angesehen, wo mobil gearbeitet werden muss und Datenabruf und -visualisierung bedeutsam sind, die Eingabe sich aber vor allem auf Formular-basierte Datenerfassung konzentriert. Endgültige Entscheidungen sind diesbezüglich allerdings noch nicht gefallen.

Schließlich wollten wir wissen, ob die bisherigen Entwicklungsbestrebungen – insbesondere im Hinblick auf die Unterstützung der als relevant angesehenen Plattformen – *zufriedenstellend* war.

Zwar wollten sich noch nicht alle Unternehmen positionieren, doch ließ sich eine generelle Zufriedenheit ausmachen. Dies ist umso bemerkenswerter, als die Erfahrungen mit der App-Entwicklung häufig noch spärlich sind, die Festlegung auf zu unterstützende Plattformen schwer fällt und es Kritik an vorhandener Hardware und Software gibt. Gleichzeitig spiegelt die Zufriedenheit das Potential wieder, das in der Nutzung mobiler Endgeräte gesehen wird.

In diesem Zusammenhang sind Einschätzungen mitunter konträr zu denen, die bezüglich des Entscheidungsprozesses bei der Plattformauswahl zu hören waren. Wurde dort etwa iOS für seine Restriktionen kritisiert, beurteilen die Firmen, die für

iPhone entwickeln, die Ergonomie sehr positiv. Vor allem der Schulungsaufwand sei bei der Einführung von Smartphones gering, da Mitarbeiter intuitiv mit den Geräten arbeiten können. Dies verringere auch die Zahl eingehender Support-Anfragen.

Bei aller Euphorie wurde allerdings angemerkt, dass „Luft nach oben“ bestehe. Derzeitige Lösungen fühlten sich noch „nicht nach 100% an“. Mit anderen Worten: der aktuelle Stand wird nicht nur als verbesserungswürdig gesehen, sondern auch als verbesserbar. Details hierzu finden sich in Kapitel 4.3.4

4.3.2.4 Distribution

Applikationen für mobile Endgeräte *können* auf anderen Wegen auf die Geräte gelangen, als dies bei klassischer Software der Fall ist. Datenträger als Installationsquelle entfallen aller Regel nach. Endverbraucher beziehen Apps entweder direkt von Webseiten oder aus sogenannten Appstores. Diese bieten eine Übersicht verfügbarer Anwendungen und sind ähnlich aufgebaut wie Online-Shops. Für Unternehmen bietet sich darüber hinaus die Möglichkeit, Mechanismen der Softwareverteilung zu nutzen. Diese sind allerdings derzeit weit weniger ausgereift als im Desktop-Bereich. Die Mächtigkeit der Programme kann zudem je nach Plattform unterschiedlich ausfallen, insbesondere bei Unterstützung mehrerer Plattformen.

Grundsätzlich stehen die Unternehmen dem Konzept des Appstores positiv gegenüber. Er ermögliche eine einfache Distribution und sei für Kunden komfortabel zu nutzen. Auch die Möglichkeit, Apps zu bewerten wurde grundsätzlich positiv eingeschätzt. Dies erzeuge „Druck, hochwertige Apps zu entwickeln“. Zu bedenken ist allerdings auch das Risiko von – nicht notwendigerweise gerechtfertigter – negativer Kritik auf diesem Wege.

Für eine Reihe von Unternehmen stellt der Appstore sogar den einzig gewünschten Kanal zu Endkunden dar. Ein „einfacher Link z. B. auf eine mobile Web-Applikation“ wird „negativ für das Image“ gesehen. Ein gut funktionierender Appstore wird als wichtiges Kriterium für die Verbreitung einer Plattform betrachtet.

Auch für die interne Verteilung spielen Appstores eine Rolle. Wir wurden darauf hingewiesen, dass mehrere Anbieter entsprechende Lösungen im Programm haben. Diese ließen auch die Umsetzung spezifischer Anforderungen zu.

Kritisch anzumerken ist die Durchlaufzeit beim Deployment. Es sei z. B. für BlackBerry „recht umständlich“ und erfordere „bis zu zwei Tage“ Zeit – bei *jedem* Deployment. Das Ausbringen von Apps in Apples Store könne sich verzögern, wenn die dabei stattfindende Prüfung negativ ausfalle. Auch bei positiver Prüfung könne die Freischaltung neuer Apps bis zu eine Woche Zeit in Anspruch nehmen.

Anbindungen an bestehende Softwareverteilungssysteme gestalten sich bisher noch schwierig. Zwar streben alle Unternehmen, die intern Apps nutzen möchten, die automatisierte Erstinstallation (*Betankung*) derselben auf den Endgeräten an; der Fortschritt bei der Realisierung ist bisher aber sehr eingeschränkt. Als Stichworte wurden hier „Mobile Device Management Platform“ (siehe auch Kapitel 7.10) so-

wie „Enterprise Appstore“ genannt. Insbesondere für die Aktualisierung bestehender Apps wird eine wenig aufwendige Möglichkeit, bestehende Installationen zentral zu verwalten, für wichtig erachtet. Hier spielen wiederum auch Sicherheitsaspekte eine große Rolle, etwa die Möglichkeit, Richtlinien (*Policies*) durchzusetzen. Offenbar ist BlackBerry (bzw. der Hersteller RIM) hier positiv hervorzuheben, das durch die längere Verbreitung im kommerziellen Umfeld auf entsprechende Anforderungen gut vorbereitet ist. Apple hingegen müsse noch im Unternehmensbereich „ankommen“, ließe aber derartige Bestrebungen zunehmend erkennen.

Entwicklung und Test sind derzeit von MDM-Systemen noch entkoppelt. Ein simuliertes *Roll-Out* oder die *Betankung* bestimmter Schlüsselgruppen werden somit erst in Zukunft möglich werden.

Interessanterweise wurden Virtualisierungslösungen nicht angesprochen. Im Rahmen der BYOD-Diskussion merkte zwar ein Teilnehmer an, man würde für Telearbeit ein *Citrix*-System nutzen; diese Konzepte auf mobile Endgeräte zu übertragen kam allerdings nicht zur Sprache. Prinzipiell wäre es allerdings durchaus denkbar, im sicheren Kontext des Unternehmens laufende Apps auf privaten, potentiell unsicheren mobilen Endgeräten aufzurufen – ähnlich, wie dies auch bei Terminal-Server-Lösungen wie der oben erwähnten funktioniert. Ein solches Vorgehen kann Sicherheitsrisiken reduzieren und die Geräteverwaltung vereinfachen. Ob ein solcher Einsatz in Zukunft bedeutsam wird, bleibt abzuwarten.

Wenn Apps in Form von Web-Apps angeboten werden, wird angestrebt, diese nicht nur per Webseitenaufruf zugreifbar zu machen. Die Möglichkeit, dass Nutzer die App auf gewohntem Wege „installieren“ und als Symbol in der Übersicht der selbst aufgespielten Programme erhalten, wird als wichtig erachtet.

Anzumerken ist abschließend, dass die Unternehmen derzeit nicht planen, Premiumdienste zu betreiben oder Apps kostenpflichtig anzubieten. Zu bezahlende Optionen in Apps sind erst für die Zukunft und auch nur bei einigen Unternehmen geplant. Für die Mehrzahl werden Apps Instrumente bleiben, die den eigenen Mitarbeitern, der Prozesseffizienz oder der Kundenbindung dienen, selbst aber keinen Umsatz erzeugen.

4.3.2.5 Zweck von Apps

Eine Besonderheit der gewählten Forschungsmethodik ist, dass Ergebnisse kaum antizipierbar sind. Dies kann zwar dazu führen, dass für die Untersuchung vorgesehene Bereiche nur unzureichend bearbeitet werden können, ermöglicht wie in Kapitel 2.4 erläutert aber auch, sehr weitreichende Schlüsse zu ziehen. So planten wir etwa, den empfundenen Nutzen von Apps explizit in den Interviews zu thematisieren. Dies lag in der Regel allerdings aufgrund des Verlaufs nicht nahe und wurde daher übersprungen.

Auch wenn dies jeweils eine bewusste Entscheidung durch die Interviewenden war, sind zwei Schlüsse möglich. Erstens befinden sich Apps noch in der Erprobung. Die

Frage nach dem Nutzen ist daher zunächst zweitrangig. Zweitens haben Unternehmen bereits implizit Nutzen festgestellt oder aber sehen Nachteile, wenn sie sich nicht mit dem Thema auseinandersetzen.

Dies deckt sich auch mit den Aussagen, die wir in unmittelbarem Bezug zum Nutzen festhalten konnten. Als impliziter Nutzen wird etwa eine Steigerung des Kundennutzens erhofft, eine Verbesserung der Servicequalität erwartet oder eine Vereinfachung von Arbeitsprozessen angenommen. Dementsprechend wird Apps zugetraut, die Wirtschaftlichkeit betrieblicher Prozesse und des gesamten Unternehmens steigern zu können. Ganz im Gegenteil dazu wurde mitunter aber angemerkt, Wirtschaftlichkeit spiele „keine Rolle“, sobald Marketingargumente für die Entwicklung von Apps sprächen. So wird häufig eine vermutete Imagesteigerung angeführt. Dies scheint mitunter als extrinsischer Zwang wahrgenommen zu werden. Apps werden dann zwar als wichtig für das Bild des Unternehmens in der Öffentlichkeit bzw. bei den Kunden beurteilt, allerdings weniger aufgrund vermuteten Nutzens als aus Sorge. So könnte das eigene Unternehmen als „rückständig“ wahrgenommen werden, wenn Konkurrenten Apps in bestimmten Bereichen anbieten, die vom eigenen Unternehmen nicht abgedeckt werden. Diese Beobachtung geht eng einher mit der bereits beschriebenen *Notwendigkeit*, populäre Plattformen abzudecken, um bei Kunden keinen negativen Eindruck zu hinterlassen.

Dementsprechend wurde auch überspitzt formuliert, es müsse aufgepasst werden, aktuelle Entwicklungen nicht zu „verschlafen“. Wenn diese Gefahr gesehen wird, obwohl das Thema bereits auf der Agenda steht, drückt dies ein hohes Maß an Unsicherheit aus. Dennoch werden Apps als „unbedingt notwendig für zukünftige Umsätze“ gesehen.

Empfundener Nutzen ist in vielen Fällen noch recht unspezifisch. Die meisten Unternehmen sehen großes Potenzial für Apps in der Informationsbeschaffung. Tatsächliche positive Effekte lassen sich – wenn überhaupt – erst beobachten, nicht aber messen.

4.3.2.6 Entwicklung

Die Entwicklung von Apps für mehrere Zielplattformen wirft insbesondere technische Fragen auf. Aus diesem Grund wurde in den Interviews ein Blick auf die genutzten Entwicklungsmethoden und Besonderheiten bei der App-Entwicklung geworfen. Anders als von uns zu Beginn des Projekts erwartet, aber in Übereinstimmung mit den in den vorherigen Abschnitten zusammengetragenen Ergebnissen, ist die Erfahrung mit Entwicklungstechniken für Apps bisher recht gering.

Insbesondere ist festzustellen, dass keine angepassten Entwicklungsprozesse existieren. Im Wesentlichen werden dieselben Prinzipien befolgt, die für die Entwicklung von Desktop-Applikationen gelten. Da viele Apps nur der Informationsbereitstellung dienen, sind vor allem Methoden zur Oberflächengestaltung relevant. Im Unterschied zur Entwicklung von Desktop-Applikationen wurden viele der jeweils ersten Apps

nicht in Teams, sondern durch einzelne Entwickler realisiert, die höchstens einen informellen Austausch pflegten. In einem Fall wurde sogar von der Maxime „Ein Entwickler: eine App“ berichtet.

Sobald sich dies großflächig ändert und auch Entwicklungsteams Apps realisieren, könnten sich deutlichere Unterschiede ergeben. Als Beispiel wurde genannt, dass es insbesondere auf Android eine klarere Trennung zwischen Definition des Graphical User Interface (GUI) und der Ablauflogik gäbe. Dies stelle einen bedeutenden Unterschied zur klassischen Entwicklung für den Desktop, aber auch das Web, dar. Während die Entwicklung klassischer Anwendungen bisher oftmals schichtenübergreifend nach fachlichen Erwägungen geschah, könnte es zukünftig organisatorische Änderungen geben. Denkbar wäre etwa eine andere Rolleneinteilung für Entwickler.

Neben den kleinen Teamgrößen war auch festzustellen, dass die Anzahl generell mit App-Entwicklung betrauter Mitarbeiter sehr gering ist. Sie lag bei fast allen Teilnehmern unabhängig ihres Hintergrunds im deutlich einstelligen Bereich. Uneinigkeit herrschte bei der Frage, wie zusätzliche Kompetenz im Unternehmen aufgebaut werden kann. Berichtet wurde von

- der Ausschreibung von Abschlussarbeiten und Rekrutierung besonders begabter Studenten,
- der Kooperation mit Universitäten und Fachhochschulen,
- dem Einstellen entsprechend vorgebildeter Fachkräfte,
- der selbstständigen Einarbeitung durch die Mitarbeiter, sowie
- internen Workshops.

Externe Schulungen wurden nicht genannt. Es ist aber davon auszugehen, dass diese in Zukunft ebenfalls eine Rolle spielen werden. Wie Mitarbeiter für das Thema fit gemacht werden können, wurde auch als ein besondere Anliegen vieler Unternehmen aufgenommen. Erste Ideen und Anregungen hierzu finden sich in Kapitel 7.5.

Die Teilnehmer gaben in einigen Fällen auch an, dass es methodisch „keine Unterschiede zu klassischer Softwareentwicklung“ gäbe. Dementsprechend ließ sich auch keine Festlegung auf bestimmte Methoden feststellen; sowohl am Wasserfallmodell angelehnte Prozesse als auch agile Methoden kommen zum Einsatz. Generell werde erwartet, dass Ergebnisse sehr schnell erzielt werden. Dies könnte die Bedeutung agiler Praktiken erhöhen.

Technisch sei eine besondere Einarbeitung vonnöten, insbesondere wenn spezielle Rahmenwerke wie jQuery Mobile genutzt würden. Gleichzeitig wird mitunter auf den Einsatz etablierter Entwicklungswerkzeuge verzichtet. So berichtete ein Teilnehmer etwa, die Entwicklung erfolge „eher unstrukturiert und chaotisch“, selbst ein Code-Repository käme nicht zum Einsatz. Möglicherweise lässt sich auch dies

durch die häufig anzutreffende Beschränkung auf Informationsbereitstellung erklären. Wird auf etablierte, mächtige Backends zurückgegriffen, können Apps zur reinen Darstellung recht schnell realisiert – und bei Bedarf auch verändert – werden.

Ein Teilnehmer wies auf die Bedeutsamkeit hin, die Cross-Plattform-Entwicklung in einem Projekt zu vereinen. Nur so könne eine einheitliche Anwendung über alle Plattformen hinweg sichergestellt werden. Cross-Plattform-Ansätze wurden allerdings nur von wenigen Unternehmen in Betracht gezogen. Bisherige Ergebnisse bei der Suche nach geeigneten Rahmenwerken waren offenbar nicht besonders zufriedenstellend. So berichtet ein Unternehmen, dass Cross-Plattform-Ansätze, die auf Web-Technologien aufbauen, kein natives Look & Feel erreichen und damit kein „App-Gefühl“ beim Kunden erzeugten. Auch sei ungeklärt, in welchen Fällen die Richtlinien eine Distribution über Appstores ermöglichen. Für Cross-Plattform-Ansätze, die ein natives Look & Feel anstreben, fehle bisher das nötige Know-How in den zugrundeliegenden Technologien.

Anders als in klassischen Entwicklungsprojekten ist die Einbindung von Designern und Marketingmitarbeitern in vielen Projekten zur App-Entwicklung geboten. Allerdings kann argumentiert werden, dass diese Feststellung vor allem aus dem Umstand resultiert, dass viele Apps gerade für die Außendarstellung des Unternehmens erstellt werden. In vergleichbaren Projekten zur Entwicklung klassischer Anwendungen wären daher ebenfalls entsprechende Mitarbeiter eingebunden.

Schwierigkeiten wurden bezüglich des Testens von Apps berichtet. Die vorhandenen Simulatoren werden als unzureichend erachtet. Insbesondere erschweren Kontextwechsel (z. B. Online/Offline) und eine große Heterogenität der Hardwareplattformen (*device fragmentation*) das Testen. Dies gelte nicht nur zwischen Plattformen: selbstverständlich bereitet es Schwierigkeiten, die Leistungsfähigkeit einer App gleichermaßen etwa für iPhone und ein Android-basiertes Gerät sicherzustellen — ähnlich schwierig kann es aber auch sein, die Unterstützung der auf dem Markt befindlichen Android-Geräte und -Versionen sicherzustellen. Im Bezug auf Kontextwechsel werden nicht nur geeignete Testmethoden, sondern auch Entwicklungspraktiken benötigt. Ansonsten wird die sich ergebende Komplexität kaum beherrschbar sein.

Zwei teilnehmende Unternehmen nutzen in der Anwendungsentwicklung modellgetriebene Verfahren. Beide würden diese gerne auf die App-Entwicklung übertragen. Umfangreiche Erfahrungen hierzu lassen sich noch nicht beschreiben.

Neben der Erfahrung mit den eigenen Entwicklungsprozessen erhielten wir auch Hinweise auf externe Dienste und Dokumente. So wurde die Kontrolle von Apps vor der Bereitstellung im App Store als „positiv restriktiv“ empfunden. Eine „grobe Fehlerliste mit Verbesserungsvorschlägen“ könne tatsächlich helfen, Apps zu verbessern. Diese Einschätzung steht allerdings konträr zur Kritik an langen Deployment-Zeiten (siehe oben). Bei Android-Apps gäbe es größere Freiheitsgrade, aber dafür keine Qualitätskontrollen. Zur iOS-Dokumentation fanden sich positive Stimmen,

zu der für Android überwiegend auch.

Zu erwähnen ist schließlich, dass bisher nur sehr wenige Mitarbeiter in den Unternehmen Erfahrungen mit der App-Entwicklung gemacht haben. Dies deckt sich mit den Zahlen, die in der Umfrage ermittelt wurden (vgl. Kapitel 4.2.4). In den meisten Unternehmen sind es nur einzelne Mitarbeiter. Gleichzeitig wurde aber betont, dass die Einarbeitung (für Entwickler) „relativ einfach“ sei. Bezüglich der Motivation machte ein Unternehmen die Erfahrung, dass die Art des Themas zwar sehr motivierend wirke, der Alltag aber schnell wieder einkehre – etwa durch Probleme mit neuen Frameworks.

Generell ist zu erwarten, dass sich in Zukunft deutliche Änderungen in den Entwicklungsprozessen ergeben. Während diese bisher in vielen Fällen vom prototypischen Charakter der Apps geprägt waren, kann eine Professionalisierung erwartet werden. In diesem Rahmen zeichnet sich ab, dass sich Änderungen herauskristallisieren werden, wie sie z. B. auch beim Vergleich der Entwicklung von Desktop- und Server-Applikationen offenbar werden.

4.3.2.7 Best Practices und Probleme

Auch auf Basis des geringen Erfahrungsschatzes lassen sich bereits erste erfolgreiche Praktiken beschreiben. Auch haben Unternehmen Erfahrungen gemacht, welche Fehler in der App-Entwicklung begangen werden können. In Form von Handlungsempfehlungen können alle teilnehmenden Unternehmen von diesen Berichten profitieren.

Im Gegensatz zur Broschüre zu Softwaretests [MK10][4] liegt der Fokus hier allerdings nicht auf umfangreichen Handlungsempfehlungen, die in einem Rahmenwerk zur Anwendbarkeit präsentiert und mit kontextbezogenen Hinweisen versenden sind. Hierfür ist der Erkenntnisgewinn bezüglich der App-Entwicklung schlicht noch nicht weit genug. Möglicherweise wird es aber mittelfristig sinnvoll werden, die dann in den Unternehmen der Region verfolgten Ansätze auszuwerten und eine Broschüre mit best practices zu erstellen.

Die Handlungsempfehlungen, die sich bereits herauskristallisiert haben, sind in Kapitel 6 zusammengetragen, da sie für sich stehen und von der Auswertung des Status quo entkoppelt betrachtet und genutzt werden können.

Neben den aus positiven Erfahrungen ableitbaren Handlungsempfehlungen haben Unternehmen allerdings auch eine Reihe schlechter Erfahrungen gemacht und sind auf Probleme gestoßen. Zum Teil wurden diese im Rahmen der Auswertung bereits angesprochen. Im Folgenden werden darüber hinausgehende Auffälligkeiten aufgezeigt. Diese sollen dabei helfen, Fehler zu vermeiden und Probleme zu umgehen.

Die Teilnehmer waren sich einig, dass zum gegenwärtigen Zeitpunkt Kompromisse eingegangen werden müssen. Hierfür lassen sich einige Beispiele anführen:

[4] Die genannte Broschüre entstand im Rahmen eines vorangegangenen Projekts des IAI.

- Viele Projekte erscheinen hinreichend klein, sodass zur Entwicklung gerne studentische Praktikanten eingesetzt werden. Anders als von Entscheidungsträgern vermutet, sei der direkte produktive Einsatz der auf solche Weise entstandenen Apps häufig nicht möglich.

- Einerseits werden Apps als einfach zu entwickeln angesehen. Die Themen Sicherheit und Benutzerfreundlichkeit erfordern andererseits ein über das übliche Maß hinausgehendes Know-How.

- Grundsätzlich werde es geschätzt, dass über Appstores Rückmeldungen und Kommentare zu eigenen Apps erhalten werden können – eine Möglichkeit, auf diese zu antworten, fehle aber.

- Trotz des begrenzten Umfangs von Apps wurde von negativen Erfahrungen mit Outsourcing berichtet.

- Auf der einen Seite bieten Geräte wie Tablets neue Eingabemöglichkeiten, die deutlich zweckadäquater als Tastatur und Maus sein können. Auf der anderen Seite herrscht bei einigen Unternehmen Skepsis, ob Apps für betriebliche Prozesse geeignet sind, die umfangreiche Datenerfassung erfordern.

- Einerseits seien Teile der Technik bereits sehr ausgereift, andererseits sei Android und noch stärker Apple die Verankerung im Consumer-Markt anzumerken. Ein „Fokus auf Enterprise-Probleme und Lösungen“ wird vermisst.

- Während Smartphones und Tablets ein Hype-Thema sind, zeige sich im unternehmerischen Einsatz, dass sie nicht für jeden Anwendungsfall akzeptiert werden.

- Einerseits sei ein Teil der Mitarbeiter vom Einsatz mobiler Endgeräte begeistert. Andererseits sehe ein anderer Teil keine Veranlassung, diese zu nutzen.

- Zwar ist die Entwicklung mit einem ad-hoc Ansatz möglich, der flexibel auf neue Anforderungen reagieren kann. Aber dies führe zu einem unstrukturiertem Vorgehen und einer Vernachlässigung *sauberen* Software Engineerings.

- Smartphones und Tablets sind einfach in der Benutzung und können „herumgereicht werden“. Das Konzept der (personalisierten) Nutzung durch mehrere Anwender ist aller Regel nach aber ausgeschlossen.

Beklagt wurde darüber hinaus, dass Apps mitunter „symptomatisch“ entwickelt würden. Die Modifikation einer App, um dem Vorstand eine bestimmte Funktionalität zu demonstrieren, ist sicherlich in Einzelfällen sinnvoll und zulässig, führt aber in wiederholten Fällen zu Ineffizienz. Schlimmer noch, ein etwaiger „Wildwuchs“

kann fehlende Nachhaltigkeit nach sich ziehen und gegenläufig zu Projekten zur Komplexitätsreduktion (etwa durch die Konsolidierung von Servern) wirken.

Mehrere Unternehmen beklagten Sicherheitsprobleme. So sei die Vernachlässigung entsprechender Aspekte eher die Regel als die Ausnahme. Als Beispiele wurden die Benutzung verschlüsselter Verbindungen, die Sperrung bei Nichtaktivität sowie die sichere lokale Datenspeicherung genannt. Die Vernachlässigung dieser Richtlinien könnte fatale Folgen haben, insbesondere vor dem Hintergrund der zunehmenden Berichterstattung über „Smartphone-Viren“ und Ausspähung durch Apps in den Medien. Dem Eindruck, dass Smartphones zunehmend in den Fokus von Aktivitäten der Computerkriminalität rücken, ist sicherlich zuzustimmen.[5] Darüber hinaus sind Schäden für Unternehmen durch Datenverlust, -diebstahl und -manipulation zu befürchten, wenn nicht mit derselben Sorgfalt auf Sicherheitsaspekte eingegangen wird wie bei der Entwicklung klassischer Software. Natürlich ist hierbei zu beachten, dass viele Apps gar nicht oder nur in geringem Maße sensible Daten verarbeiten. Umso mehr ist aber darauf zu achten, dass bei der Verarbeitung ebensolcher entsprechende Standards und Bestimmungen eingehalten werden – und dass Apps keinen über ihren Einsatzzweck hinausgehenden Zugriff auf Unternehmensressourcen haben.

4.3.3 Anforderungen an die App-Entwicklung

Anforderungen spielen in der Softwareentwicklung eine wichtige Rolle: ist die Anforderungsermittlung nicht zuverlässig oder werden Anforderungen nicht korrekt umgesetzt, unterscheidet sich die Software funktional von dem Produkt, das der Auftraggeber erwartet hatte. Das sogenannte Requirements Engineering, ein Teilgebiet des Software Engineering, hat in den letzten Jahren hohe Aufmerksamkeit in Forschung und Praxis erhalten. Im Rahmen des Projekts wollten wir zunächst erheben, welche Anforderungen an Apps generell gestellt werden. In einem zweiten Schritt wurde dann diskutiert, inwiefern die Anforderungserhebung für Apps Unterschiede zu den bisher hierfür angewandten Prozessen aufweist oder sogar besonders behandelt wird.

Generell wurde darauf hingewiesen, dass Anforderungen kontextabhängig zu ermitteln seien. Eine Besonderheit ist dabei, dass für den Kontext auch gerätespezifische Funktionen zu beachten sind wie z. B. die Nutzung von Kamera oder Geolokation, etwa mittels GPS. Auch die Verbindung zu Geräten im unmittelbaren Umfeld (*Near Field Communication*) wurde als potentiell bedeutsam identifiziert. Mitunter wurde argumentiert, Apps sollten durch Nutzung solcher Funktionen und durch ein Eingehen auf die Bedürfnisse der Benutzer einen „Wow-Effekt“ erzeugen, also ihre Nutzer begeistern. Die Besonderheit in der Anforderungsermittlung ist daher nicht in der Nutzung bestimmter Funktionen zu sehen, sondern vielmehr in der er-

[5] Wenn Smartphones zukünftig breite Anwendung bei Zahlungsvorgängen finden [Bir12, Ros12], wird das Risiko weiter steigen.

wünschten Wirkung auf die Nutzer. Zu erwarten ist allerdings, dass insbesondere im innerbetrieblichen Kontext mit der Zeit eine nüchterne Betrachtungsweise Einzug halten wird.

Smartphones und Tablets gelten als besonders intuitiv zu bedienen. Dementsprechend wurde als Anforderung genannt, dass Apps für wenig technikaffine Mitarbeiter interessant sein müssen. Dabei sei zu bedenken, dass Nutzerfreundlichkeit abhängig von den jeweiligen Plattformen definiert würde. Dies folge nicht notwendigerweise ergonomischen Erkenntnissen, vielmehr gäbe es Gewöhnungseffekte.

Anforderungen wie *gutes Aussehen*, *hohe Performance*, *ausreichende Sicherheit* und *Einhalten des Datenschutzes* wurden häufig genannt. Bezüglich der Notwendigkeit eines *nativen Look & Feel* herrschte Uneinigkeit. Für viele Unternehmen ist dies eine der wichtigsten Anforderungen überhaupt, andere halten sie gerade im unternehmensinternen Einsatz von Apps für verzichtbar. Eine bisher wenig beachtete Anforderung, die in Zukunft Bedeutung gewinnen könnte, ist der *Energieverbrauch*. Die Implementierung einer App hat erheblichen Einfluss darauf, wie stark eine App bei längerer Laufzeit den Akku des Endgeräts belastet. Dies wiederum hat potentiell Einfluss auf die Nutzerakzeptanz. Als weiterer Punkt wird häufig gefordert, Apps in kurzer Zeit entwickeln zu können, auch wenn dies im Widerspruch zu einigen anderen Anforderungen steht.

Zum jetzigen Zeitpunkt werden Apps häufig aus einer begrenzten Perspektive betrachtet. Eine App soll *eine* Funktion erfüllen. Dies schlägt sich notwendigerweise auch in der Anforderungsanalyse nieder. Dies gilt aber nicht für alle Unternehmen; es ist zudem fragwürdig, ob es sich hierbei um eine App-spezifische Besonderheit handelt.

Eine besondere Bedeutung im Rahmen von Apps haben Datenanbindung und -verfügbarkeit. Man könnte für Apps unterstellen, dass sie entweder vollständig offline laufen *oder* permanent auf ein Backend zugreifen können. Benötigt werden allerdings in vielen Fällen Apps, die zwar Informationen nachladen, aber – etwa bei schlechter Netzabdeckung – auch offline arbeiten können. Je nach Einsatzzweck ist dabei eine automatische Synchronisation nötig, etwa um während einer Offlinephase eingegebene Daten zum Server zu schicken. Darüber hinaus sollten Anwendungen robust gegenüber einem Netzwechsel (etwa von WLAN auf UMTS) sein.

Die Besonderheiten in der Dateneingabe (*Touch*) im Vergleich zu einer Tastatur erfordern, Eingabefehler automatisch korrigieren zu können. Während derartige Hilfestellungen zum Teil auch für Anwendungen für PCs gefordert werden, kommt ihnen bei Apps eine erhöhte Bedeutung zu. Generell müssten Apps besonders robust sein.

Als für Endkunden wichtig wurde ein *Single-Sign-On*-Ansatz gesehen. Anders als auf dem PC wird hierunter aber üblicherweise verstanden, dass bei der Einrichtung einer App einmalig Zugangsdaten eingegeben werden, bei nachfolgender Verwendung der App aber nur noch die Sicherheitsfunktionen des Endgeräts genutzt werden.

Dies scheint auch der gängigen Erwartung von Nutzern zu entsprechen, ist aber mit Sicherheitsbedenken verbunden.

Zum jetzigen Zeitpunkt vermischt sich die Anforderungsermittlung noch mit der Technologieevaluation. Von HTML5 beispielsweise erwarten die Unternehmen in der Regel deutliche Verbesserungen für die Programmierung von Web-Apps. Daher werden im Hinblick auf die in HTML5 enthaltenen Features Anforderungen formuliert, die aber zum jetzigen Zeitpunkt nicht vollständig zu befriedigen sind.

