Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Konstruktion eines Kriterienkatalogs zur Bewertung nativer mobiler Enterprise Applikationen

Masterarbeit

Leipzig, Februar 2016

vorgelegt von:
Björn Buchwald
Studiengang Master Informatik

Betreuer:
Prof. Dr. Ing. habil. Klaus-Peter Fähnrich
Fakultät für Mathematik und Informatik, Abteilung Betriebliche Informationssysteme
Dipl. Inf. Ruslan Hrushchak
Fakultät für Mathematik und Informatik, Abteilung Betriebliche Informationssysteme
appPlant UG, Leipzig

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

AES ..... Advanced Encryption Standard
API ..... Application Programming Interface
APNS ..... Apple Push Notification Service
App ..... Mobile Application
ARM ..... Advanced RISC Machines
ASLR ..... Address Space Layout Randomization
BDSG ..... Bundesdatenschutzgesetz
BES ..... Blackberry Enterprise Server
BS ..... Betriebssystem
BYOD ..... Bring Your Own Device
CLR ..... Common Language Runtime
COBO ..... Corporate Owned Business Only
COPE ..... Corporate Owned Personally Enabled
CRM ..... Customer Relationship Management
CSS ..... Cascading Style Sheets
DEP ..... Data Execution Prevention
EAS ..... Exchange ActiveSync
EC2 ..... Elastic Cloud Computing
EMM ..... Enterprise Mobility Management
ERP ..... Enterprise Resource Planning
FIPS ..... Federal Information Processing Standard
GCM ..... Google Cloud Messaging Service
GDR ..... General Distribution Release
GID ..... Gerätegruppen Identifier
HTTPS ..... HyperText Transfer Protocol Secure
IaaS ..... Infrastructure as a Service
IDE ..... Integrated Development Environment
IPsec ..... IP Security Architecture
JS ..... JavaScript
JSON ..... JavaScript Object Notation
L2TP ..... Layer 2 Tunnel Protocol
LOB ..... Line-Of-Business
LTE ..... Long Term Evolution
MADP ..... Mobile Application Development Platform
MAM ..... Mobile Application Management
MCC ..... Mobile Cloud Computing
MCM ..... Mobile Content Management
MDM ..... Mobile Device Management
MEAP ..... Mobil Enterprise Application Platforms
MIME ..... Multipurpose Internet Mail Extensions
MVC ..... Model-View-Controller
MVP ..... Model-View-Presenter
MVVM ..... Model-View-ViewModel
NFC ..... Near Field Communication

OSI ..... Open Systems Interconnection
OTA ..... Over-the-Air
PaaS ..... Platform as a Service
PARC ..... Palo Alto Research Center
PLM ..... Product Lifecycle Management
PM ..... Presentation Model
PPG ..... Push Proxy Gateway
PPTP ..... Point-to-Point-Tunneling Protocol
QML ..... Qt Modeling Language
S/MIME ..... Secure / Multipurpose Internet Mail Extensions
S3 ..... Simple Storage Service
SaaS ..... Software as a Service
SASL ..... Simple Authentication and Security Layer
SCM ..... Supply Chain Management
SD ..... Smart Digital
SDK ..... Software Development Kit
SoC ..... System-on-a-Chip
SSL ..... Secure Sockets Layer
SSO ..... Single Sing-on
SyncML ..... Synchronization Markup Language
TLS ..... Transport Layer Security
TPM ..... Tamper-Resistant Security Processor
UEFI ..... Unified Extensible Firmware Interface
UI ..... User Interface
UID ..... Unique Identifier
UMTS ..... Universal Mobile Telecommunications System
URI ..... Uniform Ressource Identifier
VPN ..... Virtuelles Privates Netzwerk
WinJS ..... Windows Library for JavaScript
WNS ..... Windows Notification Service
WPF ..... Windows Presentation Foundation
WYSIWYG ..... What You See Is What You Get
XAML ..... Extensible Application Markup Language
XHTML ..... Extensible HyperText Markup Language
XMPP ..... Extensible Messaging and Presence Protocol

1 Einführung und Zielsetzung der Arbeit

Die meisten Unternehmen und Institutionen besitzen heutzutage eine umfassende IT-Infrastruktur, die jedoch ein deutliches Erweiterungspotenzial in der hier vorgestellten Thematik der Mobilität aufweist.
Unternehmen setzen täglich eine Vielzahl von, im Enterprise Sektor etablierten, Systemen ein. Sie müssen zum Beispiel Fertigungsprozesse steuern, Informationen analysieren sowie ein sicheres Kundenmanagement, eine sichere Datenhaltung und Kommunikation realisieren. Sie greifen dabei auf Systeme zum Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Product Lifecycle Management (PLM) und Supply Chain Management (SCM) zurück. Ein Aspekt, der bis vor wenigen Jahren vernachlässigt wurde und nun von den Unternehmen entdeckt wird, ist die Mobilität. Sie stellt neue Herausforderungen an die IT-Infrastruktur und zwingt die großen Software-Hersteller wie IBM, Oracle, SAP und HP, neue skalierbare Ansätze für mobile Endgeräte zu integrieren.
Die Mobilität wird in diesem Dokument als Sektor innerhalb der Informationstechnik verstanden, geprägt durch das Management von mobilen Geräten, Anwendungen und Inhalten mit Hilfe dedizierter Systeme, auch bezeichnet als mobile Systeme oder Plattformen.
Das Interesse an diesem Sektor seitens der Unternehmen und das Vorantreiben seiner Entwicklung erwachsen aus folgenden Fragen:

- Wo werden unternehmenskritische Entscheidungen getroffen?
- Wo werden beispielweise Daten über Qualität und aktuelle Beschaffenheit von Produkten aufgenommen?
- Wo arbeiten Angestellte an innovativen Ideen?
- Wie können Mitarbeiter zeit- und ortsunabhängig dem Unternehmen ihre Arbeitskraft zur Verfügung stellen?
- Wie groß ist der Gewinn, wenn Informationen am Ort des Geschehens erhoben werden, ohne Zeitverzögerung übermittelt und im Unternehmen ausgewertet werden können?

Im Business-to-Consumer Bereich sind folgende Fragen relevant:

- Können Produktdaten den Kunden zeit- und ortsunabhängig zur Verfügung gestellt werden?
- Welche Informationen sind für den Kunden relevant während er mobil ist?

Es stellt sich heraus, dass der Sektor Mobilität erstaunliches Potenzial bietet. Rund 23 % der Unternehmer gingen in einer Umfrage von MOBILE HELIX & VANSON BOURNE (2013) von einer Produktivitätssteigerung von 40 % aus, sollten alle Enterprise Applikationen mobil zur Verfügung gestellt werden. Zudem ist die Grundvoraussetzung einer Umsetzung bereits in jeder Hosentasche vorhanden. In ARNS, BESSING, BUGGISCH & GRACKLAUER (2014) heißt es: „Die Erfolgsmeldungen im mobilen Umfeld sprechen für sich: 90 Prozent aller Deutschen über vierzehn Jahre haben ein Mobiltelefon, mehr als 40 Prozent ein Smartphone. Bei den Neuverkäufen sind rund 80 Prozent der Geräte Mobiltelefone mit Touchscreen und Internetfähigkeit.“ Smartphones und Tablets als Oberbegriffe für eine unüberschaubare Anzahl von mobilen Geräten verschiedener Größen, mit verschiedenen mobilen Betriebssystemen (BSs) können ideal für die Zwecke der Großunternehmen genutzt werden. Im Idealfall sind

keine zusätzlichen Investitionen in mobile Endgeräte notwendig, da das persönliche Gerät alle Anforderungen einer betrieblichen Nutzung erfüllt. Kollaborative Prozesse zeit- und ortsunabhängig erhöhen die Produktivität, lassen eine sofortige Aufnahme und Analyse neuer Produktdaten zu und führen zu schnelleren Lösungen für den Kunden. Dieser greift mobil auf Daten zu und schließt seine Käufe und Verträge von unterwegs aus ab. Es ergibt sich ein Vorteil sowohl für das Unternehmen als auch den Kunden.
Nach dem Aufzeigen einiger Vorteile gestaltet sich die Umsetzung als nicht zu unterschätzende Herausforderung. Es sind umfangreiche Investitionen nötig, um eine mobile IT-Infrastruktur zu schaffen. Nur wenige Unternehmen besitzen eigene IT-Abteilungen, um die Integration und das Management von mobilen Systemen zu verwirklichen. Diese Probleme führten zur Herausbildung einer breiten Landschaft von Integrations-, Management und Entwicklungsplattformen für mobile Geräte und Applikationen.
Die Anbieter versuchen, zentrale Anforderungen der Großunternehmen bei der Entwicklung mobiler Enterprise Applikationen zu erfüllen. Dazu gehören unter anderem Echtzeit-Kommunikation, gleichzeitige Nutzung von persönlichen und Unternehmensdaten auf einem Gerät, die Integration vorhandener Dienste, online und offline Bearbeitung von Informationen, das Umgehen mit Netzwerkelastizität, übersichtliche Benutzerschnittstellen, vereinfachte und schnelle Entwicklung für eine oder mehrere Plattformen sowie ausreichende Performanz. Hervorzuheben sind Aspekte der Sicherheit in Bezug auf alle genannten Anforderungen. (Vgl. MOTWIN, 2013) Beispiele sind die Verschlüsselung von Daten auf den Endgeräten sowie der Aufbau von VPN-Verbindungen zum eigenen Firmennetzwerk, um nur einen Bruchteil zu nennen.
Auch die Entwickler mobiler BSs haben die Nachfrage ihrer Systeme im Enterprise Segment erkannt. Nutzern, die ihr Smartphone bisher vorwiegend für den privaten Gebrauch einsetzten, werden nur selten die eingebauten Enterprise Funktionalitäten auffallen und tatsächlich sind Android, Windows Phone und iOS auf den ersten Blick für den Privatkunden ausgelegt. BlackBerry ist der bekannteste Vertreter für die Enterprise Nutzung. Die neueren BS-Versionen bemühen sich zunehmend, native Funktionen für den Unternehmenseinsatz zur Verfügung zu stellen.
Wie sich herausstellt, existieren bedeutende Unterschiede im Entwicklungsstand der Enterprise Funktionalitäten und der Umsetzung von Unternehmensanforderungen innerhalb mobiler BSs und Applikationen. Daraus ergibt sich die Frage nach Eignung dieser Systeme für verschiedene Businessszenarien. Hier setzt die vorliegende Arbeit an, welche durch Analyse von Anforderungen relevante Kriterien im Unternehmensbereich bestimmt sowie eine Möglichkeit zur Bewertung nativer mobiler Applikationen aufzeigt.

1.1 Motivation und Zielsetzung

Literaturrecherchen zeigen, dass eine Analyse der Anforderungen an mobile Enterprise Applikationen, betrachtet über die verschiedenen Teilbereiche der Mobilität, bisher nicht gegeben ist. Analysen beziehen sich meist auf die Allgemeinheit der mobilen Applikationen, auf einzelne mobile Plattformen, oder befassen sich ausschließlich mit Teilaspekten wie Sicherheit und Mobile Device Management (MDM).
Da vorhandene Analysen von Teilbereichen und Anforderungen unzureichend sind, gestaltet es sich für Entwickler als auch für Unternehmen und die dort verantwortlichen

Sicherheitsbeauftragten als schwierig, Einschätzungen über die Tauglichkeit einer Applikation zu treffen.
Außerdem fehlen Betrachtungen, welche die sich in den letzten Jahren etablierten Plattformen und Technologien, sowohl für Entwickler als auch für Unternehmen, im mobilen Bereich zusammenfassend darstellen. Dazu zählen z.B. die mobilen Plattformen der Betriebssystemhersteller, Enterprise Mobility Management (EMM)-Systeme und Mobile Application Development Platforms (MADPs), Entwicklungsplattformen sowie Technologien im Bereich MDM, Verschlüsselung, VPN, Berechtigungssysteme und Realisierung von Benutzerinterfaces.
Aus diesen Gründen ist das Ziel der vorliegenden Arbeit, die grundlegenden Anforderungen mobiler Enterprise Applikationen zu analysieren und in einem Kriterienkatalog für mobile Enterprise Applikationen festzuhalten. Auf Grundlage dieses Katalogs ist eine Analyse und Bewertung mobiler Applikationen möglich, die mögliche Schwächen und Verbesserungsmöglichkeiten aufzeigt.
Des Weiteren entsteht im Rahmen der Arbeit eine mobile Enterprise Applikation zur Erfassung von Arbeitszeiten für die Windows Phone 8.1 Plattform. Sie dient der exemplarischen Anwendung des Katalogs.
Durch den iterativen Prozess von der Beschreibung aktueller Herausforderungen im Sektor Mobilität über die Auseinandersetzung der besonderen Herausforderungen mobiler Applikationen im Businessumfeld, bis hin zur Aufstellung eines Kriterienkatalogs, wird ein zusammenfassender Überblick aktueller Technologien im Sektor Mobilität für Enterprises gegeben.
Die Erarbeitung eines Kriterienkatalogs und Entwicklung einer Applikation haben neben dem genannten Nutzen eine konkrete praktisch motivierte Zielsetzung. Das Unternehmen appPlant[1] erweitert sein Know-How im Bereich der Entwicklung mobiler Enterprise Applikationen und nutzt die entstehende Beispielapplikation „appPlant Tempo“ zur Demonstration seines Portfolios.

1.2 Methodisches Vorgehen

Dieser Abschnitt gibt Auskunft über den Entstehungsprozess dieser Arbeit. Es werden die einzelnen Schritte erläutert. Parallel zu den Phasen I bis VII fand die Entwicklung einer mobilen Enterprise Applikation zur Zeiterfassung statt. Dabei wurde sich an den Phasen der Softwareentwicklung nach (BALZERT, 2000) orientiert.

I. Recherche von Konzepten mobiler Anwendungen im Unternehmensumfeld

Zu Beginn erfolgte die Recherche von grundlegenden Begriffen, Aussagen und Statistiken in Zusammenhang mit Mobilität und mobilen Applikationen. Die Literaturrecherche erfolgte im Bestand der Universitätsbibliothek Leipzig, durch Metasuchmaschinen wie Google Schoolar, in IT-Fachmagazinen (online) wie TechTarget und ZDNet, in Angeboten von Research Groups wie Gartner und Info-Tech sowie bei Verlagen wie Springer Link. Nach dem Identifizieren der führenden Hersteller und Anbieter mobiler Dienstleistungen wurde nach Whitepapern auf den entsprechenden Internetpräsenzen gesucht. Besonders bewährte Quellen sind BlackBerry, Apple und MoTwin.

[1] http://www.appplant.de/

Mit dem Anspruch, die Bedürfnisse der Unternehmen zu analysieren, erfolgte eine Untersuchung einzelner Branchen auf typische Enterprise Applikationen. Dabei wurden die plattformspezifischen Mobile Application (App) Stores der Hersteller Apple, Microsoft, Google und Blackberry durchsucht. Schließlich erfolgte eine erste Ableitung typischer Funktionen und Anforderungen einzelner Branchen.

II. Verdichtung der Rechercheergebnisse

Im zweiten Schritt wurden die Ergebnisse in Zusammenhang gebracht, verdichtet und ein konzeptionelles Gesamtbild erarbeitet. Es erfolgte eine erste Aufteilung der Konzepte in Teilbereiche und eine Zuordnung von Technologien. Wichtig war es, ein Verständnis beider Sichten, sowohl der Unternehmen als auch des Entwicklers, auf die App-Entwicklung zu erhalten. Aus diesem Grund fand eine vertiefte Recherche der realen Umsetzung von Geschäftsprozessen mit Hilfe mobiler Enterprise Apps im Unternehmensumfeld statt. Besonders hilfreich war der Beitragsband von VERCLAS & LINNHOFF-POPIEN (2012b). Dort geben über 38 Publikationen aus verschiedenen Sichten einen Einblick in die Thematik.

III. Recherche, Analyse und Beschreibung grundlegender Anforderungen

Die einzelnen analysierten Teilbereiche und Technologien wurden im Weiteren vertieft betrachtet und durch nachfolgende Recherchen in einem iterativen Prozess erweitert und beschrieben. Gleichzeitig erfolgte eine Recherche nach konkreten Realisierungen der gefundenen Technologien auf mobilen Plattformen und das Ergänzen der Technologien durch konkrete Beispielrealisierungen.

IV. Aufstellen von Kriterien

Zunächst erfolgte die Erarbeitung eines Konzeptes zur Realisierung des Kriterienkataloges anhand der gefundenen Anforderungen. Dabei fand eine Orientierung an den bereits eingeführten Teilbereichen statt. Außerdem wurde der Katalog nach genauer Analyse in zwei Phasen aufgeteilt und um ein Bewertungssystem erweitert. Schließlich folgte die Ableitung von Kriterien aus den gefundenen Anforderungen an mobile Enterprise Applikationen pro Teilbereich. Die Kriterien und das Bewertungssystem inklusive der Gewichtung einzelner Kriterien wurden parallel schrittweise angepasst und erweitert.

V. Bewertung mobiler Betriebssysteme und sukzessive Anpassung der Kriterien

Nach der Ableitung von Kriterien in einer ersten Version erfolgte die Bewertung der mobilen BSs. Dabei wurde bevorzugt nach Whitepapern der Hersteller recherchiert. Außerdem wurden wissenschaftliche Paper sowie Berichte von IT-Fachmagazinen zu einzelnen Funktionen der BSs und MDM-Systeme herangezogen. In einem iterativen Prozess wurden Bewertung und Kriterien weiter verfeinert.

VI. Anwendung des Kataloges

Dieser Schritt umfasst die Konzeption einer Fallstudie, welche die entwickelte mobile Enterprise Applikation integriert und eine exemplarische Anwendung des Kriterienkataloges ermöglicht. Das Lastenheft des Softwareentwicklungsprozesses wird an dieser Stelle integriert.

Schließlich werden die Ergebnisse der Arbeit sowie des Entwicklungsprozesses der mobilen Beispielapplikation angewendet.

VII. Zusammenfassung der Ergebnisse

Das Zusammenfassen und Analysieren der Ergebnisse sowie die Aufstellung von Verbesserungsvorschlägen fand in diesem Schritt statt.

1.3 Aufbau der Arbeit

Der Schwerpunkt der Arbeit besteht in der Erarbeitung eines Kriterienkatalogs zur Bewertung mobiler Anwendungen. Die schrittweise Einarbeitung in den umfassenden Themenkomplex und Annährung an einzelne Kriterien wird über die Aufteilung in Kapitel erreicht.
Eine Einführung in grundlegende Konzepte der Mobilität für Unternehmen wie Netzwerkelastizität, On- und Offline-Nutzung, Bring Your Own Device (BYOD), MDM sowie in mobile Enterprise Applikationen und mobile Plattformen wird in Kapitel 2 gegeben. Dies führt zu einem grundlegenden Verständnis der Thematik und der Einordnung verschiedener Begriffe der Mobilität.
Kapitel 3 bedient sich der grundlegenden Begriffe, bringt sie mit dem Begriff der mobilen Enterprise Applikation in Verbindung und fasst die entstehende Umgebung in einem Schaubild zusammen. Daraufhin werden sowohl die Sicht des Entwicklers, als auch die Sicht des Unternehmens auf diese Umgebung erläutert. Damit werden Zusammenhänge zwischen beschriebenen Technologien aufgezeigt und ein Verständnis für verschiedene Sichtweisen geschaffen.
Auf Grundlage von Kapitel 2 und 3 werden zentrale Aspekte der App-Entwicklung im Enterprise-Bereich ausgewählt und jeweils die relevanten Technologien und Konzepte für jeden Teilbereich erläutert. Beispiele der Umsetzung werden anhand aktueller mobiler Plattformen ausgewählt. Kapitel 4 bildet den Kern der Arbeit und die Grundlage für die Aufstellung des Kriterienkatalogs.
In Kapitel 5 wird der Kriterienkatalog aufgestellt auf Grundlage von Kapitel 4. Es erfolgt eine Aufteilung des Katalogs in die Vorentwicklungsphase, welche Kriterien für die zugrundeliegende mobile Plattform der Applikation umfasst und die Entwicklungs- und Einsatzphase, welche Kriterien für die Bewertung der Entwicklung und Umgebung der mobilen Applikation enthält. Schließlich wird ein Bewertungssystem eingeführt, um eine Aussage über die Tauglichkeit einer mobilen Enterprise Applikation für den Unternehmenseinsatz treffen zu können.
Kapitel 6 bedient sich des Kriterienkataloges und der entwickelten Beispielapplikation für das Zeitmanagement. Anhand einer Fallstudie werden grundlegende Entscheidungen für die Entwicklung der Applikation aufgezeigt und die Umsetzung einzelner Kriterien des Katalogs beschrieben. Schließlich wird die mobile Applikation anhand des Kataloges bewertet. Damit wird der Einsatz des Kataloges und der Bewertung anhand einer realen Enterprise Applikation aufgezeigt.
Eine Zusammenfassung der Ergebnisse und das Einbringen von Kritik sowie Verbesserungsvorschlägen erfolgt in Kapitel 7.

2 Herausforderungen bei der Entwicklung

Dieser Abschnitt gibt eine Einführung in die grundlegenden Konzepte und Herausforderungen der Entwicklung Mobiler Applikationen im Unternehmensumfeld. Zunächst erfolgt eine Analyse, welche Konzepte für die Unternehmen im Sektor Mobilität von Bedeutung sind und welche Anforderungen und Probleme bei der Umsetzung mobiler Lösungen entstehen. Daraufhin wird der Begriff der mobilen Enterprise Applikation eingeführt. Schließlich werden grundlegende Charakteristiken wie Arten, Entwicklungsmöglichkeiten, Plattformen der Entwicklung und des Managements, aufgezeigt.

2.1 Mobilität

Die Mobilität verspricht den Unternehmen immense Vorteile. Entscheidend für eine erfolgreiche Umsetzung einer mobilen Lösung und damit für das Erzielen eines Wettbewerbsvorteils ist das Erfüllen der grundlegenden Anforderungen der Unternehmen. Diese entstammen beispielsweise den Bereichen Sicherheit, Zuverlässigkeit, Benutzerfreundlichkeit usw. Außerdem müssen die aktuell vorhandenen Probleme im Sektor Mobilität hinreichend gut gelöst werden.

2.1.1 Bedeutung von Mobilität für Unternehmen

Unternehmen streben nach höherer Produktivität, besserem Kundenservice und zufriedenen Mitarbeitern. Der Sektor Mobilität und seine Umsetzung durch modernste mobile Plattformen im Unternehmen verspricht diese Bestrebungen zu erfüllen. Mobilität für Unternehmen kann in zwei Konzepte gefasst werden. Man unterscheidet zum einen Mobile Business. Darunter wird „[…] die Nutzung mobiler Anwendungen und -services zur Unterstützung der Unternehmensziele und zur Förderung des Geschäftserfolgs verstanden“ (ARNS, BESSING, BUGGISCH & GRACKLAUER, 2014). Zum anderen existiert das Konzept der Enterprise Mobility, welches „das Management der mobilen Geräte, Anwendungen und Inhalte innerhalb von Unternehmen“ (ARNS, BESSING, BUGGISCH & GRACKLAUER, 2014) bezeichnet. Beispiele sind der Trend zu BYOD, die Mobilisierung der eigenen Website oder die Entwicklung eigener mobiler Applikationen.
Die Verbreitung mobiler Endgeräte als Ende des bisher gebräuchlichen Desktop PCs zwingt die Unternehmen neue Wege einzuschlagen und bisherige Konzepte zu überdenken. Sichtbar wird dies durch Geschäftsprozesse, die inzwischen vermehrt auf mobilen Plattformen realisiert werden. „»Mobile« ist längst viel mehr als der Zugriff auf E-Mails, Kalender und Kontakte.“ (ARNS, BESSING, BUGGISCH & GRACKLAUER, 2014)

2.1.2 Bedürfnisse der Unternehmen im Sektor Mobilität

Die Gliederungspunkte sind angelehnt an (MOTWIN, 2013). Dort wurden Unternehmen nach kritischen Anforderung an eine mobile Entwicklungsplattform gefragt.

a. Sicherheitserfordernisse

Die Sicherheit von Kommunikation und Daten steht bei den Unternehmern an oberster Stelle der Anforderungen. Hierzu zählen beispielsweise die sichere Anbindung des eigenen

Firmennetzwerks über Secure Sockets Layer (SSL)-VPN-Verbindungen oder die Verschlüsselung von sicherheitskritischen Daten auf dem mobilen Endgerät.

b. Netzwerkelastizität

Das Umgehen mobiler Lösungen mit unterschiedlicher Netzabdeckung und Verbindungsstandards wird als Grundvoraussetzung angesehen.

c. Intuitive und einheitliche Bedienbarkeit der Lösung

Gesten und Navigation sollten den Erfahrungen der Benutzer entsprechen. Soll beispielsweise dasselbe App-Konzept auf verschiedenen BSs umgesetzt werden, so wird trotzdem eine einheitliche Bedienbarkeit auf den Plattformen gefordert, um die Bedienung durch eine vorhandene Erfahrung des Nutzers zu erleichtern.

d. Innovativ wahrgenommene Benutzeroberfläche

Die Plattformhersteller wie Microsoft, Apple oder Google verfolgen eigene Strategien und Konzepte, um dem Benutzer eine einzigartige und innovative Oberfläche zu bieten.

e. Hohe Integrationsfähigkeit mobiler Dienste

Die Anbindung mobiler Anwendungen an vorhandene Back-End-Systeme ist eine Kernanforderung bei der Entwicklung im Sektor Mobilität.

f. Intelligente Konzepte zur Online- und Offline-Nutzung

Die Netzwerkelastizität bedingt, dass Daten bei der Online-Nutzung geladen werden, damit sie im Offline-Betrieb zur Verfügung stehen. Mobile Anwendungen sollten nach Möglichkeit Daten auch im Offline-Betrieb zur Verfügung stellen und eine Bearbeitung zulassen. Die Synchronisation mit dem Back-End erfolgt bei der nächstmöglichen Netzwerkverbindung.

g. Integrierte Plattformen und intelligente Clients

Unternehmen suchen mobile Plattformen mit ausgereiften Funktionen und Clients. Hierzu zählen MDM-Clients, die durch eine schnelle und einfache Konfiguration das Managementsystem des Unternehmens anbinden. Weitere Beispiele sind integrierte Verschlüsselungsmechanismen, VPN-Clients sowie die Anbindungen an Bezahlsysteme und Cloud-Lösungen.

h. BYOD

Die Nutzung des Mitarbeitergerätes für Unternehmenszwecke durch das Verwalten eines privaten und eines geschäftlichen Bereichs ist durch die bereits vorhandenen Geräte und darauf befindlichen Systeme ein Schwerpunkt. Weitere Strategien wie Corporate Owned Personally Enabled (COPE) und Corporate Owned Business Only (COBO) sind ebenfalls verbreitet.

i. Management von Devices, Applikationen und Content

Das Management von Geräten, die innerhalb eines Unternehmens mit unterschiedlichen BSs und Anwendungen ausgestattet sein können, spielt insbesondere bei BYOD-Strategien eine große Rolle.

j. Kommunikation in Echtzeit

Besonders in unternehmenskritischen Bereichen ist eine garantierte und schnelle Kommunikation der Teilnehmer gefragt. Verschiedene Push- und Pull-Mechanismen werden von den Plattformanbietern angeboten.

k. Einfachheit der Entwicklung (write once, use across platforms)

Eine Lösung soll bei möglichst wenig Budget einen möglichst großen Kundenkreis ansprechen. Dies erfordert eine Art der Entwicklung, in welcher einmal geschriebener Anwendungscode auf möglichst vielen Plattformen lauffähig ist.

l. Performanz

Der Auftraggeber verlangt, dass die Lösung auf allen Zielplattformen und damit verbundenen Gerätekonfigurationen verzögerungsfrei und fehlerfrei lauffähig ist. Probleme entstehen durch die Fragmentierung des Marktes.

m. Personalisierung

Eine häufige Anforderung ist die Möglichkeit der Anpassung einer mobilen Lösung an das Unternehmen bzw. die Mitarbeiter oder den Endkunden. Dazu gehören die Anpassung der Benutzeroberfläche (z.B. verschiedene Designs und Sprachen) bis hin zur Integration vorhandener Mail- oder Cloud-Storage-Konten.

2.1.3 Herausforderungen bei der Umsetzung mobiler Lösungen

Dieser Abschnitt befasst sich mit den derzeitig im Sektor Mobilität auftretenden Problemen sowohl für Entwickler als auch Unternehmen. Die Gliederungspunkte stehen im Vergleich zu WIEMANN (2013).

a. Fragmentierung:

Sowohl die Entscheidung der Unternehmen, welche mobile Plattform die richtige ist, als auch die Entwicklung der benötigten Anwendungen, werden durch die stetig steigende Anzahl an BSs und Entwicklungsplattformen erschwert. Wurden die zu unterstützenden BSs wie iOS, Android, Windows Phone und BlackBerry erst einmal festgelegt, so folgt die Auswahl einer geeigneten Entwicklungs- sowie Managementplattform. Mit deren Hilfe sollte eine komfortable Entwicklung nach dem „write-once-use-accross-paltforms“ oder ähnlichem Prinzip umsetzbar sein. Dies hält den Entwicklungsaufwand so gering wie möglich. Das Problem vergrößert sich, wenn neue Betriebssystemversionen bzw. Application Programming Interface (API)-Level der vorhandenen Systeme auf den Markt kommen. Neue Geräte werden mit diesen Versionen ausgeliefert. Es dauert nicht lange, bis neue Versionen durch neu

angeschaffte Geräte in den Unternehmen Einzug finden. Schließlich sollen die vorhandenen Apps ebenfalls auf diesen Geräten laufen, was häufig bereits innerhalb einer Plattform eine Überarbeitung des Quellcodes erforderlich macht. Bereits vorhandene Applikationen müssen ständig auf ihre Kompatibilität zu neuen Versionen geprüft werden.

b. Unterschiedliche Browser-Versionen, Display-Auflösungen und Hardware-Ausstattungen

Vor allem bei Web-Apps sind verschiedene Browserversionen ein Problem, da der Browser die Laufzeitumgebung darstellt. Die Versionen unterstützen beispielsweise unterschiedliche SSL-Versionen oder Teilmengen des HTML 5-Standards.
Verschiedene Display-Auflösungen verursachen zusätzlichen Aufwand bei der Entwicklung der Benutzeroberfläche. Apps für die Apple- oder Android-Plattform werden auf Tablets oder Smartphones mit Auflösungen von 854x480 Pixel bis 1920x1080 Pixel oder mehr dargestellt.
Verschiedene Hardwareausstattungen können bereits innerhalb der Geräte einer Betriebssystemversion zu Performanceproblemen führen. Von Single-Core bis Quad-Core Prozessoren und mehr sind alle Konfigurationen vorhanden. Unterschiedliche Geräte besitzen verschiedene Sensoren mit unterschiedlicher Genauigkeit und Abtastrate, verschiedene Standards für die drahtlose Kommunikation, verschiedene Akkulaufzeiten usw. Diese Unterschiede erfordern einen deutlichen Mehraufwand bei der Planung und Entwicklung mobiler Anwendungen.

c. Offline-Funktionalität

Das Bereitstellen von Inhalten im Offline-Betrieb, wie im vorherigen Abschnitt geschildert, stellt eine Herausforderung dar. Es muss beispielsweise zwischen verfügbaren und nicht verfügbaren Funktionen unterschieden werden. Durch die begrenzten Ressourcen wie Bandbreite, Speicher, Netzwerkverfügbarkeit können nur eingeschränkt Daten im Vorfeld geladen werden.

d. Mangel an Standards

Viele Standards im Sektor Mobility werden nur unzureichend umgesetzt. Der HTML5 Standard ist ein Beispiel. Er wird nicht einheitlich und unvollständig von Browsern unterstützt.

e. Integration und Verknüpfung mit bestehenden Systemumgebungen

Durch die komplexen Systemumgebungen im Bereich Mobility werden die Integration und Verknüpfung von mobilen Lösungen in vorhandene Systeme oft unterschätzt. Sollen beispielsweise vorhandene ERP-, CRM- und PLM-Systeme an das Tablet oder Smartphone angebunden werden, so sind komplexe Kommunikations- sowie Sicherheitsaspekte zu betrachten. Anbindung an verschiedene Datenbanksysteme sowie Online-/Offline-Datenhaltung und Synchronisation sind mit vorhandenen Systemen zu koordinieren.

f. Mangel an Marktführern für Mobile App Tools

Die Entwicklung mobiler Applikationen als native, hybride oder web App hat viele Entwicklungs- und Managementplattformen entstehen lassen. Die Anbieter gehen teilweise

vollkommen verschiedene Wege. Ein Trend lässt sich hin zur Entwicklung von HTML5/JavaScript (JS)-basierten Web-Apps feststellen.

2.2 Mobile Enterprise Applikationen

Im Folgenden findet eine Einführung in relevante Konzepte mobiler Enterprise Applikationen statt. Dazu gehören mobile Geräte, BSs, Applikationen und Plattformen für das Management und die Entwicklung.

2.2.1 Einordnung und Definition

Unter mobilen Applikationen für Enterprises (Mobile Enterprise Applikationen) versteht man rollen-basierte Applikationen, da sie für bestimmte Rollen und Funktionen innerhalb einer Organisation entwickelt werden.
Diese Definition und die folgenden drei Absätze beruhen auf BAL (2013), der sich ausführlich mit dem Einsatz mobiler Enterprise Apps auseinandersetzt. Es ist zu bedenken, dass nicht jede Funktion die Eignung besitzt, als mobile Applikation umgesetzt zu werden. Ein Ingenieur beispielsweise besitzt dutzende Aufgaben, doch nur wenige eignen sich für den mobilen Einsatz. Beispiele wären die Konstruktion eines Werkstücks mit einer Computer-Aided Design Software auf einem Tablet oder das Ausführen von Berechnungen auf dem Smartphone. Er benötigt jedoch keine Applikation, die sein gesamtes Aufgabenspektrum abdeckt, komplexer Navigationsverläufe bedarf und viele Stunden an Übung erfordert.
Enterprise Applikationen sind nützliche Einzelfunktionen für eine bestimmte Rolle mit intuitiver Bedienung, die eine schnelle Akzeptanz beim Nutzer hervorrufen.
Unternehmen können demnach viele solcher Apps besitzen. Werden diese über ein zentrales Repository zur Verfügung gestellt, so kann jeder Mitarbeiter, Kunde und Lieferant die für seine Rolle passende App zeit- und ortsunabhängig herunterladen. Sind die bereitgestellten Funktionen für den mobilen Einsatz geeignet, ergibt sich nach eine Produktivitätssteigerung.
Apps können von verschiedensten Perspektiven aus kategorisiert und beschrieben werden. Um ein Verständnis zu schaffen, welche Apps als mobile Enterprise Applikationen betrachtet werden können, wird die folgende Sichtweise eingeführt. Mobile Enterprise Applikationen können für Mitarbeiter, Maschinen und den Endkunden entwickelt werden. Die folgenden Beschreibungen diesbezüglich stehen im Vergleich zu VERCLAS & LINNHOFF-POPIEN, 2012a.

a. Mitarbeiter und Maschinen

Bereiche, in denen Mobile Enterprise Apps für Maschinen eingesetzt werden, sind beispielsweise die Logistik, die Industrie oder Krankenhäuser. Maschinen wie LKWs bis hin zu medizinischen Geräten sind zu nennen. Charakteristisch sind hohe Anforderungen an die Hardware bezüglich dem Standhalten gegenüber Temperatur, Erschütterungen und Feuchtigkeit. Außerdem ist die lange Lebensdauer solcher Maschinen zu bedenken (10-15 Jahre), die weit über der eines Smartphones liegt. Sowohl Apps für Mitarbeiter, als auch für Maschinen, sind in die IT-Infrastruktur des Unternehmens einzubinden.
Apps für Mitarbeiter kennzeichnet der Drang zur Nutzung im privaten und im geschäftlichen Bereich. Entsprechende Container-Strategien und Authentifizierungsmaßnahmen sowie eine stärkere Trennung der Nutzeridentität von der mobilen Hardware sind umzusetzen. (Vgl. VERCLAS & LINNHOFF-POPIEN, 2012a)

b. Endkunden

Während Apps für Mitarbeiter auf einer im Unternehmen etablierten mobilen Plattform entwickelt werden können, werden Apps für den Endkunden nach Möglichkeit plattformunabhängig realisiert. Sie sollen viele Endgeräte am Markt erreichen. Eine einfache Nutzeroberfläche und Bedienbarkeit, um eine schnelle Akzeptanz zu erreichen wird vorausgesetzt. Ähnlich zu Apps für Mitarbeiter sind höchste Sicherheitsanforderungen zu erfüllen. Beispiele sind mobile Applikationen in der Finanz- oder Versicherungsbranche. (Vgl. VERCLAS & LINNHOFF-POPIEN, 2012a)

Kritisch für den erfolgreichen Einsatz mobiler Applikationen ist der Prozess der Analyse von Geschäftsprozessen und Systemen. BAL (2013) empfiehlt einen strukturierten Prozess, in welchem durch Aggregation und Priorisierung entsprechende Eigenschaften und Funktionalitäten für Rollen-basierte Anwendungen generiert werden. Auf dieser Grundlage entsteht eine Roadmap für Apps, die den maximalen Gewinn mit hoher Nutzerzufriedenheit hervorrufen.
Der Prozess sollte vorhandene Systeme mit einbeziehen und die unter 2.1.2 aufgezeigten Bedürfnisse ausreichend befriedigen. Dies wird durch eine Reihe von mobilen Systemen wie MADPs und EMM Suites ermöglicht. Die Entwicklung und das Management mobiler Anwendungen sind nur ein Bruchteil Ihrer Funktionalitäten.

2.2.2 Einführung in mobile Plattformen

Die Auswahl einer geeigneten Plattform für das eigene Unternehmen stellt eine große Herausforderung dar, da viele Teilbereiche betrachtet werden müssen. Die wichtigsten Bestandteile mobiler Plattformen sind die mobilen Endgeräte mit den darauf installierten BSs. Ein Gerät ist meist für ein bestimmtes BS ausgelegt. Zum Beispiel ist das iPhone der Firma Apple für den Betrieb mit dem firmeneigenen BS iOS konstruiert, während die Nokia Lumia Modellserie für den Betrieb mit Windows Phone ausgelegt ist. Auf den Sony Xperia Modellen ist Googles Android System lauffähig und BlackBerry 10 läuft beispielsweise auf der firmeneigenen BlackBerry Z-Serie. Bei der Betrachtung verschiedener BS sollte Folgendes beachtet werden. Bereits die, hier erwähnten, vier etablierten Systeme verursachen durch die Vielzahl der verbreiteten Versionen (auch Fragmentierung genannt) einen enormen Managementaufwand. Gleiches gilt für die Vielzahl an Gerätetypen mit unterschiedlicher Hardwareausstattung. Vor allem die unterschiedlichen Displaygrößen sind bei der Entwicklung einer intuitiven benutzerfreundlichen Nutzeroberfläche eine große Herausforderung.
Je nach der mobilen Strategie des Unternehmens stellt entweder das Unternehmen die mobilen Endgeräte bereit oder die Mitarbeiter nutzen ihre privaten Geräte für Unternehmenszwecke (BYOD). Stellt das Unternehmen die Geräte bereit und kann der Nutzer diese ebenfalls für den privaten Einsatz nutzen, so spricht man von COPE. Ist die private Nutzung untersagt, so wird ein sogenannter COBO Ansatz gefahren. Die Vor- und Nachteile sind in Tabelle 1 dargestellt.
Je nach Ansatz des Unternehmens können demnach persönliche und/oder geschäftliche Anwendungen, siehe Abschnitt 2.1.3, auf den Geräten betrieben werden. Diese Anwendungen stellen einen weiteren Kernbestandteil der mobilen Plattformen dar. Dazu gehört die Bereitstellung von Applikationen in App-Stores, genauso wie Werkzeuge zur Entwicklung von mobilen Enterprise Apps.