Hingewiesen wurde auf konfliktäre bzw. unerfüllbare Anforderungen, wenn Anforderungskataloge und Richtlinien für klassische Anwendungen und Systeme unreflektiert übernommen werden. Zum Beispiel wird für gewöhnlich gefordert, auf PCs Virenscanner und andere Schutzsoftware zu betreiben. Diese ist für mobile Endgeräte nicht in vergleichbarer Form verfügbar. Inwiefern diese Tatsache die Anforderungen an Apps beeinflusst – z. B. könnte unterstellt werden, dass diese eigene Maßnahmen treffen müssen – ist derzeit unklar. Ein Datenleck in einer App, die Kundendaten verarbeitet, wurde allerdings als „Super-GAU“ angegeben.

Über die Sicherheitsanforderungen hinaus, die denen an PC-Anwendungen ähnlich sind, muss die mobile Nutzung besonders beachtet werden. So besteht bei einigen Anwendungen die Gefahr des Missbrauchs grundsätzlich legitimer Daten. Eine App etwa, mit der Nutzern standortbezogene Informationen abrufen können, würde gemeinhin als unkritisch angesehen werden. Sollte allerdings die Übermittlung der Standorte an ein Backend unverschlüsselt erfolgen, wäre es möglich, aus diesen Daten ein Bewegungsprofil zu erstellen. Folglich muss bei der App-Entwicklung die Vermeidung kontextabhängiger Sicherheitsproblematiken eine explizite Anforderung werden.

Im Allgemeinen gaben alle Unternehmen an, dass (noch) keine Anforderungskataloge speziell für Apps existieren. Hier gab es nur eine Ausnahme in Form eines einfachen Kataloges, der die Kommunikation zwischen Fachabteilung und IT verbessere. Trotz der oben aufgeführten Besonderheiten sei die Entwicklung in der Regel projektbezogen. Unternehmensweite Vorgaben gelten im Allgemeinen auch für Apps.

Aufbauend auf der Ermittlung besonderer Anforderungen wurde erörtert, inwiefern sich das Konzept der Apps auf die Anforderungsanalyse als solche auswirke. Im Allgemeinen ließen sich nur wenige Besonderheiten feststellen; der Mehrzahl der Unternehmen zufolge sei es hierzu noch „zu früh“.

Wie oben erwähnt wurde zum Teil die Anforderungsanalyse mit einer Technologieevaluation verbunden. Ausgehend von in Erwägung gezogenen Techniken wurde in diesen Fällen geprüft, ob überhaupt korrespondierende Anforderungen vorstellbar sind.

Aufgrund der Wünsche hinsichtlich der Bedienbarkeit von Apps wurde vorgeschlagen, Plattformspezifika bei der Anforderungsermittlung mit aufzunehmen. Folglich wäre es nötig, dass die damit betrauten Mitarbeiter die Möglichkeiten, Einschränkungen und Unterschiede der Plattformen kennen und bei Gesprächen mit Auf-

traggebern und Fachabteilungen auf sich ergebende Besonderheiten oder potentielle Inkonsistenzen in der Bedienung hinweisen.

4.3.4 Pläne für die Zukunft und Resümee der Unternehmen

Wesentlich für ein Projekt, in dem der Status quo der Nutzung einer Technologie erarbeitet wird, ist der Zukunftsausblick. Aus diesem Grund wurden in der Interviews vier Themenbereiche behandelt, die Hinweise auf die Zukunft der App-Nutzung geben können. Die Ergebnisse werden im Folgenden dargestellt.

4.3.4.1 Strategische Bedeutung

Für alle beteiligten Unternehmen ist zumindest die Nutzung, in vielen Fällen aber auch die Entwicklung von IT strategisch bedeutend. Deshalb wurde diskutiert, inwiefern auch Apps eine derart hohe Bedeutung zukommt.

Insgesamt ergab sich ein gemischtes Bild. Einige Unternehmen konnten sich spontan positionieren, andere waren abwartender. Bei der klaren Positionierung fand sich sowohl die Bestätigung, dass die Entwicklung von Apps als strategische Aufgabe gesehen wird, als auch die Feststellung, dass sie rein operative Werkzeuge seien. Darüber hinaus wiesen einige Teilnehmer darauf hin, dass für sie vor allem wichtig sei, welche Bedeutung ihre Kunden Apps zumäßen. Hier würde mitunter bereits jetzt eine strategische Ausrichtung wahrgenommen.

Die Unternehmen, bei denen Apps nur eine operative Bedeutung haben, nutzen sie z. B. zur Vertriebsunterstützung und Kundenansprache. Den bereits weiter oben geschilderten Überlegungen folgend, z. B. Notebooks durch Tablets ersetzen zu können, ist eine zukünftige Änderung der Bedeutsamkeit denkbar. Daher wird ein Wandel hin zu einer strategischen Bedeutung zumindest als Möglichkeit gesehen. Zum Teil wird noch nach geeigneten Anwendungsfällen gesucht, bei denen Tablets Notebooks ersetzen könnten.

Die abwartenden Unternehmen verwiesen vorwiegend auf die bisher geringen Erfahrungen. So solle etwa noch die Evaluation von Prototypen abgewartet werden. Insgesamt werde aber von einer stark steigenden Bedeutung ausgegangen. Der Markt treibe diese Entwicklung auch voran.

Unternehmen, die Apps eine strategische Bedeutung zumessen, verbinden damit häufig eine Reihe von Erwartungen an die Zukunft. Beispiele sind

- die Extrapolation, dass die Nutzung von PCs und Notebooks rückläufig sein wird,

- die Annahme, dass ein Großteil der Bevölkerung zukünftig über einen praktisch permanenten Internet-Zugriff verfügen wird,

- die Vermutung, dass Web-basierte Lösungen eine extrem hohe Zugänglichkeit besitzen werden, sowie

- die Prognose, dass die Kommunikation mit Kunden verstärkt bidirektional verlaufen wird und Kunden damit Einfluss auf die Produktgestaltung nehmen.

Maßnahmen wie die Einrichtung von Kompetenzzentren und der Aufbau eines Mitarbeiterstamms mit Kenntnissen in der App-Entwicklung unterstreichen die hohe Bedeutung. Zu beobachten ist dabei viel Optimismus. So wird Apps zugetraut, Innovationen zu beschleunigen und zu Prozessverbesserungen beitragen zu können bzw. diese zu begünstigen. Insbesondere diese Betrachtung als *Katalysator* illustriert auch die Bedeutung, die Mitarbeitern hierbei beigemessen wird. Sie sollen, inspiriert durch sich ergebende neue Möglichkeiten, Verbesserungsvorschläge einbringen.

Abschließend ist noch zu erwähnen, dass sich nicht alle Unternehmen eindeutig festgelegt haben. So wurde beispielsweise berichtet, Apps für Kunden würden strategisch entwickelt, intern sei die Nutzung aber rein operativ.

4.3.4.2 Zusätzliche Investitionen

Software zu entwickeln erfordert ein entsprechendes Budget; der Aufbau von benötigtem Wissen zieht zumindest kalkulatorische Kosten nach sich. Aus diesem Grund haben wir uns erkundigt, ob Unternehmen bereit sind, in die Entwicklung von Apps bzw. in die Voraussetzungen hierfür zu investieren.

Zwei unterschiedliche Herangehensweisen waren zu beobachten. Ein Teil der Firmen plant eine Umverteilung; ein etwa gleich großer Teil wird zusätzlich investieren.

Der Umfang zukünftiger Investitionen war bei allen Teilnehmern allerdings noch unklar. Die Zahl bereits ausgeschriebener Stellen ist in diesem Rahmen äußerst gering. Interessanterweise wird dabei von den meisten Unternehmen der interne Aufbau von Expertise angestrebt; Outsourcing gilt als wenig attraktiv.

Die Unternehmen, die keine zusätzlichen Investitionen sondern eine Umverteilung von Mitteln planen, gehen davon aus, dass Apps zum Teil andere Anwendungen ersetzen bzw. dass sie nur ein Teil zukünftiger Entwicklungsprojekte werden. Da gleichzeitig mit (stark) fallenden Preisen für Infrastruktur, Hardware und Software gerechnet wird, wäre demnach keine Erhöhung des Gesamtinvestitionsvolumens nötig, selbst wenn in Zukunft deutlich mehr Apps entwickelt werden sollen.

Mitunter lassen sich zusätzliche Investitionen und Umverteilungen nicht bedingungslos trennen. Werden etwa Entwickler projektspezifisch auch für die Realisierung von Apps eingesetzt, während die Zahl der Mitarbeiter in der Entwicklung steigt, lässt sich nicht ohne weiteres sagen, ob dies als Investition in die Anwendungsentwicklung im Allgemeinen oder die App-Entwicklung im Speziellen zu werten ist. Ähnliches gilt für Infrastrukturmaßnahmen wie etwa die Modernisierung bestehender Entwicklungswerkzeuge. Selbst wenn dies aufgrund der Besonderheiten

von Apps erfolgen sollte, bliebe es vom Charakter her eine Maßnahme, die ohnehin turnusmäßig durchgeführt wird. Dementsprechend wäre sie nicht als Investition in Apps, sondern als Mittel für effektivere Entwicklung zu sehen.

4.3.4.3 Zufriedenheit mit dem Status quo

Zum Abschluss der Interviews stellte sich die Frage, ob die teilnehmenden Unternehmen mit dem Status quo der App-Entwicklung in ihrem Hause zufrieden sind. Diese Einschätzung soll helfen, die Erkenntnisse in diesem Kapitel ins Verhältnis zu setzen. Schließlich stellt es eine Momentaufnahme dar, die relativ zu sehen ist.

Die meisten Teilnehmer sind mit dem Status quo und der aktuellen Entwicklung zufrieden. Dies gilt vor allem in betriebswirtschaftlicher und anwendungsbezogener Hinsicht. Negative Einschätzungen finden sich nur hinsichtlich der Umsetzung und zu technischen Aspekten. So berichtete ein Teilnehmer, App-Projekte würden zu „Hauruck-Aktion“ verkommen und bei der Technik seien fortdauernd Kompromisse im Spannungsfeld zwischen Entwicklungszeit und Qualität zu schließen.

Wirtschaftlich sind die Unternehmen sowohl bei der internen Nutzung als auch bei Apps für Kunden zufrieden. So wurde von diversen Apps berichtet, die etwa durch Zeitersparnisse kalkulatorisch ihren Entwicklungsaufwand wieder erwirtschaftet hätten. Folglich wird auch weiteres Innovationspotential gesehen.

Bisher getätigte Investitionen hätten sich schnell bezahlt gemacht, da die Lernkurve steil sei. Dazu käme, dass typischerweise junge Entwickler bzw. Berufseinsteiger mit nicht allzu hohem Gehalt zum Einsatz kämen. Das verringere die Kosten.

Auch wenn einzelne Projekte nicht erfolgreich gewesen sind, wird dies nicht als Scheitern betrachtet. Vielmehr gilt die Erprobung der Möglichkeiten als notwendiger Schritt.

Als nicht vollständig geklärt wird die tatsächliche Akzeptanz von Apps beschrieben. Hier seien noch zusätzliche Erfahrungen nötig. Nicht allen Kunden sei zudem zu vermitteln, warum Apps in bestimmten Bereichen sinnvoll eingesetzt werden könnten. Aber auch die typischen Anforderungen von Kunden müssten noch besser verstanden werden.

4.3.4.4 Verbesserungspotential und Abschluss

Der allgemeinen Zufriedenheit zum Trotz war zu erwarten, dass Unternehmen Verbesserungspotential beschreiben bzw. zum Abschluss Wünsche für die Zukunft beschreiben konnten. Tatsächlich wurden einige Punkte genannt, wenn auch viele Unternehmen die aktuelle Entwicklung auf sich zukommen lassen, da sie deutliche Verbesserungen im Rahmen der Weiterentwicklung der Technologien erwarten.

Verbesserungspotential wird in verschiedensten Bereichen gesehen:

- Wenn die Spracherkennung und -steuerung verbessert wird, eröffnete dies umfangreiche neue Möglichkeiten. Dies gilt sowohl für die Bedienung von Apps im

Allgemeinen als auch für sich erst durch die Verwendung natürlicher Sprache ergebende Nutzungsarten.

- Sicherheit und Datenschutz könnten verbessert werden. Hier werden vor allem Apple und Google in der Pflicht gesehen. Dabei galt BlackBerry einerseits als Vorbild, andererseits wurde deren als „Security by Obscurity“ empfundenes Modell auch kritisiert. Sicherheit im Allgemeinen ist ein Thema, bei dem der Wunsch nach Fortschritt deutlich geäußert wurde (siehe auch Kapitel 7.3).

- Es sollte einfacher werden, mit beschränktem Budget möglichst viele Plattformen zu bedienen – ganz im Sinne des dieser Broschüre zugrundeliegenden Projekts.

- Die mobil verfügbaren Übertragungsgeschwindigkeiten für Daten sollten sich verbessern. Somit würde die routinemäßig mobile Nutzung von Videokonferenzen und Ähnlichem denkbar.

- Die Synchronisation zwischen Apps, die Offline-Unterstützung bieten, und ihrem Backend sollte durch mächtigere Werkzeuge oder Methoden unterstützt werden.

- Ähnlich wie im PC-Umfeld seit Langem üblich muss eine geeignete „Enterprise-Unterstützung“ gefunden werden. Die heutige Technik sei häufig zu stark vom Consumer-Umfeld geprägt.

- Ein Teilnehmer erklärte, er könne sich vorstellen, dass sich Business Apps sinnvoll aus Programmbausteinen zusammensetzen ließen. Hier besteht demnach Forschungsbedarf.

- Generell wünschten sich viele Unternehmen technische bzw. technologische Innovationen, die es erlauben, weniger Kompromisse bei der Entwicklung von Apps eingehen zu müssen.

Darüber hinaus wurden Möglichkeiten erwünscht, die Anforderungsanalyse besser zu gestalten. Allgemein sehnen sich die Unternehmen zudem nach Investitionssicherheit: eine Konvergenz der Technologien in nicht allzu ferner Zukunft wäre sehr hilfreich.

In Bezug auf die zukünftige Entwicklung betonten mehrere Teilnehmer, dass sie sehr gespannt auf die Ergebnisse des Projekts seien. Einige betonten, dass auch Ergebnisse, die über ihre eigenen Anforderungen hinausgehen, von Interesse sind. Als Beispiel wurde eine Auseinandersetzung mit am Markt verfügbaren Cross-Plattform-Ansätzen genannt. Offenbar verbirgt sich dahinter der Wunsch, die aktuelle Entwicklung möglichst detailliert nachvollziehen zu können und technisch auf dem aktuellsten Stand zu bleiben.

Mehrere Teilnehmer erklärten uns zudem, dass es wertvoll wäre, genauer zu verstehen, welche Anforderungen Kunden an Apps haben und welche Nutzungsszenarien es für Apps gibt. Dem folgend wurde die Beobachtung des Markts als wichtige Aufgabe identifiziert.

4.4 Zwischenfazit

Aufbauend auf den Ergebnissen der Umfrage und der Interviews lässt sich ein kurzes Zwischenfazit ziehen.

Herauszuheben ist zunächst das grundsätzliche Interesse an dem Thema. Für die Unternehmen im IHK-Bezirk Nord Westfalen sind Apps nicht nur von grundsätzlichem Interesse für die geschäftliche Anwendung, sondern ihre Nutzung wird häufig konkret geplant. Gleichzeitig ließ sich Besonnenheit beobachten: die teilnehmenden Unternehmen laufen nicht Gefahr, unreflektiert einem eventuellen Hype zu erliegen. Nichtsdestotrotz misst eine Reihe der Unternehmen Apps für die Zukunft eine strategische Bedeutung zu.

Bemerkenswert ist, dass einerseits bisherige Entwicklungsprojekte recht unstrukturiert verliefen und Mitarbeiter kaum geschult wurden, andererseits das Interesse gerade auf Seiten technikaffiner Mitarbeiter sehr hoch ist.

Für alle Unternehmen ist es wichtig, mit ihren Apps alle *relevanten* Plattformen zu unterstützen. Welche Plattformen aber als bedeutsam erachtet werden, ist individuell sehr verschieden. Insgesamt wird iOS und Android eine sehr hohe Relevanz zugemessen. Ein natives Look & Feel von Apps spielt für viele Unternehmen eine bedeutende Rolle, einige legen darauf aber keinen Wert.

Festzustellen war insbesondere in den Interviews, dass es bezüglich Business Apps nicht *eine* Schwierigkeit oder *wenige* Herausforderungen gibt. Vielmehr gibt es eine ganze Reihe von ungelösten Fragen sowie Wünschen hinsichtlich der zukünftigen Entwicklung.

Sobald man von der Betrachtung grober Erkenntnisse auf eine feinere Sicht wechselt, zeigt sich ein Höchstmaß an Heterogenität. Bei Fragen zur Entwicklung oder zur Erhebung von Anforderungen etwa lässt sich zum gegenwärtigen Zeitpunkt noch kein homogenes Bild zeichnen.

Die bei der Erhebung des Status quo vereinzelt identifizierten Best Practices greift Kapitel 6 auf. Kapitel 7 betrachtet ausgewählte Aspekte, die im Rahmen der Interviews spezielle Erwähnung fanden. Bei diesen konnte erheblicher Forschungsbedarf festgestellt werden.

Kapitel 5

Cross-Plattform-Entwicklung

Inhaltsübersicht

Schon zu Beginn des Projekt und erneut während der Interviewphase wurde deutlich, dass die Entwicklung mobiler Anwendungen für mehrere Plattformen ein zentrales Anliegen ist. Dieses Kapitel stellt deshalb zunächst grundsätzliche Überlegungen an und anschließend eine Kategorisierung von Cross-Plattform-Ansätzen vor. Entscheidungskriterien für die Auswahl eines Cross-Plattform-Frameworks nebst einer Evaluation prominenter Frameworks führen schließlich zu einer Liste vorläufiger Handlungsempfehlungen und Ansätzen für zukünftige Forschung und Entwicklung.

5.1 Grundsätzliche Überlegungen und Einordnung

Auf dem Smartphone-Markt konkurrieren unterschiedliche Plattformen mit jeweils eigener Hardware, eigenem Betriebssystem und eigenen Frameworks um die Gunst der Kunden und der Entwickler (siehe Kapitel 3). Die Heterogenität auf Ebene der Hardware und der Software ist eine wesentliche Herausforderung für die Entwicklung von Anwendungen für mobile Endgeräte. Die Hardware-Heterogenität gleicht zu einem großen Teil das jeweilige Betriebssystem aus. Auch wenn einzelne Plattformen wie z. B. Android vereinzelt Probleme beim Umgang mit ihrer Hardwarevielfalt aufweisen, fällt diese für einen Großteil der Anwendungsszenarien nichts ins

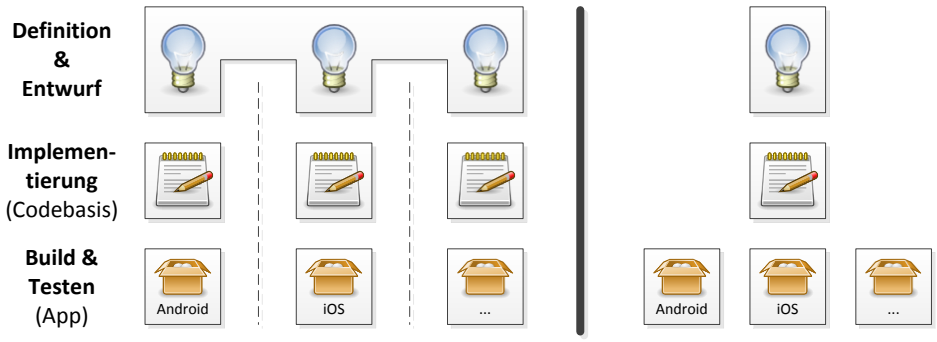

Abbildung 5.1: Gegenüberstellung des nativen (links) und des Cross-Plattform-Entwicklungsprozesses (rechts)

Gewicht und Entwickler können pro Plattform auf eine einheitliche Abstraktionsebene unabhängig von der Hardware zugreifen. Wesentlich stärker beeinflussen die unterschiedlichen Betriebssysteme den Entwicklungsprozess.

Eng verbunden mit dem Betriebssystem sind die bevorzugte Programmiersprache der Plattform und die von der Plattform bereitgestellten Frameworks und APIs zum Zugriff auf Funktionen des Betriebssystems. Die Gesamtheit dieser für die Entwicklung einer App relevanten Elemente wird unter dem Begriff SDK zusammengefasst. Nutzt man die von der Plattform vorgeschlagenen Prozesse und Methoden und programmiert die Anwendung für eine Plattform separat, so spricht man von *nativer Entwicklung*. Die linke Hälfte von Abbildung 5.1 skizziert den nativen Prozess bei mehreren zu unterstützenden Plattformen. Die Entwicklung greift auf die von der Plattform angebotenen Elemente zurück und ist demzufolge immer stark auf diese Plattform fokussiert und beschränkt. Die Ergebnisse des Entwicklungsprozesses sind eng mit den plattformspezfischen Elementen verbunden und zu großen Teilen nicht auf anderen Plattformen wiederverwendbar. Dies betrifft nicht nur Quelltext und unterstützende Artefakte wie Tests, sondern auch Teile des Entwurfs, sofern es sich um von der Plattform beeinflusste Design-Entscheidungen handelt. Den Vorteilen einer auf eine Plattform angepassten Entwicklung steht der deutlich erhöhte Aufwand gegenüber, wenn mehr als eine Plattform unterstützt werden soll.

Die Notwendigkeit, eine Business App für mehr als eine Plattform bereitzustellen, ergibt sich aus der historischen, aktuellen und erwarteten Verteilung der Plattformen. Die historischen und aktuellen Marktanteile bestimmen, welche Plattformen derzeit (noch) in der Zielgruppe im Einsatz sind. Die Prognosen über zukünftige Marktanteile in Verbindung mit den aktuellen Verkaufstrends beeinflussen die langfristige Tragbarkeit und Zukunftssicherheit der Plattform-Strategie eines App-Entwicklers. In der aktuellen Marktsituation dominieren die beiden großen Plattformen Android und iOS [27]. Je nach Zielgruppe müssen aber zusätzlich auch BlackBerry oder die Symbian-Plattform von Nokia berücksichtigt werden, falls sie in der

Zielgruppe einen relevanten, nicht vernachlässigbaren Anteil an den aktuell im Einsatz befindlichen Geräten aufweisen. Bei strategischen Investitionen und Entwicklungsprojekten spielt der erwartete Erfolg von Plattformen wie Microsofts Windows Phone eine Rolle. Unabhängig von der konkreten Ausrichtung des App-Projekts werden somit in den meisten Fällen mindestens die zwei dominierenden Plattformen zu berücksichtigen sein, gegebenenfalls aber deutlich mehr. Schon Android und iOS unterscheiden sich erheblich, nicht zuletzt in den bevorzugten Programmiersprachen, sodass Entwicklungsansätze attraktiv erscheinen, die von vornherein auf mehrere Plattformen ausgerichtet sind. Bei der *plattformübergreifenden Entwicklung* wird prinzipiell eine gemeinsame Codebasis für alle zu unterstützenden Plattformen erstellt (Abbildung 5.1, rechte Hälfte). Sie bietet somit das Potenzial, den Entwicklungsaufwand deutlich zu verringern. Statt sich in mehrere Plattformen einarbeiten zu müssen und die Anwendung für jede Plattform vollständig separat zu implementieren, können Entwickler sich bei einem plattformübergreifenden Entwicklungsansatz auf *eine* übergreifende Plattform beschränken. Demgegenüber sind je nach verwendetem Ansatz gegebenenfalls Einschränkungen hinsichtlich unterstützter Features oder in Bezug auf die Gestaltung der Benutzeroberfläche hinzunehmen. Die meisten plattformübergreifenden Ansätze vereinheitlichen Entwurf und Implementierung. Das *Packaging* der Apps und das Testen muss aufgrund der Unterschiede zwischen den Plattformen weiter separat erfolgen.

Solche plattformübergreifenden Ansätze im mobilen Umfeld, ihre Vor- und Nachteile sowie Einsatzgebiete sind der Fokus dieses Kapitels. Historisch gesehen stand die Entwicklung von Desktop-Applikationen vor ähnlichen Problemen, die sich aus dem Nebeneinander unterschiedlicher Desktop-Betriebssysteme wie Windows, Linux oder Mac OS ergaben. Es wurden zahlreiche Lösungsstrategien entwickelt; Kompatibilitätsprobleme bestehen dennoch fort. Neben der Tatsache, dass die mobilen Plattformen nicht auf diesen Betriebssystemen, sondern auf speziellen Mobil-Betriebssystemen laufen, unterscheiden sich die Problemstellungen hinsichtlich mindestens zwei weiterer gravierender Punkte, sodass sie nicht vollständig vergleichbar sind und sich die Lösungsansätze unterscheiden. Zum einen dominierten Windows-Betriebssysteme auf dem Desktop lange Zeit und in Teilen bis heute deutlich stärker als irgendeine Plattform im mobilen Umfeld, sodass viele Desktop-Anwendungen insbesondere für Verbraucher nicht mehr als ein System unterstützen müssen. Zum anderen beschränken sich die zu unterstützenden Hardware-nahen Funktionen bei vielen Desktop-Anwendungen im Wesentlichen auf die grafische Benutzeroberfläche, das Dateisystem und die Netzwerkschnittstellen. Zu diesen treten bei Apps eine Vielzahl weiterer Hardwarefunktionen wie Kamera oder GPS.

Nichtsdestotrotz sind einige Gemeinsamkeiten zwischen der aktuellen Situation im mobilen Umfeld und der Situation im Desktopbereich feststellbar. Einen Ansatz zum Umgang mit der Plattformvielfalt auf dem Desktop stellt die Java-Technologie dar. In den letzten Jahren, spätestens seit dem Aufkommen des Web 2.0, fanden

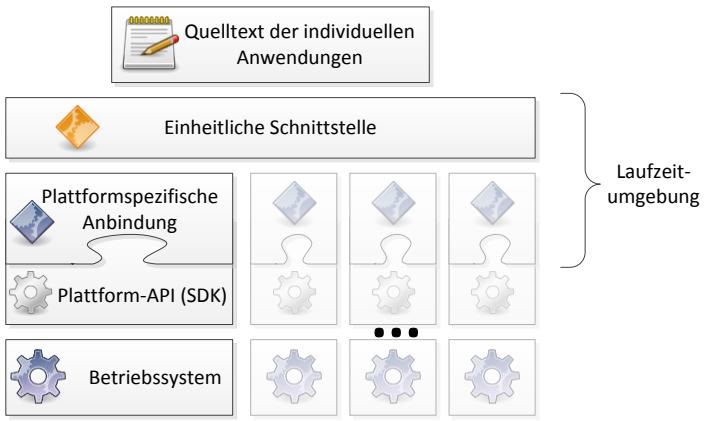

Abbildung 5.2: Konzeptionelles Schema von Laufzeitumgebungen für die Cross-Plattform-Entwicklung mobiler Anwendungen

auf dem Desktop auch Web-basierte Anwendungen Zuspruch. Beiden Ansätzen ist gemein, dass die Anwendungen nicht direkt auf der Hardware des Zielsystems laufen, sondern in einer separaten Laufzeitumgebung. Die Laufzeitumgebung fungiert als Kompatibilitätsschicht, auf der die Anwendungen entwickelt und ausgeführt werden. Im Fall von Java handelt es sich um die Java Virtual Machine, Web-Anwendungen laufen im Browser. In letzterem Fall kann zudem ein Teil der Logik *entfernt* (etwa auf einem Server) abgebildet werden.

5.2 Mögliche Ansätze

Die Verwendung einer separaten Laufzeitumgebung ist im mobilen Umfeld die vorherrschende Technik für plattformübergreifende Entwicklung. Wie in Abbildung 5.2 dargestellt, abstrahiert eine Umgebung von Unterschieden zwischen den Plattformen durch eine einheitliche Schnittstelle, auf der die Anwendungen aufbauen können. Für jede mobile Plattform, auf der die Laufzeitumgebung zur Verfügung stehen soll, existiert eine Anbindung und Umsetzung des plattformübergreifenden Systems an die Spezifika der jeweiligen Plattform. Die Laufzeitumgebung bietet Entwicklern konkreter Anwendungen eine gemeinsame Abstraktionsebene über alle unterstützten Plattformen, auf der die Entwickler ihre Anwendungen aufbauen können. Es existieren verschiedene Ansätze, wie eine derartige Laufzeitumgebung umgesetzt werden kann. Der verwendete Ansatz determiniert, wie der zu entwickelnde plattformübergreifende Quelltext aussieht. Die einfachste Herangehensweise, diese Kompatibilitätsschicht nur als eine gemeinsame API im Sinne eines Adapters für die Plattform-SDKs zu konzipieren, entfällt aufgrund der unterschiedlichen Programmiersprachen auf mobilen Plattformen.

Die offensichtlichste Plattformen übergreifende Entwicklungsmethode im mobilen Umfeld greift auf den Web-Browser als Laufzeitumgebung zurück. Die in diesem Rahmen erstellten *mobilen Web-Apps* basieren auf HTML, Cascading Style Sheets (CSS) und JavaScript und sind auf den meisten Plattformen mit modernem Browser lauffähig. Der Quelltext der mobilen Web-App wird im Browser ausgeführt und kann durch Verwendung neuer Web-Technologien wie HTML5 unter Umständen auch auf fortgeschrittene Funktionen der Plattformen zurückgreifen, z. B. solche zur Bestimmung der geografischen Position. Die Unterstützung derartiger Funktionen variiert allerdings von Browser zu Browser. Da die Apps zudem in jedem Fall im Browser laufen, der als *Sandbox* konzipiert ist und Zugriffe auf das System minimieren soll, ist nicht damit zu rechnen, dass Web-Anwendungen jemals umfassenden Zugriff auf alle Funktionen des Betriebssystems erhalten werden. Neben dem eingeschränkten Funktionsumfang ist die Benutzeroberfläche von Web-Anwendungen zudem inhärent Web-basiert und unterscheidet sich in Aussehen und Verhalten von nativen mobilen Anwendungen. Nichtsdestotrotz stellen Web-Apps für viele Einsatzzwecke eine ernstzunehmende und leicht zu realisierende Entwicklungsalternative dar. Sie profitieren von den ausgereiften Web-Technologien und dem reichhaltigen Angebot an Web-Frameworks. Zudem existiert eine Vielzahl von Frameworks für die Entwicklung mobiler Webseiten, die Entwickler hinsichtlich häufig auftretender Anforderungen mit CSS und JavaScript unterstützen. Eine detailliertere Betrachtung erfolgt in Unterabschnitt 5.4.1 dieses Kapitels.