	Bring-Your-Own-Device (BYOD)	Company Owned, Personal Enabled (COPE)	Company Owned, Business Only (COBO)
Mitarbeiter	Freie Entscheidung für ein Gerät	Nutzung vorgeschriebener Geräte	Nutzung vorgeschriebener Geräte für geschäftlichen Bereich
	Ein Gerät für geschäftlichen und persönlichen Bereich	Ein Gerät für geschäftlichen und persönlichen Bereich	Zwei Geräte für geschäftlichen und persönlichen Bereich
Unternehmer	Keine Anschaffungskosten	Anschaffungskosten	Anschaffungskosten
	Hoher Verwaltungsaufwand durch heterogene Gerätelandschaft	Geringerer Verwaltungsaufwand als BYOD, je nach verwendeten Geräten	Geringerer Verwaltungsaufwand als BYOD, je nach verwendeten Geräten
	Geringer Einfluss auf Geräte und Konfiguration	Starker Einfluss auf Geräte und Konfiguration	Volle Kontrolle über Geräte
	Hohe Kosten für Sicherheitseinrichtungen, da Trennung von geschäftlichen und persönlichen Daten	Geringere Kosten für Sicherheitseinrichtungen als BYOD, da höhere Kontrolle über Geräte	Normale Kosten für Sicherheitseinrichtungen, da keine Trennung von geschäftlichen und persönlichen Daten

Tabelle 1 Einfluss von Konzepten eines Mobile Device Managements auf Unternehmer und Mitarbeiter

Nach der Einführung von Geräten, BSs und Anwendungen gilt es eine Strategie zu finden, diese zu verknüpfen und zu verwalten. Damit wird das Management als weitere Komponente mobiler Plattformen identifiziert. Das Management von Geräten wird als MDM bezeichnet, analog die Verwaltung von Anwendungen als Mobil Application Management (MAM). Ein etablierter Bestandteil ist inzwischen das Mobile Content Management (MCM) zur Verwaltung von Inhalten auf verschiedensten Geräten.
Info Tech Research Group empfiehlt außerdem den Hersteller einer Plattform als Bestandteil anzusehen und diesen zur Entscheidungsfindung genauer zu untersuchen. Welche Beziehungen bestehen zu anderen Entwicklern? Wendet man sich vorzugsweise einem etablierten Entwickler mit zuverlässigem Support, breiter Anbindung an Back-End Systeme, weitreichende Unterstützung der Geräte-, BS-, Anwendungskonfigurationen und eventuell höheren Kosten zu oder sind Start-Ups mit innovativen neuen Herangehensweisen geeigneter?
Die Kernbestandteile mobiler Plattformen sind auf Abbildung 1 zusammenfassend visualisiert. Mobile Plattformen können unterteilt werden in solche, die sich mit der Entwicklung mobiler Applikationen befassen und denen, die für ihr Management verantwortlich sind. Eine genaue Abgrenzung existiert meist nicht. Eine Erläuterung dieser Ausprägungen mobiler Plattformen wird in Abschnitt 2.2.4 gegeben.

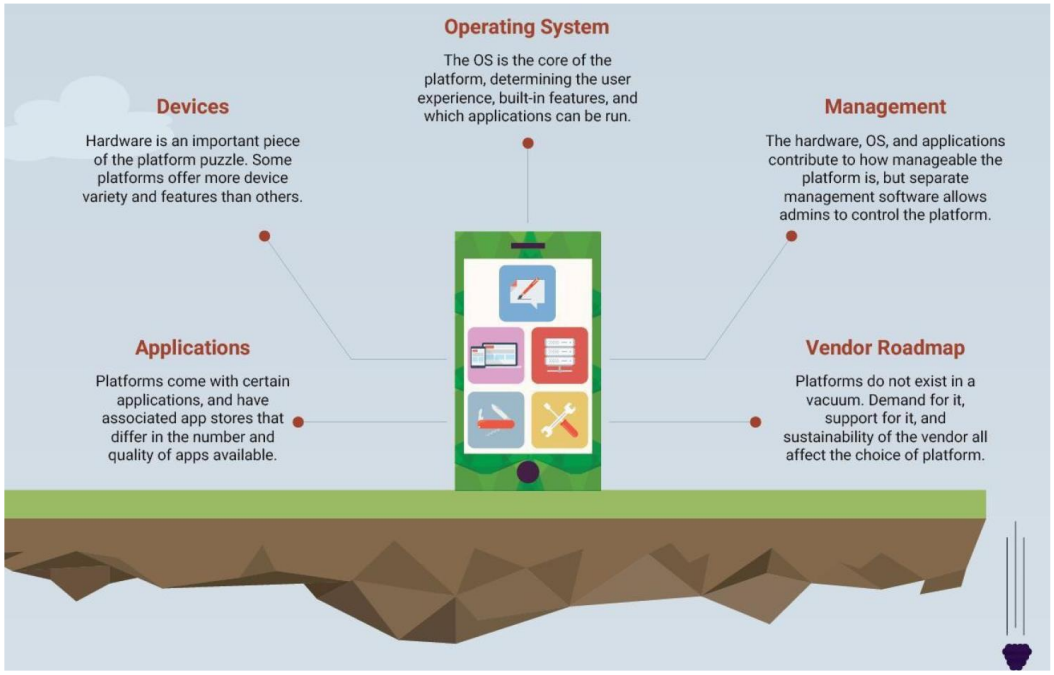

Abbildung 1 Bestandteile einer mobilen Plattform (INFO-TECH RESEARCH GROUP, 2014)

Der folgende Abschnitt befasst sich mit der Entwicklung und den Arten mobiler Applikationen. Dabei sind die Entwicklungsplattformen von entscheidender Bedeutung. Der Umfang dieser mobilen Plattformen reicht von der Unterstützung einer nativen Plattform bis zu dem Ziel, mehrere native Plattformen sowie Arten von Applikationen, wie nativ, hybrid und web Apps, zu verwalten. Außerdem kann die Möglichkeit einer cross-plattform Entwicklung zu den Aufgaben dieser Plattformen gehören. Apple beispielsweise unterstützt mit seiner Entwicklungsplattform ausschließlich die Entwicklung für das eigene iOS-System und bietet den besten Support für iOS-Geräte im Vergleich zu anderen Plattformen. Dies ist für bestimmte Anwendungskategorien vorteilhafter, als eine cross-platform Lösung. PhoneGap konzentriert sich auf das Erstellen hybrider Apps. Sencha gilt als Verfechter für die web- sowie hybride-App-Entwicklung. (Vgl. VALDES, VAN L. BAKER, MARSHALL & WONG, 2014)

2.2.3 Arten der Entwicklung

Mobile Enterprise Applikationen lassen sich nach der Art ihrer Entwicklung in drei Kategorien einteilen. Es existieren native, hybride und Web-Applikationen. Im Folgenden wird eine detaillierte Unterscheidung der Arten mobiler Applikationen vorgenommen. Diese beruht auf dem Vergleich der Entwicklungsarten von HILDEBRANDT, LUTHIGER, STAMM & YEREAZTIAN (2012). Die Autoren zeigen zusätzlich anhand von Beispielapplikationen, wann welche Art von App eingesetzt werden sollte.

a. Nativ

Die native Entwicklung mobiler Applikationen befasst sich mit der Entwicklung speziell für eine entsprechende Zielplattform. Die Anwendung wird direkt auf dem Gerät ausgeführt. (Vgl. HILDEBRANDT, LUTHIGER, STAMM & YEREAZTIAN, 2012). Beispiele sind das Schreiben von C#/Extensible Application Markup Language (XAML) Code für die Windows Phone 8.1-Plattform oder die Entwicklung in Swift für die iOS 8-Plattform und die damit verbundenen Geräte.

Tabelle 2 zeigt die Entwicklungswerkzeuge der vier größten mobilen Plattformen. Die aktuelle API (Stand Oktober 2014) stellt dem Entwickler die neuesten Funktionen für die entsprechende Zielplattform zur Verfügung. Die Integrated Development Environments (IDEs) beinhalten Editor, Debugger, Compiler usw. Aufgelistet sind die vom Hersteller der Zielplattform bereitgestellten Umgebungen, falls vorhanden. Oftmals sind IDEs zu verschiedenen Zielplattformen kompatibel. Beispielsweise existiert ein Plug-In für Visual Studio, welches die Entwicklung von BlackBerry Native Apps erlaubt. Jede Zielplattform unterstützt eine andere Teilmenge von Programmiersprachen. Auffällig ist, dass viele Hersteller eigene Standards etablieren. iOS stellt mit der neuen Betriebssystemversion 8 die eigens entwickelte objektorientierte Programmiersprache Swift vor. Microsoft bietet zur Beschreibung der Benutzeroberfläche die XAML an. Außerdem stellen die Hersteller Emulatoren bereit, mit denen die entwickelten Apps auf dem Zielsystem getestet werden können.

Native Entwicklungs-werkzeuge	Google Android	Apple iOS	Microsoft Windows Phone	BlackBerry Native
Aktuelles API-Level	Android 5.0	iOS 8	Windows Phone 8.1	BlackBerry Native 10.3
Entwicklungs-umgebung	jede Java IDE, z.B. Eclipse, Android Studio	Xcode 7	Microsoft Visual Studio 2013	Momentics IDE 2.1.1
Programmier-sprachen (direkt vom Hersteller unterstützt)	Java	Objective-C, Swift	C# oder C++ mit XAML, C++ mit DirectX, JavaScript/CSS/HTML	C/C++, JavaScript/ CSS/ HTML, Java, ActionScript/AIR
Emulator	Android Emulator	iOS Simulator	Windows Phone 8.1 Emulator	BlackBerry 10 Device Simulator

Tabelle 2 Die Entwicklungswerkzeuge der vier verbreitetsten mobilen Plattformen

Die native Entwicklung mobiler Applikationen bietet einige Vorteile. Die Hardware der Geräte und Funktionen der Plattform können in vollem Umfang angesprochen werden, indem die betriebssystemeigene API genutzt wird. In Windows Phone wird dem Entwickler durch die direkte Programmierung mit dem .NET-Framework eine maximale Flexibilität und Funktionalität bereitgestellt ohne eine dazwischenliegende Abstraktionsschicht. Dies ermöglicht eine maximale Performanz bzw. Mächtigkeit der Plattform bei der Umsetzung der Lösung zu nutzen.

Jede Plattform besitzt eine eigene Philosophie, wenn es um die Gestaltung der Benutzeroberfläche, Navigation und Nutzerinteraktion geht. Zahlreiche Guidelines sollen dem Entwickler beim Design eines für die jeweilige Plattform typischen Look-and-Feel verhelfen. Nativ erstellte Apps können das Look-and-Feel der jeweiligen Zielplattform vollständig umsetzen.

Bei der Entwicklung für eine spezielle Plattform ist der Code nicht auf eine andere Plattform übertragbar. Das Portieren einer Windows Phone Applikation auf die Android-Plattform ist durch unterschiedliche Programmiersprachen, Design-Pattern, Hardwareausstattung der Geräte und Funktionen usw. nur durch eine komplette Neuentwicklung möglich. Die Unterstützung jeder weiteren Plattform erfordert beträchtlichen Aufwand.

Die Verbreitung nativer Apps geschieht durch App-Stores. Jeder Plattformentwickler stellt seinen eigenen Store bereit. Beispiele sind Googles „Play Store“ oder BlackBerrys „BlackBerry World“. Auf allen Plattformen durchlaufen Apps einen umfangreichen Validierungstest, um beispielsweise missbräuchliche Inhalte oder Funktionen auszuschließen. Die Dauer eines Tests für eine eingereichte App beträgt mehrere Tage, sodass eine neue Version oder ein Update der App nicht sofort dem Nutzer zur Verfügung steht.
Kandidaten für die native Entwicklung sind Applikationen, die ein umfangreiches User-Interface mit zahlreichen Interaktionen benötigen; DirectX-Schnittstelle ansprechen; spezielle Sensoren mit schneller Abtastung benötigen; intensive Berechnungen ausführen oder hohe Sicherheitsanforderungen stellen. Dazu zählen beispielsweise Spiele, Kamera-Apps und Chat-Programme.

b. Hybrid

Hybride Apps besitzen eine native Komponente und eine Web-Komponente. „Die native Komponente kann direkt auf die betriebssystemspezifischen Funktionalitäten zugreifen, während die Web-Komponente in einem entsprechenden Container läuft.“ (HILDEBRANDT, LUTHIGER, STAMM & YEREAZTIAN, 2012) Der Zugriff auf Sensoren kann damit über den nativen Teil erfolgen, sodass plattformspezifische Funktionen genutzt werden können.
Die Kommunikation zwischen den zwei Komponenten wird meist über ein Framework realisiert. Dieses Framework legt gleichzeitig die zu nutzende Programmiersprache fest. Typische Standards sind HTML, JavaScript, Cascading Style Sheets (CSS) und C#. Durch diese zusätzliche Kommunikationsschicht besitzt eine hybride App meist eine geringere Performanz als ein nativer Vertreter.
Plattformunabhängiger Code sollte in der Web-Komponente realisiert werden. Dieser Code wird plattformübergreifend unterstützt. Damit ist der Aufwand für die Umsetzung einer App auf unterschiedlichen Plattformen vorwiegend unabhängig von deren Anzahl.
„Das Framework liefert den In-Between-Code, der den plattformunabhängigen Code in die native API übersetzt.“ (HILDEBRANDT, LUTHIGER, STAMM & YEREAZTIAN, 2012) Dadurch kann nur für Plattformen entwickelt werden, die das Framework unterstützt. Der Entwickler ist auf die ständige Weiterentwicklung von diesem angewiesen.
Das Look-and-Feel der Anwendung wird ebenfalls durch das genutzte Framework festgelegt. Deshalb entspricht die erstellte Anwendung meist dem Aussehen einer speziellen nativen Plattform. Der Entwickler muss außerdem beachten, dass eine Bedienung auf allen zu unterstützenden Plattformen bzw. Geräten möglich ist. Beispielsweise besitzt das iPhone keinen Back-Button wie ein Windows Phone.
Hybride Apps werden ähnlich nativen Apps meist in App Stores verbreitet, sodass hierbei die gleichen Probleme bezüglich Bereitstellung und Updateverhalten entstehen.

c. Web

Web-Applikationen laufen auf einem Web-Server. Der Client ist der Browser des Nutzers. Er wird als Benutzerinterface verwendet. Durch die Nutzung von JavaScript und HTML5 verlagert sich die Ausführung des Codes hin zum Client. Diese Art von Apps besitzt keinen direkten Zugriff auf Betriebssystemfunktionen. Das Ansprechen von Hardware und Kommunikationsschnittstellen ist damit stark eingeschränkt. Lediglich die durch JavaScript und HTML5 bereitgestellten Zugriffe auf standardisierte Schnittstellen können einige Sensoren

auslesen. Die Performanz ist dabei wesentlich schlechter, als bei den nativen oder hybriden Vertretern. (Vgl. HILDEBRANDT, LUTHIGER, STAMM & YEREAZTIAN, 2012)
Durch die Nutzung des Browsers als standardisiertes Applikationsfenster kann die App auf jeder Plattform ausgeführt werden. Beschränkungen entstehen jedoch durch eine unterschiedlich große Unterstützung des HTML5-Standards der Browser. Der potenzielle Anwenderkreis erhöht sich dennoch beträchtlich, im Vergleich zu den bereits vorgestellten Entwicklungsarten.
Das Look-and-Feel entspricht nicht dem nativer Anwendungen. Die typischen User Interface (UI)-Elemente des HTML5-Standards unterscheiden sich von nativen Vertretern. Ein einheitliches oder angepasstes Nutzerinterface ist nicht ohne Erweiterungen möglich. JQuery Mobile als JavaScript Framework verspricht jedoch eine an iOS angelehnte UI bereitzustellen. Die Nutzung von nicht nativen UI-Elementen muss durch die CPU aufwendig berechnet werden. Dies führt zu weiteren Performanceeinbußen, vor allem bei komplexen Benutzerschnittstellen. (Vgl. HILDEBRANDT, LUTHIGER, STAMM & YEREAZTIAN, 2012)
Die Bereitstellung von Web-Apps erfolgt über den Web-Server. Neue Applikationen oder Updates werden auf dem Server installiert und stehen dem Endanwender sofort zur Verfügung. Ein kostenpflichtiger und zeitaufwendiger Validierungsprozess entfällt.

	nativ	hybrid	web
Performance	Sehr gut	Gut	Befriedigend
Funktionalität abhängig von	Nativer Plattform	Frameworks wie PhoneGap und Titanium	Browser (JavaScript/HTML5 Unterstützung)
Hardwarezugriffe und Funktionalitäten	Uneingeschränkt	Eingeschränkt	Stark eingeschränkt
Look-and-Feel	Entspricht der jeweiligen nativen Plattform	Angepasst an maximal eine native Plattform	Unterscheidung von nativen Plattformen
Aufwand für mehrere Plattformen	Ansteigender Aufwand mit jeder weiteren Plattform	Mehrheitlich unabhängig von Anzahl unterstützter Plattformen	Unabhängig von Anzahl unterstützter Plattformen
Veröffentlichung	Über App-Store	Über App-Store	Auf Web-Server
Bereitstellungsdauer/ Updates	Verzögerung um einige Tage	Verzögerung um einige Tage	Aktualisierung auf Web-Server

Tabelle 3 Eigenschaften der Arten mobiler Applikationen

Tabelle 3 gibt einen zusammenfassenden Überblick der drei Arten mobiler Applikationen auf Grundlage der wichtigsten Unterscheidungskriterien.

2.2.4 Entwicklung und Management im Unternehmen

a. Relevante Plattformen im Enterprise Mobility Sektor

Gartner veröffentlicht jährlich seine Marktanalysen etablierter IT-Systeme, den sogenannten „Magic Quadrant for …“. Im Jahr 2014 wurden im Bereich Enterprise Mobilität zwei Arten Mobiler Plattformen analysiert, die im Kontext dieser Ausarbeitung von Bedeutung sind. Dazu gehören MADPs sowie EMM-Systeme. EMM-Systeme sind eine Weiterentwicklung der

MDM-Plattformen, die nicht nur Geräte und Anwendungen verwalten, sondern erweiterte Funktionalitäten wie MCM, ausgeweitetes MAM sowie BYOD-Aspekte mit weitreichenden Sicherheitsmerkmalen aufweisen. Bis 2013 wurde der „Magic Quadrant for Mobil Device Management“ jedes Jahr veröffentlicht. Durch die Entwicklung des Marktes hat sich der Fokus vom reinen MDM zu EMM verschoben. Dies zeigt die Aktualität und Bedeutung der Plattformen im Bereich Enterprise Mobility und begründet die weitere Auseinandersetzung mit diesen im folgenden Abschnitt. MADPs sind Plattformen, die sich mit der Entwicklung und Verwaltung mobiler Applikationen für verschiedene BSs und Geräte beschäftigen. Sie sind Bestandteil sogenannter Mobil Enterprise Application Platforms (MEAPs), welche für die technische Anbindung zwischen Back-End-Services und mobilen Applikationen verantwortlich sind. Da sie die Grundlage für das EMM bilden, erfolgt eine Einführung zu diesen Plattformen ebenfalls im folgenden Abschnitt. Bei den einzelnen Begriffen sollte beachtet werden, dass die entsprechenden Systeme mit ihren Funktionalitäten einer ständigen Weiterentwicklung unterliegen und die verschiedenen Hersteller unterschiedliche Begriffsdefinitionen anbieten. Eine klare Abgrenzung der Begriffe existiert derzeit nicht.

b. MEAPs sowie deren Abgrenzung zu MADPs und EMMs

Aufgrund der heterogenen Geräte und Systemlandschaft gestaltet sich die Entwicklung mobiler Applikationen für das eigene Unternehmen schwierig. Zentrale Fragen des Unternehmens sind:

- Welche mobilen Plattformen sollen im Unternehmen unterstützt werden? (Google Android, Apple iOS, Windows 8 Phone, Blackberry OS etc.)
- Wie können zu entwickelnde Enterprise Applikationen auf diesen Systemen gleichermaßen umgesetzt werden? Nach Möglichkeit sollten sich die Erfahrungen der Nutzer beim Einsatz der Applikation auf den verschiedenen BSs nur geringfügig unterscheiden.

Einen Ansatz bieten MEAPs. Sie stellen Entwicklungswerkzeuge für mobile Anwendungen bereit. Das Ziel dabei ist, Anwendungscode einmal zu schreiben und diesen, durch möglichst geringfügige Modifikationen, für verschiedenste Geräte zur Verfügung stellen zu können.
WIEMANN (2013) gibt folgende Definition einer solchen Plattform. „Eine Mobile Enterprise Application Platform (MEAP) deckt alle Bereiche der Entwicklung, der Einrichtung und dem Betrieb/Management mobiler Anwendungssysteme ab. Sie ermöglicht die Bereitstellung von Anwendungen auf heterogenen Endgeräten und heterogenen Betriebssystemen aus einem Quellcode.“ Die Definition spricht die Unterstützung verschiedener BSs an. Wie bereits im Abschnitt zu mobilen Plattformen erwähnt, sind die verwendeten Begriffe im Sektor Mobility nicht klar abgegrenzt. Blackberry definiert: „[…]MEAPs bieten alle relevanten Komponenten und die Software-Infrastruktur für den Support eines großen Einsatzes von mobilen Applikationen.“ (BLACKBERRY, 2014a) Dabei ist nicht festgelegt, dass verschiedene BSs unterstützt werden müssen. BAL (2013) schreibt: “Mobile Enterprise Application Platforms are pre-built environments that allow an individual to fabricate a mobile application with the intended purpose of deploying the application to multiple mobile operating systems.”
MEAPs enthalten damit nicht nur Komponenten zur Entwicklung, sondern auch zur Verbreitung und zum Management mobiler Applikationen. Die Umsetzung erfolgt über das Bereitstellen einer Middleware, die Applikationen auf den jeweiligen Endgeräten verwaltet und

die Verbindung zu diversen Back-End-Services bereitstellt. Dies können bereits vorhandene ERP-Systeme, Cloud-Dienste oder Ähnliches sein.
Weiterhin hat sich der Begriff der MADPs etabliert. Der Unterschied zwischen MEAP und MADP lässt sich schwer fassen, falls er existiert. Die Definition nach Gartner "The mobile application development platform (MADP) market offers tools, technologies, components and services […]. The MADP enables an enterprise to design, develop, deploy, distribute and manage a portfolio of mobile applications running on a range of devices […]." (VALDES, VAN L. BAKER, MARSHALL & WONG, 2014) kommt dem Begriffsbild von WIEMANN (2013) sehr nahe. Meist wird der Begriff MADP genutzt, wenn der Fokus auf der Entwicklung von Enterprise Applications liegt. MEAPs werden häufig mit zusätzlichen Management Funktionalitäten in Verbindung gebracht. In diesem Dokument werden die Begriffe als gleichwertig angesehen. Eine weitere Möglichkeit bestünde darin, MADPs als Bestandteile von MEAP anzusehen.
Die Abgrenzung zu EMM-Systemen wird wie folgt vorgenommen. „Idealerweise erfolgt die technische Integration mobiler Anwendungen über eine entsprechende Mobile Enterprise Application Plattform (MEAP) auf Basis einer serviceorientierten Schnittstellenlandschaft. Die Verwaltung der Geräte, Applikationen und Daten sollte durch eine Enterprise-Mobility-Management-Lösung (EMM) erfolgen." (ARNS, BESSING, BUGGISCH & GRACKLAUER, 2014) Damit erfolgt die Kommunikation der entwickelten Applikationen mit der vorhandenen IT-Landschaft über MEAPs. Das Management erfolgt durch EMMs. Wie bereits erwähnt, dienen MEAPs ebenfalls der Verwaltung von Applikationen. Dies wird als Schnittmenge der beiden Plattformarten identifiziert.

c. Funktionalitäten

Die zur Verfügung gestellten Werkzeuge und Funktionalitäten einer MEAP sollen dem Entwickler eine schnelle und einfache Entwicklung von Anwendungen für verschiedene Systemkonfigurationen erlauben ("code once, deploy many").
Zu den Entwicklungswerkzeugen sollte ein Debugger und Emulator gehören. Damit wird eine Fehleranalyse und eine wirklichkeitsgetreue Ansicht und Laufzeit der Anwendung, ohne dass ein entsprechendes Geräte der jeweiligen Plattform vorhanden sein muss, gewährleistet.
Schließlich sollte der Entwickler bei der Wahl einer IDE wenige Einschränkungen erfahren. Werkzeuge wie Eclipse, Android Studio, Visual Studio und Xcode sind weit verbreitet. Einige Hersteller bieten Software Development Kits (SDKs) an, die in verschiedenen IDEs verwendet werden können. What You See Is What You Get (WYSIWYG) Editoren mit Drag & Drop Funktion zur Erstellung von UIs gehören zum Standard.
Für die Erstellung der ersten Apps im Privatgebrauch eines Entwicklers, mit eventueller Veröffentlichung in einem der renommierten App-Stores, mögen diese Werkzeuge ausreichen. Die Entwicklung im Unternehmensbereich erfordert allerdings die Anforderungen solcher Applikationen in die Entwicklung miteinzubeziehen und tiefgreifende Maßnahmen und Strategien zu verfolgen, um sichere, benutzerfreundliche und innovative Apps entstehen zu lassen. Nach BLACKBERRY (2014d) sind folgende Kategorien von Funktionalitäten für eine erfolgreiche Plattform entscheidend.

i. Konnektivität

Folgende vier Eigenschaften sollte eine MADP zur Verfügung stellen:
Die Entwicklung von Apps, die

- nicht Session-basiert sind.
- eine sichere Verbindung zum Firmennetzwerk erlauben.
- bei unterschiedlicher Netzwerkabdeckung arbeiten.
- die Akkulebensdauer schonen.

Mobile Enterprise Applikationen greifen auf Firmendaten zu, die hinter einer Firewall geschützt sind („behind-the-firewall data“). Um eine Verbindung zu ermöglichen, werden VPN-Verbindungen benutzt. Diese sind Session-basiert. Bei einer permanenten und zuverlässigen Verbindung zum Netzwerk ist diese Technik sicher und komfortabel. Mobile Applikationen werden jedoch in unterschiedlichen mobilen Systemumgebungen genutzt. Es muss damit gerechnet werden, dass Verbindungen oft unterbrochen sind. Dies führt bei VPN-Verbindungen zu einem erneuten Log-In-Prozess. Für den Endanwender stellt dieses Verhalten eine deutliche Verminderung der Benutzbarkeit dar. Lösungen, die ein ständiges Senden von Keep-Alives vorschlagen, übersehen, dass dies die Akkulaufzeit des mobilen Endgerätes vermindert. (Vgl. BLACKBERRY, 2014d)

ii. Data on Demand

MADPs sollten für diesen Bereich die folgenden Eigenschaften vorweisen:

- Bereitstellung von Benutzerbenachrichtigungen (Notifications),
- Push- und Sync-Technologien sowie
- Background-Tasks

Mobile Enterprise Applikationen sollten den Benutzer über wichtige Ereignisse benachrichtigen, die seine Aufmerksamkeit bedürfen oder sein Handeln voraussetzen. Solche Nachrichten werden als Notifications bezeichnet.
Die Synchronisation der Daten mit dem Back-End kann über verschiedene Strategien erfolgen. Eine Möglichkeit besteht darin, durch einen Push-Mechanismus alle Enterprise-Daten auf das Gerät zu laden. Im Falle einer Offline-Nutzung der App sind die Daten sofort nutzbar. Alternativ kann der Nutzer benachrichtigt werden, dass eine Synchronisation nötig ist, wenn er auf bestimmte Daten zugreifen möchte. Hier muss er sich in einen Bereich begeben, der eine Netzwerkverbindung zulässt, um die benötigten Daten zu laden. Der Vorteil dieser Methode besteht im geringeren Speicherverbrauch auf dem mobilen Endgerät.
Kleinere Datenmengen werden durch eine Push-Nachricht sofort auf das Endgerät transportiert. Eine Datenkompression hilft die Größe der Daten klein zu halten. Eine verschlüsselte Ende-zu-Ende-Verbindung sorgt für die nötige Sicherheit bei der Übertragung. Um das Transportieren größerer Datenmengen zu möglichen, wird ein push-and-pull Ansatz genutzt. Hier wird in einer initialen Push-Nachricht von einem App-Server eine URL für den darauf folgenden Download (Pull) der Daten gesendet. Damit wird das anfängliche Datenlimit in Push-Nachrichten umgangen.
Ein Background-Task wird benötigt, um die Daten einer Push-Nachricht automatisch entgegenzunehmen und zu beantworten, ähnlich einem Server-Socket, das auf eingehende

Verbindungen wartet. Nicht alle mobilen BSs lassen Background-Tasks zu, was wiederum die Wahl des BS von der Art der mobilen Applikation abhängig macht.

iii. Integriertes Mobile Application Management

Mobilität hat deutlich höhere Ansprüche an das Management von Applikationen, als dies von Desktop-Applikationen bekannt ist. Gründe sind BYOD-Umgebungen und die Fragmentierung im Bereich mobiler BSs und Anwendungen. Ein MAM ist zuständig für das Management des App-Lebenszykluses. Dies umfasst das Testen, Veröffentlichen, Updaten und Absetzen der Applikation. Weitere Bestandteile sind die Software-Auslieferung, -Lizensierung, -Konfiguration und das Nutzer-Tracking. Zu den kritischen Anforderungen eines MAM gehören die Kontrolle des Datenaustauschs zwischen den mobilen Applikationen und der ferngesteuerte Zugriff auf Unternehmensdaten. Ein Kernbestandteil ist der Enterprise Application Store aus dem Mitarbeiter selbstständig zur Verfügung gestellte Apps des Unternehmens herunterladen können. Die Entwicklung einer App sollte standardmäßig die Möglichkeit des mobilen Managements integrieren.

iv. Offene Standards & Open Source

Die Entwicklung von nativen Apps, Web-Apps und hybriden Apps sowie die Unterstützung von Cross-Plattform Werkzeugen gehört im Bereich der MADPs zum Standard. Durch den steigenden Trend zu HTML5 sollte die Möglichkeit der Web-App Entwicklung dem Entwickler in jedem Fall zur Verfügung stehen.

v. Gleichartige User Experience

Kriterien, die eine Herausforderung für den Entwickler darstellen, sind:

- die Vielfalt an mobilen BSs und Geräten,
- verschiedene Displaygrößen,
- die Batterielebensdauer und
- die Nutzungsbedingungen.

Eine MADP sollte die Entwicklung von Apps mit einer gleichartigen Benutzererfahrung zwischen verschiedenen Plattformen ermöglichen. Dabei sind Konzepte für an die Nutzer anpassbare Oberflächen zu verwenden.
Eine einfache Bedienung, die es ermöglicht, mit wenigen Interaktionen eine Aufgabe zu lösen oder Informationen zu beschaffen, ist notwendig.
Durch kleine Displaygrößen, unterschiedliche Hardware, verschiedene Versionen der BSs sowie Einschränkungen innerhalb der BSs wird es dem Entwickler erschwert, die genannten Anforderungen umzusetzen. Aus diesen Gründen unterscheidet sich der Aufwand für die Erstellung einer gleichartigen Benutzererfahrung im mobilen Sektor gegenüber dem Desktop-Bereich.

vi. Integrierte Sicherheit

Da Apps auf „behind-the-firewall“-Daten zugreifen, sind integrierte Sicherheitsmechanismen besonders wichtig. Sie sollten eng mit dem Entwicklungsprozess verknüpft werden. (Vgl. BLACKBERRY, 2014d)
Von besonderer Bedeutung sind Werkzeuge zur Umsetzung des BYOD-Gedankens. Unternehmensdaten sollten stets in einer sicheren Umgebung aufbewahrt werden. Dafür existieren unterschiedliche Ansätze. Bei der Containerization werden sowohl persönliche als auch Unternehmensdaten in einem separaten Container aufbewahrt. Damit werden zwei voneinander abgeschottete Bereiche auf einem Endgerät definiert, die keinerlei Einfluss aufeinander ausüben können. Potenziell schädlicher Software im persönlichen Bereich ist es nicht möglich, Unternehmensdaten auszulesen oder gar zu verändern. Das Unternehmen hat ausschließlich Zugriff auf den Unternehmensbereich und kann beispielsweise durch einen Remote-Zugriff Daten hinzufügen oder löschen. Die Privatsphäre im persönlichen Bereich wird gewahrt. Nachteile entstehen durch die strikte Trennung der Bereiche. Ein verschieben von Text zwischen den Bereichen ist nicht möglich, privater und Firmenkalender müssen separat geführt werden usw. Dadurch können Mitarbeiter in ihrem natürlichen Arbeitsfluss gestört werden und die Produktivität sinkt. Einige Unternehmen wenden sich von dieser Strategie mit der Begründung ab: „this isn't how my users work“ oder „this isn't how human beings work.“ (FAAS, 2013)
Ein weiterer Ansatz ist das klassische MDM. Hier werden persönliche Daten und Unternehmensdaten gemeinsam verwaltet. Das Unternehmen legt fest, welche Software auf dem Gerät ausgeführt werden darf. Dabei soll gewährleistet werden, dass Schadsoftware nicht den Weg auf das Gerät findet. Die Einschränkungen betreffen hier vor allem die Privatsphäre des Endanwenders, dessen persönliche Daten dem Unternehmen offen gelegt werden, von diesem verändert werden können und eine Ausführung bestimmter Anwendungen untersagt werden kann. Vorteile für den Nutzer sind durch die geringe Separation der Bereiche auszumachen. Ein paralleles Arbeiten an privaten und geschäftlichen Daten ist möglich. Dies bringt aber zahlreiche Sicherheitsrisiken mit sich. Der Nutzer kann beispielsweise private Anwendungen installieren, die Schadsoftware enthalten. Außerdem können geschäftliche Daten durch private Anwendungen ausgelesen werden und somit unbemerkt in fremde Hände fallen. Um dies zu verhindern, muss das MDM strikte Vorgaben liefern.

3 Das Ökosystem einer mobilen Enterprise Applikation

Endanwender bzw. Unternehmen sehnen sich nach einfach zu bedienenden mobilen Applikationen, die an ihre Bedürfnisse angepasst sind. Sie führen komfortabel die in diesem Moment benötigte Funktion aus und zeigen entscheidungskritische Informationen an. Dies erweckt den Anschein, dass ebenfalls die dahinterstehenden Mechanismen einfach realisierbar sowie überschaubar seien und der Entwicklungsaufwand geringer, im Vergleich zu Desktop-Anwendungen, eingestuft wird.
Dieses Kapitel soll ein grundlegendes Verständnis über den Aufbau und die Umgebung einer mobilen Enterprise Applikation geben. Dabei werden zwei Perspektiven dieses Ökosystems veranschaulicht. Zum einen die Sicht auf eine mobile Applikation von der Seite der Unternehmen und zum zweiten die Sicht des Entwicklers. Neben der Herausarbeitung einer Schichtenarchitektur für Enterprise Apps werden relevante Technologien und Mechanismen für jedes Teilsystem aufgezeigt. In Abhängigkeit zur betrachteten Perspektive sind verschiedene Kommunikationspartner bzw. Back-End-Systeme im Einsatz, die aufgezeigt und in die Umgebung der mobilen Applikation eingeordnet werden.
Ziel dieses Abschnitts ist es, die Komplexität der Entwicklung mobiler Applikationen durch das Aufzeigen der beteiligten Teilsysteme und Sichten greifbar zu machen. Für den Entwickler ist es notwendig, beide Sichten zu verstehen. Des Weiteren wird der Übergang zu Kapitel 4 ermöglicht, welches elementare Kriterien für die Entwicklung mobiler Enterprise Apps beschreibt.

3.1 Veranschaulichung von Aufbau und Umgebung

Abbildung 2 zeigt die Umgebung, in welcher mobile Enterprise Applikationen entwickelt und eingesetzt werden. Deutlich wird die Komplexität dieses Gesamtsystems im Hinblick auf die große Anzahl an beteiligten Teilsystemen und Technologien. Die Verknüpfungen zwischen diesen sind aus Gründen der Übersichtlichkeit stark vereinfacht dargestellt und im realen Einsatz schwer zu überschauen.
Die Darstellung unterscheidet zunächst zwischen der Entwicklersicht, in der für den Softwareentwickler relevante Aspekte dargestellt werden und der Unternehmenssicht. Diese stellt die von Unternehmen gestellten Anforderungen und eingesetzten Systeme dar. Teilsysteme wie MDAPs oder ERP-Systeme sind in einem dunkelgrau gefassten Container abgelegt. Relevante Technologien sind in weißen Containern hervorgehoben. Durch die Spiegelung zwischen den Sichten sollen relevante Aspekte innerhalb eines Themenbereichs vergleichbar dargestellt werden. Knotenpunkte bzw. Smartphone-Symbole stehen für die einzelnen Schichten des Ökosystems einer mobilen Enterprise Applikation.

3.2 Untersuchung des Ökosystems aus zwei Sichten

Dieser Abschnitt befasst sich mit den in Abschnitt 3.1 erwähnten Sichten auf eine mobile Enterprise App und bezieht sich somit unmittelbar auf Abbildung 2. Bei der Aufstellung von Anforderungen und Kriterien sowie der Entwicklung einer mobilen Applikation ist ein Verständnis beider Sichten unabdingbar.