Hybride Ansätze für die plattformübergreifende Entwicklung beabsichtigen, die Lücke zwischen den Web-Technologien und den (Hardware-)Funktionen der Plattformen zu überbrücken. Solche hybriden Ansätze, wie z. B. Apache Cordova [28] (ehemals PhoneGap) kombinieren die Fähigkeiten der mobilen Browser hinsichtlich Rendering von HTML und Ausführung von JavaScript mit nativen Elementen. Dazu betten sie die Web-Steuerelemente der jeweiligen Plattform in die App ein. Die Browser-Engine stellt die Webseite dar und führt die in JavaScript implementierte Programmlogik aus. Wann immer eine native Funktion aufgerufen werden soll, übergibt das eingebettete Browser-Element an den nativen Teil des Frameworks. Die genaue Ausgestaltung des hybriden Ansatzes hängt vom verwendeten Framework ab. In Unterabschnitt 5.4.2 wird Apache Cordova (PhoneGap) als prominentester Vertreter genauer betrachtet. Da sie eine Hybridform von Web-App und nativen Funktionen sind, gelten einige der oben erwähnten Vorteile und Einschränkungen mobiler Web-Apps auch für hybride Apps. Der hybride Ansatz adressiert im Wesentlichen den Kritikpunkt des fehlenden Zugriffs auf Plattformfunktionen. Viele Web-Frameworks, die bei mobilen Web-Apps zum Einsatz kommen, sind auch bei hybriden Apps einsetzbar.

Ein darüber hinausgehender Ansatz im Bereich der Laufzeitumgebungen greift nicht mehr auf den üblichen *Stack* an Web-Technologien und die schon auf den Plattformen vorhandenen Browser-Engines zurück, sondern stellt eine vollständig

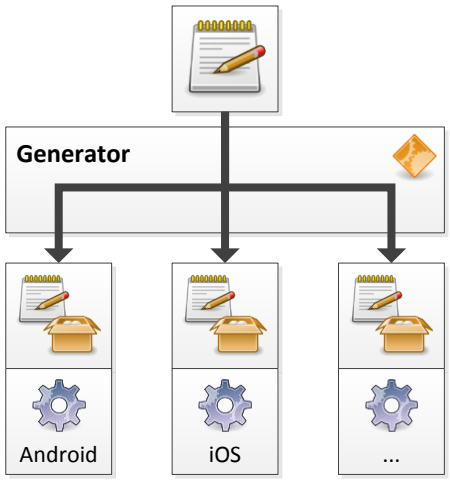

Abbildung 5.3: Funktionsweise generierender Ansätze

eigenständige Laufzeitumgebung bereit. Der Quelltext, den Entwickler auf diese individuelle Laufzeitumgebung abgestimmt entwickeln, kann in anderen Sprachen geschrieben sein und andere Technologien verwenden, als im Web-Kontext üblicherweise verfügbar sind. Die Interpretation des Programms erfolgt vollständig durch die jeweilige *Engine*. Dieser Ansatz ist prinzipiell flexibler bezüglich Einbindung und Anpassung an die Plattform, weil die Engine sämtliche Bereiche der auf ihr aufbauenden Apps kontrolliert – inklusive der Darstellung, die bei Web-basierten Ansätzen von der Browser-Engine übernommen wird. Diese größere Flexibilität ermöglicht zum Beispiel, die Benutzeroberfläche aus nativen Steuerelementen aufzubauen. Der Verzicht auf die bekannten Web-Technologien bedeutet aber im Allgemeinen auch einen höheren Einarbeitungsaufwand. Die Erstellung derartiger Frameworks ist zudem komplexer und umfangreicher als der Rückgriff auf existierende Web-Technologien. Unter Umständen sind entsprechende Frameworks aus diesem Grund weniger ausgereift oder bieten nur einen beschränkten Funktionsumfang.

Der Ansatz, eine hybride oder eigenständige Laufzeitumgebung als plattformübergreifende Kompatibilitätsschicht einzusetzen, ist derzeit der populärste und wird von den meisten Cross-Plattform-Frameworks im mobilen Umfeld verwendet. Es sind aber auch Techniken zur integrierten Entwicklung für mehrere Plattformen denkbar, die keine separate Laufzeitumgebung auf der Plattform nutzen, sondern Anwendungen generieren, die direkt in den plattformspezifischen Umgebungen laufen. Abbildung 5.3 stellt das Prinzip solcher *generierender Ansätze* wie modellgetriebener Entwicklung oder Cross-Compiling dar. Die weiterhin gemeinsame, plattformübergreifende Codebasis wird bei Generator-Ansätzen automatisch beim Erstellen der plattformspezifischen Pakete auf die jeweilige mobile Plattform portiert. Im Ergebnis entstehen hierbei vollständig native Anwendungen, die direkt auf den Plattformen lauffähig sind. Im Rahmen der Portierung generiert das Cross-Plattform-Framework

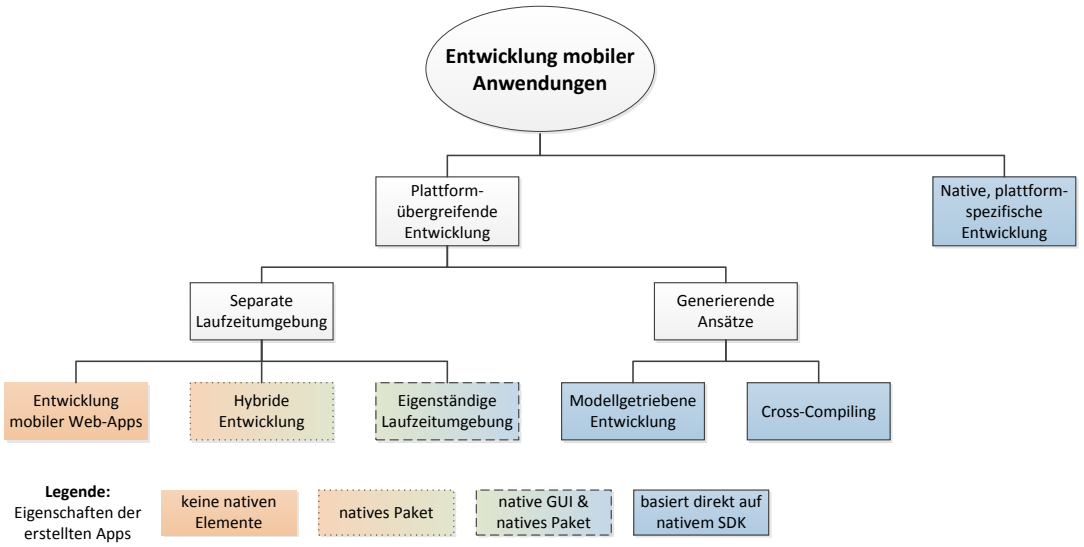

Abbildung 5.4: Hierarchie der Cross-Plattform-Entwicklungsansätze

dabei Quelltext in der plattformtypischen Sprache ausgehend von der gemeinsamen Codebasis, die in einer gemeinsamen, plattformunabhängigen Sprache geschrieben ist. Bei einem modellgetriebenen Ansatz kann der plattformübergreifende Quelltext zum Beispiel in einer speziellen domänenspezifischen Sprache für die Beschreibung mobiler Anwendungen geschrieben sein. Code-Generatoren transformieren das so beschriebene Modell der App dann für jede Plattform in den entsprechenden Programmcode, d.h. nach Java für Android und nach Objective-C für iOS, unter Verwendung der entsprechenden Plattform-APIs. Entsprechende Ansätze wie z. B. applause [29] oder XMLVM [30] sind noch in frühen Stadien der Entwicklung. Das Unterkapitel 5.6 stellt diese Frameworks sowie das als Ableger des Business-Apps-Projekts entstandene modellgetriebene Framework MD^2 kurz vor.

Abbildung 5.4 stellt die betrachteten Entwicklungsansätze in Form einer Hierarchie dar. Tabelle 5.1 fasst die wichtigsten Eigenschaften der verschiedenen Ansätze zusammen. Die vorgestellten Cross-Plattform-Ansätze unterscheiden sich hinsichtlich des Deployment-Formats der erstellten App. Bei der mobilen Web-Entwicklung werden Web-Apps erstellt, auf die Nutzer über den Browser zugreifen. Die übrigen Ansätze produzieren eine App im nativen Format, wie sie auch bei der nativen Entwicklung erstellt wird. Apps im nativen Deployment-Format (z. B. eine apk-Datei für Android) lassen sich auf den mobilen Geräten installieren und über die typischen Vertriebskanäle dieser Plattformen, die Appstores, verteilen.

Die in der Hierarchie abgebildeten Ansätze unterscheiden sich ferner in Bezug auf die grafische Benutzeroberfläche der erstellten Apps. Während Apps, die mit generierenden Ansätzen erstellt wurden, direkt auf dem nativen SDK der Plattformen

Beschreibung des Ansatzes	Beispiele	Native Elemente
Mobile Web-Apps: Webseiten optimiert für mobile Geräte, abrufbar über den mobilen Browser, unterstützt durch spezialisierte mobile Frameworks	HTML5, jQueryMobile, Sencha Touch	Keine
Hybride Entwicklung: Web-Komponente eingebettet in natives Framework, das Zugriff auf Plattformfunktionen bereitstellt; Apps als Webseite zusammen mit Framework-Engine als natives Paket	Apache Cordova, Rhomobile	App in nativem Deployment-Format
Eigenständige Laufzeitumgebung: Framework stellt eigenständige Engine bereit; Apps werden mit Engine gebündelt, die zur Laufzeit den Quelltext interpretiert	Appcelerator Titanium Mobile	Natives Deployment-Format; native GUI möglich
Generierende Ansätze: Apps werden in gemeinsamer Codebasis plattformübergreifend entwickelt und dann in native Apps transformiert	MD^2, applause, XMLVM	Generierte App basiert auf nativem SDK
Native Entwicklung: für jede Plattform wird eine separate App unter Verwendung der jeweiligen plattformtypischen Programmiersprache und APIs entwickelt	Android, iOS, BlackBerry	Verwendet direkt das jeweilige native SDK

Tabelle 5.1: Übersicht über Cross-Plattform-Entwicklungsansätze

basieren, können eigenständige Laufzeitumgebungen zumindest native Komponenten zur Erstellung der GUI einsetzen, auch wenn damit entwickelte Apps keinen direkten Zugriff auf das native SDK haben. Web-basierte Ansätze verwenden unmittelbar keine nativen Elemente und haben deshalb zunächst ein anderes Look & Feel. Mit zusätzlichem Aufwand ist es gegebenenfalls möglich, ein solches zu imitieren, Unterschiede werden aber meist sichtbar und spürbar sein.

5.3 Auswahlkriterien für Cross-Plattform-Ansätze

Wenn Business Apps für mehrere mobile Plattformen entwickelt werden sollen, hängt die Entscheidung für einen der genannten Cross-Plattform-Entwicklungsansätze – bzw. für ein konkretes Cross-Plattform-Framework – von einer Vielzahl an Kriterien ab, die im Folgenden kurz betrachtet werden. Einige Kriterien wurden schon bei der Vorstellung der prinzipiellen Ansätze im vorherigen Kapitel erwähnt. Die Kriterien sind so gewählt, dass sie möglichst gut zur Differenzierung der Frameworks geeignet

sind. Kriterien, bei deren Bewertung sich die Cross-Plattform-Ansätze kaum unterscheiden, wurden nicht aufgenommen. Dies betrifft vor allem Querschnittsthemen wie z. B. Sicherheitsbelange (siehe auch Kapitel 7.3).

Die Kriterien – sowie die Evaluation verschiedener Frameworks auf Basis der Kriterien im folgenden Kapitel 5.4 – basieren auf mehreren Quellen. Neben den Interviews im Rahmen des Projekts „Business Apps“ trugen die Forschungsarbeit der Autoren sowie mehrere Abschlussarbeiten von Studenten dazu bei, dass eine umfassende und praxisrelevante Liste von Kriterien entstanden ist, die eine fundierte Bewertung und Auswahlentscheidung ermöglicht. Weitere Informationen sind auch einem wissenschaftlichen Artikel der Autoren zum Thema zu entnehmen [HHM12].

Die Auswahl eines Entwicklungsansatzes und eines konkreten Frameworks bedingen sich hinsichtlich der Kriterien. Die folgende Zusammenstellung enthält sowohl Kriterien, die vordringlich vom Ansatz des Frameworks determiniert werden, als auch solche, die abhängig vom konkreten Framework sind, gegebenenfalls innerhalb der vom Ansatz vorgegebenen Grenzen. Der vom Framework verfolgte Entwicklungsansatz determiniert für die erste Gruppe der Kriterien direkt, in welchem Umfang das Framework diese erfüllt. Bei einer zweite Gruppe von Kriterien gibt der Ansatz den Rahmen vor, innerhalb dessen das Framework das Kriterium erfüllen kann. Eine dritte Gruppe von Kriterien ist unabhängig vom Entwicklungsansatz und vollständig vom eigentlichen Framework bestimmbar. Der Übergang zwischen den Gruppen ist dabei nicht trennscharf. Im Folgenden sind die Kriterien grob nach abnehmender Abhängigkeit vom Entwicklungsansatz sortiert.

5.3.1 Vom Entwicklungsansatz determinierte Kriterien

Distribution Die Distribution kann entweder über die entsprechenden Kanäle der mobilen Plattformen (Appstores) erfolgen, oder als Web-App. Wie im vorhergehenden Kapitel erläutert, hängt das vom Deployment-Format ab, das durch den Entwicklungsansatz festgelegt ist.

Migrationsoptionen Bei einigen Ansätzen lässt sich das Ergebnis leicht auf andere Ansätze migrieren; zum Beispiel kann eine Web-App bei neuen Anforderungen mit minimalen Aufwand zu einer hybriden App portiert werden. In engen Grenzen kann ein Framework die Erfüllung dieses Kriteriums beeinflussen, wenn es besonderen Augenmerk auf Standardisierung legt.

5.3.2 Vom Entwicklungsansatz beeinflusste Kriterien

Look & Feel Auch wenn die generelle Erscheinung einer App durch diese selbst festgelegt wird, hat das verwendete Framework einen starken Einfluss auf Aussehen und Verhalten der App. Der dem Framework zugrundeliegende Ansatz bestimmt maßgeblich, inwiefern sich die App wie eine native App verhält.

Bei Web-basierten Ansätzen ist dies zum Beispiel nur eingeschränkt möglich oder erfordert hohen Aufwand von Seiten des Frameworks oder des App-Entwicklers. App-Nutzer sind die spezifische Erscheinungsform ihrer Plattform gewohnt und erwarten von Apps, dass sie sich dieser in Aussehen und Verhalten anpassen. Das betrifft auch den Lebenszyklus einer App, der das Verhalten im Fall von Unterbrechungen oder eines Kontextwechsel definiert. Die Bedeutung dieses Kriteriums variiert je nach Zielgruppe der Business App. Während bei kundenzentrierten Apps oft auf eine professionelle Außendarstellung Wert gelegt wird, steht bei Apps für die Mitarbeiter häufig eher die Funktionalität im Mittelpunkt. Aber auch hier kann das Look & Feel für die Akzeptanz von Bedeutung sein. Nutzt man die intuitiven Verhaltensmuster in der Bedienung von Apps, ist zudem gegebenenfalls eine höhere Produktivität möglich.

Performance Eng verbunden mit dem Erscheinungsbild der Oberfläche sind die Ausführungsgeschwindigkeit der App und ihre Reaktionszeit auf Benutzerinteraktionen. Laufzeitumgebungen sind in diesem Punkt zunächst im Nachteil.

Zugriff auf plattformspezifische Funktionen Um die Möglichkeiten mobiler Geräte auszunutzen, benötigen Business Apps für viele Anwendungsfälle Zugriff auf plattformspezifische Funktionen wie die Kamera des Geräts, Ortung oder Kontaktinformationen. Die meisten Cross-Plattform-Frameworks bieten Zugriff auf die grundlegenden Funktionen. Unterschiede gibt es bei fortgeschrittenen Anforderungen, z. B. bezüglich Beschleunigungsmesser oder Near Field Communication (NFC). Web-Apps sind in diesem Punkt konzeptionell eingeschränkt und abhängig von der Unterstützung, die der Browser für fortgeschrittene HTML5-Spezifikationen z. B. zur Geolokation anbietet.

Unterstützte Plattformen Anzahl und Relevanz unterstützter Plattformen sind weitere wichtige Entscheidungsmerkmale. Mehr noch als die Tatsache, ob eine Plattform unterstützt wird, ist die Frage entscheidend, in welchem Umfang und in welcher Qualität die Funktionen auf der Plattform bereitgestellt werden. Leichtgewichtigere, Web-basierte Entwicklungsansätze sind hier im Vorteil, weil entsprechende Frameworks einfacher auf weitere Plattformen portiert werden können.

Komfort des Entwicklungsprozesses Ein intuitiver und unkomplizierter Entwicklungsprozess erleichtert die Arbeit mit einem Framework. Diese Eigenschaften werden nur zu einem geringen Anteil vom zugrundeliegenden Ansatz bestimmt. Haupteinflussfaktoren sind Architektur, Programmiersprache und API des Frameworks, die auch die Lernkurve des Frameworks prägen. Eine vollständige und verständliche Dokumentation erhöht den Komfort des Entwicklungsprozesses erheblich.

Entwicklungsgeschwindigkeit und -kosten Unterstellt man eine annähernd proportionale Beziehung zwischen der Entwicklungsdauer und den reinen Entwicklungskosten, so spielt die Geschwindigkeit, mit der Anforderungen umgesetzt werden können, eine entscheidende Rolle bei der Bewertung eines Frameworks. Dieses Kriterium wird beeinflusst vom Entwicklungsprozess und der Entwicklungsumgebung.

GUI-Design Die Erstellung der Benutzeroberfläche ist aufgrund der beschränkten Bildschirmgröße und zum Teil auch Auflösung mobiler Geräte und der besonderen Eingabegeräte (Touchscreen statt Maus und Tastatur) besonders wichtig und aufwendig. Bei der Auswahl eines Cross-Plattform-Frameworks ist deshalb die Art und Weise, wie die GUI definiert wird, entscheidungsrelevant. Dabei zählt nicht nur die generelle Struktur der GUI-Beschreibung, z. B. deklarativ oder programmatisch (d. h. imperativ), die teilweise vom Entwicklungsansatz bestimmt ist, sondern auch die Unterstützung durch GUI-Designtools.

Entwicklungsumgebung Die Verfügbarkeit und der Reifegrad der Entwicklungswerkzeuge, die für Entwicklungen mit dem jeweiligen Cross-Plattform-Tool zum Einsatz kommen können, beeinflussen die generelle Handhabbarkeit des Frameworks. Zu den Werkzeugen kann zum Beispiel eine Integrated Development Environment (IDE) mit Funktionen wie *Syntax Highlighting* und Auto-Vervollständigung gehören, ebenso wie ein Debugger und Emulator. Der Aufwand für die Einrichtung der gesamten Entwicklungsumgebung ist ebenfalls zu beachten.

5.3.3 Vom Framework eigenständig beeinflussbare Kriterien

Zukunftssicherheit Eine strategische Entscheidung für ein Cross-Plattform-Framework sollte auch berücksichtigen, wie zukunftssicher das Framework ist, d.h. ob langfristige Stabilität, Weiterentwicklung und Support wahrscheinlich sind. Indikatoren dafür sind die aktuellen Entwicklungszyklen des Frameworks, regelmäßige Bugfixes und Releases, Unterstützung für die aktuellen Plattformversionen sowie eine aktive Community[1] und eine große, diversifizierte Gruppe von Entwicklern. Erhält ein Framework professionelle und dauerhafte Unterstützung von Firmen, so ist es wahrscheinlicher, dass es auch für die Zukunft eine gute Entscheidung ist. Sofern ein Framework von einzelnen Entwickler(firmen) dominiert wird und von diesen in hohem Maße abhängig ist, besteht Gefahr für die langfristige Stabilität, sollten sich die zentralen Akteure aus dem Projekt zurückziehen. Ein Cross-Plattform-Framework, dessen Ent-

[1] Eine aktive Community zeichnet sich z. B. durch das Vorhandensein von Foren aus, in denen sich Entwickler und Nutzer austauschen.

wicklungsansatz Migrationsoptionen als letzten Ausweg bereithält (s. o.), ist etwas zukunftssicherer.

Wartbarkeit Umfang und Lesbarkeit des Quelltexts, wie sie typischerweise bei Verwendung des jeweiligen Frameworks zu beobachten sind, beeinflussen die Wartbarkeit. Dieses Kriterium hängt eng mit anderen Kriterien, die den Entwicklungsprozess betreffen, zusammen.

Skalierbarkeit Die Skalierbarkeit im Sinne der Handhabbarkeit des Frameworks bei größeren Projekten und größeren Entwicklerteams hängt davon ab, wie modularisierbar mit dem Framework erstellte Anwendungen sind und inwiefern Architektur und Konzeption des Frameworks für umfangreichere Projekte ausgelegt und einsetzbar sind.

Lizenz und Kosten des Frameworks Die Lizenz, unter der das Framework bereitgestellt wird, hat auch Einfluss auf die Kosten, die für das Framework anfallen. Zu den Einsatzkosten eines Frameworks zählen aber auch Kosten für Support und Betrieb der Infrastruktur, d. h. sämtliche Kosten, die beim Einsatz des Frameworks anfallen. Kosten für die eigentliche Entwicklung werden hingegen in einem separaten Kriterium berücksichtigt.

5.4 Vorstellung ausgewählter Rahmenwerke

Die folgenden Unterkapitel blicken genauer auf konkrete Cross-Plattform-Lösungen, stellen diese vor und bewerten sie anhand der Kriterien aus Kapitel 5.3, beginnend mit mobilen Web Apps in Kapitel 5.4.1. Es folgen die Vorstellung und Bewertung von Apache Cordova (PhoneGap) als Vertreter hybrider Lösungen in Kapitel 5.4.2 und in Kapitel 5.4.3 Appcelerator Titanium Mobile als Beispiel für eine eigenständige Laufzeitumgebung. Tabelle 5.2 gibt einen Überblick über die Evaluationsergebnisse. Die Evaluation basiert in Teilen auf einem wissenschaftlichen Artikel der Autoren [HHM12], der für weitere Informationen konsultiert werden kann.

5.4.1 Mobile Web-App

Eine mobile Web-App ist eine Webanwendung, die für die Anzeige und Bedienung auf mobilen Endgeräten optimiert ist. Es kann sich entweder um die mobile Version einer bestehenden Webanwendung handeln oder um eine eigenständig entwickelte Anwendung. Davon abzugrenzen sind mobile Webseiten, die im Unterschied zu einer Webanwendung keine komplexe Interaktion mit dem Benutzer unterstützen und hauptsächlich Informationen darstellen, wobei die Trennlinie naturgemäß unscharf verläuft. Der Fokus im Folgenden liegt ausgehend vom Thema Business Apps auf den mobilen Web-Apps. Web-Apps basieren auf standardisierten Web-Technologien:

	Mobile Web-App	Apache Cordova	Titanium Mobile
Distribution	o	+	+
Migration	+	+	-
Look & Feel	-	o	+
Performance	o	+	-
Funktionen	-	+	+
Plattformen	+	+	-
Entwicklungsprozess	+	+	o
Entwicklungsgeschwindigkeit	+	+	-
GUI-Design	+	+	-
Entwicklungsumgebung	+	+	o
Zukunftssicherheit	+	+	o
Wartbarkeit	+	+	o
Skalierbarkeit	+	+	+
Lizenz & Kosten	o	+	-

Tabelle 5.2: Vergleich von Cross-Plattform-Frameworks

HTML als Auszeichnungsprache für Inhalt und Struktur der Anwendung, CSS für die Darstellung sowie JavaScript für dynamische Elemente und die Anwendungslogik. Web-Apps sind – wie normale Webseiten – über Hypertext Transfer Protocol (HTTP) im Internet erreichbar. Der Browser des mobilen Geräts ruft die URL der Web-App direkt auf oder wird an sie weitergeleitet. Er interpretiert das als Antwort gelieferte HTML-Dokument, stellt es unter Berücksichtigung der CSS-Anweisungen dar und führt den enthaltenen JavaScript-Code aus. Insofern unterscheidet sich das Vorgehen nicht von anderen Webseiten. Der Entwicklungsprozess kann sich am Prozess für normale Web-Anwendungen orientieren, größere Anpassungen sind aufgrund der Geräte- und Plattformvielfalt allenfalls im Bereich des Testens vorzunehmen.

Grundsätzlich sind zwei Strategien zur Entwicklung mobiler Webseiten denkbar. Bei der ersten Alternative wird die mobile Seite eigenständig und unabhängig von eventuell vorhandenen Desktop-Webseiten ähnlichen Inhalts entwickelt. Dieser Ansatz erleichtert die Optimierung für mobile Geräte um den Preis duplizierten Aufwands für die Erstellung und Pflege der parallel betriebenen Webseiten. Die mobile Seite kann optimal auf die mobile Nutzung angepasst werden. Eine solche Optimierung ist insbesondere wichtig, falls es sich tatsächlich um eine Web-App handelt, die anders als eine vorrangig Informationen darstellende Webseite interaktive Funktionalität anbietet.

Die zweite Strategie besteht darin, eine gemeinsame Webseite für alle Gerätetypen von Desktop bis Smartphone anzubieten. Sie eignet sich insbesondere bei Webseiten, die vorrangig Informationen darstellen. Die gemeinsame Webseite kann dabei nach dem Paradigma des *Responsive Web Design* erstellt werden. Derartige Webseiten

besitzen ein flexibles Layout, das sich den jeweiligen gerätespezifischen Gegebenheiten anpasst. CSS3, insbesondere das Modul Media Queries, ermöglicht es, dass dieselbe Webseite sowohl auf großen als auch auf kleinen Bildschirmen jeweils optimal dargestellt wird. Für Webseiten mit nur mäßig ausgeprägtem App-Charakter ist die gemeinsame Strategie dank der Möglichkeiten im Umfeld von HTML5 eine gute Alternative. Eine Fortschreibung dieser strategischen Richtung ist die sogenannte Mobile-First-Strategie, nach der eine Webseite zunächst für mobile Geräte erstellt und anschließend an die Gegebenheiten größerer Geräte angepasst wird [Wro11]. Unterstützer dieser Strategie verweisen zur Begründung auf die gestiegene Bedeutung mobiler Nutzung und auf die höhere Qualität von Webseiten, die mit Fokus auf das Wesentliche entwickelt worden sind.

Die Optimierung für mobile Geräte muss allerdings der typischerweise kleineren Präsentationsfläche und dem unterschiedlichen Nutzungsverhalten Rechnung tragen. Bei aufwendigeren Anwendungen sind auch die limitierten Systemressourcen auf vielen mobilen Geräten zu beachten. Die geringere Auflösung und kleinere Bildschirmdiagonale auf mobilen Geräten – z. B. 1136×640, 4 Zoll beim iPhone 5 oder 1280×720, 4,8 Zoll beim Samsung Galaxy S III – lassen weniger Platz für den Inhalt der Webseite und erfordern eine komprimierte Darstellung. Mobile Geräte werden außerdem häufig im Hochformat verwendet, was den horizontal zur Verfügung stehenden Platz weiter beschränkt. Das Nutzungsverhalten wird hauptsächlich vom oft verwendeten Touchscreen anstelle der Kombination aus Tastatur und Maus geprägt. Die Steuerelemente einer Web-App müssen so dimensioniert sein, dass sie sich trotz der ungenaueren Steuerung leicht bedienen lassen. Wenn möglich, sollten Tastatureingaben durch Auswahlfelder oder andere Eingabeoptionen ersetzt werden, um den Aufwand für den mobilen Benutzer gering zu halten. Die Unterstützung von Gesten, wie sie häufig zur Steuerung in mobilen Anwendungen verwendet werden, verbessert die Bedienbarkeit und den Gesamteindruck einer Web-App. Um diese besonderen Anforderungen im mobilen Feld umzusetzen, existieren diverse Frameworks für das mobile Web, die mit einer Kombination aus CSS und JavaScript die Entwicklung mobiler Web-Apps vereinfachen. Diese Mobile-Web-Frameworks wie jQueryMobile oder Sencha Touch stellen angepasste Komponenten, Stylesheets und Event Handling bereit.

Unterstützung für einige der Problemfelder verspricht auch die nächste Version von HTML, HTML5, die sich derzeit noch im Prozess der Standardisierung befindet. Neben dem eigentlichen Kern von HTML5, d. h. der Auszeichnungssprache als Weiterentwicklung von HTML, ist unter dem Begriff HTML5 auch eine Vielzahl anderer Spezifikationen zusammengefasst, die zurzeit im Umfeld von HTML5 entwickelt werden. Dazu gehören zum Beispiel Schnittstellen zur lokalen Speicherung von Daten oder zur Ermittlung der geographischen Position sowie CSS3, die nächste Version der Formatierungssprache. Die Auszeichnungssprache HTML5 selbst wird unter anderem um semantische Elemente und Multimedia-Unterstützung erweitert.

Obwohl sich die Spezifikation noch in der Entwicklung befindet, werden wesentliche Features von HTML5 schon von einigen Browsern unterstützt.

Die Standardisierung von HTML, CSS und JavaScript ermöglicht eine plattformübergreifende Entwicklung von Web-Apps. Sofern die Browser der Zielplattformen die Spezifikationen korrekt, vollständig und einheitlich umsetzen, kann eine gemeinsame Code-Basis für alle Plattformen verwendet werden. Die Historie auf dem Desktop hat jedoch gezeigt, dass von Browser zu Browser zum Teil erhebliche Unterschiede in Darstellung und Verhalten zu verzeichnen waren. Diese Inkompatibilitäten sind auch zwischen mobilen Browsern festzustellen. Da die Standardbrowser der populärsten Plattformen, Android und iOS, allerdings mit WebKit auf derselben Layout-Engine basieren, sind die Unterschiede geringer und die Unterstützung der Standards im Allgemeinen besser als es zu frühen Zeiten des Webs auf dem Desktop der Fall war. Die wichtigsten mobilen Browser unterstützen auch HTML5 und einige der damit in Verbindung stehenden Spezifikationen. Der Grad der Unterstützung variiert diesbezüglich aber stärker.

Um noch bestehende Unterschiede zwischen Browsern zu überbrücken, kann man gegebenenfalls ein JavaScript-Framework wie Modernizr [31] einsetzen, das ermittelt, welche Funktionen der Browser unterstützt. In Kombination mit JavaScript-Bibliotheken, die fehlende Funktionalitäten soweit möglich nachrüsten, – sogenannten Polyfills – kann der Funktionsumfang der Browser einander angeglichen werden. Für diverse fortgeschrittene HTML5-Funktionen stehen solche Polyfills zur Verfügung. Insgesamt kann von einer guten Unterstützung der Web-Standards auf mobilen Plattformen gesprochen werden. Die Rendering- und JavaScript-Engines der Browser sind hinreichend schnell, um auch umfangreichere Web-Apps verarbeiten zu können.