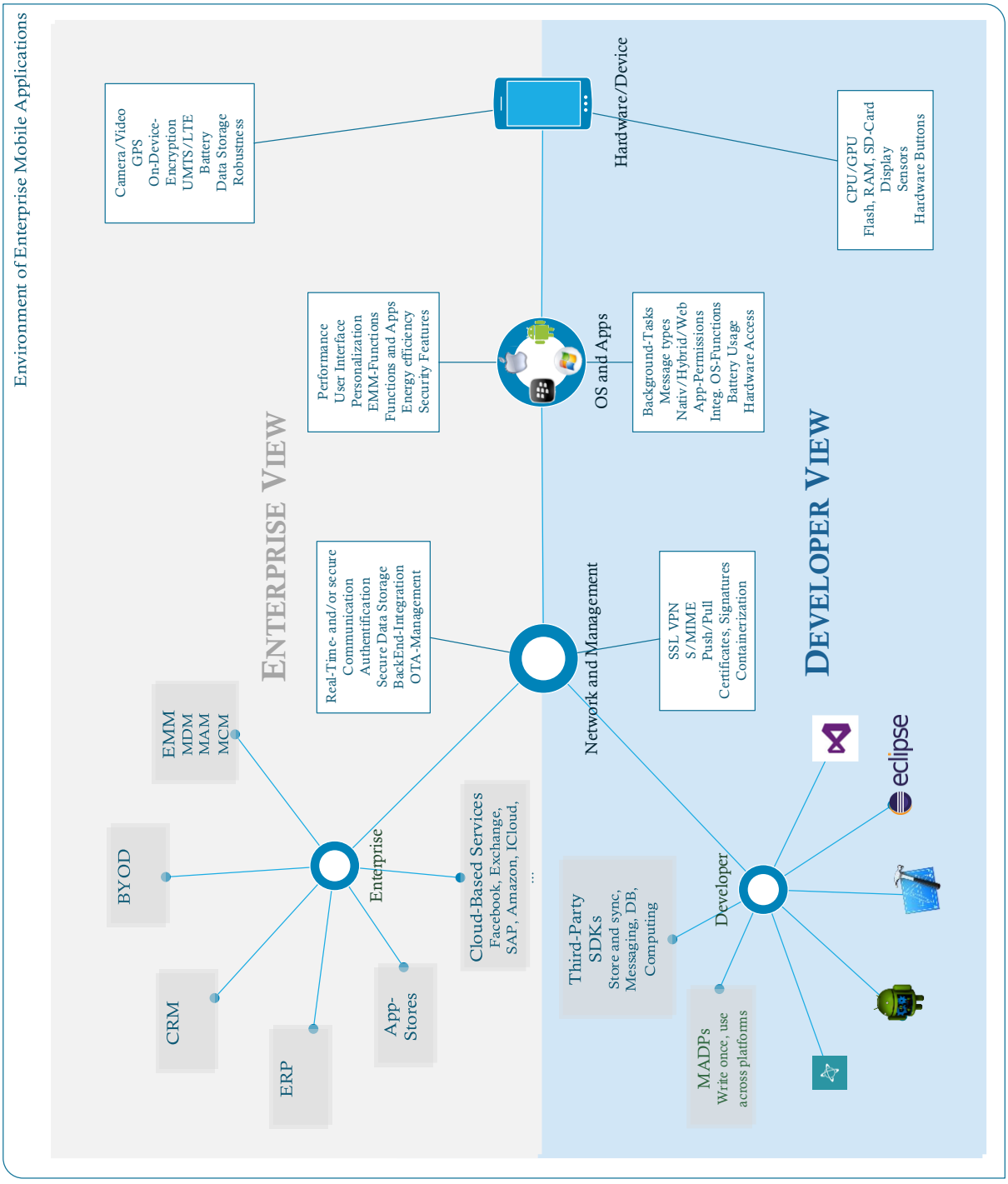

Abbildung 2 Umgebung von mobilen Enterprise Applikationen

3.2.1 Perspektive des Unternehmens

Die Unternehmensperspektive stellt die Sicht des Unternehmens auf eine mobile Applikation dar.

Geräte und Hardware

Angefangen bei der Hardware sind Enterprises an Bestandteilen interessiert, die eine bestmögliche Unterstützung des Mitarbeiters bei der Bewältigung seiner täglichen Aufgaben gewährleisten. Man stelle sich einen Techniker vor, der verschiedenste Wartungsarbeiten an Windkraftanlagen durchführt. Zur Aufnahme der einzelnen Fälle nutzt er eine vom Unternehmen bereitgestellte App. Zum Festhalten des Schadens wird die Kamera- bzw. Videofunktion benötigt. Der Ort des Schadens wird über eine GPS-Ortung automatisch eingetragen. Die Akkulaufzeit des Geräts sollte groß genug sein, um mind. einen Fall komplett ohne zwischenzeitliches Aufladen bewältigen zu können. Der Techniker muss das Gerät unter verschiedensten Wetterbedingungen einsetzen können. Es muss gegen Wasser, Staub, Stöße und ähnliches abgesichert sein. Die Aufträge eines Tages werden auf dem Gerät abgespeichert. Der Datenspeicher muss ausreichend dimensioniert werden. Außerdem verlangen die Endkunden eine vollkommen sichere Aufbewahrung der Auftragsdaten. Eine Verschlüsselung auf dem Gerät muss bereitgestellt werden. Die Vergabe von Aufträgen wird ebenfalls über die genannte App abgewickelt. Notfälle und zeitkritische Anfragen können jederzeit eingehen und eine schnelle Kommunikation ist nötig. Zur Verständigung zwischen Zentrale und Mitarbeitern wird eine Verbindung über das Mobilfunknetz per Universal Mobile Telecommunications System (UTMS) bzw. Long Term Evolution (LTE) genutzt.

Mobile Betriebssysteme und Apps

Die Wahl des mobilen BS ist eng mit der Auswahl an Hardware-Bausteinen verbunden. Auf dem Endkundenmarkt wird ein Gerät ausschließlich mit einem spezifischen BS verkauft. Dies hat den Vorteil, dass die Performance des BS mit der Performance der Hardware abgestimmt wurde. BSs, die höhere Anforderung an die Hardware stellen, werden entsprechend mit leistungsfähigeren und dafür teureren Komponenten verkauft. Dies lässt wiederum Rückschlüsse auf die Effizienz des BS und damit auf die Ausnutzung der vorhandenen Ressourcen zu. Eine Abwägung zwischen Kosten und Nutzen ist ein Kriterium bei der Auswahl eines geeigneten BS. Schließlich spielt die Benutzeroberfläche und Möglichkeiten der Personalisierung eine elementare Rolle. Ein weiteres Entscheidungskriterium besteht darin, welche mobilen Anwendungen auf dem Gerät ausgeführt werden sollen. Es stellt sich die Frage: Sind die entsprechenden Anwendungen für das ausgewählte BS verfügbar und bietet die ausgewählte Plattform ausreichend Potenzial für zu entwickelnde Apps? Dieses Potenzial kann auf verschiedene Weise interpretiert werden. Möglichkeiten sind:

- Welche Funktionen sind bereits im BS integriert? Wichtigste Vertreter sind Sicherheitsmechanismen zur Verschlüsselung und Autorisierung/Authentifizierung, die Synchronisation mit dem Back-End sowie integrierte EMM-Funktionalitäten.
- Wie groß sind der Akkuverbrauch und damit die Energieeffizienz des Systems?

Netzwerk und Management

Der Bereich Netzwerk und Management ist geprägt durch Sicherheitsaspekte. Unternehmen sind in erster Linie an der Wahrung von Firmengeheimnissen interessiert. Aus diesem Grund wird eine sichere Kommunikation, Datenübertragung und Datenhaltung gefordert. Außerdem sind Mechanismen für die Echt-Zeit-Kommunikation, d.h. garantierte und schnelle Übertragung von Ereignissen, um schnellstmögliche Entscheidungsprozesse zu erlauben, von großer Bedeutung. Die Anbindung von Back-End-Services sowie die Authentifizierung an diesen Systemen per Username und Passwort bzw. zertifikatbasierte Lösungen sind ebenfalls entscheidende Punkte aus Sicht der Unternehmen. (Vgl. KERN, 2012) Das Management der Geräte durch MDM-Funktionen des BS über das MDM-System bzw. EMM-Plattformen ist ein zentraler Bestandteil moderner mobiler Infrastrukturen.

BackEnd-Systeme

Über das Netzwerk wird die Kommunikation der mobilen Unternehmensapplikation mit verschiedensten Back-End-Systemen ermöglicht. Die Integration von vorhandener IT-Infrastruktur, wie ERP- und CRM-Systeme, ist unerlässlich, um automatisierte mobile Geschäftsprozesse zu ermöglichen. Damit sind nicht nur die internen Geschäftsprozesse der Unternehmen zu mobilisieren. Ebenfalls ändern sich stark die Geschäftsabläufe zwischen Unternehmen und Kunden. „Kunden und Geschäftspartner werden direkter und unmittelbarer in die Geschäftsabläufe einbezogen, die Kundenkontakte finden situations- und ortsabhängiger statt.“ (VERCLAS & LINNHOFF-POPIEN, 2012a) Die neu geschaffene Mobilität bedingt einigen zusätzlichen Managementaufwand. EMM-Plattformen sind aufzubauen und BYOD-Lösungen auszuwählen. Cloud-basierte Dienste wie Exchange-Server, Soziale Netzwerke und Cloud-Speicher werden fest mit der Nutzung mobiler Enterprise Applikationen verknüpft. Schließlich sind entwickelte Apps in einem App-Store des Unternehmens zur Verfügung zu stellen.

3.2.2 Perspektive des Entwicklers

Die Entwicklerperspektive auf eine mobile Enterprise Applikation ist von den speziellen Technologien und Prinzipien geprägt, durch deren Anwendung der Entwickler die genannten Anforderungen des Unternehmens umsetzt.

Geräte und Hardware

Ausgehend von der oben beschriebenen App im Störungsmanagement sind Sensoren für Video- und Kamera-Funktion sowie zur Sprachaufzeichnung von Bedeutung. Hardware-Buttons sind meist fest mit dem Benutzungskonzept des installierten BS verbunden und müssen bei der Planung der Benutzerinteraktion einbezogen werden. Gleiches gilt für die Display-Eigenschaften und dessen Größe, welche für die Planung des Benutzerinterfaces der App entscheidend ist. Vorhandene Speichermöglichkeiten wie Flash-Speicher oder Erweiterungen durch Smart Digital (SD)-Karten-Slots und deren Ausmaß sind bei der persistenten Datenhaltung zu berücksichtigen. Hardware-Bausteine wie System-on-a-Chip (SoCs) oder RAM-Bestückungen sollten in Zusammenarbeit mit dem BS die Performanceanforderungen sowie Anforderungen an das Energiemanagement der zu entwickelnden Applikation erfüllen.

Alle diese Hardware-Ressourcen stellen knappe Betriebsmittel dar und müssen sorgfältig verwaltet werden. (Vgl. BAUMGARTEN, BERNHOFER & DÖRFEL, 2012)

Mobile Betriebssysteme und Apps

Mobile BSs sollten dem Entwickler reichhaltige Möglichkeiten bei der Entwicklung zur Verfügung stellen. Die grundlegende Komponente dieser BSs bildet der Betriebssystemkern. Er übergibt dem Entwickler die grundlegenden Konzepte und Abstraktionen, zu denen unter anderem die Verwaltung der Betriebsmittel zählt. „Wichtige Unterschiede bestehen darin, welche Funktionen oder Dienste des BS den Anwendungen zur Verfügung gestellt werden sollen und wie dies geschehen soll." (BAUMGARTEN, BERNHOFER & DÖRFEL, 2012) Konzepte und Mechanismen sind beispielsweise das Multiprogramming, das Zulassen von Hintergrundprozessen, die beispielsweise selbst bei Inaktivität einer App weiterhin Benachrichtigungen empfangen oder senden können und die Verwendung von synchroner Kommunikation bzw. asynchronen Ereignissen. Elementar ist außerdem der Zugriff auf Hardwareschnittstellen. Hierunter fällt z.B. das Auslesen des GPS-Sensors zur Bestimmung der aktuellen geographischen Koordinaten. (Vgl. BAUMGARTEN, BERNHOFER & DÖRFEL, 2012) Eine wichtige Rolle spielen verschiedene Benachrichtigungsmechanismen, wie Push-Benachrichtigungen, Nachrichten in einem Notification-Center (Windows Phone) oder Hub (BlackBerry), auf dem Sperrbildschirm oder App-Symbol (Live Tile). Die mobilen BSs unterstützen davon meist nur eine Teilmenge. Schließlich ist das Energiemanagement von Bedeutung. Wann werden Apps oder Dienste in den Ruhezustand versetzt und wann werden sie beendet oder neu gestartet (App-Lebenszyklus)? Der Einsatz ressourcenschonender Mechanismen und Strategien ist unabdingbar. Durch bereits integrierte BS-Funktionen versuchen sich die Hersteller voneinander abzuheben. Immer mehr Enterprise Funktionalitäten, wie MDM, finden standardmäßigen Einzug in die BSs. BlackBerry bietet die vollständige Integration seiner EMM-Plattform (Blackberry Enterprise Server (BES) 10) an, Windows Phone 8.1 besitzt einen integrierten MDM-Client, Android 5 wird mit nativer Unterstützung für Samsungs mobile Enterprise-Lösung Knox ausgeliefert und iOS 8 enthält ebenfalls erweiterte Enterprise Features. Auf Ebene der Apps bildet das Management der App-Berechtigungen (App-Permissions) eine tragende Säule eines mobilen BS. Wann dürfen Apps auf welche Ressourcen zugreifen und wie stimmt der Endanwender dieser Nutzung zu? Nicht alle BSs stellen hier feingranulare Entscheidungsmöglichkeiten für den Nutzer zur Verfügung. Der Entwicklungsaufwand, welcher sich aus der Art der Entwicklung in nativ, hybrid oder web-App und der Wahl des mobilen BS mit seinen inhärenten Konzepten ergibt, ist ein zentraler Faktor für die späteren Entwicklungskosten.

Netzwerk und Management

Im Bereich Netzwerk und Management sind von Bedeutung sichere Verbindungen zum Firmennetzwerk durch SSL VPN, sicherer Nachrichtenaustausch durch Secure / Multipurpose Internet Mail Extensions (S/MIME) als Standard für die Verschlüsselung und Signatur von Multipurpose Internet Mail Extensions (MIME)-gekapselter E-Mails sowie verschiedene Push/Pull-Mechanismen. Banking-Software oder E-Mail Programme benötigen eine Verbindung zum Internet, um Daten nachzuladen. Der Einsatz von Transport Layer Security TLS als hybrides Verschlüsselungsprotokoll und aktueller Standards ist von größter Wichtigkeit. (Vgl. WILDT & MEISTER, 2012) Das Signieren von

Nachrichten bzw. Verifizieren einer Signatur sowie die Authentifizierung von Nutzern durch Zertifikate oder andere Verfahren sind ebenfalls in diesem Bereich angesiedelt. Der Bereich Management beschäftigt sich neben den genannten Sicherheitsmechanismen mit der Trennung und Verwaltung von persönlichen Daten und Unternehmensdaten auf dem mobilen Endgerät. Grundlegende Konzepte sind aus Abschnitt 2.2.4 c. vi. Integrierte Sicherheit zu entnehmen. Aus Entwicklersicht ist hier der Bereich der Virtualisierung von Prozessen bzw. Systemen interessant. Grundlagen zu virtuellen Maschinen zeigen SMITH & RAVI NAIR (2005). Beispiele zum Einsatz im Bereich mobiler Endgeräte werden von BAUMGARTEN, BERNHOFER & DÖRFEL (2012) dargestellt.

BackEnd-Systeme

Das mobile BS ist nur ein Teil der mobilen Plattformen, wie z.B. Googles Android-Plattform. Zusammen mit den Vertriebs- und Distributionskanälen haben sich diese Plattformen zu eigenen Ökosystemen entwickelt. Die Hersteller bieten jeweils eigene Entwicklungssysteme an. In Abbildung 2 sind die verbreitetsten Systeme aufgezeigt. BlackBerry bietet seine Momentics IDE an, Android-Entwickler sollen zukünftig Android Studio nutzen, Apple preist seine XCode Umgebung an und Microsoft bleibt bei seinem bewährten Visual Studio. Eclipse als offene IDE kann durch Plug-Ins für mehrere Plattformen eingesetzt werden. An dieser Stelle beginnt die Frustration der Entwickler. Soll eine App auf mehreren Plattformen erscheinen, so multiplizieren sich die Entwicklungskosten mit der Anzahl dieser. Dabei ist der gesamte Lebenszyklus einer App einschließlich Wartung und Pflege zu betrachten. (Vgl. WILLNECKER, ISMAILOVIĆ & MAISON, 2012) Zur Senkung der Kosten existieren Konzepte der hybriden und Web-Apps (siehe Abschnitt 2.2.3) sowie die entsprechenden Entwicklungsplattformen in Form von MADPs (siehe Abschnitt 2.2.4 b).
Die Entwicklung von Apps beinhaltet ebenfalls die Vernetzung mit verschiedenen Services. Das ist zum einen die Verbindung mit der unternehmenseigenen IT-Infrastruktur. Zum anderen spielen verstärkt Cloud Services von Drittanbietern eine Rolle. Allgemein wird von Mobile Cloud Computing (MCC) gesprochen. Eine Definition und genauere Betrachtung dieses Teilaspektes gibt Abschnitt 4.3. Die drei für den Entwickler interessanten Schichten des MCC sind Infrastructure as a Service (IaaS), Platform as a Service (PaaS) und Software as a Service (SaaS). IaaS beinhaltet Dienste wie Storage, Rechenleistung oder Virtualisierung. Beispiele sind Amazon Elastic Cloud Computing EC2 und Simple Storage Service (S3). PaaS bietet Umgebungen zum Erstellen, Testen und Verwalten von Applikationen. Hier werden Dienste wie Google App Engine, Microsoft Azure und Amazon Map Reduce angeboten. SaaS stellt Anwendungssysteme und -dienste zur Verfügung, die direkt aus der Cloud genutzt werden können ohne Installationsaufwand zu erzeugen. (Vgl. SIRTL & KOCH, 2012) Beispiele stellen die Anbieter Salesforce und Microsoft Mesh mit Diensten wie Filesharing und Synchronisation zur Verfügung. Zur Nutzung der Services stellen die Anbieter verschiedene SDKs für die unterschiedlichen Plattformen bereit. Beispiele sind das iOS bzw. Andoid SDK von Salesforce oder das Amazon Web Service SDK für Android.

3.3 Fokus auf nativer App-Entwicklung

Nachdem in den vorangegangenen Abschnitten die Grundlagen für ein Verständnis von Mobilität, ihrem Zusammenhang mit der Welt der Unternehmen und Geschäftsprozesse, der Mobilisierung dieser Prozesse durch mobile Enterprise Apps, der Zusammenhang zwischen der Unternehmenssicht und der IT-Sicht des Entwicklers beschrieben sowie die technischen Methoden und Konzepte eingeführt wurden, liegt der Fokus im Folgenden auf der Sicht des Entwicklers. Hier wird speziell auf die native Entwicklung von mobilen Enterprise Applikationen eingegangen. Als Fallstudie dient die Entwicklung einer Zeiterfassungs-App für den Unternehmensbereich. Die Entwicklungsplattform ist Windows Phone 8.1. Die native Entwicklung sorgt für die bestmögliche Unterstützung von Funktionalitäten sowie des Look-und-Feels der jeweiligen Plattform. Da eine native Entwicklung ausschließlich auf dem Ökosystem der Zielplattform erfolgt, können später direkte Rückschlüsse auf die Plattform und ihre Ausgereiftheit gezogen werden. Bei der Entwicklung von hybriden oder Web-Apps ist dies durch die Eingeschränktheit der nutzbaren Funktionalitäten der Zielplattform bzw. die Abhängigkeit zu Framework-Entwicklern nicht möglich. Zunächst erfolgt die Erstellung von Kriterien, die aus der Sicht des Entwicklers im Enterprise-Bereich von grundlegender Bedeutung sind. Die Kriterien werden erstellt auf der Grundlage des in diesem Abschnitt beschriebenen Umfeldes einer Enterprise Applikation und der aufgezeigten Teilbereiche, Technologien und Methoden der Entwicklersicht. Anhand dieser Kriterien werden die erstellte App und damit die zugrundeliegende Plattform auf ihre Tauglichkeit für den Unternehmensbereich geprüft.

4 Beschreibung grundlegender Kriterien der Entwicklung

Im Bereich der mobilen Applikationen stellen Enterprise Apps, im Vergleich zu den reinen Consumer Apps für den Massenmarkt, besondere Anforderungen an die Funktionalität und Entwicklung. In Zeiten, in denen das Firmengeheimnis oberste Priorität hat und eine ständige Produktivitätssteigerung gefordert wird, sind unter anderem Sicherheit, Kommunikation und Performanz grundlegende Aspekte der App-Entwicklung.
Kapitel 4 wird diese Hypothese untersuchen und anhand von Kriterien eine einheitliche Aufstellung der Anforderungen an mobile Enterprise Applikationen liefern. Als Grundlage und Orientierung dient die in Abbildung 2 beschriebene Umgebung mobiler Enterprise Apps.

4.1 Sicherheit

Der Sektor Sicherheit wird als wichtigste Anforderung an mobile Applikationen im Unternehmensbereich erkannt. In MOTWIN (2013) wird die Sicherheit als kritische Funktionalität für mobile Applikationen vom Großteil der Unternehmen eingestuft. Bei der Neuentwicklung von Mobilen Enterprise Services werden hohe Sicherheitsanforderungen neben der sich schnell ändernden mobilen Infrastruktur und der Integration von mobilen Lösungen in die vorhandene Unternehmensinfrastruktur als zentrale technologische Unsicherheit im Unternehmensbereich eingestuft. (Vgl. WEIß & SÖLLNER, 2012)
Die Sicherheit dient als wichtiges Unterscheidungskriterium im Vergleich bestehender mobiler Plattformen. Eine Enterprise-App gilt als sicher, wenn

- ein wirkungsvoller Schutz des Systems und der Apps,
- eine umfassende Kontrolle der App-Permissions,
- eine sichere Verschlüsselung der Gerätedaten,
- eine sichere Netzwerkkommunikation zum Firmennetz und angebundenen Services sowie
- eine sichere Zugriffsauthentifizierung

vorhanden sind. Diese Teilkriterien werden im Folgenden näher beschrieben.

4.1.1 Wahrung der System- und Applikationsintegrität

Mobile Geräte im Enterprise-Einsatz bieten Zugang zu Unternehmensnetzwerken und bewahren vertrauenswürdige und sensible Daten der Nutzer und Unternehmen auf. Als Zugangspunkt zu diesen Informationen außerhalb der schützenden Firewall des Unternehmens sind sie ein beliebtes Ziel für bösartige Software (Malware). Angreifer sehen mobile Geräte als unsicheren Angriffspunkt, mit Hilfe dessen sie die eigentlichen Sicherheitsvorkehrungen des Unternehmens umgehen können. Aus diesem Grund ist es von äußerster Wichtigkeit, dass moderne mobile Geräte und BSs integrierte Hardware sowie Softwaremerkmale zum Schutz gegen Malware aufweisen. Dieser Abschnitt erläutert grundlegende Mechanismen zum sicheren Booten, zum Schutz vor dem Ausführen unerwünschten Programmcodes und zum Schutz sowie der Isolation der ausgeführten Apps, welche moderne mobile Plattformen zur Verfügung stellen.

Secure Boot

Ein elementarer Sicherheitsmechanismus des BS ist der Secure-Boot. Zunächst werden alle für die Ausführung des BS erforderlichen Softwarekomponenten wie Bootloader, Kernel, Boot-Treiber, Startup-Dateien, usw. mit einer digitalen Signatur versehen. Während des Boot-Prozesses werden die einzelnen Komponenten anhand der Signatur auf ihre Integrität hin überprüft. Wird festgestellt, dass eine Komponente z.B. durch Malware verändert wurde, um bösartigen Code auszuführen, so wird das restliche System geschützt, indem der Start dieser Software verhindert wird.
Windows Phone 8.1 nutzt das Unified Extensible Firmware Interface (UEFI)[2] und dessen integrierte Secure-Boot-Komponente, um einen sicheren Boot-Prozess (Trusted Boot) zu gewährleisten. UEFI ersetzt das traditionelle BIOS, bietet dieselben Funktionen und fügt einige Sicherheitsfunktionen und andere Mechanismen hinzu. UEFI startet nicht nur den Bootloader, sondern prüft außerdem dessen Integrität. Dadurch wird ein Jailbreaking verhindert, welches einem Angreifer oder dem Nutzer das Verändern und Installieren von unautorisierten Apps ermöglichen würde. UEFI überprüft die digitale Signatur der Firmware inklusive optionaler ROM-Komponenten des Geräts. Ausschließlich der Hersteller der Hardware kann eine valide Signatur dieser Komponenten erstellen, sodass ein Schutz gegen Firmware- und Master-Boot-Record-Rootkits besteht. Dies ist der erste Schritt in der sogenannten „Chain Of Trust“. (Vgl. MICROSOFT, 2014h)
Der Trusted Boot-Prozess beginnt mit der Überprüfung des Bootloaders durch UEFI. Daraufhin verifiziert der Bootloader die Signatur des Windows Phone Kernels. Dieser prüft alle anderen Komponenten des Windows-Startprozesses.
Nachdem dieser Prozess beendet wurde, werden die Systemkomponenten und Apps geladen. Diese besitzen ebenfalls eine digitale Signatur und werden geprüft. Windows Phone Apps müssen eine Signatur des Stores oder eines autorisierten Entwicklers besitzen. Wird schadhafter Code innerhalb der Komponenten oder Apps entdeckt, werden diese nicht ausgeführt. (Vgl. MICROSOFT, 2014h)
Dieser Prozess wird ebenfalls von iOS[3], Android[4] sowie Blackberry[5] unterstützt. Durch die fortwährende Signierung und Prüfung aller Komponenten wird es Angreifern erschwert, Schadsoftware in das System einzuschleusen.

Address Space Layout Randomization (ASLR) und Data Execution Prevention (DEP)

ASLR ist eine Technologie zum Schutz des Speichers des BS, welche den Ort, an welchen ausführbarer Code geladen wird, randomisiert. Es entsteht ein Schutz gegen Buffer-Overflow-Attacken. Ist es für einen Angreifer möglich vorherzusagen, an welchen Stellen im Speicher Code abgelegt wird, so kann er diese Stellen durch schadhaften Code überschreiben und Buffer-Overflows ausnutzen. Durch ASLR ist es möglich, Zielspeicheradressen an nicht vorhersagbare Stellen im Speicher zu verlagern. (Vgl. MARGARET ROUSE, 2014)
Mit Hilfe der DEP werden Speicherbereiche als ausführbar oder nicht ausführbar deklariert. Ausführbarer Code muss in einem als ausführbar gekennzeichneten Bereich deklariert sein, um

[2] http://www.uefi.org/about
[3] APPLE (2015b)
[4] https://source.android.com/devices/tech/security/verifiedboot/index.html
[5] http://bizblog.blackberry.com/2015/02/blackberry-security-starts-at-endpoints/

gestartet werden zu können. Sollte ein Angreifer Schadsoftware in den Datenbereich (nicht ausführbar) des Speichers einschleusen und durch einen Buffer-Overflow versuchen diese aufzurufen, so wird dies durch DEP unterbunden. Der Mechanismus wird in kompatiblen CPUs, wie Advanced RISC Machines (ARM)-Prozessoren über das Execute Never Bit realisiert. (Vgl. MICROSOFT, 2014h)
Die genannten Mechanismen sind fester Bestandteil der mobilen BSs seit den Versionen Android 4.0, iOS 5, Windows Phone 8 und BlackBerry 10.

Sandboxing

Moderne mobile BSs stellen ein Konzept bereit, um den Zugriff von Apps untereinander und auf Systemressourcen zu beschränken. Sollte eine App trotz der bereits genannten Sicherheitsvorkehrungen bösartigen Code ausführen, so wird ihr Zugriff und damit der potenzielle Schaden auf ein Minimum beschränkt. Dieser Mechanismus wird Sandboxing genannt. Apps wird der Zugriff auf Dateien, welche andere Apps speichern und auf Einstellungen des Systems untersagt. Meist können sie lediglich auf ein ihnen zugewiesenes Verzeichnis zugreifen, um Daten abzulegen. Das Sandboxing sorgt für eine Isolation der Apps. Dies wird realisiert, indem jede App zunächst in einem Zustand ohne Berechtigungen ausgeführt wird. Um Zugriff auf Systemressourcen wie GPS, Internet, Kamera oder zu anderen Apps zu erlangen, muss die App Berechtigungen definieren, denen der Nutzer zustimmen muss. Werden diese vom Benutzer erteilt, so werden die Rechte der App um diese Berechtigungen erweitert. Details zu Berechtigungssystemen der einzelnen mobilen BSs sind Abschnitt 4.1.2 zu entnehmen.
Unter iOS befinden sich alle Drittanbieter-Apps in einer Sandbox. Jede App besitzt ein eigenes Home-Verzeichnis zur Ablage ihrer Daten, welches bei der Installation zugewiesen wird. Die Apps laufen in einem Modus ohne Privilegien namens „mobile“. Die gesamte BS-Partition wird in einem Nur-Lese-Modus gemounted. Der Zugriff von Drittanbieter-Apps auf Nutzerinformationen, Services wie iCloud, usw. wird über sogenannte Entitlements geregelt. Diese Entitlements werden benötigt, um privilegierte Operationen auszuführen, welche Root-Rechte erfordern. Sie sind digital signiert und können somit nicht geändert werden. (Vgl. APPLE, 2015b)
Windows Phone 8.1 setzt das Sandboxing-Konzept mit Hilfe sogenannter AppContainer um. Jede App und große Teile des BS laufen innerhalb von AppContainern. Für jeden AppContainer ist eine Sicherheitsrichtlinie deklariert, welche aus mehreren Capabilities besteht. Das sind Ressourcen wie GPS, Kamera, Mikrophone, Netzwerk oder Sensoren, auf welche die App zugreifen darf. Standardmäßig wird jedem AppContainer eine kleine Menge an Capabilities gewährt, wie der Zugriff auf den eigenen Speicherbereich namens Isolated Storage. Der Zugriff auf zusätzliche Capabilities wird im Applikationscode deklariert und kann zur Laufzeit nicht geändert werden. Dabei gilt das Prinzip „least privilege“. Das heißt, Apps erhalten ein Minimum an Berechtigungen, um ihre vorgesehene Aufgabe zu erfüllen. (Vgl. MICROSOFT, 2014h)
Das Sandboxing unter Blackberry weist keine entscheidenden Unterschiede auf. Zu bemerken ist die integrierte BYOD-Lösung, die einen geschäftlichen und einen persönlichen Bereich bietet. Hier werden Apps, welche in beiden Bereichen vorhanden sind, ebenfalls vollständig voneinander isoliert. Eine Vorstellung der BYOD-Lösung wird im Abschnitt 4.4.1 b gegeben.

Die Realisierung unter Android ist ähnlich zu den genannten Systemen. Jede App läuft in einer Sandbox und besitzt einen eigenen isolierten Speicherbereich. Applikationen desselben Entwicklers besitzen allerdings ebenfalls Zugriff auf diesen Speicherbereich. Über sogenannte Intents können Daten von einer Applikation zu einer anderen weitergeleitet werden. Außerdem kann eine App auf Services, welche andere Applikationen zur Verfügung stellen, zugreifen. ORTHACKER et al. (2012) zeigen, dass durch diese Mechanismen sensible Nutzerdaten ausgespäht werden können. Angenommen, es existieren zwei Apps. Die erste App dient dem Zugriff auf ein soziales Netzwerk und besitzt die Berechtigung auf das Internet zuzugreifen. Die zweite App ermittelt die GPS-Koordinaten des Nutzers. Sie besitzt das Recht zum Zugriff auf das GPS, jedoch nicht auf das Internet. Außerdem stellt sie die Ermittlung der Koordinaten als Android Service bereit. Nun ist es möglich, dass App eins den Service von App zwei nutzt, die aktuellen Koordinaten ausliest und diese dem sozialen Netzwerk zur Verfügung stellt, ohne dass der Nutzer eine entsprechende Berechtigung vergeben hat. Dies wird auch als Permission Spreading bezeichnet. (Vgl. ORTHACKER et al., 2012)

4.1.2 Kontrolle von Geräte-Administrierten Rechten

Geräte-Administrierte Rechte bzw. Permissions sind Berechtigungen für bestimmte Systemzugriffe, die das mobile BS den installierten Applikationen erteilen kann. Die Rechte beziehen sich auf Datenzugriffe auf bestimmte Bereiche, wie das Auslesen von Kontakten, gespeicherten E-Mails oder Daten anderer Apps. Des Weiteren werden Zugriffe auf Hardwarebausteine, wie Kamera, Beschleunigungsmesser, WLAN, Near Field Communication (NFC) oder GPS durch Rechte verwaltet. Außerdem wird die Ausführung von Background-Tasks, die Nutzung der in diesen Tasks entgegengenommenen Systemereignisse und der Empfang von Benachrichtigungen (Notifications) durch Rechte verwaltet. Die Plattformentwickler gehen unterschiedliche Wege, wenn es darum geht, wie der Entwickler diese Rechte anfordert und wie der Nutzung dieser Rechte zugestimmt wird. Tabelle 4 zeigt, welche Philosophie die einzelnen BS verfolgen und wie groß der Einfluss des Nutzers auf die Vergabe von Berechtigungen ist.

Berechtigungen unter Android

GOOGLE schreibt über sein Android System: „A central design point of the Android security architecture is that no application, by default, has permission to perform any operations that would adversely impact other applications, the operating system, or the user.“ (GOOGLE, 2015c) Jede Applikation wird in einer Sandbox ausgeführt. Um zu kommunizieren, müssen die Anwendungen dies über Berechtigungen explizit angeben. Der Entwickler fordert die Berechtigung statisch in der Datei „AndroidManifest.xml“ an. In Listing 1 wird einer Applikation gestattet, SMS-Nachrichten zu versenden.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.loginApp" >
    <uses-permission android:name="android.permission.SEND_SMS" />
    ...
</manifest>
```

Listing 1 Deklaration einer Berechtigung unter Android in der Manifest-Datei (vgl. GOOGLE, 2015c)

Der Nutzer wird zur Installationszeit gefragt, ob er den deklarierten Berechtigungen zusagt siehe Tabelle 4. Um die App nutzen zu können, muss er allen Permissions zustimmen und kann seine Zustimmung später nicht widerrufen. CHIA, YAMAMOTO & ASOKAN (2012) bezeichnen Android als „user-consent permission system“. Dem Nutzer wird demnach die Verantwortung übertragen, das Risiko bzw. Gefahrenpotenzial der App einzuschätzen. Als Nachteil dieser Art von Systemen wird angebracht, dass sich der Nutzer sehr schnell an die Anfrage zur Installationszeit gewöhnt und dieser unbekümmert zustimmt. Der Vorteil ist die vollständige Kontrolle des Nutzers über die Vergabe der Berechtigungen.

	Android	iOS[6]	Windows Phone[7]	Blackberry[8]
Zeitpunkt der Abfrage von Permissions	Bei Installation der App	Bei Installation (basic-Permissions wie Internetzugriff) und wenn benötigt (z.B. GPS, Kontakte, Kalender)	Wenn benötigt (nur bei sensiblen oder vertraulichen Infos, wie Anmeldedaten oder Position); sonst keine Abfrage, sondern Info auf Seite des App Stores	Bei Erstmaligem Start der App
Abwählen einzelner Permissions bei Abfrage	Nein	Ja	Nein	Ja
Angabe von Gründen, wieso Permission benötigt wird	Nein, allgemeine Beschreibung	Ja, kontextbezogene Beschreibung	Nein	Nein, allgemeine Beschreibung
Nachträgliches Abwählen/ Zustimmen einzelner Permissions	Nein, nur über Drittanbieter-Apps	Ja	Nein (vsl. WP 8.1 GDR 2: Abwählen einzelner Permissions in den Settings)	Ja (Settings > Security and Privacy > Application Permissions)

Tabelle 4 Vergleich der mobilen BSs beim Informieren des Nutzers über App-Permissions

Berechtigungen unter iOS

Den Gegensatz zu dieser Strategie verfolgen die „centralized permission systems“. Ein Beispiel für dieses System ist das Apple iOS. Hier wird die Zustimmung der Rechte von einer zentralen Autorität ausgeführt. (Vgl. CHIA, YAMAMOTO & ASOKAN, 2012) Im Falle von Apple ist dies der Apple App-Store. Da Apple die Installation von Apps ausschließlich über den App-Store erlaubt, gibt es einen zentralen Punkt, an dem die Zugriffskontrolle erfolgt. Apple führt einen aufwendigen Review-Prozess durch. Die vom Entwickler genutzten Systemfunktionen werden auf ihre ordnungsgemäße Verwendung hin untersucht. Gegebenenfalls wird die App abgelehnt. Damit ist der Entwickler in der Verantwortung, Systemzugriffe sorgfältig zu planen und ausschließlich geeignete Zugriffe auszuführen, damit sein Produkt veröffentlicht werden kann. Zusätzlich liegt die Zugriffskontrolle vollständig in den Händen des Herstellers, der den Apps ausschließlich die benötigten Zugriffsrechte erteilt. Unter iOS wird der Nutzer auf verschiedene

[6] Details: http://www.howtogeek.com/211623/how-to-manage-app-permissions-on-your-iPhone-or-ipad/
[7] Details: https://www.windowsphone.com/de-de/how-to/wp8/apps/how-can-i-tell-if-an-app-has-requirements
[8] Details: https://developer.blackberry.com/native/documentation/dev/tools/app_permissions.html

Arten nach Berechtigungen gefragt. Zur Installationszeit wird der Nutzer ausschließlich nach rudimentären Rechten, wie dem Zugriff auf das Internet befragt. Feingranulare Anfragen, wie das Erheben von Ortsinformationen des Nutzers, werden eingeholt, sobald diese benötigt werden. Der Nutzer kann einzelne Berechtigungen zu jedem Zeitpunkt entziehen oder erteilen. Jede iOS App besitzt ein Property List File (Info.plist), in welchem Konfigurationsinformationen festgehalten werden. Es handelt sich um ein Dictionary mit einer Menge von Key-Value-Paaren, die im XML-Format abgespeichert sind. Es werden Informationen wie z.B. von der App benötigte Device-Eigenschaften, Icons, Projektname und –version sowie Keys für die benötigten Zugriffsrechte der App festgehalten. Die Eigenschaften können vom Entwickler über den „Xcode property list editor“ geändert werden. Die folgenden Beschreibungen zu Keys stehen im Vergleich zu APPLE (2015a). Einige Keys werden vorausgesetzt und von iOS automatisch definiert. Sie beginnen mit CF. Beispiele sind CFBundleDevelopmentRegion, CFBundleDisplayName, CFBundleExecutable und CFBundleIconFiles. Ein Beispiel für Zugriffsrechte, die vom Entwickler gesetzt werden, sind die Location Services. Diese nutzen GPS, Bluetooth, Wi-Fi-Hotspots and Informationen über Mobilfunkzellen, um die aktuelle Position des Nutzers zu bestimmen. Unter iOS 8 existieren zwei Keys NSLocationWhenInUseUsageDescription und NSLocationAlwaysUsageDescription. Der zugehörige Value ist jeweils ein String, der den Grund für die Erteilung dieses Zugriffs enthält und dem Nutzer angezeigt wird, sobald die App den Service benötigt. iOS ist das Einzige der in Tabelle 4 untersuchten BSs, das dem Nutzer konkrete Hinweise liefert, warum eine Berechtigung zu einem bestimmten Zeitpunkt notwendig ist. Details, wann diese Rechte angezeigt werden, gibt APPLE (2015b). Der erstgenannte Key wird gesetzt, wenn ein Zugriff benötigt wird, solange sich die App im Vordergrund befindet. Der letztgenannte Key gibt die Berechtigung für im Hintergrund befindliche Apps. Der Entwickler erfragt die Berechtigung durch den Code siehe Listing 2 (Beispiel in Swift).