Web-Apps sind prinzipiell hinsichtlich des Zugriffs auf plattformspezifische Funktionen beschränkt, weil der Browser sie abgeschirmt vom Betriebssystem ausführt. Zwar werden im Rahmen von HTML5 für einige solcher Funktionen Schnittstellen spezifiziert, die generelle, plattformübergreifende Verfügbarkeit ist dennoch häufig nicht gegeben. Des Weiteren befinden sich einige dieser Standardisierungsverfahren noch in einem frühen Stadium (*Working Draft*), z. B. *Web Storage* oder die *File API*. Historisch gesehen setzte PhoneGap (heute Apache Cordova, siehe folgendes Unterkapitel) an dieser Stelle an, um die fehlende Funktionalität zu ergänzen, solange die Spezifikationen nicht die notwendige Verbreitung aufweisen.

Das zweite wichtige Kriterium, das man bei der Entscheidung für mobile Web-Apps untersuchen sollte, ist das der Distribution. Mobile Web-Apps unterscheiden sich signifikant von den anderen Ansätzen in der Art, wie sie an den Nutzer verteilt werden. Nutzer erhalten mobile Web-Apps nicht wie sonstige mobile Apps in Form eines Installationspakets über einen Appstore. Eine Installation ist ebenfalls nicht möglich, auch wenn ein Link in Verbindung mit Offline-Speicherfunktionalitäten ein ähnliches Nutzungserlebnis bezüglich der Verfügbarkeit erschaffen kann. Eine

garantierte Offline-Verfügbarkeit ist aber nur bei auf dem Gerät installierten Anwendungen gewährleistet. Vor diesem Hintergrund sind Web-Apps für den Einsatz in Situationen mit instabiler oder fehlender Netzwerkverbindung nicht unmittelbar geeignet.

Für den Benutzer deutlich erkennbar läuft eine Web-App immer im Browser, was mitunter einen weniger professionellen Eindruck hinterlässt. Die Interviews im Rahmen des zugrundeliegenden Projekts haben gezeigt, dass auch einige Firmen diese Vorbehalte auf Seiten ihrer Nutzer erkennen und eine Verteilung ihrer Apps in der für eine Plattform üblichen Form über die vorgesehenen Kanäle bevorzugen. Auch hier kann ein hybrider Ansatz wie Apache Cordova Abhilfe verschaffen. Auf der anderen Seite hat der unkomplizierte Zugang zu einer Web-App über das Internet ohne Installation auch den Vorteil eines einfacheren Einstiegs. Aktualisierungen können zudem transparent und sofort vorgenommen werden, da keine Änderungen auf dem Client notwendig sind. Dieser ruft automatisch immer die aktuelle Version auf.

Mobile Web-Apps sind entscheidend dadurch charakterisiert, dass ihre Benutzeroberfläche durch den Browser dargestellt wird. Ihre GUI verwendet also die vom Browser bereitgestellten Elemente anstelle nativer Komponenten. Das Aussehen und bis zu einem gewissen Grad auch das Verhalten einer Web-App unterscheiden sich aus diesem Grund initial von nativen Apps. Diesem Umstand kann nur bis zu einem gewissen Grad mit dem Einsatz spezifischer CSS-Regeln begegnet werden, ein Unterschied ist weiterhin erkennbar. Des Weiteren erhöht der Einsatz von plattformspezifischem CSS den Entwicklungs- und Testaufwand.

Hinsichtlich der weiteren Kriterien schneiden mobile Web-Apps nahezu durchgehend positiv ab. Sie unterstützen dank der weiten Verbreitung von Web-Standards alle Plattformen und sind auch langfristig eine zukunftssichere Strategie. Zudem kann man vom Web-App-Ansatz relativ schnell zur hybriden Entwicklung wechseln, sodass Migrationsoptionen zur Verfügung stehen, sollten zusätzliche Anforderungen auftreten. Der Entwicklungsprozess unterscheidet sich nicht wesentlich von dem normaler Web-Anwendungen. Aufgrund des hohen Verbreitungsgrades von HTML, CSS und JavaScript stehen ausgereifte Entwicklungswerkzeuge für GUI und Anwendungslogik zur Verfügung. Der hohe Bekanntheitsgrad der verwendeten Technologien wirkt sich positiv auf den Komfort des Entwicklungsprozesses und die Entwicklungszeiten aus. Zusammengefasst und im Vergleich zum Beispiel zur nativen Entwicklung erlaubt der Web-App-Ansatz eine eingängige, flexible und schnelle Entwicklung.

Die Performance mobiler Web-Anwendungen ist begrenzt durch die zur Verfügung stehende Bandbreite. Dem kann durch intelligentes Caching mit HTML5-Technologien zu einem gewissen Grad begegnet werden. Web-Anwendungen erfordern zunächst keine Lizenzkosten; fortgeschrittene Web-Frameworks sind aber gegebenenfalls nur kommerziell verfügbar. Das Hosting von Webseiten erfordert zusätzliche, dauerhafte Aufwendungen.

Zusammengefasst sind mobile Web-Apps als Einstieg in die mobile Entwicklung zu empfehlen oder für Anwendungen mit beschränktem Funktionsumfang. Sobald plattformspezifische Funktionen benötigt werden oder die Anwendung im nativen Paketformat verteilt werden soll, sind andere Ansätze zu bevorzugen. Ein plattformspezifisches, natives Look & Feel ist mit Web-Apps kaum möglich.

5.4.2 Apache Cordova (PhoneGap)

Apache Cordova [28], auch bekannt unter seinem ursprünglichen Namen PhoneGap, ist eine hybride Plattform zur Erstellung nativer mobiler Apps mit HTML, CSS und JavaScript. Der Begriff *native App* bezeichnet in diesem Zusammenhang eine App, die Zugriff auf native Funktionen hat und als natives Paket auf die Plattformen ausgeliefert wird, deren Benutzeroberfläche aber nicht notwendigerweise aus nativen Komponenten besteht.

Das Projekt PhoneGap wurde 2009 von Nitobi gegründet, um die namensgebende Lücke zwischen Web- und nativen Anwendungen auf mobilen Plattformen zu überbrücken. Neben den ursprünglichen Entwicklern von Nitobi wird die Entwicklung auch von größeren Firmen wie IBM oder Microsoft und einer breiten Entwicklercommunity getragen. 2011 kaufte Adobe Systems Nitobi. Die Entwicklung von PhoneGap wird seitdem unter dem Dach der Apache Foundation und unter dem Namen Apache Cordova weitergeführt. Der Begriff PhoneGap bezeichnet nun eine Distribution des Cordova-Frameworks. Apache Cordova/PhoneGap ist Open Source unter der Apache License, Version 2.0.

Apache Cordova fällt in die Kategorie der hybriden Entwicklungsansätze, die die Browser-Engine der Zielplattform als Hauptbestandteil ihrer Laufzeitumgebung verwenden. Der Quelltext von Anwendungen, die auf Cordova als Cross-Plattform-Tool aufsetzen, besteht wie der gewöhnlicher Web-Apps aus HTML(5)-Dokumenten, CSS und JavaScript und kann auch auf gewöhnliche Web-Frameworks zurückgreifen. Cordova stellt den Anwendungen eine plattformübergreifend einheitliche JavaScript-API zum Zugriff auf Plattform- und Gerätefunktionen zur Verfügung. Der Quelltext und weitere Ressourcen der Anwendung wie Web-Frameworks und Bilder – d. h. der plattformunabhängige, gemeinsame Teil – werden mit der jeweiligen plattformspezifischen Cordova-Umgebung kombiniert, sodass die Anwendung für jede Plattform als natives, eigenständiges Paket zur Verfügung steht. Die Anwendung selbst ist aber plattformunabhängig und hat eine einheitliche Codebasis für alle Plattformen. Die Laufzeitumgebung von Cordova besteht aus einem JavaScript-Teil und einem nativen Part. Abbildung 5.5 stellt idealtypisch die Architektur von Anwendungen dar, die Apache Cordova verwenden.

Beim Start einer Cordova-Anwendung wird vom nativen Teil eine Web-Komponente erstellt, in der die Webseite mitsamt dem JavaScript-Code der Anwendung und der Laufzeitumgebung ausgeführt wird. Cordova nutzt dabei eine von der Platt-

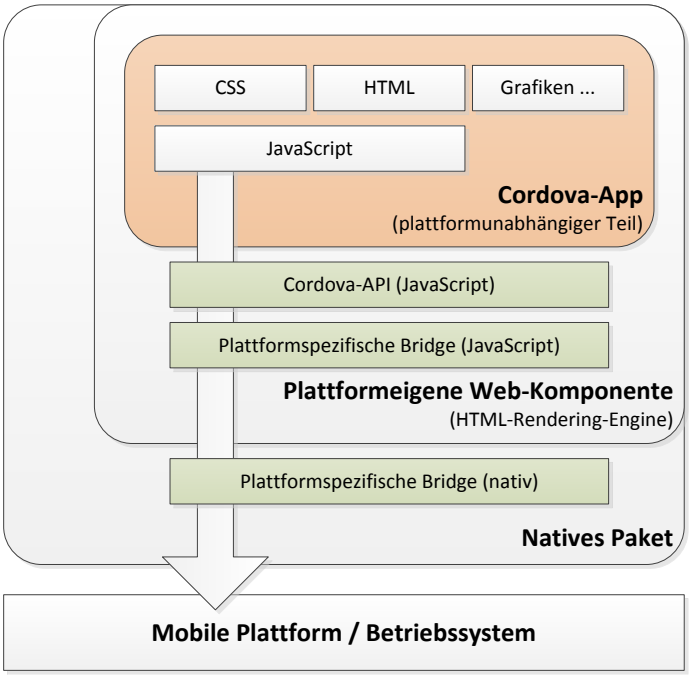

Abbildung 5.5: Architektur von Anwendungen auf Basis von Apache Cordova

form zur Verfügung gestellte Web-Komponente. Aufrufe der Cordova-JavaScript-API werden von einer plattformspezifischen *Bridge* in JavaScript an den nativen Teil der Laufzeitumgebung weitergereicht. Die dazu verwendeten Techniken variieren von Plattform zu Plattform. Die jeweilige nativ implementierte Funktion ruft dann die entsprechende Methode der Plattform auf und gibt das Ergebnis zurück an die Web-Komponente. Des Weiteren reicht der native Wrapper Plattformereignisse an die Web-Komponente weiter, wo sie in JavaScript an registrierte Event-Handler übermittelt werden. Über die Ereignisbehandlung von Cordova können sich Anwendungen zum Beispiel benachrichtigen lassen, wenn der Nutzer einen der Hardware-Buttons drückt, sich der Ladezustand der Batterie ändert oder das Gerät sich mit einem Netzwerk verbindet.

Die API von Apache Cordova bietet einheitlichen Zugriff auf folgende Gerätefunktionen: Geolokation, Kamera, Aufnahme von Audio und Video, Beschleunigungssensor (Accelerometer), Kompass. Darüber hinaus enthält die API-Methoden für die folgenden Plattformfunktionen, die zunächst unabhängig von der Geräte-Hardware sind: lokale Speicherung nach W3C-Standards (Web Storage, Web SQL Database), Zugriff auf das Dateisystem und die Kontakte, Benachrichtigungen (inkl. Vibration), Netzwerkverbindung, Abspielen von Audiomedien. Apache Cordova unterstützt nahezu alle wichtigen mobilen Plattformen. Der Umfang der unterstützen Funktionen schwankt allerdings von Plattform zu Plattform, nicht alle prinzipiell von Cordo-

va angebotenen Funktionen sind auf allen Plattformen verfügbar. Die vollständige API wird auf iOS, Android und Windows Phone 7 unterstützt. Neuere BlackBerry-Versionen erfahren eine nahezu vollständige Unterstützung, während auf Symbian, Bada und WebOS – in dieser Reihenfolge – Einschränkungen, insbesondere hinsichtlich des Dateisystems und der lokalen Speicherung, hingenommen werden müssen. Ältere Versionen der erwähnten Plattformen sind gegebenenfalls weitergehenden Einschränkungen unterworfen.

Cordova bietet eine Plugin-Architektur, um weitere Funktionen nachzurüsten, die auf nativen Code zurückgreifen. Ein Plugin kann auf fortgeschrittene Plattformfunktionen zugreifen, die nicht von Cordova selbst unterstützt werden, oder umfangreichere und algorithmisch anspruchsvolle Programmteile kapseln, deren Implementierung in JavaScript zu langsam wäre. Ein Cordova-Plugin besteht aus dem nativen Code und einem JavaScript-Teil. Der JavaScript-Teil stellt den in JavaScript implementierten Apps die API zum Zugriff auf die Funktionen des Plugins bereit und ruft den nativen Code mit Hilfe entsprechender Schnittstellen auf, die Cordova bereitstellt. Die Kombination aus JavaScript- und nativem Code ist für jede Plattform zu implementieren, die durch das Plugin unterstützt werden soll. Der JavaScript-Teil wird dabei jeweils über alle Plattformen hinweg dieselbe Schnittstelle nach außen anbieten, während die interne Umsetzung in JavaScript und nativem Code von Plattform zu Plattform differiert.

Die einheitliche API von Cordova muss die Unterschiede zwischen den Plattformen berücksichtigen und eine gemeinsame Abstraktionsebene über alle Plattformen finden. Dementsprechend besteht prinzipiell eine Tendenz zum kleinsten gemeinsamen Nenner. In der Praxis sind die Funktionen von Cordova aber in den meisten Fällen ausreichend. Naturgemäß stehen nicht alle Feinheiten in dem Maße zur Verfügung, wie sie vom Plattform-SDK bereitgestellt werden. Bei der Gestaltung der API orientiert sich Apache Cordova, wo möglich, an existierenden Spezifikationen der W3C. Das gilt zum Beispiel für die Geolokation und die Storage API. Dieser Ansatz entspricht Cordovas Projektziel, keine vollständig neue Plattform aufzubauen, sondern die momentan existierende Lücke zwischen den Anforderungen an mobile Apps und den Möglichkeiten von HTML5 zu überbrücken. Wenn eine Plattform – genauer: die plattformeigene Web-Komponente – eine entsprechende Spezifikation schon umsetzt, implementiert Cordova diese nicht neu, sondern lässt die von der Plattform bereitgestellte Implementierung intakt. Da sich die Cordova-API unter Anderem hinsichtlich der Benennung und Strukturierung von Funktionen an die Spezifikationen hält, geschieht dies für den Entwickler transparent.

Das Konzept der hybriden Cross-Plattform-Entwicklung im Allgemeinen und Apache Cordova als Realisierung dieses Konzepts im Speziellen vereinen die Vorteile mobiler Web-Apps mit der Möglichkeit, auf native Funktionen zuzugreifen, sowie mit einem nativen Distributionsformat. Mit Ausnahme dieser beiden Kriterien gelten die Beurteilungen des vorherigen Unterkapitels zu Web-Apps. Dank der breiten Un-

terstützung der wichtigsten Gerätefunktionen mobiler Plattformen lassen sich mit Cordova auch fortgeschrittene Szenarien als mobile App umsetzen. Zudem können Cordova-basierte Apps über die üblichen Distributionskanäle der mobilen Plattformen verteilt und installiert werden, da sie im nativen Format gepackt werden. Dieser Vorzug macht Apache Cordova auch dann als Alternative zu Web-Apps interessant, wenn keine plattformspezifischen Funktionen benötigt werden, aber eine installierbare App erforderlich ist. Eine auf dem Gerät installierte App hat auch den Vorteil, dass keine Internetverbindung zum Aufruf benötigt wird, sofern alle Einzelseiten der Anwendung in der Paketdatei enthalten sind. Für die Distribution über die entsprechenden Appstores sind die einschlägigen Konditionen zu beachten.

Zwei der größten Kritikpunkte an mobilen Web-Apps entfallen demnach bei hybriden Apps. Bei den weiteren Kriterien ist, mit Ausnahme einer allgemein leicht gestiegenen Komplexität, mindestens keine Verschlechterung festzustellen, da die verwendeten Technologien deckungsgleich zum Web-Apps-Ansatz sind. Da hybride Apps mit Cordova nicht im Browser laufen, sondern nur die Web-Komponente der Plattform einbetten, sind sie für die Nutzer nicht mehr anhand dieses Merkmals als Web-Anwendungen zu erkennen. Es werden keine Steuerelemente des Browsers wie Adresszeile oder Menü angezeigt. Allerdings basiert die Benutzeroberfläche genau wie bei Web-Apps auf HTML und CSS, sodass das Look & Feel ebenso Unterschiede zu einer nativen Oberfläche aufweist. Aufgrund der erweiterten Interaktionsmöglichkeiten mit der Plattform und dem Nutzer ist es möglich, das Verhalten einer Cordova-App rudimentär ähnlicher zu nativen Apps zu gestalten als das mit Web-Apps möglich ist.

Der Entwicklungsprozess für Cordova-basierte Apps ähnelt dem von Web-Apps. Die Cordova-API erfordert nur eine kurze Einarbeitungszeit. Die plattformspezfische Paketierung sorgt für eine geringe zusätzliche Komplexität des Build-Prozesses. Die Verwendung plattformspezifischer Funktionen erhöht naturgemäß den Testaufwand. Im Übrigen gilt die Evaluierung von Web-Apps bezüglich der entwicklungsbezogenen Kriterien. Alle wichtigen Plattformen werden hinreichend unterstützt. Die Performance ist besser als bei Web-Apps, da die Übertragungszeiten entfallen und ansonsten die gleiche Rendering-Engine genutzt wird. Die Zukunftssicherheit sollte dank der breiten Unterstützung für Apache Cordova nach gegenwärtigem Standpunkt gesichert sein. Nichtsdestotrotz muss man die weitere Entwicklung aufmerksam verfolgen, da eine gewisse Abhängigkeit besteht. Wie erwähnt bemüht sich Cordova allerdings um standardisierte Schnittstellen, die die Transition zu reinem HTML5 in der Zukunft vereinfachen würden. Lizenzkosten fallen aufgrund der Open-Source-Natur des Projekts nicht an. Für die Verteilung der App über die plattformeigenen Marktplätze muss man zumeist dem entsprechenden Entwicklerprogramm kostenpflichtig als Mitglied beitreten.

Zusammengefasst ist Apache Cordova zu empfehlen, wenn mindestens einer der Vorteile gegenüber Web-Apps von Bedeutung ist: der Zugriff auf plattformspezifi-

sche Funktionen oder das native Distributionsformat. Falls ein natives Look & Feel eine unerlässliche Anforderung ist, sollten hingegen andere Alternativen in Betracht gezogen werden. Bei derartig strikten Anforderungen muss gegebenenfalls für jede Plattform nativ entwickelt werden.

5.4.3 Appcelerator Titanium Mobile

Titanium Mobile ist ein Cross-Plattform-Framework der Firma Appcelerator [32]. Es ist der Klasse der Frameworks zuzurechnen, die auf einer eigenständigen, frameworkspezifischen Laufzeitumgebung basieren. Titanium folgt dem Prinzip, eine JavaScript-Anwendung zur Laufzeit zu interpretieren. Die Apps haben ein dezidiert natives Aussehen und Verhalten. Mit Titanium Mobile können primär Apps für iOS und Android erstellt werden.

Entwickler implementieren Titanium-basierte Anwendungen vollständig in JavaScript, inklusive der grafischen Benutzeroberfläche. Dazu stellt Titanium JavaScript-Funktionen zur programmatischen Erstellung der GUI und zum Zugriff auf plattformspezifische Funktionen bereit. Der JavaScript-Quelltext der Anwendung wird dann zusammen mit der Titanium-Engine, die prinzipiell ein JavaScript-Interpreter ist, im nativen Deployment-Format gepackt. Die Titanium-Engine ist plattformspezifisch und führt zur Laufzeit den JavaScript-Code der App aus. Dabei sorgt sie als Brücke zur nativen API der Plattform dafür, dass die korrespondierenden nativen Funktionen aufgerufen werden. Auf diese Weise erstellt die Engine die grafische Oberfläche der App vollständig mit nativen Komponenten anhand der programmatischen Beschreibung in JavaScript.

Titanium unterstützt derzeit die Entwicklung von Apps für iOS und Android. Mit Version 2.0 wurde zudem die Transformation der Titanium-Apps in mobile Web-Apps ermöglicht, mit denen andere Plattformen abgedeckt werden können. Die Vorteile von Titanium bezüglich Look & Feel sind bei solchen Web-Apps aber nicht mehr gegeben. Eine native Unterstützung von BlackBerry ist seit längerer Zeit in der Beta-Phase. Es ist unklar, ob mit einer baldigen BlackBerry-Unterstützung gerechnet werden kann.

Die Basisversion von Titanium Mobile ist als Open Source unter der Apache License, Version 2.0, verfügbar. Fortgeschrittene Versionen, die unter anderem Debugging-Unterstützung enthalten, erfordern Support-Verträge von Appcelerator. Die Weiterentwicklung von Titanium Mobile ist nahezu vollständig in der Hand von Appcelerator.

Der wesentliche Vorteil von Titanium Mobile gegenüber den zuvor betrachteten Ansätzen ist das native Look & Feel der damit erstellten Anwendungen. Da die Titanium-Engine die GUI aus nativen Komponenten erstellt, ist diese inhärent nativ. Die Unterstützung von Gerätefunktionen ist ebenfalls gut. Apps werden im nativen Deployment-Format bereitgestellt. Nachteilig ist die geringe Zahl unterstützter

Plattformen. Auch wenn die beiden wichtigsten unterstützt werden und Apps in rudimentärer Form als Web-Apps für die anderen Plattformen bereitgestellt werden können, ist dies je nach Einsatzszenario eine gewichtige Einschränkung.

Titanium Mobile ist als System erkennbar weniger offen und diversifiziert als andere Ansätze, bedingt durch die starke Abhängigkeit von Appcelerator und deren Geschäftsmodell. Neben höheren Lizenzkosten im Fall von erweiterten Anforderungen steigert das auch das Lock-In-Risiko. Bei Verwendung von Titanium bindet man sich stark an ein einzelnes Unternehmen und es fallen gegebenenfalls hohe Kosten für den Wechsel der Plattform an. Verstärkt wird dieser Effekt durch die proprietäre Titanium-API, auf die Titanium-basierte Anwendungen in hohem Maße zurückgreifen. Der Umstieg auf eine andere Cross-Plattform-Lösung erfordert zumeist eine vollständige Neu-Implementierung der Anwendung. Obwohl Titanium im Rahmen des Build-Vorgangs der nativen Pakete das JavaScript optimiert, um die Ausführungsgeschwindigkeit zu steigern, erschienen Titanium-Anwendungen in Praxistests mit vielen Bedienelementen als weniger performant.

Ein wesentlicher Kritikpunkt an Titanium ist der umständliche Entwicklungsprozess. Neben fehlender Werkzeugunterstützung in der Basis-Version, z. B. beim Debugging, ist dies vor allem der umständlichen GUI-Beschreibung in JavaScript geschuldet. Eine deklarative Beschreibung wie in HTML ist bei komplexeren Oberflächen oftmals angenehmer, insbesondere wenn kein GUI-Builder zur Verfügung steht, der den umfangreichen, aber zumeist simplen Quelltext zum programmatischen Aufbau der GUI automatisch erstellt. Des Weiteren ist der Einarbeitungsaufwand in die umfangreiche Titanium-API nicht zu unterschätzen. Titanium vertritt zudem die Philosophie, nur für den Kern der Plattformaspekte eine einheitliche API bereitzustellen. Für fortgeschrittene Elemente existieren nur jeweils iPhone-, iPad- und Android-spezifische APIs, sodass bei Verwendung solcher Elemente keine plattformübergreifend identischer Quelltext mehr möglich ist, bzw. bedingte, plattformspezifische Blöcke im Quelltext nötig sind.

Insgesamt erfordert die Entwicklung mit Titanium mehr Aufwand als mit anderen Ansätzen. Dem steht als wesentlicher Vorteil nur das native Look & Feel gegenüber. Allerdings sollte für den Fall, dass natives Look & Feel eine strikte Anforderung ist, ebenfalls in Betracht gezogen werden, die mobile Anwendung separat nativ für jede Plattform zu entwickeln.

Nicht alle der erwähnten Kritikpunkte gelten für Cross-Plattform-Entwicklungsansätze mit eigenständiger Laufzeitumgebung im Allgemeinen. Vom konkreten Framework abhängig ist zum Beispiel der Komfort des Entwicklungsprozesses oder die prinzipielle Offenheit und damit einhergehende Kosten. Hinsichtlich dieser Punkte ist es denkbar, dass andere Frameworks, die einen ähnlichen Ansatz wie Titanium verfolgen, eine bessere Unterstützung bieten. Die Kritik hinsichtlich mangelnder Migrationsoptionen wird hingegen zumeist auch auf andere Laufzeitumgebungen zutreffen, da entsprechende Apps spezifisch für die jeweilige eigenständige Lauf-

zeitumgebung implementiert werden. Cross-Plattform-Lösungen wie PhoneGap, die stattdessen auf eine Web-Engine oder eine ähnlich standardisierte Umgebung zurückgreifen, ermöglichen es hingegen, Anwendungen zu implementieren, die leichter auf andere Ansätze migriert werden können. Die Entwicklung einer eigenständigen Laufzeitumgebung wird zudem immer aufwendiger sein als eine vorhandene für eigene Zwecke zu nutzen. Cross-Plattform-Frameworks mit eigenständiger Laufzeitumgebung sind deshalb tendenziell im Nachteil was Funktionsumfang, Qualität und/oder Geschäftsmodell betrifft. Dem steht der Vorteil gegenüber, die Laufzeitumgebung vollständig nach eigenen Vorstellungen gestalten zu können.

5.5 Vorläufige Handlungsempfehlungen

Im Rahmen des Projekts wurden verschiedene Cross-Plattform-Frameworks evaluiert, die zusammen alle in Abschnitt 5.2 beschriebenen Kategorien abdecken. Aufbauend auf dieser Evaluation können einige Handlungsempfehlungen zur Entwicklung von Anwendungen für mehrere Plattformen gegeben werden. Diese Handlungsempfehlungen sind nicht als allgemeingültig zu verstehen, da die Auswahlentscheidung immer von den Anforderungen im Einzelfall abhängig gemacht werden muss. Deshalb orientieren sich die folgenden Empfehlungen auch an typischen Anforderungen und zeigen jeweils auf, wann ein bestimmter Ansatz zu bevorzugen ist.

Apache Cordova (PhoneGap) hinterließ einen positiven Eindruck. Die leichtgewichtige Laufzeitumgebung unter Rückgriff auf Web-Technologien ermöglicht die Umsetzung von Anwendungen mit fortgeschrittenen Anforderungen, auch hinsichtlich des Zugriffs auf plattformspezifische Funktionen. Zugleich sorgt das Konzept von Cordova für geringen Overhead und minimale zusätzliche Komplexität. Auf Cordova basierende Apps zu entwickeln ist nicht wesentlich aufwendiger als die Entwicklung von mobilen Web-Apps. Cordova-Apps verwenden bekannte Web-Technologien, bieten aber in diesem Rahmen auch Zugriff auf Plattformfunktionen. Gegenüber reinen Web-Apps hat Cordova hauptsächlich die Vorteile, dass Apps auf plattformspezifische Funktionen zugreifen können und im nativen Format gepackt werden. Bezüglich der Distribution und Installation über die plattformeigenen Mechanismen ist zu beachten, dass dazu eine – gegebenenfalls kostenpflichtige – Mitgliedschaft in den jeweiligen Entwicklerprogrammen notwendig ist und dass man die Geschäftsbedingungen der entsprechenden Anbieter akzeptieren und befolgen muss. Mobile Web-Apps, die über das Internet aufrufbar sind, können den plattformeigenen Distributionskanal nicht nutzen – mit allen damit verbundenen Nachteilen –, sind aber im Gegenzug nicht diesen Bedingungen unterworfen. Nichtsdestotrotz und aufgrund der genannten Vorteile empfehlen wir die Verwendung von Apache Cordova, sofern die Anforderungen des jeweiligen Projekts dem nicht entgegenstehen.

Mit zunehmendem Fortschritt bei Web-Technologien sind auch reine Web-Apps eine vielversprechende Option, momentan aber noch beim Zugriff auf Plattformfunk-

tionen und bei der Verfügbarkeit eingeschränkt. Mobile Web-Apps sind als Einstieg oder für einfache Szenarien zu empfehlen. Ebenfalls zu überlegen ist es, Web-Apps als Erweiterung bestehender Webseiten zu nutzen, wenn diese mobil verfügbar gemacht werden sollen, dafür aber in der ursprünglichen Fassung wenig geeignet sind.

Eine Anforderung, die mit den Ansätzen mobiler Web-Apps beziehungsweise hybrider Apps nicht vereinbar ist, ist ein natives Look & Feel. Da beide Ansätze auf Web-Technologien basieren und ihre Benutzeroberfläche im Prinzip aus Webseiten besteht, ist es kaum möglich, das Aussehen plattformeigener GUIs nachzubilden oder deren Verhalten zu imitieren. Alternative Cross-Plattform-Tools, die diese Anforderung umsetzen, sind aktuell nur vereinzelt zu finden. Ein prominentes Beispiel ist Appcelerator Titanium Mobile. Prinzipiell dazu geeignet, ein natives Look & Feel durch Verwendung plattformeigener GUI-Komponenten zu erzeugen, sind Cross-Plattform-Ansätze, die eine eigenständige Laufzeitumgebung einsetzen, oder solche, die plattformspezifischen Code generieren, zum Beispiel modellgetrieben. In der gegenwärtigen Situation kann aber kein entsprechendes Framework im Hinblick auf den Entwicklungsprozess uneingeschränkt empfohlen werden (siehe das folgende Kapitel). Bezüglich der Anforderung eines nativen Look & Feels als solche ist festzustellen, dass diese bei speziellen Zielgruppen oder besonderen Einsatzszenarien nachvollziehbar ist. Zugleich kann aber konstatiert werden, dass ein natives Look & Feel zunächst kein Wert an sich ist – auch Web-Oberflächen sehen auf mobilen Geräten prinzipiell *vernünftig* aus und lassen sich bei entsprechender Sorgfalt komfortabel bedienen. Aussehen und Verhalten unterscheiden sich zwar vom gewohnten nativen Pendant, sind aber nicht per se *schlechter*.

Falls für bestimmte Plattformen ein natives Look & Feel gefordert wird und für andere ein Web-basiertes Layout akzeptabel ist, ist eine weitere Option, nur für erstere die Anwendung nativ zu entwickeln und für die letzteren Plattformen auf Cordova zurückzugreifen.