```
if CLLocationManager.authorizationStatus() == .NotDetermined {
    manager.requestWhenInUseAuthorization()
}
```

Listing 2 Erfragen der Berechtigung einer App für den Location Service unter iOS 8 (vgl. NAHAVANDIPOOR, 2014)

Der LocationManager gibt die aktuelle Position zurück. Er muss vorher auf eine bestehende Autorisierung befragt werden. Durch die „requestWhenInUseAuthorization“- bzw. „requestAlwaysAuthorization“-Methode wird, analog zu den Keys, die Berechtigung für den Location Service bei einer im Vordergrund bzw. im Hintergrund ausgeführten App abgefragt. (Vgl. APPLE, 2014a)

Berechtigungen unter Windows Phone und BlackBerry

Nach Tabelle 4 verfolgt Windows Phone eine im Vergleich zu Android ähnliche Strategie, wenn es um das Informieren des Nutzers über Berechtigungen geht. Ist der Nutzer mit einer Permission nicht einverstanden, so sollte er die App nicht installieren. Es ist nicht möglich, Rechte nachträglich zu entziehen. Außerdem werden einzelne Berechtigungen nicht für den Nutzer verständlich beschrieben. Dieser muss sich vor der Installation auf der App-Detailseite des Windows Phone Stores über die erforderlichen Berechtigungen informieren. Während der Installation entfällt ein Hinweis. Allerdings werden ähnlich iOS einige Permissions abgefragt, wenn sie benötigt werden. Dies beinhaltet sensible bzw. vertrauenswürdige Informationen.

Außerdem ist anzumerken, dass auch Mircosoft ähnlich Apple einen Review-Prozess aller Apps durchführt. Die zentrale Kontrolle über Berechtigungen liegt beim Plattformhersteller. Die Verantwortung über Systemzugriffe trägt demnach vorwiegend Mircrosoft, sodass sich der Nutzer auf die Richtlinien des Windows Phone Stores verlassen kann.
Blackberry wird unter dem Aspekt der Permission-Kontrolle neben iOS auf einer Ebene angesehen. Beim ersten Start einer App werden Berechtigungen abgefragt. Permissions sind in Hierarchien mit feingranularen Berechtigungen untergliedert. Hier können entweder die gesamte Hauptgruppe oder einzelne Rechte entzogen oder vergeben werden. Ein späteres Anpassen der Einstellungen ist ohne Probleme möglich.

Kontrolle durch Einschränkung der Installationsquelle von Apps

Ähnlich zu iOS, welches ausschließlich Apps aus dem iTunes-App-Store oder zertifizierte Enterprise Apps zulässt, ist Windows Phone auf Apps aus dem hauseigenen App-Store bzw. zugelassene Enterprise Apps, sogenannte Line-Of-Business (LOB) Apps, beschränkt. Blackberry beschränkt die Installation von Apps auf zwei Stores[9]. Es existieren die „Blackberry World“ für Business- und Produktivitäts-Apps und der „Blackberry World for Work“-Store für Enterprise-Apps. Die Kontrolle der Apps führt stets der Hersteller durch. Damit sinkt die Wahrscheinlichkeit von Schadsoftware. Diese kann in seriösen Stores schnell gemeldet und entfernt werden. Bei Android gilt eine Einschränkung auf Googles Play-Store nicht. Schadsoftware aus Drittanbieter-App-Stores kann ohne Probleme auf das Endgerät gelangen, solange keine Einschränkungen durch ein MDM vorgenommen werden. Durch den hohen Marktanteil von Android-Geräten von rund 80 %, gemessen am Endkundenabsatz von Smartphones im ersten Quartal 2015 (siehe GARTNER (2015)), ist Android ein besonders beliebtes Angriffsziel von Schadsoftware. Selbst im Play-Store wird trotz Sicherheitsmaßnahmen immer wieder vom erfolgreichen Einschleusen von Malware berichtet[10]. Hier fällt die nichtumgesetzte Kontrolle des Nutzers über einzelne Permissions besonders stark ins Gewicht.

4.1.3 Verschlüsselung von Gerätedaten

Die Verschlüsselung des Dateisystems und die Verfügbarkeit applikationsspezifischer Verschlüsselungssysteme sind Unterscheidungskriterien innerhalb der Kategorie Sicherheit. In (KREUZHUBER, TEUFL & ZEFFERER, 2014) werden verschiedene Sicherheitsfunktionen einzelner mobiler BSs analysiert. Der folgende Abschnitt ist in Anlehnung an diese Quelle entstanden.
„Unterschiede ergeben sich zunächst in der Speicherung der zur Verschlüsselung genutzten geheimen Schlüssel.“ (KREUZHUBER, TEUFL & ZEFFERER, 2014) Eine Möglichkeit ist die Verschlüsselung mit Hilfe eines Hardware-Elements. Dies hat den Vorteil, dass Dateien ausschließlich offline (direkt am Gerät) entschlüsselt werden können. Wird das Gerät allerdings durch einen Jailbreak gerootet, so kann in unverschlüsselter Form auf die Daten zugegriffen werden, „da kein von der Benutzerin bzw. vom Benutzer gewählter externer Eingabewert in die

[9] Zusätzlich existiert der Amazon App-Store für Android-Apps. Hier führt Amazon entsprechende Review Prozesse durch.
[10] https://blog.avast.com/de/2015/05/04/klick-app-schlich-sich-bei-google-play-ein-und-imitierte-populare-dubsmash-app/

Verschlüsselung miteingeht.“ (KREUZHUBER, TEUFL & ZEFFERER, 2014) Sowohl Bildschirmsperre, als auch Verschlüsselung, können in diesem Fall umgangen werden.
Des Weiteren besteht die Möglichkeit, einen vom Passcode des Benutzers abgeleiteten Schlüssel für die Dateisystemverschlüsselung zu verwenden. Nachteil ist hierbei die Durchführbarkeit von online Angriffen (nicht direkt am Gerät). Zum Beispiel „können über Rechen-Cluster parallelisiert Brute-Force-Attacken“ (KREUZHUBER, TEUFL & ZEFFERER, 2014) durchgeführt werden. Die benötigte Zeit für einen erfolgreichen Angriff ist in diesen Fall von der Komplexität und Länge des Passcodes abhängig.
Die verschlüsselte Ablage der Daten hat einen Einfluss darauf, auf welche Art Daten per Fernzugriff gelöscht werden können. Verschlüsselte Daten können durch das einfache Löschen des kryptographischen Schlüssels nicht mehr entschlüsselt werden. Unverschlüsselte Daten müssen aufwendig gelöscht werden.
Zusätzlich existieren Methoden, die Daten einzelner Applikationen spezifisch zu verschlüsseln. Letztendlich ist der Entwickler dafür verantwortlich, neben der Dateisystemverschlüsselung, dieses zusätzliche Verschlüsselungssystem anzuwenden. Bei einem Jailbreak sind die Applikationsdaten dann dennoch geschützt. Nicht jede mobile Plattform bietet diese Möglichkeit der Verschlüsselung. (Vgl. KREUZHUBER, TEUFL & ZEFFERER, 2014)

Verschlüsselung unter iOS

Als Vertreter der Verschlüsselung mit Hilfe eines Hardware Elements, werden im Folgenden die Verschlüsselungsmechanismen des iOS 8 BS erklärt. Als Grundlage dient APPLE (2015b).
Apple stattet seine Geräte seit der Apple A7 Prozessorserie mit einem Coprozessor zur Unterstützung von komplexen Verschlüsselungs- und Authentifizierungsmechanismen aus. Dieser Coprozessor wird als „Secure Enclave“ bezeichnet. Beispielsweise wird die Authentifizierung des Nutzers über Apples Fingerabdrucksensor (Touch ID), eingeführt mit dem iPhone 5S, über diesen Prozessor gesteuert. Diese Technologie wird in Abschnitt 4.1.5 näher beschrieben.
Weiterhin besitzt jedes Gerät eine spezifische „AES 256 crypto engine“. Diese wurde in den DMA Pfad zwischen Flash-Speicher und Hauptspeicher integriert und sorgt für die schnelle Ausführung von kryptographischen Operationen.
Jedes Gerät verfügt über eine eindeutige Unique ID (UID) und eine Gerätegruppen ID (GID), welche fest verdrahtet im Hauptprozessor hinterlegt werden. Die UID ist geräteweit eindeutig. Die GID ist eindeutig für eine Klasse von Geräten mit gleichem Hauptprozessor. Diese Identifier können weder von der Firmware noch von der Software direkt ausgelesen werden. Selbst Apple sind diese nicht bekannt. Daten können somit an ein spezifisches Gerät gebunden werden. Eine Übertragung der Hardware (z.B. Flash-Speicher) und das Auslesen auf einem anderen Gerät sind nicht möglich.
Auf Grundlage der Hardwareverschlüsselung und einer Hierarchie der verschiedenen Hardware- und Softwareschlüssel ist es nun möglich, die Daten des Flash-Speichers effektiv zu schützen (siehe Abbildung 3). Apple spricht hier von „Storage Protection“. Jedes neue File erhält einen 256-bit Key. Über diesen wird das File durch die Advanced Encryption Standard (AES) Engine verschlüsselt und schließlich auf den Flash-Speicher geschrieben. Der „per-file“ Key wird mit einem sogenannten „class key“ gewrappt. Dieser Klassenschlüssel gibt die Umstände an, unter welchen auf das File zugegriffen werden darf (Schutzklasse). Er wird wiederum vom Entwickler festgelegt. Mögliche Schlüsselwerte sind NSFileProtectionNone,

NSFileProtectionComplete, NSFileProtectionCompleteUnlessOpen, NSFileProtectionCompleteUntilFirstUser. Ist der Zugriff auf ein File von der Entsperrung des Geräts durch den Nutzer abhängig, so wird der „class key“ neben der UID durch den Passcode (vergleichbar mit der PIN des Geräts) des Benutzers geschützt. Der durch das Wrapping entstandene Schlüssel wird in den Metadaten des Files gespeichert. Diese werden wiederum über einen zufällig generierten „File System Key“ verschlüsselt.

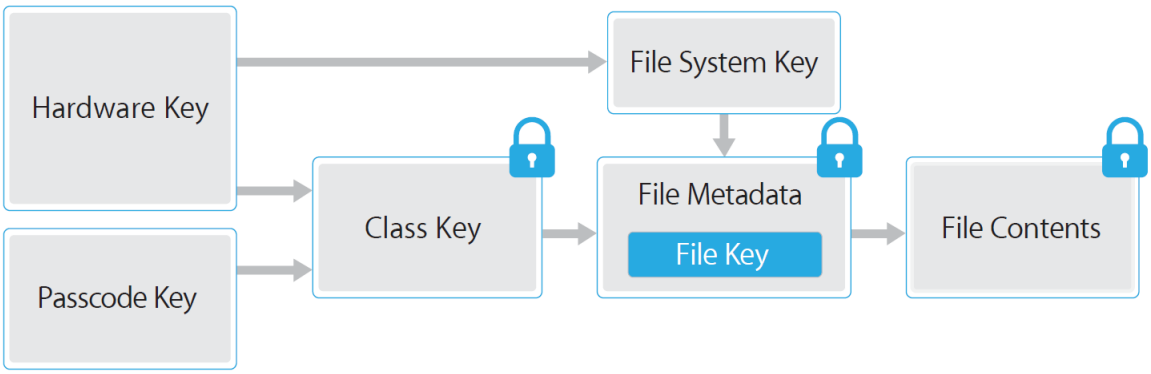

Abbildung 3 iOS 8 Architektur von Sicherheitsschlüsseln innerhalb der File Data Protection (APPLE, 2015b)

Er wird in einem sicher löschbaren Speicherbereich (Effaceable Storage) gehalten. Dieser Schlüssel dient nicht der Gewährleistung der Sicherheit, sondern einem schnellen Löschen der Daten. Wird der Schlüssel beispielsweise durch einen Fernzugriff durch einen Administrator innerhalb eines MDM gelöscht, so sind die Daten nicht mehr zugreifbar, also gelöscht.
Der Passcode oder Benutzercode zur Entsperrung des Geräts dient nicht nur als Schutz vor unbefugtem Zugriff. Er dient ebenfalls zur Verschlüsselung sensibler Daten, wie Abbildung 3 zeigt. Wird der Passcode vom Benutzer verwendet, so ist die „Storage Protection“ automatisch aktiviert. Anstelle des Passcodes sind andere Authentifizierungsmaßnahmen denkbar. Apple stellt außerdem iTouch zur Verfügung. (Vgl. APPLE, 2015b)
Aus Platzgründen wird hier nicht weiter auf Mechanismen wie die „Key Chain“ als sicherer Key-Value-Store für App-Daten oder „Effaceable Storage“ für das sichere Löschen von Daten in Flash-Speichern eingegangen. Sie können APPLE (2015b) entnommen werden. Es sei an dieser Stelle ausschließlich darauf hingewiesen, dass iOS die Verschlüsselung applikationsspezifischer Daten unterstützt.
Da selbst Apple die Hardware Keys innerhalb des Geräts nicht kennt, kann keine Behörde eine Einsicht in die Daten erwirken. Daten, die mit Apples Cloud Storage iCloud synchronisiert werden, sind allerdings nach wie vor im Ernstfall einsehbar (siehe Abschnitt 4.3).

Verschlüsselung unter Android

Android besitzt seit Version 3.0 die Möglichkeit der Dateisystemverschlüsselung. Dabei wird der Dateisystemschlüssel durch eine kryptographische Operation mit dem Passcode des Nutzers generiert (vgl. MERZ, 2014). Unter Android-Versionen kleiner 4.4 ist die Dateisystemverschlüsselung standardmäßig deaktiviert. Sie kann vom Nutzer oder durch MDM-Regeln erzwungen werden. Google kündigte für Android Version 5 eine standardmäßige Verschlüsselung an. Dies wurde jedoch bisher nicht umgesetzt. Als Grund nennt Google Performanz-Probleme bei der Verschlüsselung durch zu leistungsschwache Hardware in den Geräten zahlreicher Hersteller. (Vgl. CUNNINGHAM, 2015) Da Android kein Hardware-Element

für die Verschlüsselung verwendet, kann die Berechnung des Dateisystemschlüssels aus dem Passcode außerhalb des Gerätes erfolgen. Eine genaue Analyse der Schlüsselableitung gibt TEUFL, FITZEK, HEIN, MARSALEK, OPRISNIK & ZEFFERER (2014). Hier wird außerdem ein Überblick über Dauer und Kosten von Brute-Force-Angriffen bei der Anmietung von Amazon EC2 Cloud Instanzen gegeben. Seit Version 4.0 existiert die Möglichkeit der applikationsspezifischen Verschlüsselung über eine KeyChain. Hier wird der Passcode zur Ableitung des Verschlüsselungsschlüssels herangezogen. Seit Version 4.3 kann ein Hardware-Element zur sicheren Aufbewahrung des Schlüssels genutzt werden.[11]

Verschlüsselung unter Windows Phone 8.1

Unter Windows Phone 8.1 besteht die Möglichkeit einer Dateisystemverschlüsselung und einer applikationsspezifischen Verschlüsselung. Die Dateisystemverschlüsselung basiert auf der BitLocker-Technologie. Windows Phone nutzt einen Tamper-Resistant Security Processor (TPM) als Hardware-Element, um den Dateisystemschlüssel zu schützen. Der Passcode des Benutzers wird nicht für die Ableitung des Schlüssels genutzt. Zu beachten ist, dass eine Verschlüsselung ausschließlich über ein Exchange ActiveSync (EAS) oder MDM erfolgen kann und ausschließlich für Unternehmenskunden zugänglich ist. (Vgl. MICROSOFT, 2014h)
Die applikationsspezifische Verschlüsselung erfolgt über die Data Protection API. Über diese können einzelne Dateien im Speicherbereich verschlüsselt werden. Das Hardware-Element wird dabei nicht in die Schlüsselableitung einbezogen.

Verschlüsselung unter BlackBerry 10

BlackBerry bietet als einzige Plattform die Trennung von persönlichen und geschäftlichen Daten auf Dateisystemebene. Durch zwei unterschiedliche Verschlüsselungssysteme kann jeder dieser Bereiche getrennt voneinander geschützt werden. Der sogenannte Work Space für geschäftliche Daten wird standardmäßig und der Personal Space manuell oder über MDM-Regeln verschlüsselt. Der kryptographische Schlüssel wird unter beiden Bereichen nicht mit Unterstützung des Passcodes generiert. (Vgl. KREUZHUBER, TEUFL & ZEFFERER, 2014)

Federal Information Processing Standard (FIPS) 140-2

„FIPS (Federal Information Processing Standard) 140-2 ist ein Standard der US-Regierung und beschreibt die Verschlüsselung und die zugehörigen Sicherheitsanforderungen, die IT-Produkte zur vertraulichen, aber nicht klassifizierten Nutzung erfüllen sollten." (SEAGATE, 2010)
Der Standard gilt als Grundvoraussetzung für die Zulassung von Geräten in Regierungseinrichtungen in den USA und in Kanada.
Er beschäftigt sich speziell mit dem sicheren Design und der sicheren Realisierung kryptographischer Module und Algorithmen. Beispiele sind die Spezifikation, Ports und Interfaces kryptographischer Module, Services, Authentifizierung, physikalische Sicherheit, kryptographisches Schlüsselmanagement, Umgang mit elektromagnetischen Interferenzen, Selbsttests und die Minderung anderer Sicherheitsrisiken.
Es werden vier Sicherheitslevels definiert. Level 1 befasst sich mit den Basisanforderungen. Alle Komponenten müssen einen hohen Fertigungsgrad besitzen und dürfen keine

[11] http://developer.android.com/about/versions/jelly-bean.html

Sicherheitslücken aufweisen. Auskunft über die weiteren Stufen gibt die Federal Information Processing Standards Publication. (Vgl. U.S.DEPARTMENT OF COMMERCE/NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, 2002)
Auf Grundlage dieser Prüfung eignet sich der Standard zur Bewertung der Sicherheitsstandards kryptographischer Module mobiler BSs und Geräte und gibt Auskunft über die Eignung für Großunternehmen und Regierungseinrichtungen.

4.1.4 Sicherung der Verbindung zu Firmennetzwerken

Seit mobile Geräte im Geschäftsumfeld zum Standard gehören und sich der Trend zu BYOD-Lösungen durchsetzt, müssen Unternehmensdaten per Fernzugriff erreichbar sein. Natürlich sollte dabei eine sichere Verbindung bis zum Firmennetzwerk der Standard sein. Dies wird heutzutage über VPNs realisiert.

Einsatz von VPNs in Unternehmen

VPN-Verbindungen werden in unterschiedlichen Szenarien in Enterprises eingesetzt. Ein Einsatzgebiet besteht in der Verbindung von Außenstellen oder Büros über eine meist öffentliche Infrastruktur mit der Zentrale. Dieses Verfahren wird als Intranet-VPN oder Side-to-Side Verbindung bezeichnet. Eine Anbindung von Mitarbeitern, Zulieferern, Partnern, Kunden sowie anderen am Intranet-Zugang der Firma interessierten Gruppen, ist über ein Extranet-VPN möglich, auch End-to-Side genannt. Möchte ein Mitarbeiter eine Verbindung zum Firmennetz aufbauen, so benötigt er eine Internetverbindung sowie eine VPN-Software. Auf der Seite der Firma wird ein VPN-Gateway als Zugangspunkt eingesetzt. Der Mitarbeiter stellt mit Hilfe der VPN-Software eine Verbindung zum VPN-Gateway her. Nach einer Authentifizierung hat er Zugriff auf das Firmennetzwerk, als säße er an seinem Arbeitsplatzrechner im Büro. (Vgl. BÖHMER, 2005)
Im Kontext dieser Arbeit sind Entwicklungen interessant, die speziell mobilen Endgeräten den Zugang erleichtern wie das „Mobile VPN“. Bei klassischen End-to-Side VPNs ist der Nutzer fest an einen bestimmten Ort und Zugangspunkt gebunden. Beim „Mobile VPN“ wird davon ausgegangen, dass sich der Nutzer in Bewegung befindet. Dabei ändern sich die Netzwerkabdeckung, der Zugangspunkt, der Netzwerkstandard und die IP-Adresse. Eine klassische VPN-Verbindung ist nicht für Mobilität ausgelegt und bricht bei einer Änderung dieser Parameter ab. Der Nutzer muss sich ständig neu anmelden. Grund ist vor allem die sich ändernde physikalische IP-Adresse. Aus diesem Grund wird in einem Mobile VPN ein zusätzlicher VPN-Server eingesetzt. Hierbei wird jeder VPN-Tunnel an eine logische IP-Adresse gebunden. Folgendes Beispiel (vgl. PHIFER, 2006) ist damit realisierbar:

- Roaming von einem drahtlosen Access Point zu einem anderen Wi-Fi Hotspot
- Umschalten von Wi-Fi in ein Mobilfunknetz, z.B. 3G
- Umschalten von einem 3G Netz in ein langsameres 2G-Netz
- Umschalten von 3G auf das Firmeninterne Ethernet LAN

Probleme wie Subnetz-Roaming, Ändern des Netzwerkadapters, Stromsparmodi des mobilen Gerätes oder fehlende Netzabdeckung, werden durch den VPN-Server abgefangen, indem er den Netzwerkzustand des Endgeräts speichert und ihn bei erneuter Verbindung wiederherstellt. (Vgl. PHIFER, 2006)

VPN-Standards der mobilen Betriebssysteme

Interessant für den Entwickler sind die Verwendeten und unterstützten Standards der mobilen BSs. Diese müssen drei Dienste gewährleisten:

- Datenverschlüsselung
- Validierung der Datenintegrität
- Authentifizierung der Quelle

Heute gelten IP Security Architecture (IPsec)-basierte VPNs und SSL VPNs als Standard. Beide nutzen verschiedene Protokolle, um die oben genannten Services zu erbringen.
IPsec garantiert einen sicheren Tunnel durch unsichere Netzwerke zwischen dem Remote-Gerät und dem Firmennetz. Es arbeitet auf der Netzwerkschicht (Open Systems Interconnection (OSI) Layer 3). Aus diesem Grund wird es für Anwendungen wie Voice over IP, Multimedia und andere auf der Netzwerkschicht angesiedelte Anwendungen eingesetzt. (Vgl. BEDELL, 2010b)
IPsec wird häufig zusammen mit dem Layer 2 Tunnel Protocol (L2TP) verwendet. Das L2TP dient dabei der Kommunikation auf der Sicherungsschicht. IPsec wird hier für die Verschlüsselung des Datenverkehrs eingesetzt. (Vgl. BÖHMER, 2005)
SSL VPN ist zwischen der HTTP und TCP-Schicht angesiedelt (OSI Layer 4-7). Fernzugriffe erfolgen hierbei über den Web-Browser oder einen Java- bzw. ActiveX-Agent, somit ist kein Client notwendig. Der Zugriff auf Web-basierte Anwendungen ist ohne Probleme möglich. (Vgl. BEDELL, 2010a)
Es existiert eine Vielzahl von VPN-Typen, die wiederum verschiedenste Protokolle einsetzen, um die oben genannten Dienste sicherzustellen. Aus Platzgründen wird hier ausschließlich eine beispielhafte Auswahl vorgestellt. Zu erwähnen wäre zusätzlich das Point-to-Point-Tunneling Protocol (PPTP).
Die mobilen BSs stellen meist einen internen VPN-Client zur Verfügung, der die gängigsten Protokolle und Verschlüsselungen unterstützt. Probleme entstehen, wenn die unterstützten Protokolle nicht ausreichen. Bei der Verbindung mit speziellen Geräten wie SonicWALL oder Cisco sind meist eigene Clients zu installieren. Unter Android sind für diese Clients oftmals Root-Rechte notwendig. Die korrekte Funktionsweise kann sich dennoch von Gerät zu Gerät unterscheiden. Eine Verbindung zu einem sicheren Cisco-Router ist bei der Voraussetzung eines Endgeräts mit Root-Rechten wenig hilfreich, da der Nutzer daraufhin mit Administratorrechten am Gerät arbeitet. Schadsoftware hat somit keine Probleme, die Schutzmechanismen des BS zu umgehen. (Vgl. JOOS, 2015)
Die Auswahl des richtigen mobilen BS für die jeweilige vorhandene Infrastruktur bzw. VPN-Typen und geplanten mobilen Anwendungen kann für den Entwickler demnach entscheidend sein. iOS beispielweise bietet in seinem internen VPN-Client standardmäßig Unterstützung für spezielle Geräte an (siehe APPLE (2015b)).
Neben den unterstützten VPN-Typen unterscheiden sich die mobilen BSs in der Unterstützung zusätzlicher Hilfsfunktionen für VPN-Nutzer. VPN On Demand gewährleistet bei einer zertifikatbasierten-Authentifizierung die automatische Herstellung einer Verbindung, wenn auf vordefinierte Domains zugegriffen wird. Eine nahtlose VPN-Konnektivität wird sichergestellt. (Vgl. APPLE, 2012b)
Besonders hilfreich ist ein sogenanntes Per-App-VPN. Hier kann auf feingranulare Weise eine Verbindung für jede App bzw. Domain festgelegt werden. Die Konfiguration kann über ein

MDM erfolgen. Dabei wird sichergestellt, dass ausschließlich sichere Daten der Unternehmens-App in das Netzwerk fließen und der Zugriff außerhalb der App nicht möglich ist.
iOS 8 stellt außerdem die Möglichkeit des Always-On VPN vor. Hier wird einer Organisation vollkommene Kontrolle über den Datenverkehr eines Geräts gegeben, indem der gesamte IP-Verkehr über einen VPN-Tunnel zur Organisation erfolgt. Es besteht die Möglichkeit der Aufzeichnung und des Filterns des Datenverkehrs zu und von dem mobilen Gerät. Damit wird ein sicherer Datenverkehr bereits vom Unternehmen aus sichergestellt. (Vgl. APPLE, 2015b) Diese Funktion steht außerdem auf Android-Geräten mindestens seit Version 4.2 und unter Windows Phone 8.1 zur Verfügung.

4.1.5 Verfahren für eine robuste Zugriffsauthentifizierung

Eine steigende Anzahl von Menschen nutzt das Smartphone oder Tablet, um Rechnungen zu bezahlen, einzukaufen, persönliche Informationen aufzubewahren. Über NFC werden bargeldlose Einkäufe möglich. Außerdem kann sich autorisiertes Personal an Türen Zugang verschaffen. Hat ein potenzieller Angreifer einmal Zugang zu einem Gerät, so kann er bedenklichen Schaden anrichten. Das mobile Endgerät als Geldbörse, Schlüssel und für andere sicherheitskritische Anwendungen setzt besondere Maßnahmen für eine sichere Authentifizierung voraus. Bekannt sind das klassische Passwort (Passcode), Gestenerkennung durch die Verbindung von Punkten auf einem Raster und Gestenerkennung innerhalb von Bildern, wie Picture Password unter Windows 8 oder Blackberry 10. Gemein ist diesen Mechanismen, dass ein dritter das Passwort oder die Geste durch ausspähen (Shoulder Surfing) reproduzieren kann. Des Weiteren sehen es Nutzer als benutzerunfreundlich an, zum Entsperren des Geräts jedes Mal das Passwort eingeben zu müssen. Neben den genannten Methoden existieren biometrische Verfahren, die den Fingerabdruck oder das Gesicht als Anhaltspunkt verwenden.

Beispiele für Biometrische Verfahren

Apple setzt seit einiger Zeit auf einen biometrischen Fingerabdrucksensor. Die Technologie namens iTouch wird neben dem Passcode, also dem klassischen Passwort, zur Authentifizierung des Nutzers eingesetzt. Dieselbe Technik existiert ebenfalls im Galaxy S5 bzw. S6 unter Android. Mit Hilfe von iTouch können Einkäufe im iTunes Store, App Store, iBooks Store und Apple Pay getätigt werden, ohne die Apple ID eingeben zu müssen. Interessant für den App-Entwickler ist die Möglichkeit, iTouch für Drittanbieter-Apps zur Authentifizierung nutzen zu können. Hier werden entsprechende APIs zur Verfügung gestellt. Aus Sicherheitsgründen kann die App niemals direkt auf die mit dem Fingerabdruck in Verbindung stehenden Daten zugreifen. Ausschließlich die Abfrage auf eine erfolgreiche Authentifizierung ist möglich. (Vgl. APPLE, 2015b)
Nachteil dieser Technik und die gleichzeitige Nutzung in installierten sicherheitskritischen Apps ist, dass ein Angreifer, sollte er den Schutz umgehen, Zugriff auf alle Informationen und Apps besitzt. Eine vollkommene Sicherheit bietet das Verfahren nicht. Es wurde bereits gezeigt, dass durch eine Kopie des Fingerabdrucks sowohl der Sensor im iPhone als auch der des Galaxy S5 ohne Probleme umgangen werden können. Es bestehen unterschiedliche Ansichten, wie der dafür betriebene Aufwand zu bewerten ist. An dieser Stelle sei ausschließlich bemerkt, dass das Verfahren Schwächen aufweist.

Die Gesichtserkennung als weiteres biometrisches Verfahren unterliegt bisher einigen Schwachstellen. Das klassische Verfahren der 2D Gesichtserkennung, welches beispielsweise in Android 4.0 realisiert wurde, konnte durch eine Fotographie des Gesichts leicht getäuscht werden (Photo Attack). Eine 3D Gesichtserkennung wurde von einigen Toshiba Laptops umgesetzt. Diese benötigt allerdings ca. 30 Sekunden pro Authentifizierung, da das Gesicht aus vier verschiedenen Perspektiven aufgenommen werden muss. Dies ist für die meisten Nutzer weniger praktikabel als die Eingabe eines Passwortes. CHEN, PANDE & MOHAPATRA (2011) stellen einen Ansatz der 2D Gesichtserkennung vor, der den existierenden 3D und 2D Verfahren deutlich überlegen ist. Mit Hilfe von Bewegungs- und Lichtsensoren des Smartphones können die klassischen Angriffspunkte erkannt werden. Die Schnelligkeit ist mit der klassischen Passworteingabe vergleichbar. „It has the potential to change the current state of smartphone authentications.“ (CHEN, PANDE & MOHAPATRA, 2011)

Einsatz von Freiformgesten

Ein Verfahren, welches sich in der Entwicklungsphase befindet, ist die Nutzung von Freiformgesten für die Zugriffsauthentifizierung. SHERMAN et al. (2011) untersuchen, ob eine praktische Nutzung möglich ist und wie sicher sich diese gegenüber Attacken wie Shoulder Surfing erweist. Möglich sind Ein-Finger- und Multi-Touch-Gesten. Freiformgesten haben den Nachteil, dass sie schwieriger einzuprägen sind, was allerdings ebenfalls das Nachahmen durch Dritte erschwert. Es werden Strategien für sichere und einprägsame Gesten aufgezeigt. Schließlich wird das Verfahren als robuste Methode für die mobile Authentifizierung identifiziert.

Entwickler sicherheitskritischer Apps sind auf eine sichere Zugriffsauthentifizierung angewiesen. Damit steigt das Vertrauen des Nutzers in ihre Software, was zu einer schnelleren Verbreitung von mobilen Bezahlsystemen, mobilem Banking und mobilen Steuerungssystemen und ähnlichen sicherheitskritischen mobilen Anwendungen führt.

4.1.6 Sicherung der Netzwerkkommunikation

Die Zeiten, in denen Apps alle benötigten Daten offline halten, sind längst gezählt. Meist bestehen mehrere Verbindungen zu externen Services in der Cloud oder zum Unternehmensnetzwerk. Daten werden über eine Client-Server-Architektur ausgetauscht. Diese kabellose oder kabelgebundene Übertragung ist stets anfällig, da das übertragende Medium leicht von dritten abgehört werden kann. Passwörter, Banking-Daten, Firmengeheimnisse usw. dürfen nicht ohne eine geeignete Verschlüsselung versendet werden. Es ist die Pflicht des Entwicklers dafür zu sorgen, dass sensible Daten stets über sichere Netzwerkverbindungen übertragen werden.
Mobile Plattformen bieten standardmäßig die verschlüsselte Übertragung mittels SSL-Sockets und Hyper Text Transfer Protocol Secure (HTTPS)-Requests an. Eine sichere Verbindung wird durch die verwendete Verschlüsselung und die Art der Authentifizierung des Servers charakterisiert. Folgende Aspekte sollten beachtet werden (siehe HP (2013)):

- zwingende Nutzung von SSL/TLS (OSI Layer 5) auf allen Transportkanälen, die sensitive Daten übertragen
- Wahl starker Verschlüsselungsalgorithmen mit geeigneter Schlüssellänge (z.B. AES)
- Nutzen von signierten Zertifikaten von zugelassenen Zertifizierungsstellen
- Unterbinden der Annahme von selbst-signierten Zertifikaten
- Unbedingte vollständige Durchführung der SSL-Verifikation auf Client-Seite
- Zulassen einer sicheren Verbindung nur, wenn die Identität des Servers mit einem vertrauenswürdigen Zertifikat sichergestellt wurde
- Informieren des Nutzers über invalide Zertifikate

Es wird empfohlen, auf Server-Seite eine möglichst hohe Verschlüsselung (z.B. AES mit mind. 128 bit Schlüssellänge) und die aktuellste Version TLSv1.2 zu wählen. In TLSv1, welche SSLv3 entspricht, wurde 2009 eine Sicherheitslücke entdeckt, sodass Server auf das Schließen dieser Lücke überprüft werden sollten. Zertifikate müssen nicht notwendigerweise von öffentlichen Zertifizierungsstellen ausgestellt sein, es sei denn, das Zielsystem ist unbekannt. Meist ist einer Applikation jedoch genau bekannt, zu welchem Back-End sie eine Verbindung aufbaut. Hier genügt eine „private“ Public-Key-Infrastruktur. Diese Zertifikate werden beispielsweise von Regierungen, Unternehmen oder Forschungsinstituten für deren Privatgebrauch verwendet. (Vgl. NOWSECURE, 2014)
Für höhere Sicherheitsanforderungen kann ein Certificate Pinning eingesetzt werden. Hierbei wird das vom Client erhaltene Zertifikat des Servers gegen vom Client aufbewahrte vertrauenswürdige Validierungsdaten geprüft. Diese Daten sind beispielweise eine Kopie des Serverzertifikats.
Als weitere Maßnahme gegen Angreifer kann SSL/TLS im Wechsel mit asymmetrischen bzw. symmetrischen Schlüsselverfahren kombiniert werden. (Vgl. NOWSECURE, 2014)
Um unter Android das SSL/TLS-Protokoll verwenden zu können, wird das „Java.net.ssl“ Package zur Verfügung gestellt. Über die API können beispielweise die SSL/TLS Version und der Verschlüsselungsmechanismus ausgewählt werden. Der X509TrustManager erlaubt die Anpassung von als vertrauenswürdig eingestuften Zertifikaten. Das „X509KeyManager“–Interface und die „X509ExtendedKeyManager“-Klasse erlauben die Spezifikation von Zertifikaten, welche zum jeweiligen Kommunikationspartner geschickt werden sollen.[12] Seit TLSv1.2 sind sowohl Client als auch Server in der Lage zu spezifizieren, welche Hash bzw. Signaturalgorithmen sie akzeptieren. Die Auswahl der zu sendenden Authentifizierungsinformation an die Gegenstelle muss auf einem X509-Zertifikat sowie den vom Kommunikationspartner akzeptierten Hash- bzw. Signaturalgorithmen beruhen.[13] Unter iOS stellt die „Certificate, Key and Trust Service API“ grundlegende Dienste zur Verschlüsselung, Signierung und Verifikation von Datenblöcken bereit. Dafür werden die Blöcke über verschiedene Sicherheitsschlüssel-Funktionen (SecKey*), wie „SecKeyEncrypt“, „SecKeyDecrypt“, „SecKeyRawSign“ und „SecKeyRawVerify“, der jeweiligen Operation unterworfen. (Vgl. APPLE, 2014c)

[12] http://developer.android.com/reference/javax/net/ssl/package-summary.html
[13] http://129.33.205.81/support/knowledgecenter/SSYKE2_7.0.0/com.ibm.java.security.component.70.doc/security-component/jsse2Docs/x509extkeymanager.html

4.1.7 Schlussfolgerung

Nach einer detaillierten Untersuchung des Kriteriums Sicherheit, lässt sich folgendes schlussfolgern:
Eine App gilt als sicher, wenn die zugrundeliegende Plattform wirkungsvolle Mechanismen für den Schutz des Systems und der Apps bereitstellt. Dazu zählt ein Secure-Boot-Prozess mit vollständiger Chain-Of-Trust von den beteiligten Hardware- bis zu den Softwaremodulen. Des Weiteren wird die Unterstützung von ASLR und DEP als Speicherschutz sowie ein Sandboxing-Mechanismus zur Isolation von Apps vorausgesetzt. Die Kommunikation von Apps muss Transparent für den Nutzer erfolgen.
Eine App sollte ausschließlich die für ihre Funktionalität benötigten Systemzugriffe durchführen bzw. ausschließlich Benutzerrechte einfordern, die in angemessenem Verhältnis zu ihrer Funktionalität stehen. Der Einsatz der unterschiedlichen Permission-Systeme hat dabei jeweils Vor- als auch Nachteile. Nach Möglichkeit sollte sowohl der Plattformhersteller einen umfangreichen Review-Prozess für Apps durchführen, als auch der Nutzer eine maximale Kontrolle über App-Permissions besitzen. Dies ist im Moment bei iOS und Blackberry der Fall. Windows Phone fragt zumindest bei einem Zugriff auf vertrauenswürdige Informationen den Nutzer, was einen Schritt in die richtige Richtung darstellt.
Das mobile BS sollte solide Verschlüsselungsmechanismen - sowohl software- als auch hardwareseitig besitzen. Hardwareseitige Kryptomodule, die für die Aufbewahrung von Verschlüsselungsschlüsseln genutzt werden, sorgen für ein deutlich höheres Sicherheitslevel. Bei einem Austausch der Hardware ist es nicht möglich, die Software zu entschlüsseln. Softwaremechanismen sind leichter zu kompromittieren.
Die sichere Verbindung zu Firmennetzwerken bieten IPsec-basierte VPNs bzw. SSL VPNs je nach Anwendungsszenario. Dafür sind bei modernen mobilen BSs meist die Standard VPN-Clients ausreichend. Erweiterungen wie Per-App VPN oder Always-On VPN sorgen zusätzlich für einen sicheren Datenverkehr und Zugriff auf das Firmennetz.
Eine sichere Zugriffsauthentifizierung sollte Angriffspunkte wie Shoulder Surfing oder ähnliches erschweren. Biometrische Verfahren sind vielversprechend. Allerdings sind auch diese Verfahren mit akzeptablem Aufwand auszuhebeln. Eine Kombination aus mehreren Verfahren wie z.B. iTouch plus die Eingabe des Passcodes bietet derzeit den größten Schutz.
Sichere Netzwerkverbindungen erfordern die Nutzung von SSL/TLS (TLSv1.2 empfohlen) auf allen Transportkanälen, einen starken Verschlüsselungsalgorithmus mit hoher Schlüssellänge (z.B. AES mind. 128 bit) und das ausschließliche Akzeptieren von vertrauenswürdigen Zertifikaten bei der Authentifizierung. Zusätzliche Sicherheit bietet ein Certificate Pinning.

4.2 Benachrichtigungen

Benachrichtigungen auch bezeichnet als Notifications, sind Nachrichten, die den Benutzer über Ereignisse informieren. Apps können über Background-Tasks im Hintergrund laufen sowie mit Back-End-Systemen kommunizieren und befinden sich oftmals in einer zeitabhängigen und gekoppelten Umgebung. Sie benötigen eine Möglichkeit, den Nutzer über Ereignisse zu informieren, auch wenn sie sich im Moment des Auftretens eines Events nicht im Vordergrund befinden. Notifications stellen diesen Mechanismus zur Verfügung.
Die verschiedenen mobilen Plattformen stellen ähnliche Arten von Notifications zur Verfügung. Systeme wie Android 5, Windows Phone 8.1 und iOS 8 nähern sich deutlich

einander an. Die zwei grundlegenden Arten sind Lokale und Push Notifications. Lokale Notifications werden von der App selbst erzeugt. Push Notifications dagegen erzeugt ein Remote Server und pusht diese auf das jeweilige Endgerät. Die Plattformanbieter bieten jeweils eigene Lösungen für Push-Benachrichtiungen an. Apple stellt den Apple Push Notification Service (APNS), Mircrosoft den Windows Notification Service (WNS) und Google den Google Cloud Messaging Service (GCM) bereit.
Dieser Abschnitt befasst sich detailliert mit Benachrichtigungen unter Windows Phone 8.1 und stellt die grundlegenden Eigenschaften der Push-Notification-Services der einzelnen Plattformanbieter vor, um dem Entwickler einen Eindruck der Mächtigkeit dieser Services und ihrer Tauglichkeit für Enterprise Apps zu geben.

4.2.1 Benachrichtigungen unter Windows Phone 8.1

Neben den oben genannten Lokalen und Push Notifications unterscheidet Windows Phone die Polling Notifications. Diese pullen aktuelle Daten von einem Webserver.
Windows Phone unterteilt Notifications außerdem nach der Art, wie sie auf dem Bildschirm angezeigt werden. Die Icons des WP Startbildschirms werden Tiles bzw. Kacheln genannt. Ein Klick auf eine Kachel öffnet die jeweilige App. Dementsprechend existieren Tile Notifications, um diese Startbildschirm Tiles zu aktualisieren. Unter WP 8.1 gibt es eine Vielzahl an Templates (siehe Tile Template Catalog[14]) für die verschiedenen Kachelgrößen, mit denen ein Tile an die jeweiligen Bedürfnisse und den Style der App angepasst werden kann.

```
private static void createSimpleTile()
{
    TileUpdateManager.CreateTileUpdaterForApplication().Clear();
    TileUpdateManager.CreateTileUpdaterForApplication()
        .EnableNotificationQueue(true);
    //get template for 150x150 px tile
    XmlDocument tileXml = TileUpdateManager
        .GetTemplateContent(TileTemplateType.TileSquare150x150Text01);
    //set tile text
    var tileText = tileXml.GetElementsByTagName("text");
    (tileText[0] as XmlElement).InnerText = "Project Title";
    (tileText[1] as XmlElement).InnerText = "Task";
    (tileText[2] as XmlElement).InnerText = "This is a Task for the sample tile";
    (tileText[3] as XmlElement).InnerText = "Task run since 10 min.";
    //create notification
    TileNotification tileNotification = new TileNotification(tileXml);
    //update tile
    TileUpdateManager.CreateTileUpdaterForApplication().Update(tileNotification);
}
```

Listing 3 Erstellen und Updaten eines 150x150 Pixel großen Tiles unter Windows Phone 8.1

Die Beispielmethode „createSimpleTile“ aus Listing 3 zeigt das Erstellen eines 150x150 Pixel großen Tiles. Die „TileUpdateManager“-Klasse wird zum Erstellen und Aktualisieren von Tiles verwendet. Es können bis zu fünf Tiles in einer Notification Queue aufbewahrt werden. Durch das Aktivieren der Queue („EnableNotificationQueue(true)“) rotiert Windows Phone durch die verschiedenen Kacheln. Um bestimmte Tiles innerhalb der Queue zu aktualisieren, kann diesen ein Tag zugeordnet werden. Bei einer Aktualisierung wird jedes Tile mit dem entsprechenden Tag durch die neue Version ersetzt. (Vgl. MICROSOFT, 2014c) Mittels der Vorlage

[14] https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh761491.aspx

„TileSquare150x150Text01“ wird das entsprechende Tile-Template als XMLDocument geladen. Grundsätzlich können Tiles mehrere Zeilen Text, Bilder, ein Badge (Icon) sowie einen Counter anzeigen. Das vorliegende Template stellt lediglich vier Textzeilen zur Verfügung. Das Template und die eingesetzten Daten werden im XML-Format verarbeitet und durch Tile Notifications in diesem Format aktualisiert. Über die Update-Methode des „TileUpdateManagers“ wird die Aktualisierung des Tiles angestoßen.
Zusätzlich existieren Secondary Tiles. Diese können Links zu verschiedenen Ressourcen, wie Websites, Ordner, Seiten innerhalb einer App oder ähnliches enthalten. Des Weiteren kann auf dem Lock-Screen der Text des Tiles einer App als Lock-Screen-Tile angezeigt werden. (Vgl. MICROSOFT, 2014d) Dafür muss der Entwickler einen Background-Task erstellen, diesen im App-Manifest inklusive der ihn auslösenden Events angeben und im Code eine Registrierung des Tasks durchführen. Außerdem muss der Nutzer die App zur Anzeige auf dem Lock-Screen auswählen. Zusätzlich ist es möglich, über die „ScheduledTileNotification“-Klasse Tile Notifications zu erstellen, die nach bestimmten Zeitpunkten eine Aktualisierung durchführen.[15] Diese können den Lokalen Notifications zugeordnet werden.
Weiterhin existieren Toasts. Dies sind Textnachrichten, die dem Nutzer als Pop-Up am oberen Bildschirmrand angezeigt werden. Toasts werden von Microsoft als Nutzerbenachrichtigungen definiert, die eingesetzt werden, wenn sich die jeweilige App im Moment des Auftretens der Nachricht nicht im Vordergrund befindet. Bei ihrem Auftreten kann ein Sound abgespielt werden. Andere UI-Elemente wie Buttons werden nicht unterstützt. Hier besteht ein Unterschied zu Android und iOS. Dort können neben Text benutzerdefinierte Aktionen angegeben bzw. der Style der Nachricht angepasst werden.

4.2.2 Notification Services

Notification Services erlauben es Entwicklern, Push-Benachrichtigungen von ihrem eigenen Cloud Service an eine App zu versenden. Sie dienen dabei als Zwischenspeicher, sollte das mobile Endgerät nicht empfangsbereit sein und als sicherer Kommunikationskanal zur empfangenden App. Tabelle 5 zeigt einen Vergleich der von Apple, Microsoft und Google zur Verfügung gestellten Services.

Gemeinsamkeiten und Unterschiede der Notification Services

Alle in Tabelle 5 gezeigten Services, haben gemeinsam, dass sie keine Garantien bezüglich dem Empfang einer Nachricht und der Latenz zwischen Sender und Empfänger einräumen. Apple erwähnt explizit, den Service nicht zum Versenden von Daten zu verwenden, sondern lediglich als Benachrichtigungsmechanismus einzusetzen, sobald sich Daten auf dem Cloud Server geändert haben. Diese Daten können später separat vom Server angefordert werden. Weiterhin wird der Empfang einer Nachricht nicht bestätigt. Verlorengegangene Nachrichten können nicht behandelt werden.
Die Queuing-Eigenschaften unterscheiden sich teilweise voneinander. Während APNS eine einzige Nachricht zur gleichen Zeit speichert, sind es bei Googles GCM Service bis zu 100. Unter Android kann für Nachrichten, die ausschließlich für die Synchronisation bzw. Benachrichtigung gedacht sind, ein Tag (collapsing key) gesetzt werden. Es wird pro Tag nur

[15] https://code.msdn.microsoft.com/windowsapps/Tile-Update-every-minute-68dbbbff

eine Nachricht persistent gehalten. Windows Phone wiederum differenziert das Speicherverhalten je nach Nachrichtentyp.

Funktionsweise des APNS

Die Kommunikation findet stets zwischen drei Akteuren statt. Es existiert das mobile Endgerät mit der App, der Notification Service und der Cloud Service. APNS stellt zunächst eine sichere Verbindung zwischen Notification Service und dem Gerät her. Dafür wird eine TLS Peer-to-Peer Authentifizierung verwendet, in welcher sowohl Service als auch Device Zertifikate ausgetauscht und von der Gegenstellt geprüft werden. Diese Prozedur wiederholt sich zwischen Notification Service und Cloud Service. Daraufhin muss sichergestellt werden, dass die Nachricht ausschließlich an legitimierte Endpunkte gesendet wird. Dafür erstellt der APNS ein Device Token und gibt dieses an das autorisierte Endgerät weiter. Dieses übergibt das Token an die App. Von dort aus wird es an den Cloud Service Provider weitergereicht. Dieses Token wird in jede Benachrichtigung eingefügt. Daraufhin kann sichergestellt werden, dass ein Token für das Gerät, für welches die Benachrichtigung bestimmt ist, erstellt wurde und der korrekte Endpunkt die Nachricht erhält. (Vgl. APPLE, 2015c)

Funktionsweise des WNS

WNS nutzt einen ähnlichen Mechanismus, um ein sicheres Routing sicherzustellen. Hier wird zunächst ein Push-Benachrichtigungskanal (Kanal-Uniform Ressource Identifier (URI)) verwendet. Dieser wird ebenfalls beim Notification Service erstellt und von der App an den Cloud Service weitergereicht. Dieses Weiterreichen muss vom Entwickler selbst über sichere Standards implementiert werden. Zusätzlich muss sich der Cloud Service bei WNS registrieren. Benötigt werden dafür eine Packet-Sicherheits-ID sowie ein geheimer Schlüssel. Diese werden einer App bei der Registrierung im Windows Store übergeben und ebenfalls an den Cloud Service weitergereicht. Für die zukünftige Kommunikation zwischen Cloud Service und WNS übergibt WNS (ähnlich APNS) nach erfolgter Authentifizierung ein Zugriffstoken an den Cloud Service. Für das Senden einer Benachrichtigung wird schließlich eine HTTPS-POST-Anforderung auf dem Kanal-URI ausgeführt, in dessen Header sich das Zugriffstoken befindet. (Vgl. MICROSOFT, 2014f)

Funktionsweise des GCM Service

GCM stellt zwei verschiedene Arten von Notification Servern bereit. Der Entwickler kann sich zwischen einem HTTP- und einem Extensible Messaging and Presence Protocol (XMPP)-Server entscheiden. Bei der Nutzung des XMPP-Servers wird das Simple Authentication and Security Layer (SASL) Framework für eine sichere Kommunikation eingesetzt. Dadurch wird die Arbeit des Entwicklers bei der Erstellung einer sicheren Kommunikation vereinfacht. Er muss ausschließlich eine bestehende SASL-Implementierung wählen, ohne selbst zu implementieren. Des Weiteren besteht die Möglichkeit der bidirektionalen Kommunikation, sodass das Gerät Nachrichten zurück an den Server schicken kann. Für beide Kommunikationswege kann dieselbe Verbindung genutzt werden. (Vgl. GOOGLE, 2015b) Weitere Informationen hierzu sind in den GoogleCloudMessaging APIs zu finden. Die Kommunikation über den HTTP-Server erfolgt mittels SSL-Verschlüsselung, wobei im Header ein API-Key zur Authentifizierung genutzt wird.

	APNS	WNS	GCM
Payload	2 KB seit iOS 8	5 KB	4 KB
Garantie der Zustellung	“best effort”, nicht garantiert	Zuverlässigkeit oder Latenz nicht garantiert	Zuverlässigkeit und Latenz (durch internes Throttling) nicht garantiert
Queuing, wenn Device offline	Speicherung für begrenzte Zeit, eine Nachricht pro App, neue Nachricht führt zum Löschen vorheriger	Speicherung max. 3 Tage, bei aktiviertem Queuing bis zu 5 Kachelbenachrichtigungen pro App, keine Speicherung von Toast Notifications	Speicherung max. 4 Wochen, bis zu 100 Nachrichten pro App ohne collapsing key, max. 4 collapsing keys (= löschen vorheriger Nachrichten ähnlich APNS)
Umgang mit fehlgeschlagenen Nachrichten	Keine Rückantwort	Keine Rückantwort	Keine Rückantwort pro Nachricht, Benachrichtigung, wenn Limit von 100 überschritten (löscht alle Nachrichten)
Sicherheit der Kommunikation	Connection trust (TLS mit Server und Client CA) + assurance of accurate message routing (token trust)	Push-Benachrichtigungskanal, Kommunikation Cloud Service mit WNS über HTTPS, Authentifizierung durch Paket-SID und geheimen Schlüssel	GCM-XMPP-Server: TLS-Verbindung, SASL PLAIN Authentifizierung + API-Key, GCM-HTTP-Server: SSL, Authentifizierung über API-Key
Inhalt der Nachricht	JSON-Dictionary mit: Alert message, Counter für Badge-Icon, Sound, Text, benutzdef. Werte	Kachel-, Popup- und Signalbenachrichtigungen (XML) sowie unformatierte Pushbenachrichtigungen (Rohdaten)	Rohdaten, vollständige Kontrolle der App, wie diese Daten interpretiert werden

Tabelle 5 Übersicht der Push Notifications Services APNS, WNS und GCM

Vorstellung des BlackBerry Push Notification Service

BlackBerry bietet ebenfalls einen Push Notification Service an. Im Vergleich zu den bereits behandelten Services besteht hier die Möglichkeit, mehr Informationen über den Status von versendeten Nachrichten zu erhalten. Des Weiteren gibt es keine Abhängigkeit zu einem externen Cloud Service des Plattformbetreibers. Es besteht die Möglichkeit, die Kommunikation über den Enterprise eigenen BES zu realisieren. In Abbildung 4 wird der Übertragungsprozess einer Push-Notification dargestellt. Die folgende Beschreibung steht im Vergleich zu (BLACKBERRY, 2014b).

Es sei angenommen, der Push Proxy Gateway (PPG) ist der BES. Zunächst sendet der Push Initiator seine Nachricht an den PPG mit Hilfe eines HTTPS Posts. Der PPG befindet sich, entgegen des WNS oder des HTTP/XMPP-Servers der oben behandelten Services, unter der Kontrolle des Unternehmens. Nun sendet der PPG eine Bestätigung des Empfangs und eine Benachrichtigung, ob er die Notification annimmt. Wird die Notification zurückgewiesen, so wird ebenfalls ein Grund mitgeteilt. In Schritt 3 wird die Notification an das mobile Endgerät weitergeleitet. Eine erfolgreich empfangene Nachricht muss den folgenden Kriterien genügen:

- Eingang innerhalb der vorgegebenen Zeit
- Einhaltung der Kriterien innerhalb des „quality-of-service“-Elements

Schritt 4 zeigt die Bestätigung der eingegangenen Nachricht. Daraufhin sendet der PPG eine Notification über das Resultat der Übertragung an den Push Initiator, welcher daraufhin einen umfangreichen Statusbericht über die gesendete Notification besitzt. Schließlich Bestätigt der Push Initiator den Empfang des Resultates.

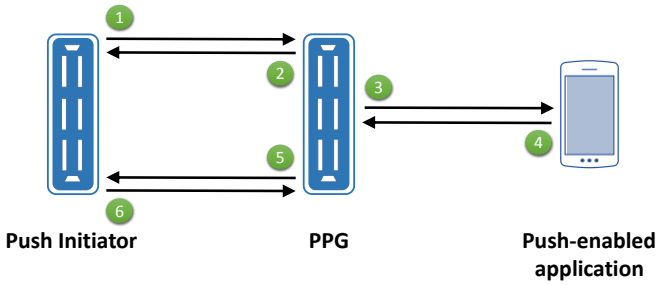

Abbildung 4 Nachrichtenablauf des Blackberry Push Notification Services, (vgl. BLACKBERRY, 2014b)

Blackberry bietet zwei Level-of-Service Optionen an. Der Push Essentials Service gibt, wie die oben beschriebenen Services, keine Rückantwort oder Statusinformationen über eine gesendete Nachricht. Der Push Plus Service ermöglicht:

- Notifications über das Resultat der Übertragung (übertragen, nicht übertragen, Zeit abgelaufen)
- Statusanfragen
- Abbrechen der Übertragung
- Einstellen der maximalen Übertragungszeit auf bis zu 8 Stunden

4.2.3 Schlussfolgerung

Zusammenfassend ist zu bemerken, dass die vorgestellten Services für die Zustellung von Benachrichtigungen an den Benutzer durchaus ausreichend sind. Alle Services bieten entsprechende Sicherheitsmechanismen und können die verschiedenen Nachrichtentypen je nach Plattform versenden. Für den Entwickler kann die Anzahl der persistent gehaltenen Nachrichten von Bedeutung sein. Die Services aus Tabelle 5 eignen sich insbesondere nicht für das Versenden von Daten, da keine Garantie für deren Übermittlung besteht. Außerdem sollten Apps, die eine Echtzeitkommunikation voraussetzen, eigene Mechanismen zum Nachrichtenaustausch implementieren. Zeitschranken können nicht garantiert werden. Im Gegensatz zu diesen Services bietet Blackberry die Möglichkeit, detaillierte Übertragungsinformationen einer Notification zu erhalten. Dies wird über einen zuverlässigen Übertragungskanal mit Hilfe von Bestätigungsnachrichten zwischen den beteiligten Services erreicht. Damit sind deutlich komplexere Nachrichtenabläufe mit Zeitschranken und der Benachrichtigung von mehreren Geräten mit Empfangsbestätigungen möglich. Aus diesem Grund werden cloud-unabhängige Services, siehe z.B. Abbildung 4, als deutlich zuverlässiger eingestuft als cloud-abhängige Services siehe Tabelle 5.

4.3 Mobile Cloud Computing

Mobile Cloud Computing (MCC) führt die Cloud und die dort angebotenen Services an mobile Endgeräte heran. Mobile Endgeräte besitzen sehr begrenzte Ressourcen, denkt man beispielsweise an Batterielebensdauer, Speicher, Bandbreite, Skalierbarkeit, Verfügbarkeit, Zuverlässigkeit und Privatsphäre. MCC ist ein Weg, diese Schwächen mobiler Geräte anzugehen, indem Services in der Cloud angeboten werden und das mobile Endgerät diese in Anspruch nimmt und somit nicht selbst erbringen muss. Das Mobile Cloud Computing Forum definiert MCC wie folgt:

"Mobile Cloud Computing at its simplest, refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just smartphone users but a much broader range of mobile subscribers." (DINH, LEE, NIYATO & WANG, 2013)

Die angebotenen Services werden in die Klassen

- Software as a Service (Salesforce CRM, Microsoft Office 365, Dropbox),
- Platform as a Service (Microsoft Azure, Google App Engine),
- Infrastructure as a Service (Amazon ES2, S3) unterteilt.

Infrastructure as a Service stellt die reine Hardware wie Speicher, Server, Virtuelle Maschinen oder Netzwerkkomponenten bereit. Abgerechnet wird meist nach der Dauer der tatsächlich genutzten Ressourcen. Die Ressourcen können dynamisch an die momentanen Anforderungen der laufenden Applikationen angepasst werden. (Vgl. MICROSOFT, 2015a)
Platform as a Service bietet, neben Hardwareservices und BSs der IaaS, die Anwendungsinfrastruktur in Form von technischen Frameworks, wie Datenbanken und Middleware oder die gesamte Entwicklungsplattform. Das Entwickeln, Testen und Warten von Anwendungen wird erleichtert. (Vgl. MICROSOFT, 2015b)
Software as a Service ist ein Software-Vertriebsmodell, bei dem Anwendungen über das Internet zur Verfügung gestellt werden. Die benötigte Infrastruktur und Anwendungen werden dabei als ein Paket bereitgestellt. (Vgl. ROUSE, 2010)
Abbildung 5 zeigt eine konzeptionelle Sicht auf das MCC. Es wird die globale Vernetzung von Endgeräten und Cloud-Service-Anbietern verdeutlicht. Zellulare und WLAN-Netzwerke sorgen für eine ständige Anbindung an das Internet bzw. an die Cloud-Infrastruktur. Diese wird auf oberster Ebene in Private und Public Clouds unterteilt. Private Clouds bieten Services für den Gebrauch eines Unternehmens oder einer Institution an. Nutzer der Cloud und Service-Anbieter sind innerhalb der Institution angesiedelt. Services von Public Clouds werden am freien Markt angeboten und können von einer unbegrenzten Anzahl von Cloud-Anwendern genutzt werden. (Vgl. BUDSZUS, BERTHOLD, FILIP, POLENZ, PROBST & THIERMANN, 2014)
Dieses Kapitel geht zunächst auf das Kriterium des Datenschutzes bei der Verwendung von Cloud-Services ein. Es soll die Frage beantwortet werden: Sind Daten bei grenzübergreifendem Datenverkehr geschützt und welche Daten sind geschützt? Daraufhin wird die Datenhaltung anhand von Cloud-Storage-Lösungen (SaaS) näher beschrieben. Anhand verschiedener Cloud-APIs wird schließlich auf die Funktionsweise von Synchronisation und Offline-Datenhaltung auf verschiedenen mobilen Endgeräten eingegangen.

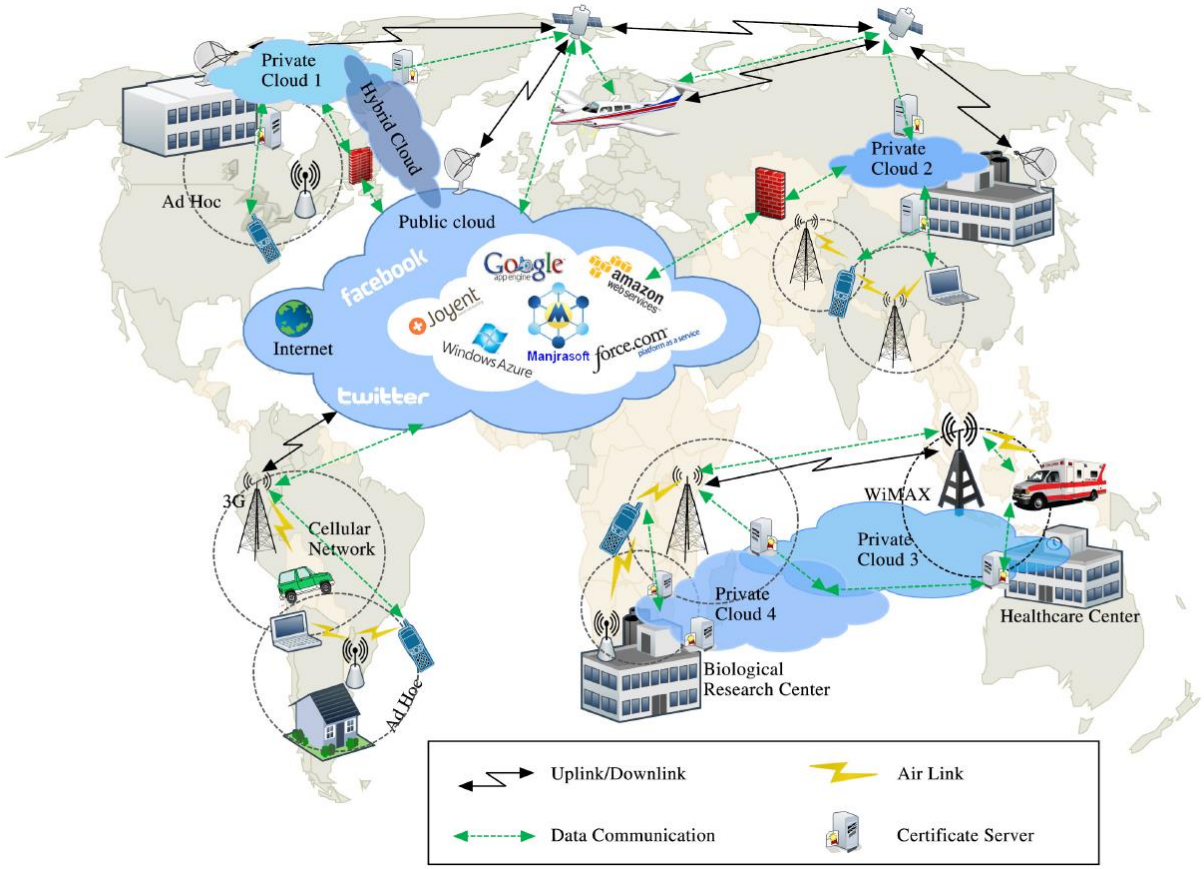

Abbildung 5 Konzeptionelle Sicht auf das MCC (SANAEI, ABOLFAZLI, GANI & BUYYA, 2014)

4.3.1 Datenschutzrechtliche Aspekte

Bei der Nutzung eines Cloud-Services durch den sogenannten Cloud-Anwender und die Erbringung des Services durch den Cloud-Anbieter, sind entsprechende datenschutzrechtliche Aspekte nach dem Bundesdatenschutzgesetz (BDSG) zu beachten. Es genügt nicht, auf reiner Vertrauensbasis personenbezogene Daten von Dritten wie z.B. den Mitarbeitern an den Cloud-Anbieter zu übermitteln und sich auf dessen Sicherheitsversprechen, die meist durch entsprechende Zertifikate gegeben werden, zu verlassen.

Regelungen zu personenbezogenen Daten

„Aus Sicht des Bundesdatenschutzgesetztes (BDSG) sind dessen Regelungen im Zusammenhang mit dem Cloud Computing nur dann anwendbar, wenn personenbezogene Daten im Inland (Ort der Datenverarbeitung) erhoben, verarbeitet oder genutzt werden (§§ 1, 3 BDSG).“ (PETZOLD, 2011) Die Anwendbarkeit auf nicht personenbezogene Daten ist somit nicht gegeben. Es kann nicht immer zweifelsfrei geklärt werden, ob die vom Cloud-Anwender übermittelten Daten personenbezogen sind oder nicht. (Vgl. PETZOLD, 2011)
Sobald der Cloud-Anwender IT-Dienstleistungen eines Cloud-Anbieters in Anspruch nimmt, gilt Letzterer als Auftragnehmer nach § 11 Abs. 1 BDSG. Die Nutzung des Cloud-Dienstes muss durch einen schriftlichen Vertrag vereinbart werden, in welchem Anforderungen wie die Berichtigung, Löschung und Sperrung von Daten dargelegt sind.
Es ist zu beachten, dass der Cloud-Anwender nach § 11 Abs. 1 BDSG stets für die Einhaltung sämtlicher datenschutzrechtlicher Bestimmungen verantwortlich bleibt. Dies ist zum Beispiel bei der Einbeziehung von Unteranbietern zu beachten. Oftmals ist es für den Cloud-Anwender

nicht transparent, bei welchen Drittanbietern und in welchem Land der eigentliche Cloud-Anbieter die ihm übermittelten Daten speichert. Der Cloud-Anbieter muss verpflichtet werden, jederzeit Auskunft über die beteiligten Unteranbieter geben zu können. Der Cloud-Anwender bleibt Herr der Daten (vgl. PETZOLD, 2011). Der Cloud-Anbieter „ist unter besonderer Berücksichtigung der Eignung der von ihm getroffenen technischen und organisatorischen Maßnahmen sorgfältig auszuwählen." (§ 11 Abs. 2 S. 1 BDSG). Dabei sind Zertifizierungen, z.B. nach ISO 27001[16] oder Gütesiegel, nur als Anhaltspunkte zu sehen, müssen jedoch auf ihre Aussagefähigkeit im Hinblick auf Cloud-spezifischen Datenschutz und IT-Sicherheitsrisiken geprüft werden.
Verschlüsselte Daten nehmen eine Sonderrolle ein, da die Frage besteht, ob es sich um personenbezogene Daten handelt. Meist gibt es Möglichkeiten, selbst verschlüsselte Daten einer Person zuzuordnen. Für eine Entscheidung müssen die genutzte Verschlüsselung und die Stärke des genutzten kryptographischen Algorithmus berücksichtigt werden. (Vgl. BUDSZUS, BERTHOLD, FILIP, POLENZ, PROBST & THIERMANN, 2014)

Regelungen für den grenzüberschreitenden Datenverkehr und ausländische Behörden

Die in diesem Abschnitt folgenden Regelungen des grenzüberschreitenden Datenverkehrs stehen beruhen auf BUDSZUS, BERTHOLD, FILIP, POLENZ, PROBST & THIERMANN (2014). Sie sollten auf Grund unterschiedlicher Datenschutzniveaus besondere Aufmerksamkeit erhalten.
Im Innereuropäischen Raum gelten die datenschutzrechtlichen Anforderungen nach der Richtlinie 95/46/EG. Orte der technischen Verarbeitung personenbezogener Daten sind eindeutig festzulegen.
Erfolgt die Datenverarbeitung außerhalb der EU bzw. des Europäischen Wirtschaftsraumes durch eine Datenverarbeitung in Drittstaaten, so gelten besondere Anforderungen der §§ 4b, 4c BDSG für den Drittstaatentransfer. Der Cloud-Anwender ist verpflichtet, sich Garantien zum Schutz des allgemeinen Persönlichkeitsrechts einzuholen, sollte in entsprechenden Drittstaaten kein angemessenes Datenschutzniveau bestehen. Cloud-Anbieter mit Sitz in den USA verpflichten sich oftmals zur Einhaltung der Safe-Harbor-Grundsätze, um ihre Vertrauenswürdigkeit zu bekräftigen. Hier ist Vorsicht geboten, da sich Cloud-Anbieter oder Unteranbieter auf freiwilliger Basis gegenüber dem US-Handelsministerium selbst zertifizieren können. Eine flächendeckende Kontrolle der Zertifizierung ist nicht gegeben.
Abschließend sei an dieser Stelle auf Entwicklungen zur Einsicht von personenbezogenen Daten durch ausländische Behörden hingewiesen. US-Behörden, wie das Federal Bureau of Investigation oder die National Security Agency, sind auf Grundlage von US-amerikanischem Recht ermächtigt, auf personenbezogene Daten in Europa zuzugreifen. Dafür ist es ausreichend, dass der Cloud-Anbieter in den USA geschäftlich tätig ist bzw. dort ein Büro unterhält. Damit könnten auch Daten einer innereuropäischen Cloud betroffen sein. Eine Datenübermittlung durch ein Unternehmen mit Sitz in Europa steht mit Art. 26 der Richtlinie 95/46/EG bzw. 4c BDSG in Konflikt. Der Cloud-Anwender würde durch die Übersendung der personenbezogenen Daten durch den Cloud-Anbieter gegen europäisches und deutsches Datenschutzrecht verstoßen. Internationale Regelungen einer solchen Datenverarbeitung sind im Moment nicht vorhanden. (Vgl. BUDSZUS, BERTHOLD, FILIP, POLENZ, PROBST & THIERMANN, 2014)

[16] ISO/IEC 27001:2013 "specifies the requirements for establishing, implementing, maintaining and continually improving an information security management system within the context of the organization." (ISO (2013))

Durch die Verantwortlichkeit des Cloud-Anwenders für die Übermittlung von personenbezogenen Daten ist die Einschätzung und Auswahl eines geeigneten Cloud-Service-Providers von großer Bedeutung. Vor allem die Regelungen des grenzüberschreitenden Datenverkehrs und die Entwicklungen zur Einsicht durch ausländische Behörden sind Entscheidungskriterien für Entwickler von Enterprise Applikationen.

4.3.2 Cloud-Storage Lösungen und Datensicherheit in der Cloud

Das Angebot an Diensten des MCC bzw. der drei Klassen IaaS, PaaS und SaaS wächst ständig und ist schwer überschaubar. Ein typischer Dienst, der sowohl von Großunternehmen als auch Privatpersonen exzessiv genutzt wird und sich durch zahlreiche APIs für verschiedene mobile Plattformen im mobilen Bereich durchgesetzt hat, ist der Cloud-Speicher. Im Folgenden werden vier weitverbreitete Cloud-Storage Lösungen vorgestellt.
Tabelle 6 zeigt die Cloud-Storage-Lösungen von Dropbox, Apple (iCloud), Google (Google Drive) und Microsoft (OneDrive). Alle Dienste, bis auf iCloud, bieten Optionen für Unternehmen an. Dabei handelt es sich meist um Services, welche über die Dienste für Privatnutzer hinausgehen wie erweiterte Dateifreigabesteuerung, Nutzerverwaltung und Wiederherstellungsfunktionen. Diese werden zu gesonderten Konditionen angeboten und bieten ein Vielfaches an Online-Speicher im Vergleich zu dem freien Speicherkontingent für Privatpersonen.
Dropbox bietet die Möglichkeit, mit Hilfe der Datastore API strukturierte Daten wie Kontakte, Aufgaben oder Bearbeitungsstände in Form von Records in einem Datastore zu Speichern. iCloud besitzt diese Funktionalität ebenfalls. Google und Microsoft bieten hier separate Cloud-Lösungen für Datenbanken an.
Die Dienste bieten zahlreiche Sicherheitsmechanismen an. Die Netzwerkkommunikation erfolgt stets über eine gesicherte SSL/TLS-Verbindung. Dropbox, Google Drive sowie OneDrive stellen für die Autorisierung das OAuth 2.0 Protokoll bereit. Alle betrachteten Dienste stellen eine Zwei-Faktor-Authentifizierung zur Verfügung. Eine Serverseitige Verschlüsselung wird ausschließlich bei iCloud versprochen (mind. AES-128). Allerdings heißt eine serverseitige Verschlüsselung nicht in jedem Fall, dass der Cloud-Anbieter die Daten nicht jederzeit entschlüsseln kann, da er theoretisch Zugriff auf die verwendeten Schlüssel des Nutzers besitzt, die ebenfalls in der Cloud abgelegt sind. Eine effektive Verschlüsselung kann somit ausschließlich über Drittanbieter wie z.B. Boxcrypter (AES-256) erfolgen, welche die Daten vor dem Einbringen in die Cloud verschlüsseln.
Dateien, die auf dem mobilen Endgerät des Nutzers abgelegt sind, entziehen sich bei der Speicherung in der Cloud der Kontrolle des Nutzers bzw. des Unternehmens. Es stellt sich die Frage, ob Zugriffsrechte, die z.B. vom Entwickler oder MDM-System festgelegt wurden, erhalten bleiben.
Unter iOS werden einzelnen Dateien Schutzklassen zugewiesen siehe Abschnitt 4.1.3. Diese Schutzklassen existieren ebenfalls für applikationsspezifische KeyChain-Einträge. Abhängig von der Schutzklasse und einem separaten Backup-Flag sind Dateien in einem iTunes- oder iCloud-Backup enthalten. Alle Dateien, welche nicht die Schutzklasse „ThisDeviceOnly“ sowie ein gesetztes Backup-Flag besitzen, werden in ein Backup integriert. Die Backup-Funktion kann über MDM-Richtlinien gesetzt werden. Trotz dieser Maßnahmen existieren Sicherheitsrisiken. Zum Beispiel ist der MDM-Administrator nicht in der Lage,

Sicherheitseinstellung des iCloud-Passwortes zu beeinflussen und damit die Sicherheit externer Daten sicherzustellen. Für sicherheitskritische Umgebungen wird deshalb geraten, das iCloud-Backup zu deaktivieren. (Vgl. TEUFL, ZEFFERER, STROMBERGER & HECHENBLAIKNER, 2013)
Unter Dropbox existiert ein separates Zugriffsverwaltungssystem. Innerhalb der Dropbox können Richtlinien für einzelne Dateien gesetzt werden. Vorhandenen Richtlinien lassen sich jedoch nicht exportieren und der MDM-Administrator hat ebenfalls keinen Einfluss auf Einstellung des Services. Aus diesem Grund wird die Installation der Dropbox-App in sicherheitskritischen Umgebungen durch das MDM-System blockiert.

	Dropbox	iCloud	Google Drive	OneDrive
Plattform SDKs	Android, iOS, Web	iOS	Android, iOS, Web, Windows Phone	Android, iOS, Web, Windows Phone
APIs	Core API, Dropbox for Business API, Datastore API, Sync API,	CloudKit API	Google Drive API	Live SDK
Authorisierung/ Authentifizierung	OAuth 2.0/ Zwei-Faktor-Authent. möglich	n.a./Zwei-Faktor-Authentifizierung möglich	OAuth 2.0/ Zwei-Faktor-Authent. möglich	OAuth 2.0/ Zwei-Faktor-Authent. möglich
Organisation der Daten	Files; Datastore (Record, Value)	Atomic persistent stores, Transactional stores (SQL), Document storage (Files), Database, Records	Files	Files
Verwalten von Apps	App Console	CloudKit Dashboard	Google Developers Console	Live SDK App Management
Freier Speicher	2GB für Files; Datastore: Max. Record Größe: 100 KiB Max. Anzahl von Records: 100K Max. Datastore Größe: 10 MiB Max. Größe einzelner sync(): 2MiB	5GB	15 GB frei für Mail, Dokumente, Accountinformationen, iOS Einstellungen und App-Daten (automatisch synchronisiert über iCloud)	15GB
Limit pro File	Kein Limit	-	10GB	2GB
Preise für Unternehmen	12 €/Nutzer/Monat, unbegrenzter Speicher	20€/Monat/1TB, keine Unternehmens-optionen	8 € pro Monat und pro Nutzer, unbegrenzter Speicher	3,80 € pro Benutzer und Monat, 1TB pro Nutzer
Funktionen für Unternehmen	Wiederherstellung von Dateien, Dateifreigabe-steuerung	-	Tools zur Nutzer-verwaltung, Geräte- und Mobilgeräte-verwaltung, Freigabe-berechtigungen	Office-Online, Data-Loss-Prevention-Funktionen
Verschlüsselung	Drittanbieter z.B. Boxcrypter AES-256	Mind. AES-128	Drittanbieter z.B. Boxcrypter AES-256	Drittanbieter z.B. Boxcrypter AES-256
Backup-Funktion für Smartphone	-	ja	ja	ja

Tabelle 6 Vergleich der Cloud-Storage Lösungen

4.3.3 Cloud-APIs und Synchronisation auf mobilen Endgeräten

Im Folgenden werden die CloudKit API der iCloud sowie die Dropbox Datastore API näher vorgestellt. Beide APIs ermöglichen es, strukturierte Informationen wie App-Daten in Form von Records in Datenbanken zu speichern und mit dem mobilen Endgeräten zu synchronisieren. Damit wird dem Problem des geringen Speichers auf mobilen Endgeräten begegnet und die Bereitstellung und Synchronisation von Daten auf den unterschiedlichen Endgeräten des Nutzers deutlich vereinfacht. Als simples Beispiel dient die Anfrage von Buch-Daten eines bestimmten Autors in der entsprechenden Datenbasis. Die Quellcodebeispiele sind in der Sprache Objective-C verfasst.

a. iOS CloudKit

CloudKit ist ein von Apple angebotener Service, mit dessen Hilfe Daten zwischen einer iOS-App und iCloud synchronisiert werden können. iOS ist in der Lage, unterschiedliche Datenmodelle zu verwalten (siehe Tabelle 6). CloudKit arbeitet auf Records. Ein Record ist ein Dictionary von Key-Value-Paaren. Ein Key definiert ein Feld eines Records. Für Values werden einfache Datentypen wie String, Number und Date unterstützt. Außerdem sind Dateien und Referenzen auf andere Records möglich. Mit CloudKit entscheidet der Entwickler genau, wann Daten ausgetauscht werden und über welche Änderungen in der iCloud er informiert werden möchte. Die folgenden Beschreibungen stehen im Vergleich zu APPLE (2014b).

i. Datenhaltung

Die ICloud speichert Daten innerhalb von Partitionen sogenannten Containern. Jeder Container wird einer App zugeordnet. Ein Container kann jedoch für mehrere Apps freigegeben werden. Eine App kann auf mehrere Container zugreifen. Ein Container besteht aus einer privaten und einer öffentlichen Datenbank. Auf die öffentliche Datenbank haben alle Instanzen einer App Zugriff. Dort werden sowohl Benutzer als auch App-Daten gespeichert. Das Lesen von Daten erfolgt dort ohne eine Autorisierung. Für jeden Nutzer einer App existiert eine private Datenbank. Eine App hat ausschließlich Zugriff auf die private Datenbank des jeweiligen Nutzers. Sowohl für Lese- als auch Schreiboperationen sind iCloud-Anmeldedaten erforderlich. Das Management einer Datenbank kann vollständig über das CloudKit Dashboard erfolgen. Selbst das Erstellen des Schemas übernimmt der CloudKit-Service. Es wird anhand der eingefügten Records automatisch erstellt. (Vgl. APPLE, 2014b)

ii. Anfrage von Records

Um einzelne Records einer Datenbank anzufragen, empfiehlt sich die Methode per Anfrage („CKQuery“). Damit werden ausschließlich die benötigten Daten übertragen. In Listing 4 bewirkt das Prädikat („NSPredicate“) die Auswahl aller in der Query definierten Buch-Records mit Autor „Andrew Stuart Tanenbaum“. Die Anfrage wird, wie in Listing 4 zu sehen, durch den Record-Typ, ein Prädikat und einen Sortier-Deskriptor definiert.