Zum Abschluss ist festzuhalten, dass das Gebiet der Cross-Plattform-Entwicklungsframeworks noch starken Veränderungen und großen Fluktuationen unterworfen ist. Regelmäßig erscheinen neue Frameworks, während bei existierenden Tools ein rascher Fortschritt festzustellen ist. Ebenso können etablierte Frameworks zurückfallen, wenn sie mit den schnellen Entwicklungen der mobilen Plattformen nicht Schritt halten. Die hier ausgesprochenen Empfehlungen sind demzufolge vorläufig und bedürfen der andauernden Reflektion, um mit der Entwicklung auf dem Markt der Cross-Plattform-Tools Schritt zu halten. Nichtsdestotrotz sind bezüglich der Vor- und Nachteile der Ansätze im Allgemeinen, unabhängig von ihrer Umsetzung in konkreten Frameworks, längerfristige Aussagen möglich. Solange die systemnäheren Funktionen von HTML5 noch keine umfassende Verbreitung gefunden haben, ist der Ansatz hybrider Entwicklung vielversprechend. In bestimmten, einfachen Anwendungsfällen sind mobile Web-Apps auch heute schon einsetzbar. Soll die App nativen Anwendungen ähneln, ist zum gegenwärtigen Zeitpunkt die native Entwicklung

in den meisten Fällen die beste Alternative. Das folgende Kapitel stellt generierende Ansätze vor, die in Zukunft möglicherweise eine Alternative sein könnten.

5.6 Modellgetriebene App-Entwicklung mit MD²

Generierende Ansätze haben das Potenzial, Cross-Plattform-Entwicklung mit nativem Look & Feel zu verbinden. Sie generieren aus einer gemeinsamen Codebasis nativen Quelltext für jede Plattform, der direkt auf die native API zugreift. Im Prinzip können so Apps entstehen, die aus Benutzersicht nicht von solchen zu unterscheiden sind, die im Rahmen nativer Entwicklung entstanden. Die Evaluation bestehender Frameworks hat jedoch gezeigt, dass generative Ansätze zum jetzigen Zeitpunkt noch nicht ausgereift genug sind, um produktiv eingesetzt werden zu können. Weder in der Entwicklung befindliche *modellgetriebene Cross-Plattform-Frameworks* noch Cross-Compiler haben derzeit einen Status erreicht, mit dem reale Anwendungen umgesetzt werden könnten. Dieses Kapitel stellt deshalb MD² vor, einen Spin-Off des Projekts „Business Apps“. Als modellgetriebenes Framework für Android- und iOS-Apps wird es seit März 2012 am Lehrstuhl für Praktische Informatik entwickelt. Zuvor sollen aber zwei Beispiele das Potenzial, aber auch die Mängel bestehender generierender Tools aufzeigen: das modellgetriebene *applause* und der Cross-Compiler *XMLVM*.

Applause [29] wurde von der itemis AG zunächst entwickelt, um auf Basis von textuellen Modellen Anwendungen für das iPhone zu entwickeln. Später wurde der Fokus auf Android und in Ansätzen Windows Phone ausgeweitet. Es handelt sich aber noch um einen Prototypen, der ausschließlich die Darstellung von Informationen in Listen und Detailansichten unterstützt. Über die Navigation hinausgehende Benutzerinteraktionen können bisher nicht umgesetzt werden.

XMLVM [30] als Beispiel für einen Cross-Compiler, der in Java geschriebene Anwendungen zur Kompilierungszeit in native Android- und iOS-Anwendungen übersetzt, ist momentan in einem sehr frühen Stadium und eher als Forschungsprototyp zu sehen. An einen produktiven Einsatz ist nicht zu denken, teilweise funktionieren nicht einmal die einfachsten Beispiele. Googles J2ObjC [33] übersetzt ebenfalls Java-Quelltext nach Objective-C, bietet aber keinerlei Unterstützung für die API von Android und iOS, sodass nur die geräteunabhängige Geschäftslogik transformiert werden kann.

5.6.1 Überblick über MD²

Die Defizite der bestehenden Frameworks und das generelle Potenzial modellgetriebener Ansätze motivierten die Entwicklung von MD². MD² ist ein modellgetriebenes Framework zur Erstellung von datengetriebenen Business Apps für Android-

und iOS-Tablets. Zentrales Element von MD^2 ist eine domänenspezifische Sprache (DSL), in der App-Entwickler die gewünschte Anwendung deklarativ auf einem hohen Abstraktionsniveau beschreiben. Dazu bietet die DSL Komponenten zur Beschreibung des Datenmodells und der grafischen Oberfläche. Beides wird durch eine Controller-Komponente nach dem Model-View-Controller-Muster (MVC) verbunden, die Geschäftslogik, Ereignisbehandlung und Navigationspfade spezifiziert. MD^{2}s Code-Generator transformiert das derart beschriebene textuelle Modell einer App schließlich in Android- und iOS-Projekte, die ohne Änderungen kompilierbar sind und direkt auf den Plattformen ausgeführt werden können.

MD^2 wird fortlaufend weiterentwickelt, kann aber schon jetzt eingesetzt werden, um einfache, formularbasierte Apps für Android und iOS gemeinsam zu entwickeln. Die kompakte DSL erlaubt dabei, komplexe Funktionalität prägnant zu beschreiben. Der in MD^{2}s DSL geschriebene Quelltext ist typischerweise um Größenordnungen kleiner als der Quelltext der generierten Android- und iOS-Apps. Zudem unterstützt MD^2 die Anbindung der Apps an ein Server-Backend über eine spezifizierte API.

Die folgenden Unterkapitel geben einen Einblick in die einzelnen Komponenten der Sprache von MD^2, beginnend mit dem Datenmodell (Kapitel 5.6.2) über die Beschreibungssprache für die grafische Oberfläche (Kapitel 5.6.3) hin zur Kontrollkomponente (Kapitel 5.6.4). Detailliertere Informationen zu MD^2 können auch zwei wissenschaftlichen Beiträgen der Autoren entnommen werden, die das Framework [HMK13a] und die DSL [HMK13b] vorstellen. Letzterer Beitrag bildet die Grundlage für die folgende Beschreibung der Sprache. Das durchgehende Beispiel der folgenden Unterkapitel beschreibt eine einfach gehaltene App zur Produktsuche und Bestellung und wird ebenfalls in dem genannten Beitrag [HMK13b] verwendet. Die Abbildungen 5.6–5.9 enthalten das textuelle Modell der App in der DSL von MD^2, aufgeteilt in die drei Sprachkomponenten *Daten*, *GUI* und *Kontrolllogik*. Die Screenshots aus Abbildung 5.8 vermitteln einen Eindruck von Aussehen und Funktionsweise der Android- und iOS-Apps, die von MD^2 aus dem Modell automatisch generiert wurden. Der App-Nutzer gibt in einer ersten Ansicht den Namen eines Produkts an, zu dem die App dann Detailinformationen auf einem Server sucht und in einer zweiten Ansicht darstellt. Dort kann der Nutzer seine E-Mail-Adresse eingeben und die Bestellung abschließen.

5.6.2 Die Datenkomponente von MD^2

Die Datenkomponente der domänenspezifischen Sprache von MD^2 folgt den Standards der Datenmodellierung hinsichtlich Begriffen und Struktur. Komplexe Datentypen werden als *Entity* definiert. Sie werden näher beschrieben durch ihre Eigenschaften, die entweder einen typischen primitiven Datentyp wie `string` oder `integer` aufweisen oder andere Entitätstypen referenzieren. Eigenschaften können als *optional* oder *mehrwertig* definiert werden, zudem sind weitergehende Einschrän-

```
1 package de.md2.bestellung.models
2 entity Produkt {
3   name : string
4   preis : integer
5   beschreibung : string(optional)
6 }
7 entity Bestellung {
8   produkt : Produkt
9   email : string
10 }
```

Abbildung 5.6: Datenmodell einer App zur Produktsuche und Bestellung in MD²

kungen ihres Wertebereichs möglich.

Das textuelle Datenmodell in Abbildung 5.6 definiert zwei komplexe Datentypen, Produkt und Bestellung, mit typischen, wenn auch vereinfachten Eigenschaften. Eine Bestellung verweist auf genau ein Produkt. Zusätzlich zu Entitäten unterstützt die Datenkomponente von MD² Enumerationen. Außerdem können alle Bezeichner im Datenmodell mit aussagekräftigen Namen und Beschreibungen versehen werden.

5.6.3 Die GUI-Komponente von MD²

Ebenso wie die übrigen Komponenten der DSL von MD² ist auch die GUI-Beschreibungssprache deklarativ. Der App-Entwickler beschreibt die grafische Oberfläche der App, indem er Inhaltselemente wie Eingabefelder, Buttons, Labels und Container verschachtelt anordnet. Container (*Panes*) stehen dabei für unterschiedliche Layouts der in ihnen enthaltenen Elemente und können ineinander geschachtelt werden. MD² stellt typische Layouts wie eine horizontale bzw. vertikale Anordnung (*FlowLayoutPane*) und eine tabellenartige Struktur (*GridLayoutPane*) bereit. *TabbedPanes* repräsentieren eine Registernavigation, wie sie für Apps typisch ist: Die Kindelemente eines solchen Container werden als einzelne Registerkarten (*Tabs*) dargestellt, die über eine gemeinsame Navigationsleiste angesteuert werden können. Die konkrete Gestaltung ist plattformabhängig: In Android findet sich die Leiste am oberen Bildschirmrand, während sie in iOS unten platziert ist.

Die Elemente der Oberfläche können im Modell an einer Stelle definiert und an anderer Stelle verwendet werden, um die Übersichtlichkeit zu erhöhen und Wiederverwendung zu ermöglichen. Um standardisierte Oberflächen für ein gegebenes Datenmodell schnell erstellen zu können, bietet MD² das Konzept der *Auto-Generatoren*. Diese stehen für eine standardmäßige Darstellung eines komplexen Datentyps, bestehend aus Labels und Eingabefeldern für die Eigenschaften des Datentyps.

Das Beispielmodell in Abbildung 5.7 definiert eine Registernavigation als die zentrale Ansicht der Bestell-App. Screenshots der daraus generierten Benutzeroberfläche sind in Abbildung 5.8 zu sehen. Die TabbedPane *AppFenster* (Zeile 2–6) besteht

```
1 package de.md2.bestellung.views
2 TabbedPane AppFenster {
3   SuchenTab -> Suchen
4   BestellenTab -> Bestellen(tabTitle "Bestellung")
5   InfoTab(tabTitle "Info")
6 }
7 FlowLayoutPane SuchenTab(vertical) {
8   Label sucheLbl { text "Suche nach Produkt" style Gross }
9   TextInput suchFeld {
10    label "Produktname"
11    tooltip "Geben Sie den Namen des Produkts ein, ..."
12  }
13  Button sucheBtn ("Suchen")
14 }
15 FlowLayoutPane BestellenTab(vertical) {
16  Label bestellenLbl { text "Bestellung aufgeben" style Gross }
17  Label info("Bitte geben Sie Ihre E-Mail-Adresse an, ...")
18  AutoGenerator bestellung { contentProvider bestellungProvider }
19  Button bestellenBtn ("Bestellen")
20 }
21 FlowLayoutPane InfoTab { ... }
22 style Gross { fontSize 20 textStyle bold }
```

Abbildung 5.7: Modell der Benutzerschnittstelle der Beispiel-App in MD^2

aus drei Tabs, die im Folgenden separat definiert werden. Die erste Registerkarte, *SuchenTab* (Zeile 7–14), ist als FlowLayout gekennzeichnet und enthält Label, Eingabefeld und Button als Inhaltselemente. Die zweite Registerkarte, *BestellenTab* (Zeile 15–20), enthält neben konkreten Inhaltselementen einen Auto-Generator für den Entitätstyp Bestellung.

Das Beispiel illustriert außerdem einige fortgeschrittene Funktionen der GUI-Komponente. So ist es möglich, Elementen einen eigenen Stil zuzuweisen (Zeile 8) oder Tooltips zu definieren (Zeile 11). Darüber hinaus unterstützt MD^2 auch die Einbindung von Grafiken und verschiedene Arten von Eingabefeldern. Der Typ eines Eingabefelds kann dabei aus dem Typ der mit ihm verknüpften Eigenschaft einer Entität abgeleitet werden (siehe das folgende Kapitel), z. B. als Datumseingabefeld.

5.6.4 Die Kontrollkomponente von MD^2

Die Kontrollkomponente der MD^2-DSL dient dazu, das dynamische Verhalten einer App zu spezifizieren und die Daten- und GUI-Komponente zu verbinden. Zugleich werden hier allgemeine Eigenschaften der modellierten App festgelegt. Wesentliches Element der Kontrollebene von MD^2 sind Aktionen. Eine Aktion kombiniert verschiedene Teilaktionen und Anweisungen zur gemeinsamen Ausführung. Aktionen können – wiederum durch eine Anweisung in einer Aktion – als Behandler für

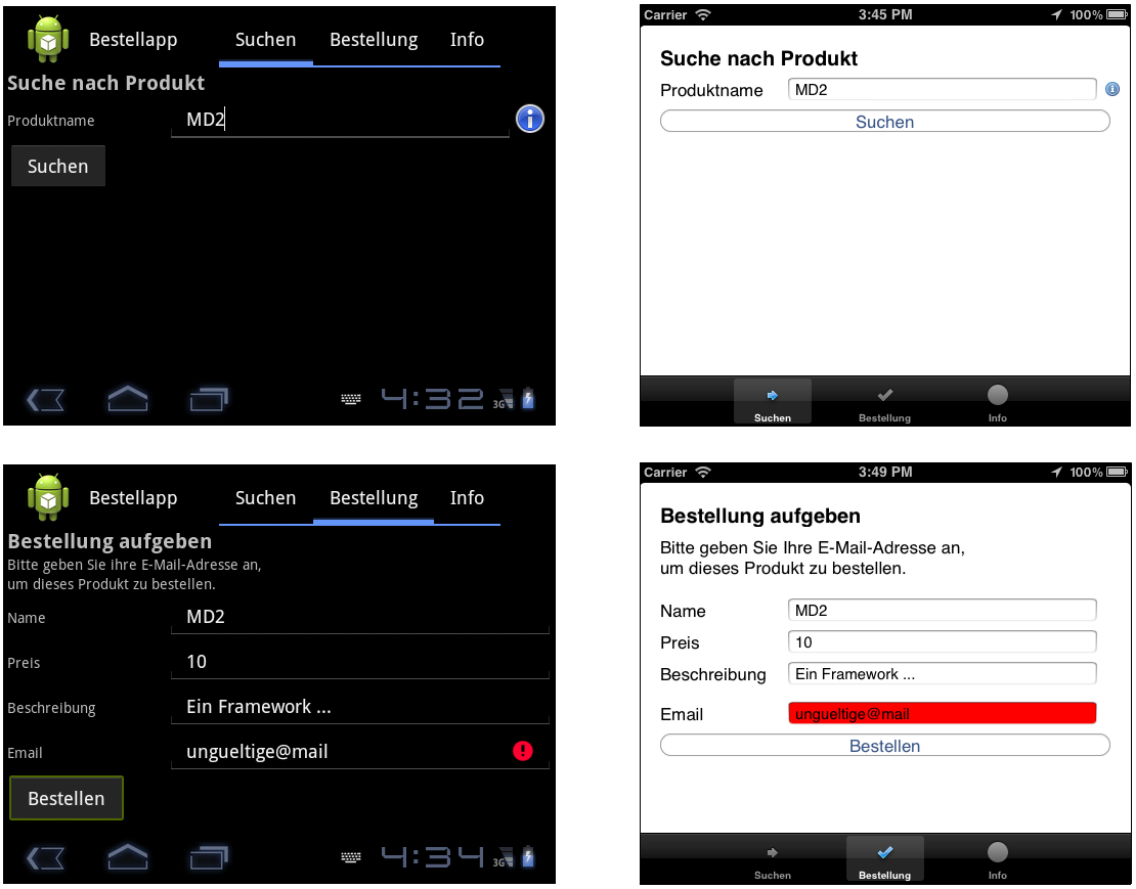

Abbildung 5.8: Screenshots der Beispiel-App auf Android (links) und iOS (rechts). Um Platz zu sparen, wurden die Screenshots gegenüber der Originaldarstellung bearbeitet und komprimiert.

bestimmte Ereignisse, z. B. zur Reaktion auf Benutzerinteraktionen oder Zustandsveränderungen, registriert werden. Im Beispiel (Abbildung 5.9) nimmt die Aktion `ereignisseRegistrieren` (Zeile 10–14) die Registrierung vor: Zwei weitere Aktionen (`produktLaden` und `bestellungAufgeben`) werden an die entsprechenden Buttons gebunden.

MD² stellt neben der Registrierung von Ereignisbehandlern eine Vielzahl weiterer Anweisungen zur Verfügung, die innerhalb von Aktionen eingesetzt werden können. Dazu gehören z. B. das Verknüpfen von Eingabefeldern mit Validatoren und ihre Anbindung an Datenobjekte im Sinne einer bidirektionalen Konsistenz, sowie der Aufruf gerätespezifischer Funktionen wie GPS. Im Beispiel werden derartige Anweisungen – mit Ausnahme eines Validators für die E-Mail-Adresse (Zeile 16) – nicht benötigt, da der zur Beschreibung der GUI verwendete Autogenerator automatisch entsprechende Datenbindungen und Validatoren vorsieht. Die Aktion `produktLaden`

```
1  package de.md2.bestellung.controllers
2  main {
3    appName "Bestellapp" appVersion "1.0" modelVersion "1.0"
4    startView AppFenster.Suchen
5    onInitialized init
6  }
7  action CombinedAction init {
8    actions ereignisseRegistrieren validatorenBinden
9  }
10 action CustomAction ereignisseRegistrieren {
11   bind action produktLaden on SuchenTab.sucheBtn.onTouch
12   bind action bestellungAufgeben
13     on BestellenTab.bestellenBtn.onTouch
14 }
15 action CustomAction validatorenBinden {
16   bind validator RegExValidator(regEx "...")
17     on BestellenTab.bestellung[Bestellung.email]
18 }
19 action CustomAction produktLaden {
20   call DataAction(load suchergebnis)
21   call NewObjectAction(bestellungProvider)
22   call AssignObjectAction
23         (use suchergebnis for bestellungProvider.produkt)
24   call GotoViewAction(AppFenster.Bestellen)
25 }
26 action CustomAction bestellungAufgeben {
27   call DataAction(save bestellungProvider)
28 }
29 contentProvider Produkt suchergebnis {
30   providerType backend
31   filter first where
32       Produkt.name equals AppFenster.Suchen->SuchenTab.suchFeld
33 }
34 contentProvider Bestellung bestellungProvider {
35   providerType backend
36 }
37 remoteConnection backend {
38   uri "http://.../de.md2.bestellung.backend/service"
39 }
```

Abbildung 5.9: Modell der Kontrolllogik der Beispiel-App in MD^2

(Zeile 19-24) illustriert die Verwendung von Datenoperationen, um Datenobjekte zu laden, zu erstellen und zu verknüpfen. Auch Speichern (Zeile 27) und Löschen werden unterstützt.

Datenobjekte, die Instanzen eines im Datenmodell definierten Entitätstyps sind, werden über Datenlieferanten (*ContentProvider*) angesprochen, die entweder eine lokale Datenbank oder eine Serverschnittstelle referenzieren. MD^2 spezifiziert eine

auf Representational State Transfer (REST) basierende Schnittstelle für den Zugriff auf Daten und generiert neben Android- und iOS-Apps auch eine entsprechende JavaEE-Implementierung. Ein Datenlieferant kann auf einen Ausschnitt der Datenquelle eingeschränkt werden, indem ein entsprechender Filter definiert wird. Im Beispiel verweist der Datenlieferant `suchergebnis` (Zeile 29-33) auf das Produkt in der Serverdatenbank, dessen Namen mit der Eingabe im Sucheingabefeld übereinstimmt.

Die Kontrollkomponente von MD^2 kann des Weiteren die Benutzeroberfläche steuern. Neben einem fortgeschrittenen Workflow-Konzept, das die Spezifizierung von Bedingungen und komplexen Navigationspfaden erlaubt, stellt die DSL auch einfache Anweisungen zur Verfügung, die zwischen verschiedenen Ansichten der GUI wechseln. Das Beispielmodell verwendet eine solche Anweisung (Zeile 24), um einen Wechsel der Registerkarte anzustoßen, nachdem ein Produkt geladen wurde.

Wie bereits erwähnt, generiert MD^2 aus einem textuellen Modell, das in der MD^2-eigenen DSL dreiteilig definiert wurde, vollständig lauffähige Apps für Android und iOS. Die Codegeneratoren erzeugen dabei Quelltext, der direkt das SDK der jeweiligen Plattform nutzt. Für Android wird demzufolge Java-Quelltext (und zusätzlich XML-Dateien für das Layout) erstellt, für iOS Objective-C. Da die Generatoren zudem die Eigenheiten der Plattformen berücksichtigen (z. B. hinsichtlich der Darstellung von Registerkarten), haben die erstellten Apps ein natives Look & Feel.

5.7 Weiteres Vorgehen

Im Feld der Cross-Plattform-Entwicklung gibt es wie dargestellt erste vielversprechende Ansätze und Lösungen. Darüber hinaus gibt es aber in vielen Teilbereichen noch offene Fragen und Probleme, deren Untersuchung, Erforschung und Lösung wünschenswert ist. Dabei dreht es sich um die Weiterentwicklung von Cross-Plattform-Ansätzen und -Lösungen, die Entscheidungsunterstützung angesichts eines wachsenden Marktes für Cross-Plattform-Lösungen sowie die Integration der Cross-Plattform-Entwicklung in existierende Entwicklungsmethodiken.

Eine aktive Forschung und Entwicklung zu Cross-Plattform-Lösungen hat im Allgemeinen keinen Platz im Tagesgeschäft der meisten IT-Abteilungen. Sie sollte Unternehmen und Einrichtungen überlassen werden, zu deren Geschäftsmodell und Ausrichtung sie passt. Eine *kontinuierliche Beobachtung* des Cross-Plattform-Marktes wird die Fortschritte dieser Unternehmen offenbaren. Eine solche Beobachtung ist deshalb angebracht, um mit den neuesten Entwicklungen Schritt zu halten und ihre Vorteile für die Unternehmen der Region zu erschließen.

Abseits der Untersuchung einzelner Cross-Plattform-Entwicklungsansätze und bereits existierender -Lösungen konnten im Projekt einige weitere Themen identifiziert werden, die im Cross-Plattform-Bereich einer genaueren Untersuchung bedürfen. Zum einen ist offenkundig, dass in jedem Entwicklungsprojekt für mehrere mobile

Plattformen ähnliche Fragestellungen und Entscheidungen anstehen. Diesbezüglich ist zu analysieren, inwiefern eine *kriteriengestützte Entscheidungshilfe* bereitgestellt werden kann. Basierend auf den Bewertungen der vorhergehenden Kapitel könnte eine Rangordnung von Cross-Plattform-Frameworks erstellt werden, nachdem Projektverantwortliche die Kriterien gemäß den Anforderungen des Projekts oder des Unternehmens gewichtet haben.

Eine weiteres Problemfeld, das bisher nur unzureichend abgedeckt wird, ist die Integration von Cross-Plattform-Lösungen in den gesamten Lebenszyklus einer mobilen Anwendung, das sogenannte *Application Lifecycle Management* (ALM). Dies betrifft die Einbettung der Tools in die unterschiedlichen Phasen des Entwicklungsprozesses wie Anforderungserhebung, Testen, Deployment oder Wartung. Momentan sind diesbezüglich noch keine Best Practices bekannt und es herrscht eine manuelle, Ad-Hoc-Integration vor. Die Cross-Plattform-Lösungen bieten nur vereinzelt und in nicht-integrierter Form Unterstützung an. Zum Beispiel ermöglicht es PhoneGap Build, in der *Cloud* plattformspezifische Pakete für Apps, die auf Cordova/PhoneGap basieren, zu erstellen. Eine durchgehende Unterstützung des Lebenszyklus fehlt aber bisher. Dies fällt insbesondere beim *Testen und Debuggen* auf, die aufgrund mangelnder Werkzeugunterstützung zumeist vollständig und manuell auf den plattformeigenen Emulatoren und Geräten erfolgen.

Kapitel 6

Best Practices der teilnehmenden Unternehmen

Inhaltsübersicht

Im Folgenden wurden erste Ansätze für Handlungsempfehlungen zusammengetragen, die auf erfolgreichen Praktiken der am Projekt beteiligten Unternehmen basieren. Durch die noch geringen Erfahrungen bleiben sie Denkanstöße für Verbesserungen. Einen Ausbau zu umfangreichen Handlungsempfehlungen inklusive Umsetzungshinweisen und Kontextinformationen könnte ein Folgeprojekt in einem oder zwei Jahren angehen.

Zu beachten ist, dass die hier zusammengestellten Handlungsempfehlungen auf Hinweisen durch die Projektteilnehmer aufbauen. Unsere Empfehlungen bezüglich der Cross-Plattform-Entwicklung hingegen finden sich in Kapitel 5.5. Darüber hinaus geben wir zum Teil Empfehlungen bezüglich der im folgenden Kapitel 7 zusammengestellten ausgewählten Aspekte.

6.1 Motivierung der App-Entwickler

Die Begeisterung, die mobilen Endgeräten im Consumer-Bereich entgegengebracht wird, lässt sich auch zur Motivierung der Entwickler nutzen. Mehrere Unternehmen berichteten, Mitarbeiter hätten sich aus eigenem Interesse in die App-Entwicklung eingearbeitet oder seien einer Einführung selbst dann aufgeschlossen, wenn Kurse und Ähnliches in die Feierabendzeit oder auf Samstage fielen. Es ist zu prüfen, ob sich die Begeisterung wecken und *nutzen* lassen kann.

Sofern Interesse der Mitarbeiter, Personalaustattung und aktuelle Projekte es zulassen, ist es sinnvoll, die Beschäftigung mit Apps weitgehend freizustellen. Mitarbeiter, die sich freiwillig mit der Thematik beschäftigen, werden tendenziell besonders motiviert sein. Des Weiteren kann es Mitarbeiter zu besonderem Engagement anspornen, wenn sie Einfluss auf Entwicklungswerkzeuge und -vorgehen nehmen können. Im Idealfall entsteht das Gefühl, an einer aktuellen Entwicklung mitwirken und die Zukunft von Apps im Unternehmen mitprägen zu können. Es sollte darauf geachtet werden, nachhaltig zu motivieren. Ein Mitarbeiter, der sich besonders engagiert, danach allerdings in Projekten arbeitet, die reine Fleißarbeit erfordern, verfällt möglicherweise schnell in einen *Alltagstrott*. Vorsicht ist auch bei der Auswahl von Anreizen geboten. Wenn es für Mitarbeiter, die keine Apps entwickeln, so erscheint, als würden den App-Entwicklern mehr Aufmerksamkeit und möglichlicherweise sogar Vorteile zuteil, die durch die betriebliche Bedeutung ihrer Arbeit nicht zu rechtfertigen sind, sind Demotivation und eventuell sogar soziale Probleme unvermeidlich. Gelingt es aber, App-Entwickler über die Art ihrer Arbeit zu motivieren, ist ein guter Beitrag zur Erreichung der Entwicklungsziele sehr wahrscheinlich.

Zu beachten ist allerdings, dass Mitarbeiter nicht „über das Ziel hinausschießen“. Die Eigenentwicklung von in guter Qualität günstig am Markt erhältlicher Komponenten in Folge großer Eifrigkeit kann kontraproduktiv sein.

6.2 Komponenten-, bzw. Service-Orientierung

Viele Apps werden derzeit als in sich geschlossene Einheiten konzipiert und entwickelt. Während die teilnehmenden Unternehmen damit gute Erfahrungen gemacht haben (siehe auch Kapitel 4.3), ist über alternative Wege nachzudenken. Dies gilt vor allem vor dem Hintergrund, dass der funktionale Umfang von Apps eher wachsen wird; viele prototypische Implementierungen hatten einen sehr geringen Umfang. Konkret bieten sich die Entwicklung im Baukastenstil sowie die Nutzung einer Service-oriented Architecture (SOA) an.

Viele Apps im geschäftlichen Umfeld dienen dazu, Informationen zu visualisieren. Zusätzlich werden einige von ihnen zur Datenerfassung genutzt. Hierbei kommen immer wieder sehr ähnliche Funktionen zum Einsatz. Beispiele wären Diagramme bei der Visualisierung und die Erfassung bestimmter Stammdaten wie etwa einer Adresse. Aus diesem Grund kann es sinnvoll sein, eine einheitliche Architektur für funktional ähnliche Apps zu entwickeln und neue Apps soweit aus bestehenden Komponenten zusammenzusetzen wie möglich. Der initiale Aufwand steigt dabei etwas, in der Folge ist aber eine höhere Entwicklungseffizienz zu erwarten. Es ist allerdings darauf zu achten, nur solche Programmmodule als wiederverwendbare Komponenten zu entwickeln, bei denen der Einsatz in mehreren Apps realistisch ist. Ein positiver Nebeneffekt kann sein, dass die Anwendungslandschaft einfacher zu überblicken und zu warten ist, wenn App einheitlich aufgebaut sind und in einer einheitlichen Art

und Weise auf Backend-Dienste zugreifen.

Bezüglich der Backend-Dienste kann es sinnvoll sein, auf SOA zu setzen. Dienstorientierte Architekturen dienen dazu, IT-Systeme mit grundlegenden Aufgaben (etwa verschiedene Datenbanken und Server) für das Angebot von Diensten zu *orchestrieren*. Während also einzeln betriebene Dienste bestehen, die recht simple aber spezialisierte Leistungen mit sehr hoher Effizienz erbringen, entstehen komplexe Leistungen durch eine geschickte Verknüpfung dieser Dienste. Insbesondere, wenn ein Unternehmen ohnehin eine SOA-Strategie verfolgt, spricht viel dafür, Apps in diese einzubinden. So können der Programmieraufwand für Apps gesenkt und die Komplexität von Apps maßgeblich verringert werden.