```
CKDatabase *myDatabase = [[CKContainer containerWithIdentifier:containerIdentifier]
publicCloudDatabase];
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"author = %@", @"Andrew
Stuart Tanenbaum"];
CKQuery *query = [[CKQuery alloc] initWithRecordType:@"Book" predicate:predicate];
[myDatabase performQuery:query inZoneWithID:nil completionHandler:^(NSArray
*results, NSError *error) {
        if (error) {
            // Error handling for failed fetch from public database
        }
        else {
            // Display the fetched records
        }
    }];
```

Listing 4 Anfrage eines Records einer iCloud-Datenbank mit dem CloudKit (vgl. APPLE, 2014b)

iii. Empfangen von Änderungen

Nun wäre es äußerst ineffizient, wenn der Client die unter ii. gestellte Query ständig ausführt, um Änderungen in Erfahrung zu bringen. Aus diesem Grund informiert die App den Cloud-Server, dass sie über Änderungen im Resultat der Query benachrichtigt werden möchte (Subscription). Die Anfrage wird nun vom Server ausgeführt und dieser teilt eine Änderung mit. (Vgl. APPLE, 2014b) Im Beispiel wird die App benachrichtigt, wenn ein neues Buch des Autors hochgeladen wird. Daraufhin kann die UI die Änderungen an den App-Nutzer weitergeben.
Eine Subscription besteht aus dem Record-Typ und dem in Listing 4 definierten Prädikat. Ein weiteres Optionsfeld gibt die Art der Änderung an, welche zu einer Benachrichtigung führt. Möglich sind Benachrichtigungen bei Erstellen, Löschen und Ändern von Records, ein einmaliges Feuern und eine Kombinationen von diesen. Daraufhin muss ein „CKNotificationInfo“-Objekt erstellt werden, welches unter anderem den Text einer Notification enthält. Dieses Objekt zeigt Listing 5.

```
CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
notificationInfo.alertLocalizationKey = @"There is a new book by your registered
author.";
notificationInfo.shouldBadge = YES;
```

Listing 5 Erstellen der Notification Beschreibung bei Änderungen der Datenbank einer iCloud (vgl. APPLE, 2014b)

Schließlich wird die Subscription („CKSubscription“) inklusive „NotificationInfo“-Objekt in der Datenbank „myDatabase“ gespeichert. Listing 6 zeigt die „saveSubscription“-Methode.

```
CKDatabase *myDatabase = [[CKContainer containerWithIdentifier:containerIdentifier]
publicCloudDatabase];
[myDatabase saveSubscription:subscription completionHandler:^(CKSubscription
*subscription, NSError *error) {
    if (error)
        // insert error handling
    }
];
```

Listing 6 Speicherung einer Subscription in der Datenbank der iCloud (vgl. APPLE, 2014b)

Ein elementarer Schritt ist die Registrierung für Push-Notifications. Das CloudKit nutzt den APNS-Service, um Subscription Notifications zu senden. Die Registrierung erfolgt in der „application:didFinishLaunchingWithOptions:“-Methode. Dabei wird dem Service ein Device

Token übergeben, welches das mobile Endgerät eindeutig gegenüber APNS identifiziert. Notifications werden ausschließlich an dieses Gerät gesendet. Die Reaktion auf eine eingegangene Notification erfolgt in der „application:didReceiveRemoteNotification:“-Methode. (Vgl. APPLE, 2014b)

b. Dropbox Datastore API für iOS

Die Datastore API dient der Synchronisation von App-Nutzerdaten zwischen verschiedenen Geräten und BSs. Dropbox speichert Daten, ähnlich der iCloud, in Form von Records. Die Values können ebenfalls einfache Datentypen wie Strings, Intergers, Booleans oder Listen von einfachen Typen enthalten. Weitere Merkmale der API sind Offline-Access und Automatische Konfliktlösung. Die folgenden Beschreibungen stehen im Vergleich zu (DROPBOX).

i. Datenhaltung

Die Datastore API kennt ähnlich der iCloud Container. Ein Container ist hier allerdings ein Datastore ähnlich einer Datenbank und besteht aus einer Menge von Tabellen. Eine Tabelle ist eine Menge von Records. Jeder Record kann eine unterschiedliche Menge von Feldern besitzen. Um sinnvoll Anfragen an eine Tabelle stellen zu können, ist es von Vorteil, wenn die Records einer Tabelle größtenteils die gleichen Felder besitzen. Allerdings gibt es kein festes Schema wie in relationalen Datenbanken oder in der iCloud. Es existiert ein Offline-Cache für Datastores, um schnelle Zugriffszeiten und Offline-Bearbeitung zu ermöglichen. iCloud arbeitet ebenfalls mit Offline-Caching der iCloud-Container. Datastores können von mehreren Accounts gemeinsam genutzt werden. (Vgl. DROPBOX)

ii. Anfrage von Records

Die Anfrage von Records (siehe Listing 7) beginnt mit dem Festlegen eines Datastores. Jede App besitzt einen Standard-Datastore. Aus diesem Grund wird dem „myDatastore“-Objekt eine Instanz des Standard-Datastores mit Hilfe des „DBDatastoreManagers“ zugewiesen. Daraufhin wird die zu durchsuchende Tabelle des Datastores abgerufen. Auf der Tabelle “Book” wird nun die Anfrage ausgeführt (siehe „getTable“-Methode).

```
@property (nonatomic, strong) DBDatastore *myDatastore;
self.myDatastore = [[DBDatastoreManager sharedManager] openDefaultDatastore:nil];

DBTable *booksTbl = [self.myDatastore getTable:@"Book"];

NSArray *results = [booksTbl query:@{ @"author": @"Andrew Stuart Tanenbaum" }
error:nil];
```

Listing 7 Anfrage von Records mit der Datastore API unter Dropbox (vgl. DROPBOX)

Die „query“-Methode nimmt eine Liste von Bedingungen (Key-Value-Paare) entgegen. Die Matches werden als Array von Records zurückgegeben. Analog des Beispiels unter CloudKit, werden Records des Typs „Book“ abgefragt. Hier wird allerdings nicht der Record-Typ festgelegt, sondern die entsprechende Tabelle dient als Typ. Schließlich werden alle Buch-Records des Autors „Andrew Stuart Tanenbaum“ zum Resultat „results“ hinzugefügt.

iii. Empfangen von Änderungen

Die Synchronisation wird unter Dropbox über die „sync“-Methode ausgelöst. Ein Datastore erhält somit Änderungen von anderen Instanzen einer App. Listing 8 zeigt die Registrierung eines Observers der „sync“-Methode.

```
__weak typeof(self) weakSelf = self;
[self.myDatastore addObserver:self block:^() {
    if (weakSelf.myDatastore.status.incoming) {
        NSDictionary *changes = [weakSelf.myDatastore sync:nil];
        // Handle the updated data
    }
}];
```

Listing 8 Empfangen von Änderungen der Datenbank mit der Datastore API unter Dropbox (vgl. DROPBOX)

Der Oberserver-Block (siehe „block“) wird aufgerufen, sobald sich der Status des Datastores durch einen Upload bzw. Download ändert. Durch das Abfragen der „datastore.status“-Eigenschaft kann überprüft werden, welche Änderungen stattgefunden haben und entsprechend geantwortet werden. Um Änderungen in den Datastore zu übernehmen, wird die „sync“-Methode aufgerufen. Das „NSDictionary“ enthält Tabellen-IDs zu Mengen von Records, die durch den „sync“-Aufruf geändert wurden. Daraufhin kann die View anhand der geänderten Records angepasst werden. (Vgl. DROPBOX)

4.3.4 Schlussfolgerung

Angefangen mit den datenschutzrechtlichen Bestimmungen ergibt sich, dass ein Konzept der Nutzung von Cloud-Diensten erarbeitet werden sollte und eine umfassende Auseinandersetzung mit diesen Diensten stattfinden muss, um ein angemessenes Datenschutzniveau je nach Branche zu erreichen. Dabei sind sowohl der Schutz personenbezogener Daten, als auch die rechtlichen Pflichten des Unternehmens als Cloud-Anwender zu bedenken. Hier gilt, der Cloud-Anwender hat stets die Pflicht, sich von der Eignung des Cloud-Anbieters zu überzeugen. Die beispielhafte Auseinandersetzung mit Cloud-Storage-Lösungen zeigt, dass Angebote für verschiedene Anwendungsszenarien zur Verfügung stehen, was eine genaue Analyse der benötigten Funktionen und des Datenaufkommens notwendig werden lässt. Die angebotenen Abrechnungsmodelle der untersuchten Anbieter ähneln sich stark. Das benötigte Datenvolumen, die Art der zu speichernden Daten und zusätzlichen Funktionalitäten sind Hauptentscheidungskriterien für einen der Dienste. Dies spiegelt sich ebenfalls bei den angebotenen APIs wieder. Die Speicherung einzelner Datensätze wird von zwei der vier untersuchten Anbieter unterstützt. Die iCloud bietet durch die Umsetzung privater und öffentlicher Datenbanken sowie dem Teilen von Containern zwischen Apps detaillierte Kontrollmöglichkeiten des Datenflusses an. Einfache Aufgaben werden von allen Cloud-Anbietern gelöst. Sind spezielle Funktionen der Kontrolle und Speicherung notwendig, so ist eine detaillierte Auseinandersetzung mit der Architektur der Dienste und Aufbau einzelner APIs unumgänglich. Zum Schluss sei bemerkt, dass eine deutliche Steigerung der Datensicherheit bei Cloud-Anbietern über eine vorherige Verschlüsselung der Daten möglich ist. Branchen wie Banken oder Versicherungen, bei denen der Schutz von personenbezogenen Daten an oberster Stelle steht, wird von der Nutzung externer Cloud-Services stets abgeraten.

4.4 Mobile Device Management und Bring Your Own Device

Die Verbreitung mobiler Anwendungen im Unternehmen bedingt das Management dieser Anwendungen und der benötigten mobilen Endgeräte. Mobile Geräte sind größeren Risiken ausgesetzt, als Rechner innerhalb des Unternehmens. Es droht der Verlust, das Kompromittieren durch Schadsoftware und das Ausspionieren von Unternehmensdaten. Da immer mehr mobile Anwendungen Zugriff auf die Netzwerke von Unternehmen besitzen, ist es unbedingt erforderlich, dass auch Endgeräte wie Tablets oder Smartphones mit den Sicherheitsrichtlinien der Unternehmen ausgestattet werden. Durch die steigende Leistungsfähigkeit der Geräte werden sicherheitskritische Daten auf ihnen gespeichert und verarbeitet. Es gilt, diese Daten zu schützen und Vorkehrungen für Verlust oder Diebstahl der Geräte zu treffen.
Das Mobile Device Management sorgt für die einheitliche Umsetzung dieser Richtlinien auf allen mobilen Endgeräten und gestattet die Umsetzung verschiedener Verwaltungsmodelle, wie in Abschnitt 2.2.2 vorgestellt. Durch eine zentrale Verwaltung können Support-Kosten verringert, Sicherheitsrisiken minimiert sowie die Produktivität der Mitarbeiter erhöht werden. Dieser Abschnitt gibt einen Überblick der grundlegenden Funktionalitäten und zeigt Beispiele der Umsetzung auf einzelnen mobilen Plattformen.

4.4.1 Anwendungsfälle

Im Folgenden werden, anhand einiger Fallbeispiele im Umgang mit Geräten in Unternehmen, die grundlegenden Funktionen eines MDM-Systems erläutert und der Lebenszyklus dieser Geräte aufgezeigt.

a. Einrichten eines neuen Geräts

Die Inbetriebnahme eines neuen Geräts im Unternehmen beginnt mit der Inventarisierung des mobilen Endgeräts. Die Inventarisierung befasst sich mit der Zusammenstellung der Geräteeigenschaften. Hierzu zählen die Identifizierung der Geräte, das Erfassen der auf den Geräten installierten Firmware, eine Liste der installierten Applikationen und Daten sowie die zur Verfügung stehende Mobilfunkanbindung. Die einzelnen Parameter werden mit den Sicherheitsrichtlinien des Unternehmens abgeglichen, um Verstöße zu erkennen und entsprechend zu reagieren. (Vgl. KERSTEN & KLETT, 2012)
Damit sind dem MDM-System alle Gerätedaten bekannt. Nun muss das Endgerät konfiguriert werden. Eine Zusammenfassung der im Folgenden beschriebenen MDM-Eigenschaften einzelner BSs zeigt Tabelle 7.
Unter Android werden ein oder mehrere MDM-Apps installiert. Diese fungieren als MDM-Administrator und führen die entsprechenden MDM-Befehle aus. MDM-Befehle sind App-Library-Funktionen, die zusätzliche Android Permissions beinhalten, die von der MDM-App abgefragt werden. Android selbst definiert sehr wenige Befehle und lässt eine Erweiterung durch Drittanbieter zu. (Vgl. SEIBEL, 2014)
Unter iOS wird ein sogenanntes Konfigurations-Profil installiert. Dieses definiert den MDM-Administrator und die auf dem Gerät nutzbaren MDM-Funktionen. Die MDM-Kommandos werden hier vom BS selbst ausgeführt. Der MDM-Client ist bereits in das BS integriert, sodass keine zusätzliche Software erforderlich ist. Apple definiert eine große Anzahl an Kommandos und übernimmt die vollständige Kontrolle über diese. (Vgl. SEIBEL, 2014)

Die Konfiguration des Endgeräts unter Windows Phone 8.1 beginnt mit der Registrierung des Geräts bei dem genutzten MDM-System über Workplace. Workplace dient zur Aufbewahrung der Account-Informationen des MDM-Systems und zur Kommunikation mit diesem. Hier ist der MDM-Client demnach ebenfalls im BS enthalten. Jedes MDM-System kann über Workplace zunächst verschiedene Informationen für die Validierung einholen. Außerdem kann es verlangen eine zusätzliche MDM-App zu installieren. Wurde das Gerät erfolgreich angemeldet, so wird der Account unter Workplace gespeichert und das Windows Phone ist erfolgreich mit dem MDM-System verbunden. Nun sendet das MDM-System ein Provisioning-File, welches die Konfiguration und Richtlinien für das Gerät enthält. (Vgl. MICROSOFT, 2014g) Verglichen mit iOS oder Blackberry, definiert Microsoft eine geringe Menge von MDM-Regeln, die eine Obermenge der unter EAS verfügbaren Regeln darstellen.
Blackberry bringt als einziges System eine integrierte BYOD-Lösung als festen Bestandteil des BS mit sich. Dies ermöglicht die Trennung zwischen privatem und geschäftlichem Bereich. „Diese Variante stellt sowohl für MDM als auch BYOD die beste Lösung dar, da die Trennung von privaten und geschäftlichen Daten bereits im BS integriert ist und somit größte Sicherheit und Verwendbarkeit für die Endbenutzerin bzw. den Endbenutzer geboten werden kann.“ (KREUZHUBER, TEUFL & ZEFFERER, 2014) Für beide Bereiche sind MDM-Regeln definiert. Für den geschäftlichen Bereich (Work Space) existiert eine große Anzahl an Regeln. Die Aktivierung eines Endgeräts wird über den Blackberry Enterprise Service (aktuell BES 12) vorgenommen (vgl. BLACKBERRY, 2015). Mit der Aktivierung wird der geschäftliche Bereich angelegt. Private Daten werden dabei nicht beeinflusst.
Konfigurationsprofile und Apps werden je nach Möglichkeit der Plattform und MDM-Strategie Over-the-Air (OTA) auf das Gerät übertragen oder auf dem Gerät vorkonfiguriert. Wird kein BYOD-Ansatz gefahren, so ist eine Vorkonfiguration vor der Auslieferung des Geräts durch den Hersteller oder das Unternehmen möglich. Hierbei entscheidet sich das Unternehmen für ein Basis-Set an Apps und die einzuhaltenden Sicherheitsrichtlinien. Bei BYOD-Strategien oder ausschließlicher OTA-Übertragung scheidet eine Vorkonfiguration aus.

b. Nutzen eines privaten und eines geschäftlichen Bereichs (BYOD)

Wie in Abschnitt 2.2.4 c beschrieben, ist eine BYOD-Strategie im Unternehmen durch ein einfaches MDM anhand von Konfigurationsprofilen realisierbar. Private und geschäftliche Daten werden gemeinsam verwaltet. Die Kontrolle bzw. ein gewisser Grad der Separation dieser Daten wird über MDM-Regeln durchgesetzt. Hierbei ergibt sich beispielsweise das Problem, dass bei einem Remote-Wipe alle Daten auf dem Gerät gelöscht werden (vgl. KREUZHUBER, TEUFL & ZEFFERER, 2014). Außerdem existieren zahlreiche Sicherheitsrisiken wie zum Beispiel bei der Installation privater Apps, die Zugriff auf geschäftliche Daten, wie Kalendereinträge oder E-Mails bekommen. Es entstehen demnach Probleme, eine Separation zu erreichen. Ein Vorteil ist der reibungslose Arbeitsablauf bei Aufgaben, die beide Bereiche beanspruchen. Ein Wechsel zwischen verschiedenen Bereichen ist in diesem Fall nicht notwendig.
Die strikte Separation der Bereiche durch getrennte Container ist ein weiterer Ansatz. Dieser kann durch die Installation einer Container-Applikation realisiert werden. Blackberry 10 enthält bereits eine vollständig integrierte Lösung. Abbildung 6 zeigt die Architektur der Blackberry BYOD-Lösung.

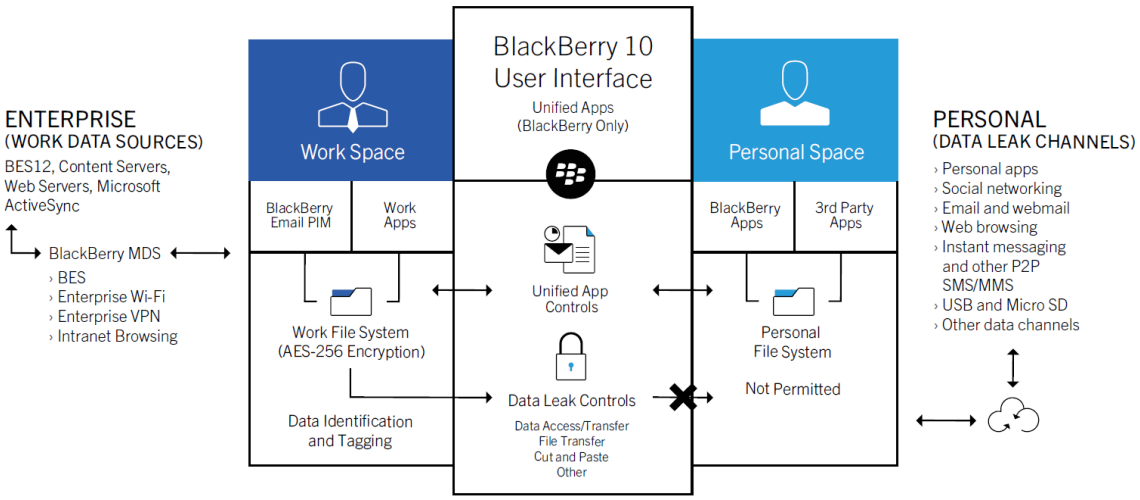

Abbildung 6 Blackberry Balance Architektur (BLACKBERRY, 2014c)

Zu sehen ist die Trennung von Arbeitsbereich und persönlichem Bereich. Zunächst werden zwei getrennte Dateisysteme verwendet. Der geschäftliche Bereich wird stets verschlüsselt. Kommunikationskanäle zum Unternehmen, wie VPN, WLAN und Enterprise Service, sind zunächst ausschließlich dem geschäftlichen Bereich zugänglich. In Ausnahmefällen kann dieses Recht auch dem persönlichen Bereich gewährt werden. Im persönlichen Bereich sind Apps von Drittanbietern oder aus der Blackberry World zugelassen, wie die rechte Seite der Abbildung zeigt. Das Blackberry Interface sorgt für die strikte Trennung der Bereiche. Der Datenaustausch vom Arbeitsbereich in den persönlichen Bereich ist nicht möglich. Das User Interface erlaubt allerdings die gemeinsame Darstellung von Daten aus beiden Bereichen. Außerdem besteht die Möglichkeit von sogenannten Dual- und Unified-Apps. Dual-Apps existieren in beiden Bereichen getrennt voneinander, sind vollständig isoliert und unabhängig. Unified-Apps bieten Einblick in beide Bereiche. Die Kalender-App ist ein Beispiel. (Vgl. BLACKBERRY, 2014c)

c. Installation einer App

Die Installation einer App kann auf verschiedenen Wegen erfolgen. Eine App kann über den In-house App-Store des Unternehmens oder aus dem öffentlichen App-Store der jeweiligen Plattform bereitgestellt werden. Das Installieren, Aktivieren, aber auch das Blockieren und Löschen von Apps erfolgt OTA über das MDM-System.
Unter iOS können Apps aus dem App Store und unternehmensspezifische (In-house) Apps OTA durch den MDM-Server bereitgestellt werden. In-house Apps werden nicht im App Store veröffentlicht und durchlaufen keinen Review-Prozess. (Vgl. APPLE, 2014d)
Windows Phone bietet zunächst die Möglichkeit, Apps aus dem Windows Phone Store zu installieren. Dabei können Listen von Apps erstellt werden, die installiert bzw. nicht installiert werden dürfen. Es besteht die Möglichkeit, dem Nutzer ausschließlich Apps anzuzeigen, die vom MDM-System zugelassen wurden. Des Weiteren ist es möglich, sogenannte Sideloaded Apps zu installieren. Dabei handelt es sich um Apps, die mit dem eigenen Enterprise Zertifikat signiert wurden. Diese können dem MDM-System hinzugefügt und ausgewählten Nutzern bzw. Geräten zur Verfügung gestellt werden. (Vgl. MICROSOFT, 2014g)
Unter Blackberry existieren zwei unterschiedliche Applikationsquellen. Im privaten Bereich können Apps aus dem „Blackberry World“-App-Store installiert werden. Der geschäftliche

Bereich bietet Zugriff auf den „Blackberry World for Work“-Store. Hier werden ausschließlich vom Unternehmen zugelassene Apps angeboten. (Vgl. REIMER, 2013)
Das Zulassen eines öffentlichen App-Stores stellt ein großes Sicherheitsrisiko dar, wie bereits in Abschnitt 4.1.2 beschrieben. Dies erlaubt dem Nutzer, aus einer schlicht endlosen Menge von Applikationen zu wählen, über deren Aktivitäten die Unternehmens-IT keinerlei Kontrolle besitzt. Jede Applikation stellt eine Gefahr für auf den Endgeräten befindliche Daten dar. Apps, welche beispielsweise Daten kopieren, Bild- oder Sprachaufzeichnungen durchführen und Backups in der Cloud ablegen, stellen ein Sicherheitsrisiko dar.
Aus diesem Grund sollte eine Einschränkung auf Apps existieren, die den Sicherheitsrichtlinien des Unternehmens entsprechen. Unternehmen haben die Möglichkeit, Apps vorzuinstallieren, oder dem Mitarbeiter über einen Enterprise Application Store die Auswahl zu überlassen. Blackberry empfiehlt einen Enterprise Application Store anzulegen, anstatt alle Apps auf jedem Endgerät zu installieren. Hier kann je nach Rolle des Mitarbeiters innerhalb der Organisation eine sichere und produktive Auswahl erfolgen. (Vgl. VIJAYAN, MANEA & LAI, 2015)

d. App- und Firmware-Updates

Das Patch Management unterscheidet zwischen den Updates der Firmware und Updates der Apps. Zunächst ist es wichtig, Updates zeitnah zu installieren, da sie meist Sicherheitslücken schließen. Die aktuellen mobilen BS unterstützen meist das Update der Firmware OTA. Updates für Apps werden aus dem App Store bezogen und entweder direkt über diesen oder über das MDM installiert. Updates dürfen die aktuellen Einstellungen und Daten des Endgerätes nicht beeinflussen. Aus diesem Grund sollte stets ein Backup der Gerätedaten erfolgen. (Vgl. KERSTEN & KLETT, 2012)

e. Verlust eines Gerätes und Inbetriebnahme eines Ersatzgerätes

Bei Verlust eines durch das MDM überwachten Gerätes sollte ein Sperren des Geräts OTA erfolgen (Remote Wipe). Dieser sollte nach KERSTEN & KLETT (2012) folgendes beinhalten:

- Aktivieren der Inbetriebnahme Sperre des Geräts (Remote Lock)
- Löschen der Verbindungseinstellungen, z.B. durch Widerruf von Zertifikaten
- Löschen von gespeicherten User-IDs und Passwörtern
- Löschen aller Daten
- Zurücksetzen auf Werkseinstellungen

Selbst wenn das Gerät nicht mit der Zentrale verbunden ist, kann der auf dem Gerät installierte MDM-Agent die oben beschriebenen Funktionen ausführen.
Falls möglich, sollte vor dem Löschen ein Backup der Daten erfolgen. Schließlich folgt ein Restore auf dem neuen Gerät. Die Möglichkeiten sind abhängig von den Backup-Eigenschaften des jeweiligen BS. iOS unterstützt seit Version 5 das Ablegen von Apps, Daten und Einstellungen in der iCloud, siehe hierzu auch Abschnitt 4.3.3. Natürlich sind die damit verbundenen Nachteile eines Cloud-Service und deren Übereinstimmung mit den Sicherheitsrichtlinien des Unternehmens zu beachten.
Beim Verlust des Gerätes ist eine Standardortbestimmung von diesem (Remote Locate) eine nützliche Funktion. Danach kann über weitere Schritte, wie einen Remote Wipe, entschieden werden. Mitarbeiter fürchten jedoch, dass dieses Recht des Unternehmens einen immensen

Eingriff in ihre Privatsphäre bedeutet. Den Unternehmen ist es damit möglich, den Standort des Gerätes und damit des Mitarbeiters zu jedem Zeitpunkt außerhalb sowie während der Arbeitszeiten zu bestimmen. (Vgl. KERSTEN & KLETT, 2012)

f. Übertagen des Gerätes an eine andere Person

Verlässt ein Mitarbeiter das Unternehmen, oder wird ihm aus bestimmten Gründen ein anderes Gerät zugeordnet, so gibt er sein Firmengerät an eine andere Person weiter. Zunächst sollte dabei ein Backup der Daten erfolgen. Schließlich müssen Einstellungen und Daten gelöscht werden.

g. Überwachen des Geräts

Um jedes an das Unternehmensnetzwerk angeschlossene Gerät mit den nötigen Rechten, Konfigurationen und Apps der Sicherheitsrichtlinien des Unternehmens auszustatten, ist eine automatische Provisionierungsfunktion von Vorteil (vgl. ZDNET, 2015). „Das MDM-System muss in der Lage sein, OTA die Richtlinien an die mobilen Endgeräte zu verteilen." (KERSTEN & KLETT, 2012) Ein Beispiel für solche Richtlinien ist ein aktiviertes Passwort mit den Parametern Freischaltungsdauer, tolerierte Fehlversuche, automatische Sperre bei Inaktivität usw. Schließlich sind die Kontrolle, das Auditieren, das Zulassen von Gruppen und Ausnahmen, das Sperren oder Löschen von Geräten sowie das Aufzeichnen und Melden von Verstößen Bereiche des MDM nach KERSTEN & KLETT (2012).
Richtlinien für den Passcode, welche die Komplexität, Länge und Gültigkeitsdauer festlegen, ein Remote Wipe aller Daten des Gerätes sowie die Konfiguration von VPN- und WLAN-Profilen, sind auf allen in Tabelle 7 behandelten BSs gegeben. Demnach sind sich die Hersteller über die Wichtigkeit des Bereichs Zugriffsschutz einig. In den Bereichen Applikationskontrolle, Cloud-Anbindung und Sicherheit existieren jedoch große Unterschiede. Beispielsweise ist es nicht unter allen BSs möglich, die Kamera zu deaktivieren, eine Whitelist von zugelassenen Applikationen zu erstellen, die Annahmen nicht vertrauenswürdiger Zertifikate zu unterbinden, Backups auf Cloud-Services zu unterbinden oder die Dateisystemverschlüsselung verbindlich festzulegen.
In diesem Zusammenhang ist ein elementarer Bereich, ob sich der Benutzer der Kontrolle des MDM entziehen kann. Wie in Tabelle 7 aufgezeigt, ist es unter Android notwendig einen MDM-Agent zu installieren. Diese MDM-App kann ohne Probleme deinstalliert werden, sodass der Administrator die Kontrolle über das Gerät verliert. Unter Windows Phone werden beispielsweise alle Unternehmenskonten und –applikationen gelöscht, sodass ein Zugang zum Unternehmensnetzwerk und zu Unternehmensdaten nicht mehr möglich ist. (Vgl. KREUZHUBER, TEUFL & ZEFFERER, 2014)
Der Zustand des Geräts muss jederzeit ausgelesen werden können, z.B. im Fehlerfall. Dazu gehören nach KERSTEN & KLETT (2012) Verbindungs-, Speicher- und Akkustatus, gespeicherte Daten, belegter Speicher, aufgelaufene Mobilfunkkosten und installierte Apps.
Sind die Geräte Eigentum des Unternehmens, so ist dieses Auslesen unproblematisch. Der Besitzer muss die Richtlinien einhalten und speichert ausschließlich Unternehmensdaten auf dem Gerät. Nach KERSTEN & KLETT müssen Unternehmen mit BYOD-Strategien in jedem Fall entsprechende Vereinbarungen mit dem Besitzer treffen. Bedenklich sei das Ermitteln des Aufenthaltsortes des Geräts oder die Übermittlung von Screenshots. Dafür ist eine Zustimmung des Besitzers einzuholen.

	Android	iOS	Windows Phone	BlackBerry
Integration im BS	MDM-Funktionalität im BS integriert, MDM-Agent notwendig (App)	integrierter MDM-Client	MDM-Funktionalität im BS integriert	MDM/BYOD-Lösung im BS integriert
MDM Administrator	Mehrere MDM-Apps pro Gerät möglich	Ein MDM-Administrator pro Gerät	Ein MDM-Administrator pro Gerät	n/a
Definition MDM-Regeln	durch Google, Erweiterung durch Drittanbieter möglich	durch Apple	durch Microsoft	durch Blackberry
Ausführung von Kommandos	MDM-Kommandos als Funktionen in der Android-Library integriert	MDM-Kommandos als wohldefiniertes Protokoll über APNS (Push-Notifications) und HTTP	MDM-Protokoll basiert auf SyncML 1.2; Robuste Push-Infrastruktur	n/a
Anzahl Kommandos	Wenige Regeln (ca. 10) meist Sicherheitsfeatures, hersteller-spezifische Erweiterungen	Große Anzahl an Regeln (über 100), die sich mit diversen Aspekte des Systems befassen	Geringe Anzahl an Regeln; Obermenge der unter Exchange ActiveSync (EAS) verfügbaren Regeln	Große Anzahl an Regeln für Work Space, Eingeschränkte Regeln für privaten Bereich
Entziehen der MDM-Kontrolle durch Benutzer	Deinstallation des Agenten möglich, damit verbundene Einschränkungen sind vom Agenten festgelegt	Vorherige Spezifikation ob Konfigurations-profil entfernt werden darf	möglich, jedoch Deinstallation aller Unternehmens-applikationen und Unternehmens-kontos	nein
BYOD-Lösung im BS Integriert	nein	nein	nein	ja (Blackberry Balance)

Tabelle 7 Vergleich der MDM Strategien mobiler BSs

h. Betrieb des Gerätes mit Verbindung zum Unternehmensnetzwerk

Mobile Unternehmens-Apps benötigen eine Verbindung zum Unternehmensnetzwerk. Über das MDM kann für eine App das Per-App VPN aktiviert werden. Bereits beim Starten der App verbindet sich diese mit dem Unternehmensnetz und tauscht Unternehmensdaten ausschließlich über eine sichere Verbindung aus.
Außerdem besteht die Möglichkeit des Always-On VPN. Damit erfolgt der gesamte Internetverkehr über einen VPN-Tunnel zur Organisation, sodass diese die vollständige Kontrolle über den Datenverkehr und die Internetnutzung erhält. Siehe für diese 2 Punkte auch Abschnitt 4.1.4.
Das Single Sing-on (SSO) sorgt dafür, dass sich der Nutzer nur einmal zu Beginn mit seinen Unternehmensdaten anmeldet und daraufhin Zugriff auf alle Unternehmens-Apps, ohne ein erneutes Anmelden pro App, besitzt.

Schließlich müssen ständig die aktuellsten Sicherheitsrichtlinien des Unternehmens auf dem Gerät umgesetzt sein und überprüft werden. Dabei wird, wie in f. beschrieben, verfahren.
Während des Betriebs sind ständige Backups durchzuführen, um bei einem Verlust oder Defekt des Geräts einem Datenverlust vorzubeugen. Dies kann, wie unter d. beschrieben, über einen Cloud-Speicher der jeweiligen Plattform oder einen Enterprise eigenen Dienst erfolgen.

i. Fehler und Probleme mit dem Gerät/ der App

Treten Probleme mit dem mobilen Gerät auf, müssen diese, unabhängig vom Aufenthaltsort des Geräts, global bearbeitet werden können. Zu beachten ist, dass der momentane Besitzer des Geräts wahrscheinlich kein IT-Experte ist und ihm ein selbstständiges Beheben nicht zugemutet werden kann. Die Lösung muss OTA durch einen Remotezugriff erfolgen. „Dazu werden Informationen über den Systemstatus des Geräts, Logfiles, Bildschirmfotos, Speicherauszüge etc. benötigt.“ (KERSTEN & KLETT, 2012) Da ein Remote-Desktop-Zugriff von den meisten Betriebssystemversionen nicht standardmäßig zur Verfügung steht, muss eine entsprechende Funktion vom MDM bereitgestellt werden. (Vgl. KERSTEN & KLETT, 2012)

j. Ausmustern des Geräts

Wird ein Gerät ausgemustert, so müssen alle Daten von diesem aus dem MDM-System entfernt werden. Außerdem ist ein sicheres Löschen des Flashspeichers erforderlich. Ein Überschreiben der Daten, wie bei herkömmlichen Festplatten, ist nicht möglich, da der Speichercontroller alle Schreibzugriffe gleichmäßig auf die Speicherzellen verteilt. Beim Löschen werden Daten ausschließlich logisch entfernt, sind jedoch physikalisch noch im Speicher vorhanden.

4.4.2 Schlussfolgerung

Jedes der betrachteten mobilen BSs besitzt einen integrierten MDM-Client. Dies zeigt die Relevanz dieser Systeme und das Bestreben der einzelnen Hersteller, ihr BS für den Unternehmenseinsatz zu rüsten. Der Lebenszyklus zeigt, dass MDM-Systeme zahlreiche Probleme bewältigen müssen. Besonders wichtig sind eine Provisionierungsfunktion, die Überwachung von Systemparametern, der Fernzugriff sowie die Kontrolle über Datenflüsse, Kommunikationskanäle und installierte Apps. Zudem muss das System Eingriffe in die Privatsphäre des Nutzers so gering wie möglich halten, um als benutzerfreundlich zu gelten.
Die mobilen BSs verfolgen unterschiedliche Ansätze der Realisierung, vom klassischen MDM, bis hin zu Containerlösungen. Außerdem existieren große Unterschiede bei der Anzahl verfügbarer Regeln. Android-Nutzern stehen nur wenige Regeln zur Verfügung, die lediglich Basisanforderungen, wie die Definition von Passcode-Richtlinien, abdecken. Hier muss sich das Unternehmen auf Drittanbieterlösungen einlassen, um eine umfassende Kontrolle zu erhalten. iOS bietet umfangreiche Regeln. Im Vergleich dazu sind unter Windows Phone 8.1 wenige Regeln umgesetzt. Diese BSs Verlangen die Nutzung eines Drittanbieter MDM-Systems. Gleiches gilt für die Umsetzung einer BYOD-Lösung. Im Vergleich zu den genannten Systemen, bietet Blackberry eine vollständige Lösung. Sowohl ein MDM, als auch ein BYOD-Ansatz ist fester Bestandteil des BS. Der hauseigene Blackberry Enterprise Service dient als Back-End. Dank einer Vielzahl an umsetzbaren Richtlinien, wird Blackberry 10 im Moment als besonders ausgereiftes System für den Unternehmenseinsatz erkannt.

Schließlich sollte bei der Konfiguration des MDM-Systems stets darauf geachtet werden, dass sich der Nutzer nicht ohne Konsequenzen den MDM-Regeln entziehen kann. Unter allen Systemen ist es möglich, hierfür Vorkehrungen zu treffen.
App-Entwickler für Enterprise-Apps sind, dank moderner MDM-Systeme, gezwungen, ihrer App ausschließlich den Zugang zu den unbedingt benötigten Ressourcen mit den erforderlichen Sicherheitsmaßnahmen zu gestatten. Stuft das MDM die App als sicherheitskritisch ein, so kann es die Installation von Apps unterbinden. Teilweise ist eine Deinstallation aus der Ferne möglich.

4.5 Benutzerinterfaces und Trennung der Anwendungslogik

Die Schnittstelle zum Benutzer ist ein entscheidendes Kriterium für den Erfolg einer App. Die einzelnen Plattformhersteller verfolgen unterschiedliche Strategien, angefangen von den Richtlinien für das App-Design und die Unterstützung von hybriden Apps, über die unterstützten UI-Elemente und Sprachen, in denen die UI definiert wird, bis hin zu den empfohlenen Design-Pattern für einen erweiterbaren und wiederverwendbaren Aufbau einer mobilen Applikation. Dieser Abschnitt gibt einen Überblick für den Entwickler, welche Möglichkeiten für die Umsetzung der UI und für die Strukturierung einer mobilen Applikation auf den einzelnen Plattformen bestehen. Schwerpunkt liegt auf dem Vergleich von Pattern, die eine Trennung von UI-Code und Anwendungslogik ermöglichen, sogenannter „Separated Presentation“-Pattern. Die Art der Trennung von UI und Anwendungslogik gibt Auskunft über die Testbarkeit, Wiederverwendbarkeit und Erweiterbarkeit einer Anwendung.

4.5.1 Übersicht „Separated Presentation“-Pattern

Dieser Abschnitt gibt eine Einführung in die auf heutigen mobilen Plattformen genutzten „Separated Presentation“-Pattern. Die hier aufgeführten Modelle bilden lediglich eine Grundlage. Es existieren zahlreiche Erweiterungen und Abwandlungen von ihnen.

a. Model-View-Controller (MVC)

Das MVC-Pattern ist eines der ersten Pattern, die eine klare Aufteilung der einzelnen Aufgabenbereiche einer Anwendung ermöglichten. Das ursprüngliche Model stammt aus dem Jahr 1979 und wurde im Xerox Palo Alto Research Center (PARC) für Smalltalk Anwendungen entwickelt (siehe REENSKAUG (2007)):

- Model: Es besteht meist aus den Anwendungsdaten, der Logik zum Laden der Daten aus verschiedenen Quellen und zum persistenten Speichern der Daten. Oftmals handelt es sich dabei um die Beschreibung eines Domain Models.
- View: Sie dient dem Zeichnen von UI-Elementen. Sie zeigt die Daten des Models an und updatet sich bei Änderungen des Models.
- Controller: Er dient dem Entgegennehmen von Benutzereingaben und dem Weiterreichen von Änderungen an das Model bzw. dem Updaten der View bei Änderungen des Models. Er wird über Nutzereingaben informiert und entscheidet über die Weiterverarbeitung dieser Information.

Das Pattern wird meist für einzelne UI-Controls eingesetzt. Heutige Controls enthalten die Logik für das Zeichnen der Komponente sowie die Nutzerinteraktion, sodass ein Controller für deren Funktionsweise nicht benötigt wird. Aus diesem Grund ist das Pattern heute weniger relevant. (Vgl. VALK, 2009b)

b. Model-View-Presenter (MVP)

Moderne Designs bestehen aus UIs mit mehreren Controls. Hinzu kommen die UI-Logik und die Daten. Man benötigt ein Pattern, welches diese Bestandteile trennt.

- Model: analog zu MVC
- View: Sie besteht aus mehreren Hierarchisch angeordneten Controls, die zusammen die UI darstellen. Der Nutzer interagiert mit der UI. Auszuführende Logik wird an den Presenter delegiert.
- Presenter: Die View-Logik wird in einem Presenter implementiert. Außerdem übernimmt er das Updaten von View bzw. Model.

Es wird zwischen zwei Varianten unterschieden (vgl. VALK, 2009b):

In der Variante „Passive View“ sind Model und View komplett unabhängig und haben keine Kenntnis voneinander. Durch die lose Kopplung der Komponenten wird ein unabhängiges Testen dieser ermöglicht.
Die Variante „Supervising Controller“ geht von der Kenntnis zwischen View und Model aus. Die View bindet Eigenschaften selbst an das Model. Hier geht die lose Kopplung verloren. Dies erfordert weniger Code in View und Presenter als in der erstgenannten Variante.
Um eine lose Kopplung zu fördern, werden meist Interfaces eingeführt, über welche die Teilkomponenten miteinander kommunizieren. Ein View-Interface ist häufig anzutreffen.

c. Unterschied MVC und MVP

Der Hauptunterschied zwischen dem MCV- und dem MVP-Pattern liegt in der Aufgabe von Controller und Presenter. Die Aufgabe des Controllers in seiner ursprünglichen Form ist im Gegensatz zu einem Presenter nicht die Entkopplung von Model und View. Er dient viel mehr als Mediator zwischen Benutzer und Anwendung. Er reagiert auf Nutzerevents. Die Entkopplung der Applikationslogik von der Präsentation wird hingegen über das Observer-Pattern realisiert. (Vgl. GREER, 2007) Die heute üblichen Views bzw. Widgets der nativen Plattformen übernehmen die Aufgabe des Controllers, indem sie Nutzerevents direkt behandeln. Der Controller wird damit überflüssig und verlagert sich in die View.

d. Presentation Model (PM)

Dieses Pattern ähnelt dem MVP-Pattern. Der Presenter wird durch ein Presentation Model ersetzt. Dieses Presentation Model hält ähnlich dem Presenter die View-Logik. Zusätzlich wird es eingesetzt, um der View die Anzeige des Models zu erleichtern, indem es Modelattribute aggregiert, erweitert, konvertiert. Speziell, wenn das Model nicht unter der Kontrolle des Entwicklers steht, ist dieser Ansatz vorteilhaft. Weitere Vorteile sind die lose Kopplung und damit unabhängige Testbarkeit der Repräsentation der UI von der View. (Vgl. VALK, 2009b)

e. Model-View-ViewModel (MVVM)

Das MVVM-Pattern ersetzt das Presentation Model durch ein ViewModel. Der einzige Unterschied zu dem PM-Pattern ist die Einführung eines Data Bindings, wie es unter Windows Presentation Foundation (WPF) zur Verfügung steht. Das Data Binding ermöglicht es, Eigenschaften und Methoden des ViewModels direkt an die View zu binden. Die View fragt gebundene Eigenschaften ab oder ruft Methoden des ViewModels, sogenannte Commands, auf. Das ViewModel updatet die View durch eine Änderung der gebundenen Attribute. Eine direkte Kommunikation, also die Manipulation von UI-Elementen durch das ViewModel, findet nicht statt. Diese Architektur führt zu einer losen Kopplung der Komponenten. Damit ist die View einfach von Designern ersetzbar, solange die Bindungen erhalten bleiben. Außerdem können die UI-Logik sowie das Model unabhängig voneinander getestet werden. Eine nähere Beschreibung der Umsetzung dieses Patterns unter WPF ist Abschnitt 4.5.3 zu entnehmen. (Vgl. VALK, 2009b)

4.5.2 Möglichkeiten der Plattformen

Die einzelnen mobilen Plattformen bieten unterschiedliche Möglichkeiten, eine App und damit die UI zu entwickeln. Dieser Abschnitt befasst sich mit den von den Plattformherstellern unterstützten Entwicklungsmöglichkeiten. Tabelle 8 zeigt, welche Arten von Apps auf den einzelnen Plattformen entwickelt werden können und mit welchen Technologien die Umsetzung erfolgt.
Die Entwicklung von Web Apps ist unter allen Plattformen möglich, da Web Apps grundsätzlich plattformunabhängig sind. Eine Auflistung erfolgt an dieser Stelle dennoch, da es plattformspezifische bzw. von den Plattformherstellern bereitgestellte Hilfsmittel gibt, die ein natives Erscheinungsbild ermöglichen. Von Microsoft stammt das Open Source Framework Windows Library for JavaScript (WinJS). Dieses stellt UI-Elemente zur Verfügung, die ein identisches Verhalten und Aussehen zu den nativen Ablegern, die unter C#/XAML eingesetzt werden, ermöglichen. Das Framework arbeitet außerdem mit der Windows Runtime zusammen und ermöglicht das Erstellen nativer JS-Apps unter Windows 8.1. Apple stellt für den hauseigenen Safari Browser und dessen mobilen Ableger unter iOS, spezifische Meta-Tags zur Verfügung. Mit diesen kann das Erscheinungsbild der App im Browser, wie die Darstellung auf unterschiedlichen Display-Größen und das Einblenden einer App-Leiste, gesteuert werden. Plattformunabhängig bietet der Extensible Hyper Text Markup Language (XHTML)-Standard verschiedene Tags an, von denen unter den einzelnen mobilen Browsern unterschiedliche Teilmengen unterstützt werden.
Hybride Apps bestehen aus einer nativen und einer Web-Komponente. Die Ausführung der Web-Komponente erfolgt in einer sogenannten WebView, einem internen Browser, der in den nativen Teil eingebettet ist. Die Funktionen dieses Browsers werden über eine plattformspezifische API gesteuert. Die Rendering Engine des internen Browsers, welche hauptsächlich die Performanz der App bestimmt, basiert auf dem plattformspezifischen Browser und ist auf mobile Applikationen spezialisiert. In iOS bis Version 7 beispielsweise, hatte die WebView API keinen Zugriff auf die Performanceoptimierungen, wie den NITRO Just-in-time JS Compiler des hauseigenen Safari Browsers. Aus diesem Grund hatten hybride Apps mit deutlichen Performanceeinbußen zu kämpfen. Seit iOS 8 wird mit der neuen WKWebView API eine dem Safari-Browser äquivalente Leistung erzielt, was die Nutzung

hybrider Apps unter iOS deutlich attraktiver gestaltet. BlackBerry setzt in Sachen hybride App-Entwicklung auf WebWorks, welches auf Apache Cordova basiert. Hier wird ebenfalls HTML5-Code in eine native WebView eingebettet. Apache Cordova als OpenSource Entwicklungsframework unterstützt das Erstellen hybrider Apps unter Android, iOS und Windows Phone. Es wird in Tabelle 8 nicht aufgeführt, da es nicht von den Plattformherstellern selbst zur Verfügung gestellt wird.

	Android 5	iOS 8	Windows Phone 8.1	BlackBerry 10.3
Web Apps	Im Browser	Im Browser, Apple-spezifische Meta-Tags	Im Browser, WinJS Framework	Im Browser
Hybride Apps Web-Content eingebettet in Web View	Web Apps in einer Web View; Web-Komponente: HTML, JS, CSS	Web Content for Safari; Web-Komponente: HTML, JS, CSS	Windows-Runtime-Apps; Web-Komponente: HTML, JS, CSS	Blackberry WebWorks beruht auf Apache Cordova; Komponente: HTML, CSS, JS
API zum Aufrufen der *Web View*	basiert auf Chromium (seit API Level 19); Webkit API	WKWebView API seit iOS 8 (Performance wie Safari)	Klasse WebView	bb/cascades/ WebView
Rendering Engine der Web View basiert auf	Chrome	Safari	Internet Explorer 11 in document mode	Blackberry 10 Browser erfüllt Ringmark Level 1 test
Native Apps	Java, XML	Objective-C, Swift	C# oder C++ mit XAML, HTML/WinJS	C/C++, QML
UI-Framework	Integriert in Android API	Cocoa Touch	Windows Presentation Foundation (WPF), WinJS	Cascades basiert auf Qt

Tabelle 8 Arten von Apps und ihre Umsetzung unter mobilen Plattformen

Die Umsetzung der UI nativer Apps geschieht auf den Plattformen mit Hilfe der nativen Sprachen und Frameworks. Android nutzt XML, unter Windows Phone besteht die Möglichkeit die UI in XAML zu definieren und BlackBerry setzt auf die Qt Modeling Language (QML). Während es sich bei XAML um eine reine Beschreibungssprache für Oberflächen basierend auf XML handelt, ist QML eine deklarative Programmiersprache für die Entwicklung von Benutzeroberflächen. Windows Phone bietet außerdem die Möglichkeit, native Anwendungen in JS mit Hilfe des WinJS-Frameworks zu definieren. Natürlich kann die UI ebenfalls unter den typischen Hochsprachen der Plattform wie Java oder C# umgesetzt werden.

4.5.3 Benutzeroberfläche in Windows Phone 8.1

a. Design

Die Grundlagen des Microsoft Designs (siehe MICROSOFT (2014a)) sind:

- Einfachheit: keine Überflüssigen Elemente, Konzentration auf die Kernfunktionen
- Internationaler Typografie-Stil: Klarheit, Lesbarkeit und Objektivität
- gutes Bewegungsdesign: Vermittlung von Leben und Eleganz

Typische Windows Phone Apps wirken meist spartanisch und übersichtlich. Die Konzentration auf die Kernfunktionen kann sehr zur Übersichtlichkeit beitragen, solange nicht zu viele Einsparungen durchgeführt werden. Dies führt oftmals zu größerem Aufwand für den Nutzer, um bestimmte Aktionen zu finden oder um zu ihnen zu gelangen.

b. Aufbau von Views

Die grundlegenden UI-Komponenten zeigt Abbildung 7. Es wird deutlich, welche Komponenten, welche Elemente enthalten können. Auf der obersten Ebene steht die Page oder das UserControl. In beiden Elementen können die Standard-Controls eingebettet werden. Pages haben den Vorteil, dass zwischen ihnen navigiert werden kann. UserControls unterstützen die leichte Wiederverwendbarkeit, da sie in Form von Templates mehrfach innerhalb von Pages eingesetzt werden können. Bei der Programmierung in C#/XAML werden die Views als Pages oder UserControls in Form von partiellen Klassen in XAML definiert. Diese verschmelzen mit den zugehörigen Code-Behind Dateien (C#) zu einer gemeinsamen Klasse. XAML ist eine von Microsoft entwickelte deklarative Sprache, mit welcher der sichtbare Teil der UI definiert werden kann. Natürlich ist die Definition ebenso in C# möglich. Die zugehörige Code-Behind-Datei dient der Behandlung von Events und dem Manipulieren von Objekten, die in XAML definiert wurden.
Innerhalb von Pages bzw. UserControls können Hub-, Pivot- und Flyout-Elemente definiert werden. Diese Elemente sind wie folgt zu verstehen:

- Hub: Unterteilt eine Page in mehrere Seiten. Für den Nutzer entsteht der Eindruck einer großen zusammengehörigen Einheit.
- Pivot: Das Registerkartenmuster wird für Benutzeroberflächen mit mehreren Seiten (Views) verwendet, zwischen denen ein Benutzer oft wechselt.
- Flyout: Eingeschobenes Menü, welches an Komponenten angeheftet und eingeblendet wird, sobald ein definiertes Event stattfindet.

Innerhalb dieser Komponenten werden Layouts wie StackPanel und Grid angeordnet. Sie können allerdings auch ohne Hub, Pivot oder Flyout innerhalb einer Page oder UserControl-Komponente eingebettet werden. Sie werden für folgende Zwecke genutzt:

- StackPanel: Enthaltene Komponenten werden als Auflistung untereinander dargestellt.
- Grid: Enthaltene Komponenten werden an einem Raster ausgerichtet.

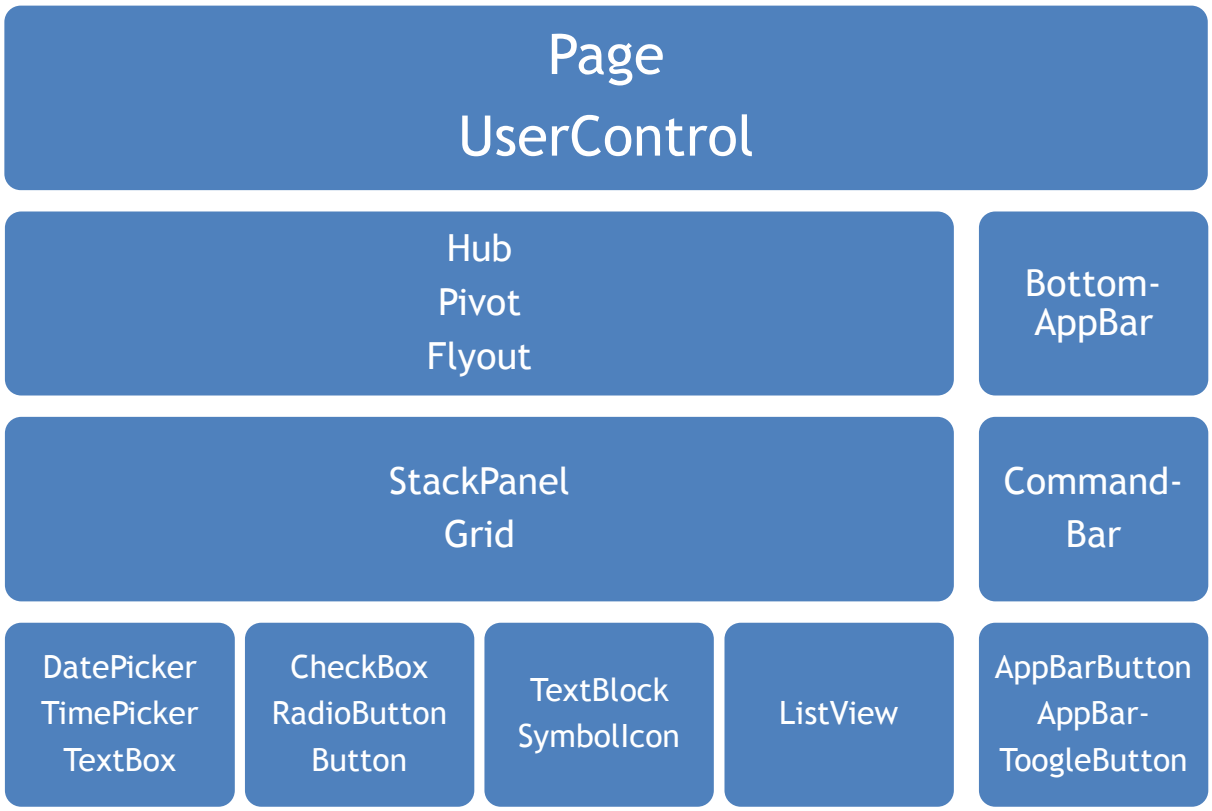

Abbildung 7 Übersicht grundlegender UI-Komponenten der Windows Phone 8.1 Plattform

In den StackPanel- und Grid-Layout-Komponenten können beliebige Elemente der untersten Ebene wie DatePicker zur Auswahl eines Datums, Buttons oder TextBlöcke zur Anzeige von Text eingebettet werden. Eine ListView zeigt eine Liste von Objekten des gleichen Typs an, wobei die Elemente entsprechend einem Template dargestellt werden. Zusätzlich existiert unter Windows Phone die Möglichkeit, eine AppBar am unteren Bildschirmrand einzublenden. Diese zeigt AppBarButtons in Form von Icons als sogenannte PrimaryCommands an. Außerdem können beliebige AppBarButtons als SecondaryCommands definiert werden, die beim Aufklappen der AppBar in Form ihrer Labels angezeigt werden.

c. Bindings

Unter Windows Phone existiert ein Binding-Mechanismus, welcher Attribute direkt an UI-Komponenten bindet. Über sogenannte OneWay-Bindings findet eine Bindung vom Attribut zur UI-Komponente statt. Ein TwoWay-Binding ermöglicht zusätzlich ein Update des Attributwertes bei Änderungen in der UI durchzuführen. Das XAML-Data Binding wird über eine Art Reflection-Mechanismus ermöglicht.
Im Folgenden wird die Realisierung eines OneWay-Binding anhand des Beispielprojektes „ProjectsExample“ erläutert. Das Beispiel besteht aus einer JavaScript Object Notation (JSON)-Datei als Datenquelle, die Projekte mit Titel und Beschreibung enthält. Das Model definiert die Klasse „Project.cs“, die Instanzen der eingelesenen Projekte zur Verfügung stellt. Die View besteht aus der XAML-Datei „ProjectsPage.xaml“ sowie der Code-Behind-Datei „ProjectsPage.cs“. Listing 9 zeigt ein Attribut „Title“ der „Project“-Modelklasse:

```
public class Project
{
    …
    private string title;
    public string Title {
        get
        {
            return this.title;
        }
        set{
            if(value != this.title)
            {
                this.title = value;
            }
        }
    }
    …
}
```

Listing 9 Beispieldefinition eines Modelattributes unter Windows Phone in C#

Die View besteht aus einem Page-Control, welches innerhalb einer StackPanel-Komponente eine TextBox namens „TitelTextBox“ definiert. Die TextBox soll an das „Title“-Attribut eines Projektes gebunden werden. Es handelt sich um ein OneWay-Binding, d.h. die TextBox wird mit dem Inhalt des „Title“-Attributes initialisiert. Änderungen des Wertes werden nicht in das Model zurückgeschrieben. Listing 10 zeigt die Definition in der zugehörigen XAML-Datei.

```
<StackPanel>
    <TextBox
        Name="TitelTextBox"
        Header="Titel:"
        Text="{Binding Title, Mode=OneWay}"/>
</StackPanel>
```

Listing 10 Beispiel der Definition eines OneWay-Bindings in einer XAML-Datei unter Windows Phone

Um das Binding zu realisieren, wird der Datenkontext des übergeordneten Page-Controls auf ein Projektobjekt gesetzt. Dies geschieht in der zugeordneten Code-Behind-Datei „ProjectsPage.cs“. Diese definiert die Methode „NavigationHelper_LoadState“. Sie wird aufgerufen, sobald die zugehörige Page geöffnet wird. Die Methode liest zunächst ein Beispielprojekt aus der Datenquelle aus. Daraufhin wird der Datenkontext der Page innerhalb der XAML-Datei auf dieses Projekt festgelegt. Alle folgenden Bindings in der XAML-Datei beziehen sich auf diesen Kontext. Die Implementierung der Methode zeigt Listing 11.

```
private async void NavigationHelper_LoadState(object sender, LoadStateEventArgs e)
{
   var sampleProject = await SampleDataSource.GetProjectAsync("Project-1");
   this.DataContext = sampleProject;
}
```

Listing 11 Setzen des Datenkontextes einer Page im Code-Behind zur Realisierung von Bindings mit der zugehörigen XAML-Datei

Durch das Binding des Attributes „Titel“ wird sein Wert in der TextBox angezeigt. Wird der Binding-Parameter Mode auf „TwoWay“ gesetzt, so werden Änderungen in der TextBox direkt in das Model durchgeschrieben. Der Wert des Attributes „Title“ des gebundenen Project-Objektes würde sich entsprechend ändern.

Ein Update der UI, sobald sich der Modelwert ändert, existiert noch nicht. Angenommen ein TextBlock-Control, welches ausschließlich zur Anzeige des Titels dient, wird an das Titelattribut gebunden. Dieses würde sich bei Änderungen des Titels nicht updaten. Der hierfür erforderliche Observer-Mechanismus wird durch das INotifyPropertyChanged-Interface realisiert. Dieses wird im nächsten Abschnitt erläutert.

d. Umsetzung des MVVM-Patterns

Das MVVM-Pattern trägt dazu bei, UI und App-Logik zu trennen. Dies führt zu einer besseren Wiederverwendbarkeit der Logik, z.B. wenn für mehrere Plattformen wie Windows Phone 8.1 und Windows 8.1 entwickelt werden soll. Außerdem fördert es eine unabhängige Testbarkeit von View und Model. Microsoft empfiehlt dieses Pattern für die Programmierung unter Windows Phone 8.1. (Vgl. MICROSOFT, 2014b) Die Umsetzung des Patterns ist auf Abbildung 8 zu sehen.

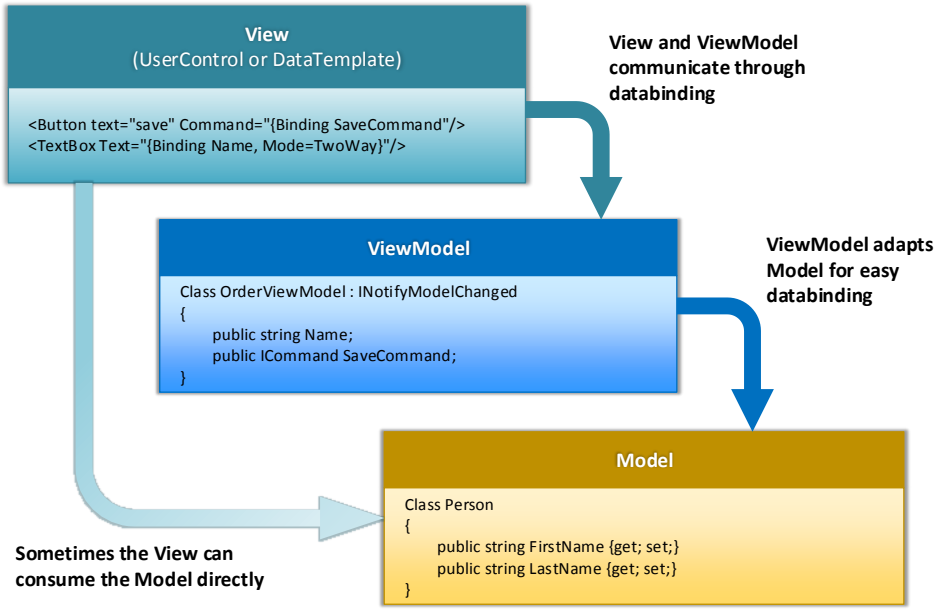

Abbildung 8 Typische Realisierung des MVVM-Patterns unter Windows Phone 8.1, (vgl. VALK, 2009a)

Die View ist über Bindings mit dem ViewModel verbunden. Sie liest oder updatet die Eigenschaften des ViewModels. Dies wird über ein TwoWay-Binding des Name-Attributes ermöglicht. Das ViewModel dient als Adapter bzw. Abstraktionsschicht zwischen View und Model. Es hält den Status der View und implementiert die View-Logik. Es stellt die Daten des Models in einer Weise zur Verfügung, in der sie möglichst einfach von der View verarbeitet werden können. Abbildung 8 zeigt die Aggregation der Modelattribute „FirstName“ und „LastName“ zu einem Attribut „Name“ im ViewModel.
Nun ist es erforderlich, dass die View bei Änderungen des Models informiert und entsprechend angepasst wird. Dies realisiert das INotifyPropertyChanged-Interface. Es wird im ViewModel implementiert und besitzt die Methode „NotifyPropertyChanged“. Sobald sich ein Modelwert ändert, wird dem Interface über den Aufruf der „NotifyPropertyChanged“-Methode das betreffende Modelattribut mitgeteilt. Daraufhin benachrichtigt dieses alle UI-Komponenten mit einem Binding zu dem entsprechenden Attribut und updatet die View.
Wie das Beispiel aus Abschnitt c. der „Project“-Klasse angepasst werden muss, um ein Updaten der UI bei einer Titeländerung im Model zu gewährleisten, zeigt Listing 12. Neben der

Implementierung des INotifyPropertyChanged-Interfaces ist in der Setter-Methode der Aufruf der „NotifyPropertyChanged“-Methode notwendig. Sobald das „LostFocus“-Ereignis der TextBox eintritt, erfährt das Model ein Update (Aufruf Setter) und die UI wird benachrichtigt.