Auf den ersten Blick weisen Apps, die in hohem Maße auf Dienste angewiesen sind, Nachteile im Hinblick auf ihren Offline-Betrieb auf. Allerdings kann gerade die Nutzung einer SOA diesbezüglich Vorteile bieten. Denn bei einer App, die Backends in einer einfachen Client-/Server-Architektur anspricht, finden für gewöhnlich sehr viele Transaktionen statt. Jede Transaktion benötigt Bandbreite und jede kann für sich fehlschlagen. Die Nutzung vieler simpler Dienste zur Erbringung komplexerer Dienste durch das mobile Endgerät erfordert überdies viel Leistung und kann damit die Akkulaufzeit verkürzen. Werden hingegen geschickt orchestrierte Dienste genutzt, kann eine App mit einer oder wenigen Transaktionen Daten anfordern, mit denen sie unmittelbar arbeiten kann. Zwar ist das Übertragungsvolumen pro Transaktion dann erhöht, der gesamte Datendurchsatz allerdings deutlich verringert. Die Leistungen werden in Rechenzentren und ähnlichen Einrichtungen erbracht, sodass an das Endgerät nur noch geringe Ansprüche gestellt werden. Auch die Latenz kann maßgeblich sinken, da viele Zugriff über drahtlose Verbindungen durch Operationen ersetzt werden können, die extrem schnell in geschlossenen Verbünden von Backend-Systemen erbracht werden. Des Weiteren vereinfacht die geringere Zahl entfernter Aufrufe Caching-Strategien.

Nicht unerwähnt bleiben darf natürlich, dass es äußerst zweifelhaft ist, ein SOA-Konzept „nur“ für Apps zu verfolgen. In jedem Fall darf die Komplexität nicht unterschätzt werden. Allerdings sind auch leichtgewichtige Ansätze denkbar, indem beispielsweise ein Server Daten zusammenträgt, die von Apps angefordert werden, sie aggregiert und eventuell notwendige Berechnungen vornimmt, um so die Zahl der Transaktionen der App mit anderen Systemen zu verringern.

Die intensive Nutzung von Backend-Systemen kann auch die Sicherheit von Apps erhöhen werden. So ist es in den meisten Fällen sinnvoll, die Speicherung sensibler Daten ausschließlich entsprechend gesicherten Servern zu überlassen. Natürlich gibt es auch hierbei Ausnahmen; in einigen Fällen gibt es Offline-Anforderungen, für die Daten auf dem Endgerät vorgehalten werden müssen. In vielen Fällen werden schlicht keine Daten verarbeitet, die so sensibel wären, dass sie besondere Schutzvorkehrungen notwendig machten. Falls ein solcher Schutz aber erforderlich ist, wird es in vielen Fällen einfacher sein, die Verbindung der App mit dem Backend ab-

zusichern und die vorhandene Expertise in der Sicherung von Daten auf Servern zu nutzen. Durch entsprechende Zugriffsregeln kann überdies der Gefahr begegnet werden, dass kompromittierte Endgeräte den Zugriff anfordern. Hierbei können typische Praktiken zum Einsatz kommen, etwa eine Begrenzung der pro Zeiteinheit abgefragten Datensätze. Der Vorteil hierbei ist, dass Sicherheitsexperten ihren gewohnten Aufgaben nachgehen können und sich – idealerweise – nicht *im Detail* mit der Sicherheit auf mobilen Endgeräten auseinandersetzen müssen. Dringend zu empfehlen ist dennoch, die Entwicklung der Sicherheit auf Smartphones und Tablets im Auge zu behalten und Entwickler entsprechend zu schulen.

6.3 Informationsverfügbarkeit

Der interne Einsatz von Apps lohnt sich vor allem dort, wo einfach Zugriff auf Information des Unternehmens geschaffen werden kann. Zwar sind mobile Endgeräte in einer Reihe von Szenarien auch für die Datenerfassung geeignet (vgl. auch Kapitel 4.3), der mobile Zugriff auf bestehende Informationen ist aber häufig mit sehr geringem Aufwand umzusetzen.

Zunächst ist zu überlegen, welche Informationen in geeigneter Form verfügbar sind bzw. mit geringem Aufwand verfügbar gemacht werden können. Dies sollte verbunden werden mit der Frage, ob es lohnt, diese Informationen mobil zur Verfügung zu stellen. *Mobil* ist dabei ein dehnbarer Begriff. Die mobile Nutzung beginnt bei Apps, die von am Schreibtisch sitzenden Mitarbeitern auf dem Smartphone genutzt werden, weil der Zugriff schneller oder komfortabler ist als über den PC. Sie kann nach Feierabend durch Mitarbeiter erfolgen, um „noch schnell“ Informationen abzurufen, etwa um einen Termin am nächsten Tag vorzubereiten. Schließlich kann mobil bedeuten, dass Mitarbeiter auf Reisen ohne festen Internetzugang Daten nutzen, die ihnen nur über das Intranet oder sogar nur durch persönliche Nachfrage zur Verfügung ständen.

Im zweiten Schritt ist zu prüfen, ob die Informationen sensibel sind; je sensibler die Informationen, desto aufwändiger ist im Zweifelsfall ihre sichere Bereitstellung. Im dritten Schritt kann die Bereitstellung dann anhand einer App umgesetzt werden.

Aus Sicht der Investitionsrechnung sind entsprechende Projekte schwierig zu bewerten. Ihr Nutzen lässt sich kaum monetär messen. Es ist allerdings zu erwarten, dass Mitarbeiter präziser und in vielen Fällen auch schneller arbeiten können.

Zu beachten ist, die Informationen in strukturierter Form zur Verfügung zu stellen. Besonders, wenn es viele Datenquellen und verschiedenste Arten von abrufbaren Informationen gibt, sollte sichergestellt werden, dass der Zugriff sinnvoll unterstützt wird. Denkbar ist etwa, eine oder wenige Apps zu verwenden, die Zugriffe bündeln, aber anpassbar sind. So könnten Mitarbeiter Abfragen anpassen oder besonders häufig genutzte Abfragen als „Favoriten“ ablegen.

Bei aller Einfachheit der Umsetzung ist zu bedenken, dass keine kritischen Daten

gefährdet werden dürfen. Dies erfordert Kontrolle darüber, was verfügbar gemacht wird: Daten werden gegebenenfalls auch erst dadurch sensibel, dass mehrere zusammenhängende, aber für sich unkritische Daten über entsprechende Funktionen zur Verknüpfung angeboten werden. Ihre gemeinsame Auswertung könnte Dritten dann unerwünschte oder sogar schädliche Einblicke in das Unternehmen gewähren.

6.4 Unterstützung von Außendienst und Vertrieb

In den Interviews wurden von mehreren Teilnehmern positive Erfahrungen im Einsatz von mobilen Endgeräten im Rahmen der Aktivitäten des Außendienstes berichtet (siehe Kapitel 4.3). Dieses Einsatzgebiet ist dabei so überzeugend, dass es als eigene Handlungsempfehlung stehen kann. Besonders geeignet sind Tätigkeiten des Vertriebs sowie – im Dienstleistungsgewerbe – des Kundendienstes.

Grundsätzlich ist zwischen zwei Aktivitäten zu unterscheiden: der Visualisierung und der Datenerfassung. Für eine erste Ausbaustufe kann mit der Visualisierung begonnen werden. Die Datenerfassung kann in einem zweiten Schritt ergänzt werden. In beiden Fällen sind Tablets die geeigneten Geräte. Smartphones eignen sich aufgrund des deutlich kleineren Bildschirms weniger. Bei geschickter Programmierung der Apps ist es allerdings möglich, Smartphones als Backup-Lösung oder für Mitarbeiter, die nur selten die betroffene App nutzen möchten, zu unterstützen.

Die hinter der Nutzung von tabletbasierten Apps stehende Vision ist die Ablösung von Papier und Notebook. Bei vielen Terminen des Außendienstes gehört beides fest zu den Arbeitsmitteln. Während das Gespräch mit dem Kunden zunächst im Vordergrund steht, werden im Laufe des Termins häufig Vertragsbestandteile, Finanzdaten und Ähnliches visualisiert. Dafür werden entweder vorbereitete Ausdrucke oder ein Notebook genutzt. Auf dem Notebook ist eine individuelle Anpassung möglich, um dem Kunden etwa die Konditionen von Finanzprodukten mit „seinen“ Zahlen zu demonstrieren. Allerdings stellt ein Notebook dabei typischerweise eine Barriere da. Es ist das Notebook des Außendienstmitarbeiters; dass der Kunde es in die Hand nimmt und eine aktive Rolle in der Begutachtung einnimmt, ist unüblich. Somit bleibt er während der Präsentation passiv. Diese Barriere gibt es bei Ausdrucken nicht; sie können in die Hand genommen, herumgereicht und beschriftet werden. Allerdings müssen vom Außendienstmitarbeiter umfangreiche Dokumente mitgeführt werden – die er zu 95% nicht benötigen wird. Trotzdem kann er nicht individuell auf die Wünsche des Kunden eingehen. Die kombinierte Nutzung von Ausdrucken und Latops kann dieses Problem verringern, aber nicht lösen. Das Mitführen eines Druckers ist wenig komfortabel.

Die Datenerfassung gestaltet sich ähnlich. Die papiergebundene Erfassung ist interaktiv und persönlich. Allerdings muss auf Komfort ebenso verzichtet werden wie auf Vor-Ort-Überprüfungen. Wenn Formulare nicht entsprechend vorbereitet sind, muss von Bestandskunden die Adresse erneut erfasst werden; eine unstimmige Post-

leitzahl fällt erst bei der Erfassung des Formulars im Unternehmen auf; usw. Schließlich stellt die papiergebundene Arbeit einen Medienbruch dar. Trotz der Möglichkeit der Automatisierung (per Scan und Texterkennung), ist hier eine Fehlerquelle zu sehen. Bei der Nutzung eines Notebooks entfallen die beschriebenen Probleme, dafür ist der Prozess deutlich weniger persönlich: mit dem Gerät interagiert nur der Außendienstmitarbeiter; den Kunden hemmt die Barriere des Notebooks schon bei der Nachverfolgung der Eingaben des Mitarbeiters.

Die geschickte Nutzung von Tablets nebst zugehörigen Apps kann in vielen Fällen das Abwägen zwischen Papier und Notebook obsolet machen. Die Visualisierungsfähigkeiten von Tablets sind mit denen von Notebooks vergleichbar. Selbst wenn viel Leistung benötigt wird, etwa zur Berechnung von Grafiken, ist ein Tablet kein Hindernis. Die Berechnung kann von Servern übernommen werden; das Tablet ruft die fertigen Grafiken dann ab. Der wesentliche Vorteil liegt in der Interaktion: das Tablet kann dem Kunden gereicht werden. Es ist für alle am Gespräch Beteiligten einfach möglich, mit dem Gerät zu interagieren. Dabei zeigt die Erfahrung, dass gerade die Touchbedienung die Zugänglichkeit des Mediums ungemein erhöht. Dies gilt selbst für ältere, wenig technikaffine Nutzer (vgl. etwa [MJLc10]). Im Gegensatz zum Notebook wird ein Tablet auch weniger als „privat“ empfunden, was das gemeinsame Betrachten von Informationen erleichtert.

Zu beachten ist allerdings, dass sich bestehende Programme, die bisher vom Außendienst auf Notebooks genutzt werden, nicht eins zu eins als Apps umsetzen lassen. Vielmehr ist es nötig, die Besonderheiten der Touchbedienung und der Bildschirmgröße zu bedenken. Gleichzeitig macht es Sinn, in angemessenem Umfang Besonderheiten mobiler Geräte wie Sensoren (etwa zur Ortsbestimmung) zu nutzen. Darüber hinaus sollte die App abgesichert werden, etwa über einen Präsentationsmodus. Nicht alle Daten sind für Kunden bestimmt. Könnte ein Kunde z. B. in wenigen Schritten – und in der Regel sogar unbeabsichtigt – zu Daten anderer Kunden navigieren, wäre dies äußerst problematisch. Diesem Risiko lässt sich durch die durchdachte Implementierung entsprechender Apps aber sehr gut begegnen.

Die Datenerfassung mittels Apps kann ähnlich gestaltet werden. Wiederum ergibt sich die Chance, das Tablet sehr interaktiv nutzen zu können. Während es sich fast wie Papier handhaben lässt, bietet es die Option, auf die Mächtigkeit computerbasierter Formularerfassung zurückzugreifen. Zudem kommen hier die zusätzlichen Funktionen typischer Tablets in Spiel. Insbesondere für die Kamera gibt es zahlreiche sinnvolle Einsatzszenarien. Bei der Datenerfassung zu bedenken ist allerdings, dass die Eingabemöglichkeiten per Touch Einschränkungen aufweisen, wenn sehr viele Daten zu erfassen sind. Einfache Formulare lassen sich schnell genug erfassen; Ähnliches gilt für Auswahlformulare, bei denen etwa auf *Ja/Nein*-Fragen zu antworten ist oder aus einer Menge von Optionen ausgewählt werden muss. Problematisch wird die Erfassung längerer Texte.[1] Hier werden Notebooks so lange ihre

[1] Einzelne Nutzer berichten, dass sie auf der virtuellen Tastatur ähnliche Anschlagszahlen wie

Stärke ausspielen, bis die Spracherkennung in entsprechenden Szenarien leistungsfähig genug ist. Im Übrigen muss die Datenerfassung mit Tablets keinesfalls durch den Kunden selbst erfolgen, um vorteilhaft zu sein: schon das Gefühl einer gemeinsamen Erfassung durch den Blick auf den Bildschirm ist geeignet, die Kommunikation zu verbessern.

Idealerweise werden Apps für den Außendienst durch Apps für Kunden ergänzt. Beide sollten eine einheitliche Gestaltung aufweisen, sodass Kunden, die sich zunächst selbst informiert haben, unmittelbar die App des Außendienstmitarbeiters „verstehen“. Umgekehrt kann ihnen die App nach dem gemeinsamen Termin an die Hand gegeben werden. Sie finden sich dann unmittelbar zurecht und können Inhalte des Gesprächs noch einmal nachvollziehen, Optionen vergleichen und so weiter. Dabei können sie sogar *After-Sales*-Aktivitäten begleiten und der Kundenbindung dienen. Idealerweise können sie sogar im Rahmen der Erfüllung des abgeschlossenen Vertrags eingesetzt werden. Im Beispiel einer Versicherung etwa könnten Schadensmeldungen zusammen mit einem per mobilen Endgerät geschossenen Foto eingereicht werden.

Allen Vorteilen zum Trotz sind Tablets im Außendienst keine risikolose Investition. Der Einsatz muss wohldurchdacht sein. Ein Mitarbeiter, der ein Notebook mit sich führt, kann auf die volle Mächtigkeit eines PCs zurückgreifen, wenn ihm die Funktionen der genutzten Anwendungen in Einzelfällen nicht ausreichen. Über Terminal-Verbindungen kann er im Zweifelsfall sogar weitreichend auf Ressourcen im Unternehmen zugreifen. Auf dem Tablet ist er hingegen weitgehend an die Möglichkeiten der installierten Apps gebunden. Aus diesem Grund ist ein Umstieg gut abzuwägen und setzt umfangreiche Probeläufe voraus. Für eine Übergangszeit ist es sinnvoll, Mitarbeiter sowohl mit Tablets als auch mit Notebooks auszustatten. Die Tablets könnten dann Papiere ersetzen und während des Verkaufsgesprächs zum Einsatz kommen. Kommt es zum Vertragsabschluss, würde zur Formularerfassung auf das Notebook gewechselt, idealerweise nach Synchronisation zuvor erhobener Daten.

6.5 Weitere Hinweise

Abschließend werden eine Reihe kurzer Tipps und Hinweise vorgestellt. Sie sollen in erster Linie als Anstoß für eigene Überlegungen dienen.

Web-Apps Steht nur ein sehr geringes Budgets zur Verfügung, kann es sinnvoll sein, eine mobile Web-App zu entwickeln. Sie ersetzt dann alle plattformspezifischen Implementierungen. Abstriche sind dabei allerdings nicht zu vermeiden, daher sollten die Auftraggeber darauf hingewiesen werden. Siehe auch Kapitel 5.4.1.

auf Computertastaturen erzielen. Verallgemeinerbar ist dies allerdings nicht.

Mehrgleisigkeit Lassen sich einzelne Plattformen in einem Kundenkontext als besonders bedeutend identifizieren, ist zu überlegen, nur für diese eine native App zu implementieren. Zusätzlich wird eine Web-App entwickelt, die Unterstützung auf allen übrigen Plattformen bietet. Es ist allerdings darauf zu achten, damit keine Kunden zu verärgern, die „ihre“ Plattform scheinbar grundlos nicht unterstützt sehen.

Prozessverbesserungen Die Entwicklung von Apps, die bestehende Geschäftsprozesse unterstützen sollen, kann genutzt werden, um Optimierungsmöglichkeiten bestehender Prozesse auszuloten oder Prozessinnovationen anzustreben.

make-or-buy Auch wenn – wie im Falle fast aller teilnehmenden Unternehmen – die Entscheidung für den Aufbau eigener Kompetenz bezüglich Apps gefallen ist, kann es situativ sinnvoll sein, Apps extern in Auftrag zu geben. Dies gilt vor allem dann, wenn Apps nicht in bestehende Geschäftsprozesse integriert oder zum Aufbau neuer Geschäftsprozesse genutzt werden. Sinnvoll ist ferner, die Entwicklung solcher Apps als Auftrag zu vergeben, deren Anforderungen in sich geschlossen sind. Insbesondere Apps mit relativ geringem Umfang und wenigen Überschneidungen zu bestehenden oder geplanten Systemen bieten sich an. Siehe auch Kapitel 7.2

MDSD Model-driven Software Development (MDSD) steht für einen Entwicklungsansatz, bei dem Software zunächst *modelliert* und danach – im Idealfall vollständig – automatisch *generiert* wird. Einige Unternehmen konnten sehr positiv vom Einsatz entsprechender Techniken berichten. Entsprechende Ansätze in der Cross-Plattform-Entwicklung wie MD^2 sind vielversprechend, aber noch in der Entwicklung (siehe Kapitel 5.6).

Kritische Grundhaltung Zweifelsohne werden mobile Endgeräte bzw. der gesamte Trend hin zu ubiquitären Anwendungen zahlreiche Nutzungsmöglichkeiten eröffnen, die für Unternehmen von großem Vorteil sein können. Gleichzeitig empfiehlt sich Besonnenheit. Während Apps für Kunden entwickelt werden, sollten gleichzeitig eigene Projekte geprüft werden. Zum jetzigen Zeitpunkt kann es sinnvoll sein, Anwendungsgebiete zu erkunden und Apps *auszuprobieren*. Dabei sollte aber nicht Gefahr gelaufen werden, neue Technologien überzubewerten.

Ablösung Werden mobile Endgeräte und neu entwickelte Apps eingesetzt, um bestehende Infrastrukturen abzulösen, ist ein gut koordiniertes Vorgehen essentiell. Die Einfachheit von Apps darf nicht dazu verleiten, die Komplexität eines Wechsels zu unterschätzen.

Coachings Auch wenn der gesamte Software-Engineering-Prozess im mobilen Umfeld profunde Besonderheiten aufweist, ist die Programmierung von Apps als

solche der Programmierung von klassischen Anwendungen recht ähnlich. Generell gilt die Einarbeitung für erfahrene Entwickler als relativ einfach. Aus diesem Grund kann die Einarbeitungszeit dadurch verkürzt werden, dass bereits in der App-Entwicklung erfahrene Mitarbeiter als Coaches eingesetzt werden. Sie wissen aus eigener Erfahrung, worauf bei der App-Entwicklung zu achten ist und können in kollegialer Atmosphäre das benötigte Wissen schnell vermitteln. Darüber hinaus kann es Sinn machen, wenig erfahrenen Entwicklern einen erfahrenen Coach zur Seite zu stellen. Apps würden dann zwar eigenständig entwickelt, aber regelmäßig zusammen mit dem erfahrenen Entwickler geprüft.

Kapitel 7

Ausgewählte Aspekte

Inhaltsübersicht

Die folgenden Unterkapitel betrachten jeweils einen ausgewählten Aspekt im Kontext Business Apps. Es handelt sich um Themenfelder, die im Rahmen der Interviews und der Auswertungsphase als relevant erkannt wurden. Wir motivieren jeden Aspekt zunächst, indem wir die Problemstellung beschreiben und deren Relevanz begründen. Dieser Abschnitt soll jeweils für die Problematik sensibilisieren. Für einige der Aspekte existieren schon Lösungen oder zumindest erste Ansätze. Wo möglich, werden diese in einem zweiten Abschnitt kurz beschrieben und auf weiterführende Literatur verwiesen. Bei Aspekten, die in Forschung und Praxis noch nicht genügend betrachtet worden sind, als dass Lösungen existieren, nennen wir wenn möglich potentielle zukünftige Ansätze und Entwicklungen.

7.1 Anforderungsanalyse für mobile Applikationen

Motivation Anforderungen ebnen den Weg zu einem Softwareprodukt. Eine effektive Anforderungsermittlung ist essentiell. Fehlerhafte Anforderungen sind im

Verlauf der Entwicklung zunehmend schwieriger zu korrigieren. Die Umsetzung der Anforderungen stellt sicher, dass die Software geliefert werden kann, die erwartet wird – vorausgesetzt, die Anforderungsermittlung und -analyse war erfolgreich.

In den Interviews wurde herausgestellt, dass die Anforderungsermittlung für Apps oftmals wenig ausgeprägt ist. Es gibt grobe Ideen, was umgesetzt werden soll, aber häufig werden diese nicht konkretisiert. Insbesondere bei der Ad-hoc-Entwicklung durch einzelne Entwickler wird dies offenbar nicht für nötig erachtet. Da zudem viele Apps nur der Erprobung dienen, nahmen die Unternehmen in Kauf, dass diese nicht den ursprünglichen Vorstellungen entsprachen. Zugleich ist die Ermittlung von Anforderungen für Apps nicht trivial: in den Interviews wurde deutlich, dass verschiedene Kontexte in Betracht gezogen werden müssen, also etwa unterschiedliche Ausprägungen der Hardware der anvisierten Zielplattform(en) sowie die Möglichkeit von Kontextwechseln (etwa der Netzgeschwindigkeit) während der Nutzung. Darüber hinaus werden Apps durch die mobile Verfügbarkeit und die neuen Eingabemöglichkeiten anders genutzt als PC-Software. Es ist davon auszugehen, dass sich dies in der Anforderungsermittlung niederschlägt.

Weiterführendes Zum jetzigen Zeitpunkt liegen keine weiterführenden Informationen vor. Vielmehr gibt es diverse offene Forschungsfragen. Es ist allerdings zu erwarten, dass die Erkenntnisse im Bereich Requirements Engineerings übertragbar sind – schließlich sind Apps Computerprogramme und unterliegen daher ähnlichen Rahmenbedingungen wie Applikationen für PCs und Server. Ein Studium der allgemeinen Literatur (z. B. [Ebe08, Poh08]) ist daher empfehlenswert.

7.2 Make-or-Buy-Entscheidungen und Outsourcing

Motivation Nicht alle der teilnehmenden Unternehmen entwickeln Software für Kunden. Folglich ist es nicht naturgegeben, dass sie Kompetenz zur App-Entwicklung im eigenen Haus aufbauen, selbst wenn der Wunsch nach Nutzung von Apps besteht. In Entwicklungsprojekten aller Art ist darüber hinaus über Outsourcing nachzudenken, also die langfristige Abgabe von Entwicklungsaufgaben an Dritte.

Grundsätzlich erscheinen Apps als gut geeignet, um extern in Auftrag gegeben zu werden. Der Aufwand pro App-Entwicklungsprojekt ist zum jetzigen Zeitpunkt aller Regel nach überschaubar. Durch die Standardisierung der Plattformen lassen sich Anforderungen recht gut beschreiben. Projekte lassen sich zudem gut von weiteren Software- oder Infrastrukturprojekten abgrenzen. Selbst wenn durch eine App auf Backend-Systeme zugegriffen werden soll, kann ein Dritter die App entwickeln, wenn ihm die Schnittstellenspezifikation zur Verfügung gestellt wird. Hinzu kommt schließlich ein aller Regel nach geringes finanzielles Volumen des einzelnen Entwicklungsprojekts – es ließe sich argumentieren, dass der *finanzielle* Schaden selbst beim Scheitern des Outsourcing-Projekts sehr begrenzt ist.

Kritisch gesehen werden kann hierbei allerdings die geringe Erfahrung mit Apps. Zum Ausprobieren von Funktionen kann die Arbeiten mit Dritten mühselig sein. Entsprechende Verträge sollten demnach flexibel gestaltet werden. Es bietet sich etwa die Verwendung agiler Entwicklungsmethoden an. Wichtig ist, dass trotz des Outsourcing derselbe Erfahrungsschatz aufgebaut werden kann, sofern das ein Ziel des Projekts ist.

Wenn Apps mit feststehendem Funktionsumfang entwickelt werden sollen, besteht das wesentliche Risiko in der Kompetenz des Anbieters. Eine vertragliche Absicherung erscheint ratsam. Generell stellt sich die Frage, ob der Aufbau eigener Kompetenz sinnvoll ist. Falls Apps strategisch genutzt werden oder als Verkaufsinstrumente dienen sollen, oder falls eine eigene Integration in bestehende Prozesse geplant ist, kann es Sinn machen, eigene Entwicklungskompetenzen intern aufzubauen. Diese sind mittelfristig flexibler einsetzbar als der „Zukauf“ von Apps.

Falls native Apps für mehrere Plattformen entwickelt werden sollen und keine der in Kapitel 5 dieser Broschüre vorgestellten Möglichkeiten hierfür geeignet erscheint, kann die Entwicklung eventuell an Dritte vergeben werden. Ein Ansatz wäre hierbei die Entwicklung für eine Plattform im Unternehmen. Die „Fleißarbeit“, eine funktional identische App für weitere Plattformen umzusetzen, könnte extern vergeben werden. Aufgrund des gegenwärtigen Ausblicks darf aber davon ausgegangen werden, dass die separate Entwicklung für alle zu unterstützenden Plattformen an Bedeutung verlieren wird.

Hinsichtlich Outsourcing gelten grundsätzlich ähnliche Bedingungen wie für Entwicklungsprojekte bei PC- oder Serversoftware. Einzelne Entwicklungsschritte an Dritte zu vergeben, erscheint aufgrund der Begrenztheit derzeitiger Projekte allerdings wenig ratsam. Lohnenswert könnte es allenfalls beim Testen von Apps sein. Dies gilt umso mehr, wenn für mehrere Plattformen entwickelt wurde, die App in verschiedenen Kontexten funktionieren muss und eine Vielzahl von Endgeräten unterstützt werden soll. Voraussetzung ist dabei, dass der Zweck der App und eventuell mit ihr verbundene Prozesse von einem Dienstleister vollständig verstanden werden können. Trivialerweise sollte der Aufwand hierfür (deutlich) unter dem für das eigene Testen veranschlagten liegen. Ferner sollten Seiteneffekte – z. B. fachliches Feedback durch Tester – ebenso bedacht werden. Zum Thema Testen vgl. auch Kapitel 7.9.

Weiterführendes Bisher liegen keine ausreichenden Erkenntnisse in Bezug auf Make-or-Buy-Entscheidungen bzw. Outsourcing vor, die es erlauben würden, Handlungsempfehlungen zu geben oder grundsätzliche Strategien zu empfehlen. Eigene Überlegungen können auf den oben diskutierten Eckpunkten aufbauen.

7.3 Sicherheit auf mobilen Endgeräten

Motivation Die Sicherheit von Unternehmensdaten und -anwendungen ist ein wichtiger Diskussionspunkt vor dem Hintergrund der zunehmenden Verbreitung mobiler Geräte in Unternehmen. Unternehmen stehen vor neuen Herausforderungen, um den Schutz sensibler Daten zu gewährleisten, die mit mobilen Geräten die Unternehmensräume verlassen. Gleichzeitig wird der entfernte Zugriff auf Kommunikations- und Unternehmensdaten immer wichtiger, um die Produktivität zu steigern und die zunehmende Mitarbeitermobilität zu unterstützen. Mit der zunehmenden Leistungsfähigkeit von Smartphones und Tablets steigt auch der Umfang sensitiver Daten. Da eine Umkehr dieser Trends nicht zu erwarten und aus gesamtunternehmerischer Sicht auch nicht zu wünschen ist, müssen die IT-Abteilungen von Unternehmen Vorkehrungen treffen, um in der veränderten IT-Landschaft weiterhin die gewohnte Sicherheit zu gewährleisten. Zu analysieren sind dabei zum einen die möglichen Einfallstore für Angriffe, die die mobilen Geräte öffnen, und zum anderen die möglichen Schäden durch erfolgreiche Angriffe.

Mobile Geräte wie Smartphones und Tablets eröffnen verschiedene Angriffswege, von denen einige auch auf anderen Systemen relevant sind, in ihren spezifischen Eigenschaften und in der Kombination aber eine neue Stufe der Sicherheitsgefährdung erreichen.

- Die ausschließlich drahtlose Netzwerkkommunikation über WLAN und Mobilfunknetze erfordert besondere Vorkehrungen beim Austausch von Daten. Verschärft wird dieser Umstand durch häufig wechselnde Netze mit oft mangelhafter Verschlüsselung. Auf Ebene der Geräte und/oder innerhalb der entwickelten Apps sind entsprechend Vorkehrungen zur Verschlüsselung des Netzwerkverkehrs erforderlich.

- Da die Geräte auf Mobilität ausgelegt sind und dementsprechend genutzt werden, sind sie einem wesentlich höheren Diebstahl- und Verlustrisiko ausgesetzt als größere, stationäre Geräte. Auf dem Gerät gespeicherte Daten und die installierten unternehmenskritischen Anwendungen sind deshalb vor dem Zugriff Unbefugter zu schützen.

- Zu prüfen ist auch, wie der Schutz von Daten gewährt werden kann, wenn Mitarbeiter aus dem Unternehmen ausscheiden, Endgeräte aber in ihren Besitz übergehen. Dies gilt auch für den Fall, dass auf privaten Geräten geschäftliche Apps installiert wurden.

- Viele Apps, insbesondere im Consumer-Bereich, bieten umfangreiche Kommunikationsmöglichkeiten und soziale Funktionen. Oftmals ist den Nutzern nicht klar, welche Daten die Anwendungen weitergeben, oder sie sind nicht entsprechend sensibilisiert und teilen sensible Daten über die Anwendungen mit

anderen. Es muss ausgeschlossen sein, dass unternehmensrelevante Daten, z. B. Kontaktdaten von Kunden, auf diese Weise in die Hände Unbefugter gelangen.

- Eine Steigerung sind Anwendungen, die explizit als Schadprogramme (*Malware*) entwickelt wurden. Dieses auch auf Desktopcomputern anzutreffende Phänomen wird auf mobilen Geräten dadurch verschlimmert, dass auf Desktoprechnern übliche Sicherheitsvorkehrungen wie Antivirenprogramme sich auf Smartphones noch nicht allgemein durchgesetzt haben bzw. sich noch in der Entwicklung befinden.

Mögliche Schäden aus Sicherheitslücken lassen sich im Wesentlichen in zwei Kategorien einteilen: den Verlust sensitiver Daten an Außenstehende und die Manipulation von Unternehmensdaten durch Außenstehende. Aufgrund der weitreichenden Fähigkeiten moderner Geräte betreffen Datenlecks oftmals nicht nur die offensichtlichen Daten, die explizit auf dem Gerät gespeichert wurden, sondern auch solche, die das Smartphone im Betrieb selbständig aufzeichnet. Aus Ortungsinformationen etwa können Angreifer in vielen Fällen Rückschlüsse auf geschäftliche Belange ziehen. Der Manipulationsaspekt bezieht sich auf die unautorisierte Veränderung von Unternehmensdaten, die beispielsweise dadurch ermöglicht wird, dass ein Angreifer Anwendungen ausführen kann, die auf Unternehmensdaten auch verändernd zugreifen.

Smartphones und Tablets ähneln in Bezug auf Sicherheitsaspekte in Grundzügen Notebooks. Auch diese werden mobil verwendet und sind entsprechend ähnlichen Gefahren ausgesetzt, insbesondere unsicheren Kommunikationskanälen und Verlustgefahr. Die Probleme sind auf Smartphone und Tablets allerdings aus mehreren Gründen verschärft:

- Die Nutzer haben oftmals ein geringeres Problembewusstsein, weil sie das Mobilgerät nicht als vollwertigen Computer wahrnehmen und den Wert der abgelegten Daten falsch einschätzen.

- Der kleinere Formfaktor erhöht die Wahrscheinlichkeit eines Verlusts oder Diebstahls.

- Die Nutzer vermischen mitunter private und berufliche Aktivitäten auf dem allgegenwärtigen Gerät.

Der letzte Punkt ist insbesondere vor dem Trend des „Bring-your-own-device“ (BYOD) hervorzuheben. Aufgrund der Popularität der neuartigen Geräte ist ein starker Wunsch der Mitarbeiter, ggf. insbesondere der Leitungsebene, festzustellen, die neuesten Smartphones und Tablets im Arbeitsumfeld einsetzen zu können (*Consumerization*). Stellt ein Unternehmen seinen Mitarbeitern nur ältere oder für den Unternehmenseinsatz vorgesehene Modelle bereit, entsteht vor diesem Hintergrund schnell das Bestreben, das private Gerät auch für berufliche Zwecke einsetzen zu

können. Dieses Bestreben geht einher damit, dass Mitarbeiter nur ein Mobilfunkgerät verwenden wollen. Aus Sicherheitsgesichtspunkten ist BYOD äußerst kritisch zu sehen, da eine Trennung zwischen beruflichen und privaten Angelegenheiten dabei kaum zu gewährleisten ist. Die Sicherheitsinfrastruktur wird deutlich komplexer, während das allgemeine Sicherheitsniveau sinkt.

Weiterführendes Sicherheit ist ein komplexes Feld, das ein hohes Maß an Expertenwissen benötigt. Wann immer möglich, sollte man deshalb auf bewährte Standardlösungen zurückgreifen, statt Eigenentwicklungen einzusetzen. Zugleich ist eine andauernde Beschäftigung mit der Sicherheit von Anwendungen und Geräten anzuraten. Die mobilen Plattformen bieten sowohl auf Geräte- und als auch auf Anwendungsebene Mittel, um einfache bis fortgeschrittene Sicherheitsanforderungen zu implementieren. Generell ist der Aspekt der Sicherheit auf mobilen Geräten an zwei Stellen zu berücksichtigen: im Rahmen der Geräteverwaltung und während der Entwicklung von Apps.

Das Mobile Device Management (MDM) bindet die Geräte sicher in das Unternehmensnetzwerk ein, wählt aus, welche Apps installiert werden können, und trifft Vorkehrungen für den Verlustfall. Diese Maßnahmen geschehen in engem Verbund mit der allgemeinen Administration der Geräte. Da die wenigsten mobilen Plattformen entsprechende Lösungen von Haus aus bereitstellen, existieren spezialisierte MDM-Lösungen (siehe auch Kapitel 7.10). Hintergrundinformationen zur allgemeinen Sicherheitsarchitektur der Plattformen liefern z. B. Apple auf [App12] und Android unter [34].

Für Entwickler von Apps sind zum einen die allgemeinen Sicherheitsvorkehrungen der Plattform relevant, weil sie das Umfeld definieren, in dem die Sicherheitskonzeption der App entworfen wird. Wenn beispielsweise die Sicherheitsarchitektur einer Plattform wie bei iOS und Android eine strikte Trennung der Daten verschiedener Apps sicherstellt, muss eine App mit mäßig sensiblen Daten gegebenenfalls keine besonderen Vorkehrungen mehr treffen, um die Datenintegrität zu gewährleisten. Zum anderen sind für fortgeschrittene Anforderungen die entsprechenden APIs der Plattform relevant. Nach dem oben erwähnten Leitsatz sind diese Eigenentwicklungen vorzuziehen, wenn sie die Sicherheitsanforderungen erfüllen. Anforderungen wie sichere Kommunikation oder zusätzlich verschlüsselte Dateien sind zum Beispiel mit den Mitteln von iOS und Android umsetzbar, ebenso stehen diverse kryptographische Algorithmen zur Verfügung. Die iOS Developer Library gibt Auskunft über die entsprechenden Frameworks [35]. Für Android finden sich Informationen unter [36].

Für verlässliche Aussagen sollte man sich in Fragen der Sicherheit nicht allein auf die Angaben und Ratschläge der Hersteller verlassen. Allgemeine Informationen aus dritter Hand zur Sicherheit mobiler Geräte gibt es zum Beispiel vom Bundesamt für Sicherheit in der Informationstechnik (BSI) [37]. Die dort bereitgestellten Informationen beziehen sich insbesondere auf die Netzinfrastruktur und Mobilfunk, die

Aussagen zu Mobilgeräten sind teilweise veraltet. Spezifische Informationen zur Sicherheit auf Smartphones, unter anderem eine Liste der größten Risiken, bietet die Europäische Agentur für Netz- und Informationssicherheit (ENISA) [38] [HD12].

Im Hinblick auf BYOD bleibt die aktuelle Entwicklung abzuwarten. Im Desktop-Bereich existieren bereits Lösungen, die etwa bei der Telearbeit zum Einsatz kommen. Virtualisierung oder der Zugriff auf Terminal-Server-Lösungen sind allerdings derzeit auf Smartphones und Tablets noch kein Thema.

7.4 Rechtliche Aspekte und Lizenzfragen

Motivation Softwareentwicklung wirft selbst in kleineren Projekten eine Reihe rechtlicher Fragen auf. Diese beziehen sich auf die Einhaltung gesetzlicher Rahmenbedingungen, Verträge mit Zulieferern und Kunden, sowie auf Lizenzen. Letztere sind sowohl bei kostenpflichtigen Softwareprodukten als auch bei freier Software zu beachten – auch für diese gibt es eine ganze Reihe von Lizenzen, die Auswirkungen auf die kommerzielle Verwendbarkeit und Weiterentwicklung haben. Dazu kommen eventuell Wartungsverträge. Schließlich können sich rechtliche Fragen im Bezug auf die Distribution der Software stellen.

Grundsätzlich sind Apps in der gleichen Weise von rechtlichen Rahmenbedingungen betroffen wie andere Software auch. Es ergeben sich allerdings zwei Themenfelder, denen ein höheres Gewicht zukommt:

- Es herrscht nur ein sehr geringes Maß an Unabhängigkeit, was unabdingbare Entwicklungswerkzeuge für die einzelnen Plattformen angeht.
- Die Distribution erfolgt – zumindest bei Apps für Endkunden bzw. die nichtfirmeninterne Nutzung – über durch Dritte zur Verfügung gestellte und verwaltete Kanäle, die Appstores.

Wenn Software für gängige PC-Plattformen wie Microsoft Windows entwickelt wird, stehen zahlreiche Entwicklungsplattformen zur Verfügung. Es kann in einer Vielzahl von Programmiersprachen entwickelt werden. Für häufig verwendete Programmiersprachen stehen nicht nur diverse Compiler, sondern sogar diverse Entwicklungsumgebungen zur Verfügung, die von unterschiedlichen Herstellern angeboten werden. Zwar gibt es eine standardisierte Windows-API, gegen die programmiert wird, aber grundsätzlich ist es möglich, auf nahezu beliebige Weise Programme für die Plattform zu erstellen. Ein *Vendor Lock-in* droht allenfalls durch die generelle Wahl der Plattform; bei Verwendung von Programmiersprachen wie C++ bzw. des auf einer virtuellen Maschine laufenden Javas hält sich aber selbst dieses Problem in Grenzen.

Im Gegensatz dazu ist die Verwendung des jeweiligen SDK bei der Entwicklung für mobile Endgeräte im Allgemeinen unumgänglich. In der Regel ist hierdurch nicht nur

die API vorgegeben. Vielmehr ist auch die Programmiersprache nicht frei wählbar. Zudem wird auf einer höheren Ebene der API angesetzt, die genutzten Funktionen bieten also einen deutlich abstrahierteren Zugang zur Plattform. Zum Teil ist die Entwicklungsumgebung einigermaßen frei wählbar, zum Teil aber auch auf ein einzelnes Produkt beschränkt (etwa Xcode von Apple). Somit ergibt sich nicht nur eine technische Abhängigkeit vom Anbieter der Plattform, sondern auch eine rechtliche. Die sich daraus ergebenden Konsequenzen sind nicht direkt kritisch – in der Regel wird kein Vertrag abgeschlossen. Nichtsdestotrotz können sich Implikationen bzw. bei Missachtung geltender Nutzungbedingungen Konflikte ergeben. Aus diesem Grund ist eine genaue Prüfung anzuraten.

Anwendungssoftware für PC und in noch stärkerem Maße Serversoftware wird in der Regel direkt vertrieben. Hinzu kommt der Vertrieb als OEM-Bundles sowie über den Einzelhandel. Das Herunterladen kostenpflichtiger Software ist eine eher neue Erscheinung und häufig ebenfalls im Direktvertrieb anzutreffen. Apps für Endkunden werden fast ausnahmslos über die Appstores vertrieben. Bei einigen Plattformen gibt es zwar theoretisch Auswahl zwischen mehreren Appstores, letztlich ist es aber zutreffend, von *einem* Appstore pro Plattform zu sprechen. Da dieser durch den jeweiligen Hersteller kontrolliert wird, müssen auch dessen Bedingungen eingehalten werden. Während die Nutzung der Appstores keine technische oder organisatorische Hürde darstellt, so ist die implizite Zustimmung zu den geltenden Rahmenbedingungen durch das Einstellen von Apps kritisch zu sehen. Das Thema kann an dieser Stelle nicht vertiefend behandelt werden, aber eine vorsichtige Prüfung ist zu empfehlen. Von einem übereilten probeweisen Einstellen von Apps ist abzuraten; vielmehr sollten die Bedingungen der Anbieters geprüft und entsprechende Fallstricke recherchiert werden.

Weiterführendes Die obigen Ausführungen sollen vor allem auf die Problematik aufmerksam machen. Eine rechtliche Beratung ist im Rahmen dieser Broschüre nicht möglich. Da rechtliche Fragen zudem nur einen Randaspekt darstellen, werden sie nicht weiterführend betrachtet.

7.5 Schulung der Mitarbeiter

Motivation In den Interviews wurde klar, dass der bisherige Ausbildungsstand bezüglich der Entwicklung von Apps vor allem auf „informellen“ Wegen erreicht wurde. So haben sich Mitarbeiter autodidaktisch in die Plattformen eingearbeitet oder von Kollegen gelernt. In einer Reihe von Fällen erfolgte die Einarbeitung sogar auf eigene Initiative in der Freizeit.

Wenn Apps regelmäßig entwickelt werden sollen, die Wartung von Apps nötig wird und generell eine ebenso professionelle Entwicklung von Apps geplant ist, wie dies bei bisher entwickelter Software der Fall ist, müssen Mitarbeiter geschult werden.

Alle anderen Ansätze sind unzureichend. Selbst, wenn Entwickler mit entsprechenden Kenntnissen neu eingestellt werden, sind Schulungen nötig, da die technische Entwicklung derzeit extrem schnell und vor allem nicht konsistent ist.

Schulungen sollten bereits geplant werden, wenn Apps in Zukunft eine größere Bedeutung zugemessen wird und diesbezügliche Entwicklungstätigkeiten demnach voraussichtlich ausgebaut werden. Hierbei sollte aufgrund der nur bedingt evolutionären Marktentwicklung besonders auf die Zukunftssicherheit geachtet werden. Die Arbeit mit benötigten APIs lässt sich auch durch das Selbststudium der dazugehörigen Dokumentation lernen. Wichtiger ist vielmehr, dass das generelle Programmieren für mobile Endgeräte, die Besonderheiten der Plattformen und Gestaltungsmöglichkeiten vermittelt werden. Schulungen sollten darauf abzielen, das Wissen zu vermitteln, das einen Anwendungsentwickler in die Lage versetzt, benutzerfreundliche Apps zu entwickeln und die Besonderheiten mobiler Endgeräte zu berücksichtigen. Ob darüber hinaus eine plattformspezifische Ausbildung erfolgen soll, muss individuell entschieden werden. Anzumerken ist allerdings, dass die Plattform-APIs relativ einfach zu überblicken und zu nutzen sind.

Weiterführendes Da die Schulung von Mitarbeitern nur ein Randthema der Broschüre ist, wird sie an dieser Stelle nicht weiter betrachtet. Es ist zudem davon auszugehen, dass es keine gravierenden Unterschiede zu Schulungen für andere Themen der Softwareentwicklung gibt und das hierbei übliche Vorgehen übertragen werden kann. Es genügt an dieser Stelle deshalb, auf die allgemeine Bedeutung aufmerksam zu machen.

7.6 Energieeffiziente Apps

Motivation Mobile Endgeräte müssen netzunabhängig mit Energie versorgt werden. Mit der Entwicklung immer leistungsfähigerer Handys einher ging die Entwicklung von Akkus, die mehr Leistung zur Verfügung stellen. Allerdings ging dies bereits bei klassischen Handys nur selten mit verlängerten Laufzeiten einher. Vielmehr wurde die zusätzliche Kapazität für die verbesserte Rechenleistung und vor allem verbesserte Displays genutzt. Ebenfalls schon bei Handys stieg die tägliche Nutzungdauer: wurden die Geräte ursprünglich nur zum Telefonieren benutzt und in minimalem Umfang auf Zusatzfunktionen zugegriffen (etwa als Ersatz der Armbanduhr zum Anzeigen der Zeit), kamen mit der Zeit zusätzliche Nutzungsmöglichkeiten hinzu. Diesen Trend führten die Smartphones fort.

Die Akkus heutiger Smartphones sind extrem leistungsfähig. Allerdings verfügen diese Geräte über eine Rechenleistung, die im Vergleich zu wenige Jahre älteren Geräten zum Teil um mehrere Größenordnungen größer ist. Displays sind häufig hochauflösend und hell. Vor allem die Nutzungsgewohnheiten haben sich geändert. Telefonieren ist zu einer Funktion unter vielen geworden. Mitunter summiert sich

die tägliche Nutzungszeit von Smartphones auf mehrere Stunden. In der Folge muss der Akku von vielen Smartphone-Nutzern täglich geladen werden – dies ist für sie lästig. Zudem schränkt es die Mobilität ein: wenn bestimmte Funktionen oder Apps besonders energiehungrig sind, sinkt deren empfundener Nutzen, da sie z. B. bei niedrigem Ladestand des Akkus nicht mehr gestartet werden oder Benutzer sich unwohl fühlen, wenn sie kein Ladekabel mit sich führen.

Aus oben genannten Gründen ist die Energieeffizienz von Apps von großer Bedeutung. Zwar wird intensiv an noch leistungsfähigeren Akkus geforscht. Ob diese aber auf kurze Sicht schneller leistungsfähig werden, als neue Smartphones die zusätzliche Energie für neue Funktionen nutzen, ist auch mit Blick auf die Historie fragwürdig. Somit spielt die Energieeffizienz besonders bei häufig genutzten Apps eine Rolle. Sie ist insbesondere dann wichtig, wenn der – tatsächliche oder empfundene – Energieverbrauch durch eine App hoch ist.

Weiterführendes Es gibt derzeit noch keine umfangreichen Erfahrungen mit der energieeffizienten Programmierung von Apps. Allerdings lassen sich dennoch einige Maßnahmen treffen:

- Mobile Plattformen wie Android bieten die Möglichkeit, den Energieverbrauch differenziert zu betrachten. Auch wenn nicht klar ist, wie genau solche Auswertungen sind, können sie als Anhaltspunkte genutzt werden.

- Der Energieverbrauch sollte beim Testen beobachtet und mit dem bekannten Verbrauch existierender Apps ins Verhältnis gesetzt werden. Ein hoher Verbrauch sollte zu einer Überprüfung der App führen. Er ist nicht nur für sich problematisch, sondern deutet auf weitere Probleme (z. B. ineffiziente Algorithmen) hin.

- Probleme können auch unüberlegte Designentscheidungen oder schlechter Programmierstil hervorrufen. Push-Dienste etwa verbrauchen typischerweise viel Energie, wenn sie als eigenständige Pull-Dienste emuliert werden. Ein Beispiel ist eine Benachrichtigung über neu eingegangene E-Mails, die intern alle 10 Sekunden den Server kontaktiert. Hierdurch werden permanent Ressourcen beansprucht und Leerlaufphasen des Geräts verhindert. Eine energiebewusste Architektur verwendet in diesem Fall z. B. die von der Plattform bereitgestellten Pushdienste,[1] die besser in das System integriert sind.

- Wenn Funktionen genutzt werden, die bekanntermaßen viel Energie verbrauchen, sollte besonders darauf geachtet werden, diese nur im absolut nötigen Maße einzubinden. Beispiele sind der Zugriff auf WLAN und Bluetooth sowie aufwendige Berechnungen.

[1] Solche Pushdienste sind z. B. Google Cloud Messaging (GCM) für Android [39] oder Apple Push Notification Service (APNS) für iOS [40].

- Ist ein hoher Energieverbrauch durch die Art der App zu erwarten, sollte der Nutzer darauf hingewiesen werden. Eine App, die beispielsweise kontinuierlich die Position des Geräts per GPS bestimmen muss, wird zwangsläufig energieintensiv sein.

Je nach Einsatzzweck ist abzuwägen, wie stark auf den Energieverbrauch geachtet werden muss. Dass ein 3D-Spiel nicht besonders energieeffizient zu realisieren ist, leuchtet ein. Nichtsdestotrotz lässt es sich eventuell an einigen Stellen optimieren – idealerweise ist das Ergebnis wahlweise eine höhere Framerate oder ein niedrigerer Stromverbrauch. Eine App, bei der ein animiertes Firmenlogo eingebunden wird (womöglich mit 3D-Effekt), ist sicherlich nett. Bei häufiger Nutzung ist ein statisches Logo aber vermutlich angebrachter. Wichtig ist allerdings, nicht zu mutmaßen. Zwar lässt sich grob unterstellen, dass alles, was typischerweise rechnerisch aufwendig ist, auch viel Energie verbraucht. Nachmessen und Ausprobieren sollten aber gerade bei umfangreicheren Apps zur Entwicklung dazugehören.

7.7 App-Roaming und Session-Management

Motivation Klassische Client-Server-Anwendungen, aber auch viele Web-basierte Anwendungen müssen Benutzersitzungen verwalten. Das *Session-Management* sorgt dafür, dass ein Benutzer nach einer Authentifizierung (*Login*) auf bestimmte Ressourcen zugreifen kann, mit vorgegebenen Rechten ausgestattet ist und gegebenenfalls eine personalisierte Ansicht erhält. In den meisten Fällen ist ein bedeutender Teil der Logik dabei serverseitig implementiert; der Client sendet Befehle und Daten und stellt als Antwort erhaltene Daten dar. Damit nicht bei jedem Arbeitsschritt (oder gar bei jeder Netzwerktransaktion) eine erneute Authentifizierung erforderlich ist, wird in der Regel eine *Sitzung* aufrecht erhalten.

Der Sitzungsbegriff ist dabei weit gefasst. Keineswegs müssen permanent Daten übertragen werden. Sitzungen können theoretisch sogar beliebig lange bestehen bleiben – praktisch werden sie aber durch einen *Timeout* begrenzt. Sitzungen sind sowohl bei zustandsbehafteten als auch bei zustandslosen Protokollen nötig. Ressourcen für eine Sitzung müssen nicht notwendigerweise während der Sitzung permanent bereitgestellt werden. Vielmehr können sie auch bei Anfragen erneut reserviert werden.

Allen Sitzungen gemein ist allerdings, dass es einen Mechanismus geben muss, der die Sitzung zwischen Server und Client aufrecht erhält. Es kann sich dabei um sehr einfache, aber auch um komplizierte Mechanismen handeln. Sie unterscheiden sich vor allem im Aufwand sie zu implementieren, in der Robustheit gegenüber Störungen und in Sicherheitsaspekten. In Bezug auf die erforderliche Rechenleistung und die benötigte Bandbreite gibt es ebenfalls Unterschiede, die aber in vielen Fällen nur eine untergeordnete Rolle spielen werden.

Ein Beispiel für ein sehr simples Sitzungsmanagement ist ein Client, der an einen

Server einen Benutzernamen und das zugehörige Passwort sendet. Sind die Anmeldedaten korrekt, erhält er daraufhin eine zufällige Zeichenkette. Bei zukünftigen Anfragen sendet er diese mit und wird so auf dem Server korrekt zugeordnet. Um bei dieser Methode ein akzeptables Maß an Sicherheit zu gewährleisten, sollte die Zeichenkette ausreichend lang und die Verbindung zuverlässig verschlüsselt sein. Für sicherheitskritische Bereiche ist sie allerdings weniger geeignet.

Weiterführendes Auf weiterführende Informationen wird an dieser Stelle verzichtet, da es nicht möglich ist, pauschale Empfehlungen zu geben. Da die Sitzungsverwaltung bereits bedeutsam für Desktop-Anwendungen ist, lassen sich keine auf Apps zugeschnittenen Hinweise geben. App-Roaming – also die Aufrechterhaltung von Sitzungen über verschiedene physische und/oder logische Kontexte hinweg – indes ist zu neu, als dass bereits Erfahrungen vorliegen würden.

7.8 Offline-Funktionalität

Motivation Mobile Anwendungen müssen in besonderem Maße Vorkehrungen für den Fall treffen, dass keine Netzwerkverbindung verfügbar ist. Dabei ist zwischen temporären, kurzzeitigen *Verbindungsabbrüchen*, z. B. aufgrund instabiler drahtloser Verbindungen, und genereller *Nichtverfügbarkeit* eines Netzzugangs zu unterscheiden. Erstere erfordern vor allem eine gute Fehlerbehandlung und eine entsprechende temporäre Reaktion auf den Verbindungsabbruch. Ist während der Ausführung einer App generell oder potenziell kein Netz verfügbar, z. B. aufgrund räumlicher Gegebenheiten, sind Strategien nötig, die auf längerfristige Offlinesituationen eingestellt sind.

Smartphones und die auf diesen laufenden Apps operieren in prekären Netzwerkumgebungen, da sie zum einen zwangsläufig drahtlos auf das Internet zugreifen und zum anderen oftmals außerhalb des jeweiligen Heimnetzwerks (firmeninternes Netz, bzw. WLAN des Verbrauchers) eingesetzt werden. Auf Reisen, insbesondere auf Auslandsreisen, kann dieser Zustand sogar längerfristig akut sein. Die Problematik ähnelt im Groben der auch bei Notebooks im mobilen Einsatz anzutreffenden Situation, unterscheidet sich aber in zwei Aspekten gegenläufiger Tendenz:

1. Die Problematik prekärer Netzkonnektivität wird dadurch verschärft, dass Smartphones und ähnliche Geräte bedeutend häufiger mobil eingesetzt werden als Notebooks. *Mobil* wird hier im Sinne von „außerhalb einer fixen Zahl bekannter Heimnetze (WLANs)“ verstanden. Ein Anschluss an kabelgebundene Netze ist bei Smartphones nicht vorgesehen. Im mobilen Betrieb ist man auf Mobilfunknetze angewiesen und muss deshalb häufig mit unterdurchschnittlicher Verbindungsqualität in Bezug auf Bandbreite und Stabilität rechnen.

2. Positiv auf die Netzwerkkonnektivität wirkt der einfache Zugriff auf Mobilfunknetze. Dank des integrierten Zugangs zum Mobilfunknetz haben Smartphones anders als Notebooks häufig auch dort eine Verbindung, wo kein bekanntes WLAN zur Verfügung steht. Ein Notebook ist in solchen Situation ohne zusätzliche UMTS-Hardware o. Ä. ohne Netzempfang.

Aufgrund dieser gegenläufigen Effekte sind zwei Denkrichtungen verständlich, wie sie unter anderem in den Interviews zum Ausdruck kamen. Teilweise schätzen die Verantwortlichen die verbesserte Konnektivität von Smartphones gegenüber anderen Geräten, aufgrund derer Apps gestaltet werden können, die stark von ständiger Erreichbarkeit profitieren. Für andere Interviewpartner war die mangelhafte Konnektivität ein entscheidender Aspekt bei der Entwicklung von Business Apps, der besonderer Berücksichtigung bedarf. Dementsprechend hängt es häufig von den Einsatzszenarien der jeweiligen App ab, in welchem Umfang und welche Vorkehrungen zum Umgang mit Verbindungsproblemen getroffen werden müssen. Eine wie auch immer geartete Rücksichtnahme darauf im Rahmen der Konzeption und Entwicklung wird aber in den meisten Fällen notwendig sein. Der folgende Abschnitt stellt kurz Strategien zum Umgang mit dieser Problematik vor.

Weiterführendes Die Problematik mangelhafter Netzwerkverbindungen ist auf allen Ebenen des Entwicklungsprozesses von Bedeutung: Im Rahmen der Anforderungserhebung sind die Szenarien zu untersuchen, in denen die App eingesetzt werden soll, und davon ausgehend ist das gewünschte Verhalten in diesen Situationen zu definieren. Während des Entwurfs sind diese Anforderungen entsprechend zu berücksichtigen, dabei hat insbesondere der Umgang mit längerfristigen Offlinephasen Einfluss auf die Architektur der Anwendung. Die Möglichkeit temporärer Verbindungsabbrüche ist im Rahmen der Implementierung überall dort einzukalkulieren, wo Netzwerkaufrufe getätigt werden. An diesen Stellen werden entsprechend robuste Implementierungen benötigt. Dem Testen kommt in diesem Rahmen eine besondere Bedeutung zu. Jede Funktion der App, die von Netzwerkverbindungen abhängig ist, sollte immer auch in Situationen getestet werden, in denen keine Verbindung verfügbar ist.[2] Hier sind jeweils unterschiedliche Konstellationen zu testen, z. B. kein Netz in Reichweite, Timeout oder Abbruch während der Übertragung. Die Unterstützung der mobilen Plattformen für das Testen mit entsprechenden Netzwerkparametern ist allerdings bisher ungenügend. Netzwerkgeschwindigkeit und -latenz des Android-Emulators sind beim Start anpassbar, aber nicht dynamisch und automatisiert zur Laufzeit. Das vollständige Abschalten der Netzwerkverbindung funktioniert nicht unter allen Android-Versionen zuverlässig. Der iOS-Simulator unterstützt ebenfalls keine dynamische und automatisierte Verwaltung der Netzwerkverbindungen. Auch nach Release der Anwendungen muss kontinuierlich geprüft werden, ob veränderte

[2] Das kann auf alle externen Ressourcen, von denen eine App abhängt, verallgemeinert werden.

Nutzungsszenarien neue Vorkehrungen gegen Verbindungsprobleme notwendig machen.

Hinsichtlich der Maßnahmen ist wiederum zwischen Verbindungsabbrüchen und längerfristiger Nichtverfügbarkeit zu unterscheiden. Temporären Konnektivitätsproblemen kann häufig durch einen sauberen Programmierstil begegnet werden, der Fehlerfälle explizit abfängt und den Verbindungsversuch angemessen wiederholt.

Für Szenarien mit längerfristiger Nichtverfügbarkeit sind Apps zu bevorzugen, die einen möglichst großen Teil des Inhalts *lokal vorhalten*. Diese Inhalte können zum Beispiel als Bestandteil des nativen Installationspakets der App auf dem Gerät installiert werden. Über den Update-Mechanismus der jeweiligen Plattform können in diesem Fall die Daten aktualisiert werden. Ein solches Update unterliegt aber den Bedingungen des jeweiligen Appstores und muss zumeist manuell vom Nutzer angestoßen werden. Eine größere Flexibilität bietet eine Speicherung der Daten nach erstmaligem Abruf durch die Anwendung selbst (*Caching*). Hierzu sind aber entsprechende architektonische Vorkehrungen zu treffen, die die Komplexität der Anwendung erhöhen, z. B. für die notwendig *Synchronisierung*. Für Funktionen, die keine Inhalte präsentieren, sondern Benutzereingaben entgegennehmen, die schlussendlich an einen externen Server übermittelt werden sollen, sind wiederum spezifische Vorkehrungen notwendig. Die Eingabedaten müssen auf dem Gerät zwischengespeichert werden, bis sie übertragen werden können. Falls sich die eingegeben Informationen auf andere Daten beziehen, die sich zwischenzeitlich geändert haben könnten (z. B. bei kollaborativen Anwendungen), sind Synchronisierungs- und Zusammenführungsmechanismen notwendig, die die Komplexität der Anwendung beträchtlich erhöhen.

Grundsätzlich haben Apps, die in Form eines nativen Pakets ausgeliefert und installiert werden, mehr Optionen, um präventiv Netzwerkprobleme zu berücksichtigen. Im Gegensatz zu Web-Apps sind sowohl native Apps als auch hybride Apps, die z. B. mit PhoneGap erstellt wurden, nach Installation ständig verfügbar. Die neuen Möglichkeiten von HTML5 im Bereich „Offline-Storage“ versprechen aber zumindest in Teilen Abhilfe. Standards wie Web Storage und Offline Web Applications (Application Cache API) ermöglichen es, besuchte Webseiten in einem Cache vorzuhalten. Die Seiten werden aber erst nach dem erstmaligen Abruf im Cache gespeichert, zudem erwies sich die Unterstützung für Offline Web Applications in einigen Browsern als unzuverlässig.

7.9 Testen

Motivation Mobile Anwendungen stellen Softwaretester vor eine Vielzahl von Herausforderungen; außerdem ist ein sorgfältiger Test mobiler Anwendungen notwendig, um auf allen Geräten einen positiven Eindruck beim Benutzer zu hinterlassen und die Funktionsfähigkeit sicherzustellen. Die besonders große Heterogenität auf Ebene der mobilen Plattformen und Geräte stellt besondere Anforderungen hinsichtlich

sorgfältiger Tests. Unter den großen mobilen Plattformen sticht insbesondere Android aufgrund der Vielzahl an unterschiedlichen Geräten hervor. Die Unterschiede beschränken sich nicht auf den Gerätehersteller, sondern betreffen nahezu alle Spezifika, die die Funktionsfähigkeit der App direkt beeinflussen: von Bildschirmgröße über Eingabegeräte bis hin zu unterstützten Funktionen.

Um das tatsächliche Benutzererlebnis vor dem Hintergrund der unterschiedlichen Ausstattung testen zu können, müssen die Anwendungen auf realen Geräten erprobt werden. Die Tatsache, dass diese von der übrigen Entwicklungsumgebung auf dem Computer der Entwickler separiert sind, erschwert Tests abseits einfacher Unit-Tests für die Geschäftslogik zusätzlich. Hier sind regelmäßig manuelle Tests auf den physischen Geräten notwendig, die nur bedingt automatisierbar sind und eine grundsätzlich andere Herangehensweise als Desktopanwendungen erfordern.

Bei der Mehrzahl von Apps ist die grafische Oberfläche ein entscheidendes Merkmal; die Spezifika mobiler Geräte erhöhen die Bedeutung einer guten Benutzeroberfläche im Vergleich zu Desktopanwendungen deutlich. Die Bedienbarkeit ist ein wesentlicher Faktor neben der allgemeinen Funktionsfähigkeit und kann vollständig nur auf dem Gerät selbst beurteilt werden. Mobile Anwendungen greifen zudem häufig auf Hardwarefunktionen wie Kamera oder GPS zurück, die ebenfalls nur auf dem Gerät selbst vollständig testbar sind. Selbst Standardelemente wie die Netzwerkverbindung erfordern spezielle Beachtung (vgl. Kapitel 7.8).

Generell zeichnet sich das Testen mobiler Endgeräte dadurch aus, dass *Kontextwechsel* betrachtet werden müssen. Beispiele sind der Ausfall der Netzwerkverbindung oder der Wechsel von WLAN auf UMTS; generell erfordern alle Arten von Ereignissen besondere Aufmerksamkeit, die zwar nicht wie Benutzereingaben auf eine App selbst wirken, aber den Kontext, in dem sie läuft, betreffen. Aus Sicht des Testens ergibt sich hierdurch eine Matrix: die denkbaren Testfälle müssen gegebenenfalls in verschiedenen Kontexten wiederholt oder sogar während eines Kontextwechsels ausgeführt werden.

Zusammenfassend ist festzuhalten, dass nahezu alle Apps gründlichen Tests auf den physischen Geräten unterzogen werden müssen. Einen einzelnen Gerätetyp zu testen, genügt im Allgemeinen nicht, um sicherzustellen, dass die App auf allen relevanten Geräten zufriedenstellend läuft.

Weiterführendes Für die meisten mobilen Plattformen stehen Emulatoren bereit, die eine Ausführung der App auf dem Computer des Entwicklers ohne Zugang zu einem physischen Gerät ermöglichen. Teilweise bieten diese Emulatoren auch die Möglichkeit, Tests zu automatisieren. Die Emulatoren sind aber vor allem dazu geeignet, die allgemeine Funktionsfähigkeit während der Entwicklung zu testen. Um eine zufriedenstellende Ausführung der App zu gewährleisten, sind in jedem Fall wiederholte, physische Tests auf den zu unterstützenden Geräten notwendig.

Um Entwicklern diese kostspielige und zeitaufwändige Arbeit zu erleichtern, gibt

es *Cloud*-basierte Lösungen für das Testen von Apps auf mobilen Geräten. Keynote DeviceAnywhere [41] bietet zum Beispiel Zugriff auf eine Vielzahl unterschiedlicher physischer Geräte, die aus der Ferne steuerbar sind. Ein derartiger Dienst stellt diverse Geräten vergleichsweise kostengünstig und schnell zur Verfügung. Neben manuellen Tests sind auch automatisierte Tests möglich. So können Regressions- und Kompatibilitätstest mit überschaubarem Aufwand und begrenztem Budget durchgeführt werden. Die entfernte Steuerung ermöglicht aber nur bedingt Aussagen über die Bedienbarkeit der Anwendungen, da sie wegen der fehlenden haptischen Rückmeldung natürlich kein „Gefühl“ für das Verhalten der Anwendungen übermitteln kann. Dazu genügt aber gegebenenfalls eine kleinere Auswahl physischer Geräte vor Ort.

Bei intern einzusetzenden Apps ist es gegebenenfalls denkbar, die Entwicklung auf einen Gerätetyp zu beschränken, wenn entsprechende Beschränkungen im Unternehmen durchgesetzt werden können. Mittelfristig wird diese Beschränkung aufgrund der Fluktuation des mobilen Gerätemarktes aber nur schwer aufrechtzuerhalten sein.

7.10 Mobile Device Management

Motivation Mit dem Aufkommen leistungsstarker mobiler Geräte rückt deren Administration verstärkt in den Blickpunkt und das Aufgabenfeld der Administratoren in Unternehmen. So wie bisher schon Desktop-PCs und Notebooks aufgesetzt und administriert werden mussten, stehen ähnliche und neue Aufgaben auch hinsichtlich mobiler Geräte an, um die sich das sogenannte MDM kümmert. MDM-Lösungen übertragen bekannte Administrationskonzepte und Softwareverteilungsprinzipien auf mobile Geräte. Der aufstrebende Markt des MDM steht dabei vor besonderen Herausforderungen.

Aufgrund der heterogenen Plattformlandschaft ist es unumgänglich, Geräte verschiedener Plattformen verwalten zu können. Der Verzicht auf plattformübergreifende Lösungen ist angesichts der Unwägbarkeiten im Markt der mobilen Plattformen keine gangbare Alternative, wenn ein langfristiger Einsatz geplant ist. Die Plattform- und Gerätevielfalt führen bei individualisierten, plattformspezifischen Lösungen zu hohen Kosten für Administration und Support. Die geschlossene Natur und die Verschiedenartigkeit existierender Plattformen, insbesondere in Bezug auf administrative Funktionalitäten, wie sie für MDM benötigt werden, erschweren die Entwicklung geeigneter MDM-Produkte. Der MDM-Markt ist dementsprechend (noch) rascher Entwicklung unterworfen.

Anforderungen an MDM umfassen zunächst die initiale Konfiguration der Geräte, die nötig ist, um das Gerät überhaupt einsetzen zu können. Dazu gehören zum Beispiel die Einrichtung des Benutzerkontos, der Telefonie und des Netzwerkzugriffs. Darüber hinaus sollte das MDM auch Zugriff auf Einstellungen geben, die es erlauben, die Oberfläche des Geräts an ein einheitliches Corporate Design anzupas-

sen. Bezüglich der Konfigurationsmöglichkeiten bedeutsamer sind aber Sicherheitseinstellungen und deren konsequente Durchsetzung auf technischer Ebene. Wie in Kapitel 7.3 angesprochen unterstützen mobile Plattformen diesbezüglich von Haus aus nicht alle Anforderungen, sodass das Mobile Device Management teilweise eigene Restriktionsmechanismen implementieren muss.

Neben der Konfiguration ist auch die Verteilung von Daten eine Anforderung an MDM. Dies umfasst zum einen Daten aus dem persönlichen Informationsmanagement (Personal Information Manager, PIM) wie E-Mails, Kontakte, Termine und Aufgaben, in einem weiteren Sinn auch Dokumente. Über eine bloße Verteilung hinaus wird eine Synchronisierung mit den jeweiligen Servern im Unternehmen erwartet: auf mobilen Geräten erstellte Termine oder Dokumente sollen nahtlos auf dem Computer im Büro zur Verfügung stehen und umgekehrt. MDM sollte außerdem die Verteilung von Apps und Updates unterstützen. Die Verteilungsfunktionalitäten erfordern wiederum abgestimmte Sicherheitsrestriktionen.

Datensicherung und Fehlerdiagnose sind weitere wünschenswerte Features. Jede der vorgenannten Funktionen sollte soweit möglich „Over-the-Air“, d.h. in Fernwartung ohne physische Anbindung des Geräts, möglich sein.

Weiterführendes Bisherige Standardlösungen wie der Rückgriff auf Microsoft Exchange *ActiveSync* zur Synchronisierung bestimmter Datenkategorien, vorwiegend aus dem PIM-Umfeld, erfüllen nur selten fortgeschrittene Anforderungen. Sie sind nur dann eine gangbare Alternative, wenn das mobile Gerät betrieblich ausschließlich als persönlicher Informationsverwalter genutzt wird. Gleichzeitig haben entsprechende Standardlösungen Defizite in der Umsetzung von Unternehmensrichtlinien zur Sicherheit. Wie erläutert machen die PIM-Funktionalitäten nur einen kleinen Teil der Anforderungen an MDM aus, sodass zumeist spezialisierte MDM-Lösungen benötigt werden.

Der aufstrebende MDM-Markt ist dabei schnellem Wandel unterworfen und noch unübersichtlich. Spezialisierte Anbieter versuchen genauso wie etablierte Softwareunternehmen, Marktanteile auf dem schnell wachsenden Markt zu erlangen und zu steigern. Einen ersten Überblick über die verfügbaren Lösungen gibt zum Beispiel das Marktforschungsunternehmen Gartner mit seinem „Magic Quadrant“ für Mobile Device Management [RGB12][42, 43], den derzeit vor allem spezialisierte Anbieter wie MobileIron [44] anführen.

Grob lassen sich zwei Varianten des MDM unterschieden: sogenannte On-Premise-Lösungen, die im Unternehmen installiert und von diesem betrieben werden, oder cloudbasierte Dienste, die gerade für kleinere Unternehmen attraktiv sein können, weil ein wesentlicher Teil des Installations- und Konfigurationsaufwands ausgelagert werden kann.

Kapitel 8

Schlussbetrachtungen

Die vorliegende Broschüre erörtert vier Aspekte der Entwicklung von Business Apps. Im Einzelnen haben vier Abschnitte

- den Status quo der App-Entwicklung in Unternehmen Nord-Westfalens beschrieben (Kapitel 4),
- einen Überblick über die Entwicklung von Apps für mehrere Plattformen geschaffen (Kapitel 5),
- erste Handlungsempfehlungen aufgezeigt (Kapitel 6) und
- Hinweise zu ausgewählten Aspekten der App-Entwicklung gegeben (Kapitel 7).

Hierzu wurden zunächst wichtige Begriffe geklärt, um dann einen Überblick über das zugrundeliegende Projekt zu geben. Dieses wurde mit Unternehmen der Region im Rahmen der Tätigkeit des Instituts für Angewandte Informatik (IAI) durchgeführt. Im Folgenden wurde der Status quo aufgezeichnet, zunächst aufbauend auf einer Umfrage und dann im Detail anhand der Ergebnisse aus geführten Experteninterviews. Festzuhalten ist ein grundsätzlich sehr großes Interesse, mobile Applikationen für geschäftliche Zwecke zu nutzen und zu entwickeln. Auch wenn der Erfahrungsstand in den Unternehmen nicht einheitlich ist, ließ er sich vielfach als rudimentär beschreiben. Derzeit gibt es noch eine Reihe von technischen und wirtschaftlichen Fragestellungen, die vor einem breiteren Einsatz von Apps geklärt werden müssen.

Ein gewichtiges Problem für die Entwicklung mobiler Anwendungen stellt die Plattformheterogenität dar. Apps für Android, iOS und ggf. weitere Plattformen separat nativ zu implementieren ist oftmals zu aufwendig. Daher wurden in dieser Broschüre mögliche Ansätze für die Cross-Plattform-Entwicklung beschrieben. Ziel ist es, den üblicherweise mit der Zahl der unterstützten Plattformen skalierenden Aufwand deutlich zu senken. Viele der zur Verfügung stehenden Lösungen sind noch nicht ausgereift oder experimentell. Nichtsdestotrotz existieren mehrere Rahmenwerke bzw. Herangehensweisen, die erfolgversprechend sind. Ausgangspunkt

sollte jeweils eine detaillierte Anforderungsanalyse sein, aufgrund derer die Art der zu entwickelnden App festgelegt werden kann. Für viele Teilnehmer ist es wichtig, Apps mit einem nativen Look & Feel bereitstellen zu können. Diese entsprechen in Aussehen und Nutzungsverhalten genau dem, was einzelne Nutzer von ihrem bevorzugten Endgerät gewohnt sind. Lösungen, die zugleich Cross-Plattform-Entwicklung und natives Look & Feel unterstützen, überzeugen bisher nicht vollständig.

Wegen des noch geringen Erfahrungsschatzes konnten nur einige Handlungsempfehlungen zur App-Entwicklung aus den Beschreibungen der Interviewpartner zu erfolgreichen Vorgehensweisen abgeleitet werden. Hier liegt ein wesentlicher Unterschied etwa im Vergleich zur IAI-Broschüre zum Thema Softwaretests [MK10]. Allerdings hat sich in den Interviews herausgestellt, dass eine Reihe von Themen im Rahmen der App-Entwicklung besondere Relevanz hat. Diese wurden als *ausgewählte Aspekte* vorgestellt. Für einige Aspekte konnten bereits erste Tipps und weiterführendende Hinweise zusammengestellt werden.

Es ist zu erwarten, dass die Verbreitung von mobilen Endgeräten weiterhin zunimmt und diese gleichzeitig noch vielseitiger und universell nutzbarer werden. Daher ist die Entwicklung von Plattformen, Rahmenwerken und Entwicklungsmethoden für Apps äußerst spannend. Zugleich ist nicht zu erwarten, dass kurzfristig Investitionssicherheit etwa bezüglich anzuschaffender Geräte oder sich anzueignender Technologien eintreten wird. Die Entwicklung muss daher genau beobachtet werden; ein besonnenes Vorgehen – wie von den Teilnehmern gewählt – erscheint ratsam.

Im Projekt und auch in der Forschungsarbeit zu mobilen Applikationen hat sich gezeigt, dass erheblicher Forschungsbedarf besteht. Dabei sind sowohl grundsätzliche Fragen zu klären als auch Erfahrungen in der Entwicklung und Nutzung von Apps zu sammeln. Anstöße hierzu geben die ausgewählten Aspekte in Kapitel 7. Einer dieser Aspekte oder eine Kombination könnten auch Ausgangspunkt eines weiteren Projekts im Rahmen des IAI sein. Hervorzuheben ist vor allem die weitgehend ungeklärte, nicht ganzheitlich gelöste Sicherheitsproblematik bei der Nutzung mobiler Endgeräte.

Anhang A

Fragebogen

Der Fragebogen auf den folgenden Seiten wurde von den Teilnehmern des Projekts im Vorfeld der Interviews ausgefüllt. Die Umfrage diente zum einen der Vorbereitung der Interviews, zum anderen geben ihre Ergebnisse in Kapitel 4.2 einen ersten Einblick in den Status quo der App-Entwicklung.

Name des Unternehmens: _____

Frage		
1.	Wie viele Mitarbeiter hat Ihr Unternehmen?	_____
2.	Wie viele Mitarbeiter Ihres Unternehmens arbeiten in der Softwareentwicklung?	_____
3.	In wessen Auftrag entwickelt Ihr Unternehmen Software?	☐ Im Kundenauftrag Für interne Auftraggeber ... ☐ ... zur internen Nutzung ☐ ... zum Angebot an Kunden
4.	Von wem wird die von Ihrem Unternehmen entwickelte Software genutzt?	☐ Eigene Mitarbeiter ☐ Mitarbeiter des Kunden ☐ Privatanwender
5.	Entwickelt Ihr Unternehmen Anwendungen für mobile Endgeräte (sog. *Apps*)?	◯ Ja - weiter mit Frage 8 ◯ Nein - weiter mit Frage 6

6.	Warum entwickelt Ihr Unternehmen bisher keine Apps?	☐ Kein Bedarf ☐ Nutzen fraglich ☐ Fehlende Erfahrung ☐ Kein Budget ☐ Andere: _____
7.	Plant Ihr Unternehmen in Zukunft Apps zu entwickeln?	◯ Ja - weiter mit Frage 14 ◯ Nein - Umfrage beendet

8.	Wie viele Apps hat Ihr Unternehmen bisher entwickelt?	_____
9.	Seit wann entwickelt Ihr Unternehmen Apps?	_____
10.	Für welche Plattformen hat Ihr Unternehmen schon Apps entwickelt?	☐ Apple iOS (iPhone, iPad, ...) ☐ Google Android ☐ Windows Mobile/Phone ☐ Symbian (Nokia) ☐ Blackberry ☐ Andere: _____
11.	Welcher Anteil der Entwickler Ihres Unternehmens hat Erfahrung in der Entwicklung mobiler Anwendungen?	_____ %
12.	Werden Apps in Ihrem Unternehmen durch gesonderte Entwickler oder eine gesonderte Abteilung entwickelt?	◯ Nein ◯ Gesonderte Entwickler ◯ Gesonderte Abteilung
13.	Wie verhält sich der Aufwand für die Entwicklung von Apps im Vergleich zum Aufwand für die Entwicklung klassischer Anwendungen? (von 1 = *deutlich aufwendiger* bis 5 = *deutlich einfacher*)	1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯

Weiter mit Frage 14 auf der nächsten Seite ...

	Wie wichtig ist Ihrem Unternehmen …? (von 1 = *sehr wichtig* bis 5 = *gar nicht wichtig*)	1	2	3	4	5
14.	… ein natives Look-and-Feel (plattformtypisches Aussehen und Bedienung) der mobilen Anwendungen?	○	○	○	○	○
15.	… der Zugriff auf die Kamera des mobilen Geräts?	○	○	○	○	○
16.	… der Zugriff auf andere gerätespezifische Funktionen, wie z.B. Geokoordinaten oder Bewegungssensoren?	○	○	○	○	○
17.	… die Unterstützung mehrerer mobiler Plattformen?	○	○	○	○	○
18.	Für welche Plattformen plant Ihr Unternehmen in Zukunft Apps zu entwickeln?	☐ Apple iOS (iPhone, iPad, ...) ☐ Google Android ☐ Windows Mobile/Phone ☐ Symbian (Nokia) ☐ Blackberry ☐ Andere:				

Vielen Dank, dass Sie sich die Zeit genommen haben, das Formular auszufüllen!

Anhang B

Interviewleitfaden

Die Interviews folgten wie in Kapitel 2.4 beschrieben grob dem folgenden Leitfaden.

1. Begrüßung
    1. Klärung der beteiligten Personen.
    2. Klärung des Begriffs „Business App“.
    3. Erläuterung des Vorgehens.
    4. Wünsche im Bezug auf das Interview
2. Status quo der App-Entwicklung im Unternehmen
    1. Werden bereits Apps entwickelt?

       Wenn ja:

        1. Welche Apps wurden entwickelt bzw. befinden sich in der Entwicklung?
        2. Welcher Art sind diese Apps, lässt sich eine Kategorisierung vornehmen?
        3. Welche Plattformen werden durch die Apps unterstützt?
        4. Wie fiel die Entscheidung hierfür?
        5. Ist diese Entscheidung zufriedenstellend?
        6. Welche Distributionswege werden gewählt?
        7. Welche Entwicklungsmethoden werden benutzt? Welche Besonderheiten gab es bei der Entwicklung? (Bei Entwicklung für mehrere Plattformen ganz konkret in Bezug hierauf.) Gestaltet sich die Entwicklung anders als für klassische Software?
        8. Wie viele Entwickler stehen für die Entwicklung mobiler Anwendungen zur Verfügung? Woher stammt die Erfahrung in diesem Bereich?
        9. Haben Sie Erfahrungen bei der Entwicklung gemacht, die Sie als erfolgreiche Strategie (best practice) benennen könnten? (Insbesondere in Bezug auf Cross-Plattform-Entwicklung.)

10. Würden Sie von bestimmten Vorgehen oder Entwicklungsmethoden aufgrund schlechter Erfahrungen abraten?
11. Ist das bisherige Vorgehen bei der Entwicklung technisch und betriebswirtschaftlich zufriedenstellend?
12. Wo wird Verbesserungspotenzial gesehen?

Wenn nein:

Gab es bisher keinen Bedarf oder gibt es andere Gründe? (Hierbei sind Probleme wirtschaftlicher und technischer Natur zu klären.)

2. Welchen Nutzen sehen Sie in der Entwicklung von Apps?
3. Sind Apps ein reines Marketingwerkzeug oder dienen Sie auch dem Erzielen von Erlösen?

3. Anforderungen an die App-Entwicklung
   1. Wer trifft die Entscheidung, ob Apps entwickelt werden sollen?
   2. Welche Anforderungen gibt es, wenn Apps entwickelt werden sollen?
   3. Wird speziell auf die Entwicklung für mehrere Plattformen geachtet? Hat dies sogar Auswirkungen auf die Anforderungsanalyse?
4. Planungen für die Zukunft
   1. Welche strategische Bedeutung haben Business Apps?
   2. Soll in Zukunft stärker auf die Kompatibilität zu mehreren Plattformen geachtet werden? (Wenn ja: welche sind besonders wichtig?)
   3. Planen Sie in Zukunft, mehr Mittel als bisher in die Entwicklung von Apps zu investieren?
5. Abschluss, Ankündigung erster Ergebnisse für den Winter

Literaturverzeichnis

[App12] Apple Inc.: *iOS Security*, 2012. – http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf

[Bir12] Birch, D.G.W.: Let a thousand currencies bloom. In: *Spectrum, IEEE* 49 (2012), june, Nr. 6, S. 30 –34. http://dx.doi.org/10.1109/MSPEC.2012.6203963. – DOI 10.1109/MSPEC.2012.6203963. – ISSN 0018–9235

[Ebe08] Ebert, Christof: *Systematisches Requirements Engineering Management*. 2nd. Heidelberg : Dpunkt, 2008

[HD12] Hogben, Giles ; Dekker, Marnix: Smartphones: Information security risks, opportunities and recommendations for users / European Network and information Security Agency (ENISA). 2012. – Forschungsbericht. – http://www.enisa.europa.eu/activities/identity-and-trust/risks-and-data-breaches/smartphones-information-security-risks-opportunities-and-recommendations-for-users

[HHM12] Heitkötter, H. ; Hanscke, S. ; Majchrzak, T. A.: Comparing Cross-platform Development Approaches for Mobile Applications. In: *8th International Conference on Web Information Systems and Technologies (WEBIST)*, 2012, S. 299–311

[HMK13a] Heitkötter, Henning ; Majchrzak, Tim A. ; Kuchen, Herbert: Cross-Platform Model-Driven Development of Mobile Applications with MD2. In: *Proc. of the 2013 ACM Symp. on Applied Computing (SAC)*, ACM, 2013

[HMK13b] Heitkötter, Henning ; Majchrzak, Tim A. ; Kuchen, Herbert: MD2-DSL – eine domänenspezifische Sprache zur Beschreibung und Generierung mobiler Anwendungen. In: *Proc. der 6. Arbeitstagung Programmiersprachen (ATPS 2013)*, GI, 2013 (LNI 215)

[LY99] Lindholm, Tim ; Yellin, Frank: *The Java Virtual Machine Specification*. 2nd. Prentice Hall, 1999. – ISBN 0201432943

[MJLc10] Majchrzak, Tim A. ; Jakubiec, Adalbert ; Lablans, Martin ; Ückert, Frank: Evaluating Mobile Ambient Assisted Living Devices and Web 2.0 Technology for a Better Social Integration. In: Becker, J. (Hrsg.) ; Backhaus, K. (Hrsg.) ; Grob, H.L. (Hrsg.) ; Hellingrath, B. (Hrsg.) ; Hoeren, T. (Hrsg.) ; Klein, S. (Hrsg.) ; Kuchen, H. (Hrsg.) ; Müller-Funk, U. (Hrsg.) ; Thonemann, U. W. (Hrsg.) ; Vossen, G. (Hrsg.): *Working Papers No. 9*, European Research Center for Information Systems (ERCIS), 2010

[MK10] Majchrzak, Tim A. ; Kuchen, Herbert: IHK-Projekt Softwaretests: Auswertung. In: *Working Papers*, Förderkreis der Angewandten Informatik an der Westfälischen Wilhelms-Universität Münster e.V., 2010 (Working Papers 2)

[Poh08] Pohl, Klaus: *Requirements Engineering*. 2nd. Heidelberg : Dpunkt, 2008

[RGB12] Redman, Phillip ; Girard, John ; Basso, Monica: Magic Quadrant for Mobile Device Management Software / Gartner Inc. 2012. – Forschungsbericht. – `http://www.gartner.com/DisplayDocument?id=2019515`

[Ros12] Ross, P.E.: Phone-y money. In: *Spectrum, IEEE* 49 (2012), june, Nr. 6, S. 60 –63. `http://dx.doi.org/10.1109/MSPEC.2012.6203971`. – DOI 10.1109/MSPEC.2012.6203971. – ISSN 0018–9235

[Wro11] Wroblewski, L.: *Mobile First*. A Book Apart, 2011 (Brief books for people who make websites). – ISBN 9781937557027

Internet-Adressen

[1] http://cs.uni-muenster.de/iai/.

[2] Marktforschungszahlen von Gartner zu 2011. http://www.gartner.com/it/page.jsp?id=1689814, http://www.gartner.com/it/page.jsp?id=1764714, http://www.gartner.com/it/page.jsp?id=1848514, http://www.gartner.com/it/page.jsp?id=1924314.

[3] https://developer.apple.com/programs/ios/.

[4] https://developer.apple.com/appstore/guidelines.html.

[5] http://www.apple.com/ipad/business/it-center/deployment-mdm.html.

[6] http://manuals.info.apple.com/en_US/Enterprise_Deployment_Guide.pdf.

[7] http://www.openhandsetalliance.com/.

[8] http://developer.android.com/guide/topics/fundamentals.html.

[9] http://developer.android.com/guide/topics/fundamentals/activities.html.

[10] http://developer.android.com/guide/topics/ui/index.html.

[11] http://developer.android.com/index.html.

[12] Java native interface. http://docs.oracle.com/javase/7/docs/technotes/guides/jni/index.html.

[13] http://developer.android.com/tools/sdk/ndk/overview.html.

[14] https://play.google.com/.

[15] http://www.androidpit.de/.

[16] http://slideme.org/.

[17] Amazon app-shop. http://www.amazon.de/gp/feature.html?docId=1000644903.

[18] Company app distribution for windows phone. http://msdn.microsoft.com/en-US/library/windowsphone/develop/jj206943(v=vs.105).aspx.

[19] https://developer.blackberry.com/devzone/platforms.

[20] https://bdsc.webapps.blackberry.com/java/.

[21] Mobile information device profile (midp). http://www.oracle.com/technetwork/java/midp-139954.html.

[22] http://docs.blackberry.com/de-de/developers/deliverables/9976/Support_for_standard_Java_APIs_446981_11.jsp.

[23] http://supportforums.blackberry.com/rim/attachments/rim/Testing_and_Deployment_of_Applications@tkb/118/1/How_And_When_To_Sign_V2.pdf.

[24] http://www.blackberrytoolkit.de/tag/javaloader/.

[25] http://phonegap.com/.

[26] http://www.sybase.de/products/mobileenterprise/sybaseunwiredplatform.

[27] http://www.gartner.com/it/page.jsp?id=1924314.

[28] http://incubator.apache.org/projects/callback.html.

[29] applause. http://applause.github.com/.

[30] http://xmlvm.org/.

[31] http://modernizr.com/.

[32] Appcelerator. http://www.appcelerator.com/.

[33] J2ObjC. http://code.google.com/p/j2objc/.

[34] http://source.android.com/tech/security/index.html.

[35] https://developer.apple.com/library/ios/#documentation/Security/Conceptual/Security_Overview/Introduction/Introduction.html.

[36] http://developer.android.com/guide/practices/security.html.

[37] https://www.bsi.bund.de/DE/Themen/weitereThemen/MobileSecurity/mobilesecurity_node.html.

[38] http://www.enisa.europa.eu/activities/application-security/smartphone-security-1/.

[39] Google cloud messaging for android. http://developer.android.com/google/gcm/index.html.

[40] Apple push notification service. https://developer.apple.com/appstore/push-notifications/index.html.

[41] http://www.keynotedeviceanywhere.com/.

[42] http://www.gartner.com/it/page.jsp?id=2010217.

[43] http://www.cio.de/strategien/2882167/.

[44] http://www.mobileiron.com/.

Alle Links wurden am 21.12.2012 überprüft.

Autoren

Die Broschüre entstand am Lehrstuhl für Praktische Informatik des Instituts für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster. Verantwortlich war ein Team aus drei Mitarbeitern, unterstützt von Prof. Dr. Herbert Kuchen.

Henning Heitkötter (heitkoetter@ercis.de) ist wissenschaftlicher Mitarbeiter am Institut für Wirtschaftsinformatik der Universität Münster. Sein Studium der Wirtschaftsinformatik an der Universität Münster schloss er 2009 mit dem Diplom ab. Seine Forschungsinteressen umfassen modellgetriebene Softwareentwicklung, insbesondere im Kontext Prozessmodellierung, sowie Themen des Software Engineering in der Entwicklung mobiler Anwendungen. Seine Ergebnisse zu diesen Themen hat er auf internationalen Konferenzen vorgestellt.

Tim A. Majchrzak (tima@ercis.de) ist Akademischer Rat am Institut für Wirtschaftsinformatik der Universität Münster und am European Research Center for Information Systems (ERCIS). Er schloss sein Studium der Wirtschaftsinformatik mit den Graden BSc und MSc ab und wurde in den Wirtschaftswissenschaften promoviert (Dr. rer. pol.). Seine Forschungsinteressen umfassen betriebliche und technische Fragestellungen des Software Engineering. Darüber hinaus hat er zu diversen interdisziplinären Themen der Wirtschaftsinformatik publiziert. Dr. Majchrzak hat seine Arbeit auf Tagungen der Informatik sowie auf *Information Systems*-Konferenzen präsentiert.

Ulrich Wolffgang war bis 2012 wissenschaftlicher Mitarbeiter am Institut für Wirtschaftsinformatik der Universität Münster, wo er mit einer Arbeit zur modellgetriebenen Entwicklung von Webanwendungen zum Dr. rer. pol. promoviert wurde. Zuvor hatte er 2008 sein Studium an der Universität Münster als Diplom-Wirtschaftsinformatiker abgeschlossen.

Herbert Kuchen ist Professor für Praktische Informatik in der Wirtschaft am Institut für Wirtschaftsinformatik der Universität Münster. Zu seinen Forschungsgebieten zählen unter anderem das Testen von Software, parallele Programmierung sowie funktionale und logische Programmierung.

Förderkreis der Angewandten Informatik
an der Westfälischen Wilhelms-Universität Münster e.V.

Leonardo-Campus 3
D-48149 Münster
Tel. : +49 251 83 38250
Fax : +49 251 83 38259

ISSN 1868-0801