```
public class Project : INotifyPropertyChanged
{
    …
    private string title;
    public string Title {
        get
        {
               return this.title;
        }
        set{
            if(value != this.title)
            {
                this.title = value;
                NotifyPropertyChanged();
            }
        }
    }
    …
}
```

Listing 12 Einsatz des INotifyPropertyChanged-Interfaces zum Updaten der UI bei Änderung des Wertes eines Modelattributes

Das ICommand-Interface ist ein weiteres, das MVVM-Pattern unterstützende Werkzeug. Das Verarbeiten von Events, wie z.B. einem Button-Click, kann in Windows Phone mit Hilfe eines EventHandlers realisiert werden. Dieser muss allerdings in dem, der View zugeordneten, Code-Behind-File implementiert werden. Da nach dem MVVM-Pattern jedoch View und App-Logik getrennt werden sollten, wurde ein Mechanismus entwickelt, der die Implementierung im ViewModel ermöglicht (vgl. VALK, 2009a). Zunächst wird im ViewModel das Kommando, siehe Abbildung 8 „SaveCommand“, als Attribut definiert. Dieses gibt ein Command-Objekt zurück. Die View kann nun an dieses Attribut binden und die Operation aus dem ViewModel heraus aufrufen. Das Interface besitzt die zwei Methoden „Execute“ und „CanExecute“. In „Execute“ wird die Logik des Kommandos, wie in Abbildung 8 das Speichern, implementiert. „CanExecute“ gibt an, ob das Kommando im aktuellen Kontext ausgeführt werden soll. Bei einem Button-Click wird zunächst die „CanExecute“- und danach die „Execute“-Methode des gebundenen Kommandos „SaveCommand“ im ViewModel ausgeführt. Liefert „CanExecute“ den Wert „false“, so wird das Attribut „Enabled“ des zugehörigen Control-Elements auf „disabled“ gesetzt. Beispielsweise wird ein Button daraufhin ausgegraut dargestellt.

4.5.4 Benutzeroberfläche in Android

a. Design

Android definiert seine Designmaßstäbe in der Version 5 neu. Mit dem sogenannten Material Design legt Google mehr Wert auf Animation, Interaktion und eine Anordnung von Elementen mit natürlich wirkender Tiefendarstellung. Bei der Interaktion mit Elementen wie Buttons, können verschiedene Animationen definiert werden. Elemente können durch einen zusätzlichen Z-Index in der Tiefe angeordnet werden und bekommen einen entsprechenden Schatten spendiert. Touch-Gesten erzeugen ein natürlich wirkendes Feedback bei Annäherung und

Entfernung des Fingers. Außerdem kümmert sich Google, um die einfachere Gestaltung von Oberflächen für verschiedene Displayauflösungen und führt neue UI-Elemente ein.

b. Aufbau von Views

Unter Android gibt es zwei Wege die UI festzulegen. Zum einen können UI-Elemente in XML definiert werden. Zum anderen besteht die Möglichkeit, Elemente als Objekte direkt im Code zu erzeugen. Grundlage der UI bilden die View-Klassen und Subklassen wie ViewGroup. Interaktive UI-Komponenten, wie Buttons oder Textfelder, erben von View. ListView oder RecyclerListView sind Unterklasssen von ViewGroup und definieren ein Layout. Sie ähneln den ListView-Elementen unter Windows Phone. Layouts halten eine Menge anderer Views. Jedes ListView-Element besitzt einen LayoutManager, der für die Position der einzelnen Views innerhalb der Liste verantwortlich ist. Zu jedem LayoutManager wird ein Adapter definiert. Dieser nimmt die Daten des Models entgegen und gibt sie zur Darstellung an den LayoutManager weiter. Weitere verfügbare Layouts sind:

- LinearLayout: Ordnet alle Elemente in einer Richtung entweder horizontal oder vertikal an.
- RelativeLayout: Zeigt Kindelemente relativ zu ihren Schwesterelementen oder Elternelementen an.
- GridView: Zeigt Elemente in einem zweidimensionalen Grid an.

c. Activities

Zu den grundlegenden Konzepten der App-Entwicklung unter Android gehören die Activities. Sie stellen die UI-Komponenten bereit, mit denen der Nutzer interagiert. Sie fördern die Umsetzung des MVC-Patterns und stellen die Wiederverwendbarkeit von Views und deren Controllern sicher. Die folgenden Beschreibungen zu Activities lehnen sich an GOOGLE (2015a) an.
Eine Activity stellt einer Anwendung eine Funktionalität zur Verfügung. Für jede Activity steht genau ein Window bereit, in welchem die UI definiert wird. Über EventHandler nimmt die Activity Nutzereingaben entgegen und führt entsprechende Logik aus. Jede App besitzt eine „main“-Activity, welche beim Start der App geladen wird und somit als Startseite dient. Von einer Activity können beliebige weitere Activities gestartet werden. Nach ihrer Aufrufreihenfolge werden Acitivies in einem „Back Stack“ gehalten. Beim Start wird eine Activity auf diesen Stack gepushed. Das oberste Element des Stacks wird angezeigt. Somit ist eine Rückwärtsnavigation z.B. durch drücken des Back-Buttons ohne Probleme möglich.
Jede Activity besitzt einen Lebenszyklus und kann drei Zustände annehmen. Sie ist „Resumed“ oder „Running“, wenn sie sich im Vordergrund und im Fokus des Benutzers befindet. Sie befindet sich im Zustand „Paused“, wenn sie im Hintergrund oder nur teilweise sichtbar ist und der Nutzerfokus auf einer anderen Activity liegt. Dabei ist sie vollständig im Hauptspeicher vorhanden, dem Window Manager zugeordnet und kann bei Ressourcenknappheit vom Betriebsystem beendet werden. Eine Activity ist im Zustand „Stopped“, wenn sie sich komplett im Hintergrund befindet. Das heißt, sie wird vorerst im Hauptspeicher gehalten, jedoch ist sie dem Window Manager nicht mehr zugeordnet und kann jederzeit beendet werden. Eine Activity implementiert verschiedene Callback-Methoden, die je nach dem Zustand, den sie im

Begriff ist anzunehmen, aufgerufen werden. Hier können zustandsspezifische Aktionen, wie das Sichern von Daten, ausgeführt werden.
Eine Activity kann von verschiedenen Anwendungen aus aufgerufen werden. Sie reagiert dabei auf verschiedene Intents und kann demnach über mehrere Anwendungen hinweg wiederverwendet werden. Intents sind Beschreibungen von auszuführenden Operationen.

d. Umsetzung des MVP-Patterns

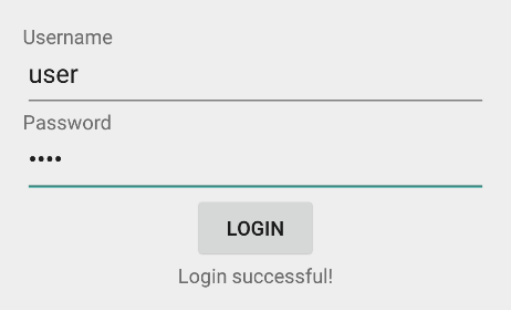

Abbildung 9 UI eines Logins auf der Android Plattform

Im Folgenden wird die Umsetzung des MVP-Patterns auf der Android Plattform anhand eines Beispielprojektes erläutert. Das Beispiel demonstriert einen Login-Prozess. Der Nutzer gibt Benutzername sowie Passwort ein und bekommt eine Rückantwort auf die Korrektheit der Eingabe. Die UI zeigt Abbildung 9. Die Nutzerdaten sind hartcodiert im Datenmodel abgelegt, da der Fokus auf der Funktionsweise des Patterns, jedoch nicht auf der Demonstration der Business Logik liegt. Abbildung 10 zeigt das zugehörige Klassendiagramm.

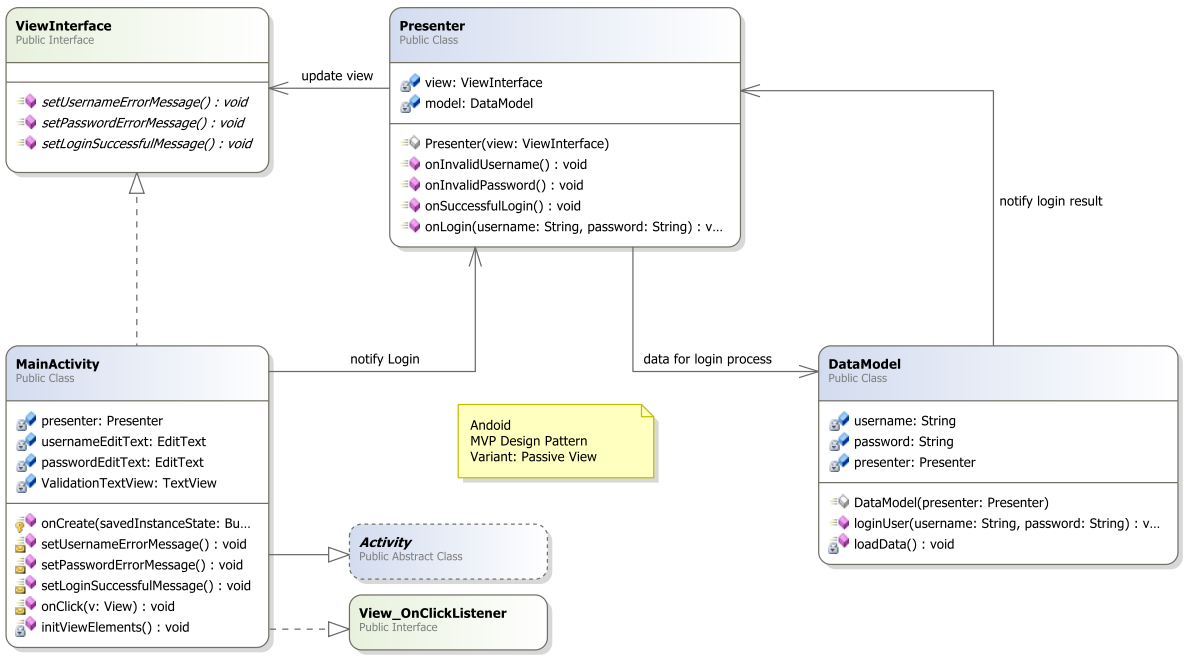

Abbildung 10 Klassendiagramm eines Login-Prozesses auf der Android-Plattform; Es wird die beispielhafte Umsetzung des MVP-Patterns aufgezeigt.

Die View besteht aus dem ViewInterface und der implementierenden Klasse MainActivity, welche eine Android-Activity darstellt. Der Presenter wird in der „onCreate“-Methode der Activity initialisiert. Er kommuniziert über das ViewInterface mit der View, sodass eine lose

Kopplung und damit leichte Testbarkeit und Austauschbarkeit der View gegeben ist. Die Realisierung der UI kann ohne Probleme durch andere Views umgesetzt werden. Der Login-Prozess wird durch das Klicken des Login-Buttons eingeleitet. Über den „OnClickListener“, zuvor auf der Button-View registriert, wird die View über das Ereignis informiert. Der „onClick“-Handler benachrichtigt den Presenter. Dieser Presenter leitet die Anfrage und damit die Nutzerdaten über die „onLogin“-Methode an das Model weiter, welches die Businesslogik enthält und die Anmeldung des Nutzers durchführt.
Das Model benachrichtigt den Presenter über das Resultat des Logins, indem es die entsprechenden Methoden auf diesem aufruft, siehe Presenter Abbildung 10. Der Presenter updatet schließlich die View, indem er die entsprechenden ViewInterface-Methoden, je nach Status des Logins, nutzt.
Es besteht keine direkte Verbindung zwischen View und Model. Ein Wissen über die Existenz der jeweils anderen Komponente ist nicht gegeben, sodass es sich hierbei um die Passive View Variante des MVP-Patterns handelt. Es ist möglich, weitere Interfaces für Presenter sowie Model hinzuzufügen. Dies fördert die Unabhängigkeit der Komponenten weiter. Die Rolle des Presenters zur Synchronisation bzw. als Mediator zwischen View und Model wird deutlich. Er interpretiert das ButtonClick-Ereignis der View als die Anfrage eines Logins des Nutzers und benachrichtigt das Model, welches die entsprechende Businesslogik ausführt.

4.5.5 Kurzübersicht des MVC-Patterns unter iOS

iOS nutzt das UI-Framework Cocoa Touch für die Gestaltung von Benutzeroberflächen. Das Framework bietet eine umfangreiche Unterstützung des MVC-Patterns. Wie bereits in Abschnitt 4.5.1 erwähnt, gibt es viele Erweiterungen und Abwandlungen dieses Patterns. Auch iOS nutzt nicht das ursprüngliche Modell, sondern empfiehlt ein dem MVP- und MVVM-Pattern ähnliches Modell. Apple stellt ebenfalls fest „Design-wise, it's best to keep model and view objects separate from each other, because that enhances their reusability.“ (APPLE, 2012a) Das entstehende Modell ist rein optisch dasselbe wie auf Abbildung 10 zu sehen. Apple bleibt allerdings bei der Bezeichnung Controller. Außerdem wird betont, dass dieses Pattern eine Ansammlung von Design-Pattern darstellt, wie Mediator, Strategy und Command-Pattern. Aus platzgründen wird an dieser Stelle nicht weiter auf diese grundlegenden Pattern eingegangen. Eine Beschreibung ist unter APPLE (2012a) zu finden. iOS unterstützt sowohl DataBindings, ähnlich Windows Phone, als auch Notifications. Somit können Modelattribut direkt an die UI gebunden werden, oder die View erhält Benachrichtigungen über die Änderung von Modelattributen. iOS stellt außerdem einheitliche Controller-Superklassen (Vermittelnde und Koordinierende Controller) zur Verfügung, von denen geerbt werden kann. Damit wird das Erstellen von Controllern, der Umgang mit Datenbindungen und das Nutzen des Patterns vereinheitlicht und erleichtert.

4.5.6 Schlussfolgerung

Nach dem Aufzeigen von Entwicklungsmöglichkeiten der UI sowie einzelner Design-Pattern zur Separierung von UI und Businesslogik auf ausgewählten mobilen Plattformen, lässt sich folgendes schließen.
Moderne UI-Frameworks bieten komfortable Möglichkeiten für Entwickler, mit Hilfe von Layouts und Views eine ausgereifte Benutzerschnittstelle für mobile Businessanwendungen zu definieren. Beispielhaft wurde dies an Windows Phone und Android gezeigt.

Dabei bieten die einzelnen Plattformen und UI-Frameworks unterschiedlichste Möglichkeiten, eine Anwendung sinnvoll zu strukturieren. Durchgesetzt hat sich die Trennung von View und Anwendungslogik. Grundlage moderner Pattern zur Separation von UI und Businesslogik bildet das MVC-Pattern. Auf dessen Grundlage geht die Entwicklung hin zu Pattern, die View und Datenmodell über einheitliche Schnittstellen separieren und eine dritte Komponente hinzufügen, die für deren Verbindung und Synchronisation verantwortlich ist. Diese wird als Controller, Presenter oder ViewModel bezeichnet. Innerhalb der einzelnen Ausprägungen kann diese Komponente geringfügig abweichende Aufgaben übernehmen.
Grundsätzlich kann jedes Pattern auf allen mobilen Plattformen umgesetzt werden. Die Entscheidung, welches Pattern genutzt wird, sollte stets von der jeweiligen Problemstellung ausgehen und nicht das Pattern als Selbstzweck haben. Ein Pattern sollte stets an die Problemstellung angepasst werden, sodass die einzelnen benötigten Komponenten durchaus variabel sind und deren hier geschilderter Aufbau lediglich als Leitlinie zu verstehen ist.
Außerdem ist anzumerken, dass einzelne Pattern auf verschiedenen Plattformen unterschiedlich gut unterstützt werden. WPF und Cocoa Touch bieten beispielsweise DataBindings, sodass die Bindung von Attributen von der UI aus direkt möglich ist. Windows Phone bietet Interfaces, die eine Benachrichtigung der View oder das Definieren von Commands außerhalb der View deutlich vereinfachen. Hier bietet sich ein Pattern wie MVVM an. Das MVP-Pattern ist für alle Plattformen geeignet und wird sowohl für Android- als auch für iOS-Anwendungen empfohlen. iOS schlägt hier bereits die Entwicklungsrichtung hin zu vereinheitlichten Controller-Klassen ein und unterstützt verschiedene Benachrichtigungsmechanismen. Dies führt zu weniger Fehlern bei der Benutzung von Pattern durch den Entwickler und bietet eine sehr gute Unterstützung für die unterschiedlichen Ausprägungen der „Separated Presentation“-Pattern.

5 Aufstellung eines Kriterienkatalogs und Bewertungssystems für mobile Enterprise Applikationen

Anhand der gezeigten Kriterien wird in diesem Abschnitt ein Kriterienkatalog für die Bewertung sowohl der mobilen Plattformen als auch einer mobilen Enterprise Applikation erstellt. Es werden die Plattformen Android 5 mit „Android for Work“, iOS 8, Windows Phone 8.1 und BlackBerry 10 behandelt.
Zusätzlich wird Android 5 mit Samsungs MDM-Lösung Knox bewertet. Android bietet die Möglichkeit, MDM-Funktionalitäten durch Drittanbieter zu erweitern. Durch die fehlenden Funktionen in der Basiskonfiguration von Android 5 ist eine deutlich schlechtere Bewertung, im Vergleich zu den anderen System, die einen fest vorgeschriebenen Regelsatz gebrauchen, zu erwarten. Um die Erweiterungsfähigkeit mit einzubeziehen, wird Samsung Knox als Beispiel für eine Erweiterung der Android-Plattform herangezogen.
Im Folgenden wird für Android 5 mit „Android for Work“ lediglich die Bezeichnung Android 5 gebraucht. Android 5 mit Samsung Knox wird durch die Bezeichnung „Samsung Knox“ abgekürzt.
Zunächst erfolgt eine Begründung der Aufteilung der Kriterien in verschiedene Phasen. Daraufhin wird ein Bewertungssystem eingeführt. Schließlich werden die einzelnen Kriterien vorgestellt und die mobilen BSs bewertet.

5.1 Erkenntnisse anhand der behandelten Kriterien

Aus den beschriebenen Kriterien ergibt sich, dass eine Bewertung der Tauglichkeit einer mobilen Applikation ausschließlich durch das Betrachten der mobilen Anwendung nicht ausreichend ist.
Beispielsweise sind Sicherheitsaspekte, Geräte-Administrierte Rechte und die Dateisystemverschlüsselung feste Bestanteile des zugrundeliegenden mobilen BS. Dies trifft ebenfalls auf MDM-Fähigkeiten und die Möglichkeiten zur Gestaltung der UI zu. Für die Tauglichkeit im Enterprise Umfeld sind diese Kriterien von höchster Bedeutung. Der Entwickler kann diese Aspekte ausschließlich durch die Plattformentscheidung in der Entwurfsphase, also vor dem Beginn der Entwicklung, beeinflussen. Während der Entwicklung der Applikation sind diese Kriterien fixiert. Lediglich das spätere Aktivieren/Deaktivieren von Funktionalitäten durch einen Administrator ist möglich.
Vielmehr muss das Gesamtsystem, bestehend aus der zugrundeliegenden mobilen Plattform, der mobilen Applikation, als auch Aspekte der Umgebung, in welcher die Applikation verwendet wird, mit einbezogen werden. Die Umgebung entspricht im Enterprise Einsatz dem Unternehmen. Aspekte der Umgebung sind beispielweise, in wie weit eine Aufklärung der Mitarbeiter über Sicherheitsmaßnahmen erfolgt und ob ein Sicherheits- und Risikoanalyse durchgeführt wird.
Durch die Betrachtung der Teilsysteme ergibt sich ein Gesamtbild, welches durch seine Bewertung ein aussagekräftiges Ergebnis der Tauglichkeit einer mobilen Applikation für den Enterprise Einsatz liefert.
Aus diesen Gründen werden zum einen Kriterien der Vorentwicklungsphase aufgestellt, welche eine Bewertung der mobilen BSs ermöglichen. Zum anderen werden Kriterien der Entwicklungs- und Einsatzphase aufgezeigt, welche die Bewertung der mobilen Applikation sowie der aktuellen Konfiguration der Umgebung dieser erlauben.

5.2 Definition des Bewertungssystems

Das hier vorgestellte Bewertungssystem wird für die Bewertung der vorgestellten Phasen genutzt. Der Kriterienkatalog für die Bewertung der einzelnen BSs und der mobilen Applikation ist den Anhängen C und D zu entnehmen. Er ist jeweils aufgeteilt in Teilaspekte wie Sicherheit, MDM, Benutzerinterface und die für diese Aspekte geltenden Kriterien.

Definition 0. Sei *n* die Anzahl der Teilaspekte und q_i die Menge der Kriterien des Teilaspektes i mit $i \in \{1, \dots, n\}$ Index über alle Teilaspekte.

Definition 1. Sei $\Omega \subset \mathbb{R}^{>0}$ ein Raum, der alle möglichen Gewichtungen eines Kriteriums enthält, wie folgt definiert:

$$\Omega := \{0{,}5; 1; 2\} \tag{5.1}$$

Das Bewertungssystem ordnet zunächst jedem Kriterium ein Gewicht des Raumes Ω zu. Einigen Kriterien, wie beispielsweise dem nachträglichen Abwählen/Zustimmen einzelner Berechtigungen, wird ein höheres Gewicht zugeordnet, da sie als besonders relevant betrachtet werden.

Definition 2. Sei $\Pi \subset \mathbb{R}^{\geq 0}$ ein Raum, der den Erfülltheitsgrad eines Kriteriums angibt (Zustandsraum), wie folgt definiert:

$$\Pi := \{0; 0{,}5; 1\} \tag{5.2}$$

Nun wird geprüft, in wie weit ein Kriterium erfüllt ist. Dafür wird eine dreistufige Skala eingeführt und jedem Kriterium ein Zustand zugeordnet. Jeder Zustand wird einem Element des Raumes Π zugeordnet und durch eine Farbe gekennzeichnet. Es existieren die Zustände „vollständig erfüllt“ = 1 (grün), „teilweise erfüllt“ = 0,5 (gelb) und „nicht erfüllt“ = 0 (rot).

Definition 3. Sei $G_i \in \Pi^{1 \times q_i}$ die Matrix aller Gewichte der Kriterien des Teilaspektes *i* mit

$$G_i = \{g_{11} \quad \cdots \quad g_{1q_i}\} \tag{5.3}$$

sowie $S_i \in \Omega^{q_i \times 1}$ die Matrix aller Zustände der Kriterien des Teilaspektes *i* mit

$$S_i = \begin{Bmatrix} s_{11} \\ \vdots \\ s_{q_i 1} \end{Bmatrix} \tag{5.4}$$

Definition 4. Sei $f: \Pi \times \Omega \to [0,1]$ Funktion zur Bewertung eines Teilaspektes gemäß

$$f(G_i, S_i) := \frac{G_i \times S_i}{\sum_{j=1}^{q_i} S_j} \tag{5.5}$$

mit $j = \{1, \dots, q_i\}$ Index über alle Kriterien des Teilaspektes *i* definiert.

Die Bewertung eines Teilaspektes geschieht mit Hilfe der Gewichtsmatrix und der Zustandsmatrix der einzelnen Kriterien durch die Funktion f. Für die einzelnen Teilaspekte, wie Sicherheit, MDM und UI, werden jeweils die einzelnen Bewertungen der Kriterien, über das Kreuzprodukt der Matrizen G_i und S_i, aufsummiert und durch die Gesamtzahl erreichbarer Bewertungspunkte geteilt. Dadurch ergibt sich der relative Anteil erreichter Bewertungspunkte des jeweiligen Teilaspektes.
Für die Berechnung erreichbarer Bewertungspunkte eines Teilaspektes ist Folgendes zu beachten. Für Kriterien, die mit „n.a.“ gekennzeichnet sind, konnte keine verwertbare

Information für eine Einordnung in die oben genannten Zustände gefunden werden. Diese werden nicht in die Gesamtzahl erreichbarer Bewertungspunkte aufgenommen. Für die Berechnung wird dem Kriterium das Element 0 des Zustandsraums Π zugeordnet. Gleiches gilt für die Kennzeichnung „nicht verfügbar“, die folgende Bedeutung besitzt: Das Kriterium, um erfüllt werden zu können, setzt das Vorhandensein einer bestimmten Technologie voraus, die das jeweilige BS oder die App nicht unterstützt. Dabei bezieht sich dieses Kriterium auf ein vorheriges, für welches das BS oder die App bereits eine geringere Bewertung erhalten hat.

Definition 5. Sei $\alpha_i \in [0,1]$ mit $\sum_{i=1}^{n} \alpha_i = 1$. Die Gesamtbewertung über alle Teilaspekte φ ergibt sich unter der Gewichtung der einzelnen Teilaspekte über die Koeffizienten α_i aus $\varphi: \mathbb{R} \to \mathbb{R}$ gemäß

$$\varphi := \sum_{i=1}^{n} (f(G_i, S_i) * \alpha_i) \tag{5.6}$$

Dabei können die einzelnen Teilaspekte, je nach Anwendungsfall, anhand der eingeschätzten Relevanz über die Koeffizienten α gewichtet werden.
Die in dieser Arbeit beschriebene Bewertung zeigt lediglich ein Beispiel. Eine individuelle Gewichtung einzelner Kriterien oder Teilaspekte, wie beispielweise die höhere Gewichtung der Sicherheit für Großunternehmen bzw. Unternehmen in sicherheitskritischen Branchen im Banken- oder Produktionssektor wird empfohlen. Denkbar ist außerdem das Entfernen einzelner Teilaspekte, sollte diesen im jeweiligen Anwendungsfall keine Relevanz zukommen.

Definition 6. Sei $g \in: \mathbb{R} \to \mathbb{R}$ ein Bewertungsschlüssel als Abbildung der Gesamtbewertung auf eine Note gemäß

$$g(\varphi) := \left\{ \begin{array}{c} 1{,}0\ f\ddot{u}r\ 0{,}95 \leq \varphi \leq 1{,}0 \\ 1{,}3\ f\ddot{u}r\ 0{,}90 \leq \varphi \leq 0{,}94 \\ 1{,}7\ f\ddot{u}r\ 0{,}85 \leq \varphi \leq 0{,}89 \\ 2{,}0\ f\ddot{u}r\ 0{,}80 \leq \varphi \leq 0{,}84 \\ 2{,}3\ f\ddot{u}r\ 0{,}75 \leq \varphi \leq 0{,}79 \\ 2{,}7\ f\ddot{u}r\ 0{,}70 \leq \varphi \leq 0{,}74 \\ 3{,}0\ f\ddot{u}r\ 0{,}65 \leq \varphi \leq 0{,}69 \\ 3{,}3\ f\ddot{u}r\ 0{,}60 \leq \varphi \leq 0{,}64 \\ 3{,}7\ f\ddot{u}r\ 0{,}55 \leq \varphi \leq 0{,}59 \\ 4{,}0\ f\ddot{u}r\ 0{,}50 \leq \varphi \leq 0{,}54 \\ 5{,}0\ f\ddot{u}r\ \varphi < 0{,}50 \end{array} \right\} \tag{5.7}$$

Den relativen Anteilen der Gesamtbewertung wird ein Bewertungsschlüssel als Abbildung auf eine Gesamtnote und Einschätzung der Tauglichkeit für Unternehmen zugrunde gelegt. Eine Empfehlung der Tauglichkeit je nach Note ist Anhang B zu entnehmen. Eine sehr gute Bewertung spricht für eine Tauglichkeit in sicherheitskritischen Umgebungen wie Banken, Versicherungen, Kreditinstituten, produzierendes Gewerbe. Das sind prinzipiell große Unternehmen mit einer großen Anzahl von Mitarbeitern. Außerdem werden Regierungseinrichtungen in diesen Bereich eingeordnet. Eine gute Bewertung ist beispielweise für den Einsatz in In-House- oder In-Car-Systemen denkbar. Außerdem ist die Verwendung für Unternehmen der Produktion oder Gesundheitsbranche mit einer mittelgroßen Anzahl an Mitarbeitern möglich. Für kleine Unternehmen mit wenigen Mitarbeitern genügen Systeme mit einer befriedigenden Bewertung. Beispiele sind Applikationen für die Lehre oder der Bereich

Medien und Design. Systeme mit einer mangelhaften Bewertung sind prinzipiell nicht für den Unternehmenseinsatz geeignet.

5.3 Kriterien für die Bewertung mobiler Betriebssysteme – Vorentwicklungsphase

Die Kriterien der mobilen BSs beschränken sich auf die Teilaspekte Sicherheit, MDM sowie Benutzerinterfaces und Trennung von der Anwendungslogik. Sie sind anhand der in Abschnitt 4 erläuterten Aspekte zusammengestellt. Ausgewählte Kriterien werden im Folgenden mit ihrer Unterstützung von den betrachteten BSs beschrieben. Außerdem wird auf die Gewichtung einzelner Kriterien nach Definition 1 eingegangen.
Für die Einschätzung der Unterstützung einzelner Funktionen der BSs wurde, neben den in Kapitel 4 genannte Quellen, folgende unterstützende Literatur genutzt. In den Teilkriterien Sicherheit (im Bereich VPN) und MDM (in den Bereichen VPN, Remote Wipe, BYOD) wurde GRUMAN (2015) als Orientierung hinzugezogen. Dort werden einige Sicherheitskriterien, aber vor allem MDM-Funktionen, einzelner Systeme aufgezeigt. Für die Einschätzung der Samsung Knox und „Android for Work“ Lösung stellte SAMSUNG (2015) hilfreiche Beschreibungen einzelner Funktionen bereit.

a. Sicherheit

Im Sektor Sicherheit wurden zunächst die Kriterien zum Abschnitt System- und App-Integrität in den Katalog übernommen. Diese stellen elementare Sicherheitsmechanismen dar, die aktuelle mobile BSs mitbringen sollten. Diese werden inzwischen von allen Systemen unterstützt (vgl. GRUMAN, 2015). Tabelle 9 zeigt eine Übersicht der Kriterien.
Es folgen die Kriterien zu Abfrage- und Einstellungsmöglichkeiten von Berechtigungen. Das nachträgliche Abwählen bzw. Zustimmen von Berechtigungen wird als besonders relevant angesehen, um dem Nutzer die vollständige Kontrolle über Datenflüsse zu gewähren. Deshalb erhält dieses Kriterium die doppelte Gewichtung. Je nach der gewählten mobilen Strategie des Unternehmens können entsprechende Einstellungen vom Unternehmen und/oder vom Nutzer vorgenommen werden. iOS und BlackBerry zeigen hier die besten Eigenschaften. Beide Systeme lassen ein nachträgliches Ändern der Berechtigungen zu. Android unterstützt diese Funktion nicht und Windows Phone wird dies erst mit dem kommenden Update 2 (auch General Distribution Release (GDR) 2) unterstützen. iOS ist das einzige System, welches kontextbezogene Beschreibungen des Entwicklers zulässt. Der Nutzer wird informiert, aus welchem Grund eine Berechtigung benötigt wird. Unterschiede gibt es außerdem beim Zeitpunkt der Abfrage von Berechtigungen. iOS befragt den Benutzer als einziges System zum Installationszeitpunkt und zum Zeitpunkt, an dem die Berechtigung benötigt wird.
Es folgen Möglichkeiten der Verschlüsselung von Daten. Die Hardwareverschlüsselung stellt einen geschützten Bereich zur Verfügung und benötigt entsprechende Hardware wie Crypto-Prozessoren. Sie wird unter Android ausschließlich auf Geräten mit entsprechender Hardware-Ausstattung unterstützt. Dies ist der großen Bandbreite an Android-Geräten unter verschiedensten Herstellern geschuldet. Die Dateisystemverschlüsselung wird von allen Systemen unterstützt. Hier ist lediglich darauf zu achten, welche Maßnahmen ergriffen werden müssen, um sie zu aktivieren. Unter Android ist diese standardmäßig nicht aktiviert und Windows Phone lässt eine Aktivierung ausschließlich über ein MDM-System zu. Die vom Entwickler beeinflussbare Verschlüsselung von Applikationsdaten, meist durch eine sogenannte KeyChain realisiert, ist auf Android, iOS und Windows Phone vorhanden.

Lediglich zu BackBerry konnte keine entsprechende Aussage gefunden werden. Der S/MIME Standard zur Verschlüsselung und Signatur von E-Mails wird inzwischen auf allen Systemen, außer Android, unterstützt. Sichere Netzwerkverbindungen über SSL/TLS sind ebenfalls auf allen Systemen möglich. Empfohlen wird der TLS 1.0 Standard oder höher. Informationen zum FIPS 140-2 Standard sind Abschnitt 4.1.3 zu entnehmen.
Es folgen Kriterien zu Möglichkeiten der VPN-Konfiguration. Ein nativer VPN-Client ist unter allen Systemen vorhanden. Lediglich Android unterstützt standardmäßig kein SSL VPN. Feingranulare VPN-Einstellungen auf Ebene einzelner Apps sind unter Android ausschließlich im gesicherten Bereich des „Android for Work“ Systems möglich (vgl. SAMSUNG, 2015). Unter BlackBerry werden das Per-App VPN, als auch das Always-On VPN, nicht unterstützt.

Sicherheit
System- und App-Integrität
ASLR
DEP
Sandboxing
Verifizierung der geladenen Softwarekomponenten während des Bootprozesses (Secure Boot)
Berechtigungen
Zeitpunkt der Abfrage von Permissions
Abwählen einzelner Permissions bei Abfrage
Angabe von Gründen, wieso Permission benötigt wird
Nachträgliches Abwählen/ Zustimmen einzelner Permissions
Verschlüsselung
Softwareverschlüsselung
Hardwareverschlüsselung
Dateisystemverschlüsselung
FIPS 140-2 Zertifizierung
S/MIME
SSL/TLS
VPN-Konfiguration
nativer VPN-Client
iPsec/SSL VPN
Per-App VPN
Always-On VPN
Authentifizierung
Biometrische Verfahren
Single Sign On
Wi-Fi Authentifizierung
Review
Review-Prozess der Apps im App-Store

Tabelle 9 Kriterien des Teilaspekts Sicherheit zur Bewertung mobiler Betriebssysteme

Die Zugriffsauthentifizierung kann unter allen Systemen über PIN oder Password realisiert werden (vgl. GRUMAN, 2015). Wichtig ist dabei ein ausreichend starkes Passwort zu wählen und gegebenenfalls MDM-Regeln für die Passwortkomplexität festzulegen. Biometrische Verfahren sind unter Android und iOS verfügbar. Sie sind allerdings nach wie vor umstritten.

Auch, wenn sie ein Shoulder Surfing ausschließen, werden diese Verfahren teilweise mit Leichtigkeit überlistet siehe Abschnitt 4.1.5. Aus diesem Grund wird hier die halbe Gewichtung gewählt. Viele Benutzer empfinden biometrische Verfahren als deutlich benutzerfreundlicher, da sie weniger Zeit in Anspruch nehmen, als die Eingabe eines Passwortes. Im Unternehmensbereich kann darauf allerdings keine Rücksicht genommen werden. Das Single-Sing-On erhöht die Benutzerfreundlichkeit bei Nutzung verschiedener Enterprise-Apps beträchtlich. Eine separate Authentifizierung für jede genutzte App entfällt. Die Unterstützung von Enterprise WLAN-Protokollen gehört mittlerweile zum Standard.
Schließlich wird das Kriterium eines Reviews von Apps des jeweiligen App-Stores der Plattform hinzugefügt. Zu einem Review gehören unter anderem das Prüfen, ob die korrekten APIs verwendet werden, ob Schadsoftware eingeschleust wurde oder werden soll und ob die korrekten Permissions verwendet wurden. Ein Review Prozess wird unter allen Plattformen durchgeführt. Auf Grund fehlender Informationen kann hier im Einzelnen nicht bewertet werden, welche Prozesse und wie diese Prozesse durchlaufen werden. Fakt ist, dass es Angreifern gelingt, Schadsoftware einzuschleusen. Es wird wenig Wert darauf gelegt, unnötige Datenübermittlungen zu unterbinden und falsch gesetzte oder missbräuchlich verwendete Berechtigungen zu entziehen. Aus diesem Grund sollte stets eine Einschränkung von Apps durch Black- oder Whitelists erfolgen. Alternativ kann ein Enterprise-App-Store zur Verfügung gestellt und der Zugriff auf öffentliche App-Stores unterbunden werden.

b. Mobile Device Management

Die Kriterien dieses Abschnitts entstammen den in Abschnitt 4.4.1 geschilderten Anwendungsfällen sowie Tabelle 7 zum Vergleich von MDM-Lösungen. Die geschilderten Kriterien beziehen sich auf MDM-Regeln. Stellt das BS entsprechende Regeln zur Verfügung, so wird dieses Kriterium als erfüllt betrachtet. Die Umsetzung der Regeln bedingt stets das Betreiben einer MDM- bzw. EMM-Lösung. Tabelle 10 gibt einen Überblick der Kriterien.
Alle Systeme bieten zum jetzigen Zeitpunkt einen integrierten MDM-Client. Unter Android ist die nachträgliche Installation eines MDM-Agenten ab Version 5 nicht mehr notwendig, da die „Android for Work“ Lösung integriert wurde. Versionen kleiner 5 müssen die „Android for Work App“ nachinstallieren (vgl. GRUMAN, 2015). iOS und BlackBerry implementieren eine große Anzahl fest vorgegebener Regeln. Android verlässt sich auf herstellerspezifische Erweiterungen. Samsung Knox nutzt diese Erweiterungsmöglichkeit und fügt über 1500 MDM APIs hinzu (vgl. SAMSUNG, 2015). Ausschließlich unter BlackBerry ist es dem Nutzer nicht möglich, die Kontrolle durch den MDM-Administrator zu unterbinden (vgl. KREUZHUBER, TEUFL & ZEFFERER, 2014). Aus Sicht des Unternehmens ist dies ein Vorteil. Zur Verfügung gestellte Geräte (COPE oder COBO) bleiben unter der Kontrolle des Unternehmens und Daten sind durch das Device Management geschützt. Bei den verbleibenden Systemen können zumindest alle unternehmensspezifischen Daten entfernt werden.
BlackBerry und Android 5 stellen integrierte BYOD-Lösungen zur Verfügung. Allerdings ist anzumerken, dass Blackberry deutlich mehr Regeln, sowohl für den privaten, als auch den geschäftlichen Bereich bereitstellt. Samsung Knox bietet als Erweiterung des Android-Systems ebenfalls eine BYOD-Lösung durch Hinzufügen eines geschäftlichen Bereichs an. Dieses System ist durch das Erweitern um zahlreiche APIs als funktionsmächtiger einzuschätzen, als die „Android for Work“-Lösung (vgl. SAMSUNG, 2015). Andere Systeme müssen hier auf

Container-Apps (Containerization) zurückgreifen. Das Kriterium der Möglichkeit von Containerization ist automatisch erfüllt, sobald das BS eine integrierte BYOD-Lösung besitzt.
Das OTA-Management der Geräte, wie das Übertragen von Konfigurationen wie VPN-, WLAN-Einstellungen und MDM-Regeln sowie das Übertragen von Firmware- und App-Updates, ist inzwischen Standard.
Das Einschränken von installierbaren Apps kann über ein Black-/Whitelisting oder das Einrichten eines Enterprise-App-Stores geschehen. Die Einrichtung eines Enterprise-App-Stores ist unter allen System möglich (vgl. KREUZHUBER, TEUFL & ZEFFERER, 2014). Unter iOS existiert keine MDM-Regel für ein Black-/Whitelisting und unter Android ist dies ausschließlich im sicheren Arbeitsbereich möglich.
Über verschiedene Services der Plattformen und MDM-Regeln sind Remote Wipe, Locate und Lock auf allen Systemen verfügbar.
Ein Backup und Restore der Gerätedaten ist jeweils durch die vom Plattformhersteller bereitgestellten Services möglich. Für den Unternehmenseinsatz ist dieser Mechanismen allerdings von zweitrangiger Bedeutung und stellt oft ein Sicherheitsrisiko dar, da Daten unbemerkt an die Plattformbetreiber abfließen. Deshalb erhält dieses Kriterium die halbe Gewichtung. Das Abschalten der Backup-Funktion ist unter Android 5 nicht möglich. Erst Samsung Knox fügt eine entsprechende Regel hinzu. Unter Blackberry kann der Zugriff verschiedener Cloud-Storage-Lösungen, wie Box oder Dropbox unterbunden werden (vgl. KREUZHUBER, TEUFL & ZEFFERER, 2014).
Die Möglichkeit, den Gerätezustand auszulesen, ist ein wichtiges Werkzeug, um Jailbreaks bzw. manipulierte Geräte und Schadsoftware zu erkennen. Unter Android ist diese Funktionalität nicht standardmäßig verfügbar.

Mobile Device Management
MDM-Client
Anzahl MDM-Regeln
Entziehen der MDM-Kontrolle durch den Benutzer
Integrierte BYOD-Lösung und Funktionsumfang
Möglichkeit von Containerization
OTA-Management
Einschränkung von Apps (Whitelist/Blacklist), Enterprise App Store
Remote Wipe
Remote Locate
Remote Lock
Passwort und Authentifizierungskontrolle
Lokale Verschlüsselung (native Unterstützung, über Software)
Konfiguration der Kommunikation wie VPN
Konfiguration der Kommunikation wie WLAN
Backup/Restore von Gerätedaten
Backups auf externe Cloud-Services unterbinden
Zustand von Geräten auslesbar
Aktivierung Per-App VPN, Always-On
Sicheres Löschen von Daten (Daten nicht mehr auf Datenträger)
Assigned Access

Tabelle 10 Kriterien des Teilaspekts MDM zur Bewertung mobiler Betriebssysteme

Unterschätzt wird häufig die Funktion des sicheren Löschens von Gerätedaten, ohne dass gelöschte Daten im Speicher zurückbleiben. Dies ist ausschließlich unter BlackBerry möglich. Andere Systeme bieten lediglich die Möglichkeit, die Daten verschlüsselt zu speichern und Geräte danach zurückzusetzen. Unter iOS ist es möglich, die Daten durch Löschen des Blockschlüssels praktisch unbrauchbar zurückzulassen, da ein Entschlüsseln nicht mehr möglich ist.

c. Benutzerinterfaces und Trennung von der Anwendungslogik

Die Kriterien dieses Teilaspektes fassen die Beobachtungen aus Abschnitt 4.5 zusammen. Einen Überblick der Kriterien gibt Tabelle 11.
Unter allen Plattformen ist das Erstellen von Web-Apps, hybriden Apps und nativen Apps möglich siehe Abschnitt 4.5.2.
Die erleichterte Trennung der UI von der Anwendungslogik ist durch die Unterstützung entsprechender Beschreibungssprachen, wie XML, XAML und QML, unter allen Plattformen möglich. Blackberry bietet als einzige Plattform eine vollständige Programmiersprache (QML) für die Beschreibung der UI.
Die Arbeit des Entwicklers wird durch die Möglichkeit der UI-Entwicklung über ein Designer-Tool deutlich erleichtert, im Gegensatz zu einer rein programmatischen Definition. Moderne Designer zeigen nicht nur die UI an, sondern lassen eine direkte Manipulation von Elementen zu und legen entsprechende Event Handler fest. Unter Android Studio, XCode und Visual Studio bestehen alle diese Möglichkeiten. Das auf Eclipse basierende Momentics von BlackBerry lässt ausschließlich ein Preview der aktuellen Konfiguration zu.
Das Strukturieren der Apps anhand verschiedener Separation-Presentation-Pattern ist grundsätzlich unter allen Plattformen möglich. Zu erwähnen ist, dass iOS das abgewandelte MVC Pattern (ähnlich MVP) sowie Windows Phone das MVVM-Pattern empfiehlt. Schließlich bieten die beiden Plattformen entsprechende Werkzeuge für die erleichterte Umsetzung des jeweiligen Patterns an.
Dieser Teilaspekt enthält, im Vergleich zu den bisher vorgestellten Aspekten, deutlich weniger Kriterien. Eine zukünftige Erweiterung ist beispielweise durch die Betrachtung der Umsetzung der Mehrsprachigkeit, oder die Unterstützung mehrerer Display-Auflösungen auf den BSs denkbar.

Benutzerinterfaces und Trennung von der Anwendungslogik
Möglichkeiten der Plattformen
Trennung von Layout, Design und Logik möglich
Beschreibungssprache für die UI
UI Entwicklung über Designer
Möglichkeit Separation-Presentation-Pattern zu implementieren
Sonstige Features zur Separation, Hilfen für Entwickler bei der Umsetzung bestimmter Pattern

Tabelle 11 Kriterien des Teilaspekts Benutzerinterface und Trennung von der Anwendungslogik zur Bewertung mobiler Betriebssysteme

5.4 Bewertung der mobilen Betriebssysteme

In diesem Abschnitt erfolgt ein Vergleich der Bewertungen der einzelnen Teilaspekte zwischen den betrachteten BSs. Die detaillierte Bewertung einzelner Kriterien ist Anlage C zu entnehmen. Im Anschluss wird das Ergebnis der Gesamtbewertung über alle Teilaspekte aufgezeigt.

Bewertung der Teilaspekte

Die Bewertung erfolgt jeweils nach Definition 4 des Bewertungssystems. Tabelle 12, Tabelle 13 und Tabelle 14 zeigen die Bewertung des jeweiligen Teilaspektes für die betrachteten mobilen BSs.

a. Sicherheit

Android 5 erreicht einen relativen Anteil von 0,65 und bleibt damit abgeschlagen hinter den anderen Systemen zurück. Gründe sind das mangelhafte Berechtigungssystem, welches es dem Nutzer nicht erlaubt, einzelne Berechtigungen zu beeinflussen. Außerdem lassen sich Defizite bei der Verschlüsselung erkennen. Die Hardwareverschlüsselung ist geräteabhängig und eine Unterstützung von S/MIME Standard sowie SSL VPN ist nicht nativ vorhanden. Außerdem fehlt der Support für SSO.
Deutlich besser schneidet das durch Samsung Knox unterstützte Android ab. Hier wurden einige APIs hinzugefügt und die oben genannten Features nachgerüstet. Nachteil ist lediglich, dass sich diese meist auf den sicheren Knox Bereich beschränken und im persönlichen Bereich nicht zur Verfügung stehen. Der erreichte relative Anteil liegt bei 0,82.
Gleichauf liegen Windows Phone 8.1 (0,86) und BlackBerry 10 (0,85). Die Vorteile von BlackBerry liegen in dem ausgereiften Berechtigungssystem und der Unterstützung sämtlicher Features im Bereich Verschlüsselung. Besonders hervorzuheben ist die Dateisystemverschlüsselung, sowohl im geschäftlichen, als auch im privaten Bereich der integrierten BYOD-Lösung. Nachteile entstehen durch das Fehlen feingranularer VPN-Konfigurationen.
Windows Phone 8.1 besitzt ähnlich Android einige Defizite im Berechtigungssystem. Berechtigungen können bisher nicht beeinflusst werden und werden auch bei der Installation nicht angezeigt. Lediglich bei der Übermittlung vertraulicher Daten erfolgt ein Einbeziehen des Nutzers. Davon abgesehen, bietet Windows Phone sowohl bei der Verschlüsselung, als auch im Bereich VPN eine lückenlose Sicherheit.
iOS 8 erhält die Bestnote (1,0). Es erfüllt alle Kriterien des Katalogs vollständig. Das Benachrichtigungssystem ist durchdacht und bietet dem Nutzer die vollständige Kontrolle.

Mobiles Betriebssystem	Bewertung
iOS 8	1,0
Windows Phone 8.1	0,86
BlackBerry 10	0,85
Android 5 + Samsung Knox	0,82
Android 5	0,65

Tabelle 12 Bewertung des Teilaspektes Sicherheit für einzelne BSs

b. Mobile Device Management

Android 5 erreicht einen relativen Anteil von 0,71 und erhält damit die schlechteste Bewertung in diesem Bereich. Gründe sind vor allem fehlende MDM-Funktionalitäten, wie das Auslesen des Gerätezustands, das Verhindern von Backups, fehlende Konfigurationsmöglichkeiten des WLAN sowie das Fehlen eines Assigned Access Modus. Positiv ist die Integration einer BYOD-Lösung zu bewerten.
iOS und Windows Phone liegen in diesem Bereich mit einer Bewertung von 0,83 bzw. 0,85 gleichauf. Zu bemängeln ist die fehlende BYOD-Funktionalität, die ausschließlich durch Drittanbieter-Apps nachgerüstet werden kann. iOS fehlt die MDM-Regel für das Erstellen von Black- bzw. Whitelists.
Mit einem relativen Anteil von 0,92 erreicht Android im Zusammenspiel mit Samsung Knox den zweiten Rang. Durch das Nachrüsten der Android Plattform durch zahlreicher APIs, ergeben sich eine umfangreiche BYOD-Lösung, eine weitreichende Unterstützung verschiedener VPN-Modi wie Per-App-VPN und Always-On sowie Assigned Access.
Spitzenreiter in diesem Bereich ist BlackBerry 10. Mit einer umfangreichen BYOD-Lösung mit Regeln für den privaten und persönlichen Bereich, expliziten Regeln für das Unterbinden von Cloud Backups, sowie der Möglichkeit eines Security Wipe zum sicheren Löschen der Daten erreicht das System die Bestnote 1,0.

Mobiles Betriebssystem	Bewertung
BlackBerry 10	1,0
Android 5 + Samsung Knox	0,92
Windows Phone 8.1	0,85
iOS	0,83
Android 5	0,71

Tabelle 13 Bewertung des Teilaspektes MDM für einzelne BSs

c. Benutzerinterfaces und Trennung von der Anwendungslogik

In diesem Bereich sind ausschließlich kleine Unterschiede zu erkennen Windows Phone und iOS erhalten sich mit einem relativen Anteil von 0,94 eine leichte Führung, die durch die Unterstützung spezieller Separation-Presentation-Pattern und deren erleichterte Umsetzung durch zusätzliche Mechanismen zu erklären ist. Android sowie BlackBerry liegen mit 83% der Punkte dicht dahinter.

Mobiles Betriebssystem	Bewertung
iOS 8	0,94
Windows Phone 8.1	0,94
BlackBerry 10	0,83
Android 5 + Samsung Knox	0,83
Android 5	0,83

Tabelle 14 Bewertung des Teilaspektes Benutzerinterface und Trennung von der Anwendungslogik für einzelne BSs

Gesamtbewertung

Zunächst wird die Gewichtung der einzelnen Teilaspekte vorgenommen. Da der Teilaspekt „Benutzerinterface und Trennung von der Anwendungslogik“, im Vergleich zu den anderen Bereichen, deutlich weniger Kriterien enthält, wird ihm ein geringeres Gewicht von $\alpha_{UI} = {}^{1}/_{6}$ beigemessen. Sicherheit und MDM werden jeweils mit einem α von $\frac{5}{12}$ gleich gewichtet. Damit ergibt sich über Definition 5 die Gesamtbewertung siehe Tabelle 15. Die Note und die Einsatzmöglichkeit folgen aus Definition 6 bzw. dem Bewertungsschlüssel siehe Anlage B.

Rang	Betriebssystem	Bewertung	Note	Einsatz
1	iOS 8	0,92	Sehr gut	Höchste Anforderungen wie Banken und Produktion bzw. große Unternehmen
2	BlackBerry 10	0,91	Sehr gut	
3	Windows Phone 8.1	0,86	Gut	Mittlere Anforderungen wie Automotive und Smart-Home bzw. mittel große Unternehmen
3	Android 5 + Samsung Knox	0,86	Gut	
4	Android 5	0,71	Befriedigend	Geringe Anforderungen wie Lehranstalten bzw. kleine Unternehmen

Tabelle 15 Gesamtbewertung mobiler BSs

5.5 Kriterien für die Bewertung mobiler Applikationen – Entwicklungs- und Einsatzphase

In die Kriterien zur Bewertung der mobilen Applikationen fließen alle in Abschnitt 4 genannten Hauptaspekte ein. Diese Phase definiert Kriterien, die der Entwickler während der Erstellung und das Unternehmen während des Einsatzes der App direkt beeinflussen können, unabhängig von der zugrundeliegenden mobilen Plattform. Eine Übersicht der Kriterien des jeweiligen Teilaspektes ist Tabelle 16, Tabelle 17, Tabelle 18, Tabelle 19 und Tabelle 20 zu entnehmen. Auf ausgewählte Kriterien wird im Folgenden jeweils genauer eingegangen.

a. Sicherheit

Der Teilaspekt Sicherheit befasst sich zunächst mit dem Verhältnis zwischen gewährten Permissions und der Funktionalität der App. Für jede Permission muss es möglich sein, eine allgemeine Beschreibung, eine Beschreibung im aktuellen Kontext und eine ausführliche Beschreibung für den Benutzer zu formulieren, sodass die Permission gerechtfertigt werden kann. Weitere Überlegungen, wann eine Berechtigung gerechtfertigt ist, werden in Abschnitt 6.6.1 bei der konkreten Bewertung eines Softwareprojektes genannt.
Die Verschlüsselung von App-Daten, wie Anmeldedaten und jede Art von personenbezogenen Daten, sollte über die KeyChain erfolgen.
Verbindungen zum Unternehmensnetzwerk sind stets über gesicherte VPN-Verbindungen zu realisieren.
Es besteht die Möglichkeit, neben den Authentifizierungsmethoden des BS, eine eigene Authentifizierung für die App zu implementieren. Besonders sicher ist diese Methode, wenn die Anmeldedaten nicht auf dem mobilen Gerät, sondern dem Unternehmensserver gesichert vorliegen. Die Authentifizierung ist in diesem Fall allerdings nur möglich, wenn eine Verbindung zum Unternehmens-Back-End existiert. Wird die externe Authentifizierung nicht

gewählt, so ist für eine sichere Verschlüsselung der Anmeldedaten zu sorgen. Eine 2-Faktor-Authentifizierung sorgt für zusätzliche Sicherheit und wird heute bereits bei den meisten Online-Services realisiert. Sie besteht meist aus einem Passwort und einem Code, welcher per SMS zugeschickt wird oder einer Smart Card.
Schließlich sollte zu allen Back-End-Systemen eine sichere SSL/TLS Verbindung bestehen.
Die Verteilung der App sollte über den App-Store der Plattform oder bestenfalls über eine eigens eingerichteten Enterprise App-Store erfolgen. Eine Verteilung über Drittanbieter, ohne Review-Prozess und entsprechende Überprüfung angebotener Apps, ist nicht zu empfehlen.
Die zu bewertende App sollte einen Review-Prozess durchlaufen. Dieser kann vom Plattformanbieter selbst erfolgen, oder vom Unternehmen über entsprechende Richtlinien realisiert werden. Apps sind unter anderem zu überprüfen auf:

- das Einschleusen von Schadsoftware
- die korrekte Verwendung von API-Funktionen
- korrekt umgesetzte Verschlüsselungsparameter wie iOS Schutzklasse, Speicherort von Passwörtern, Schlüsseln und Zertifikaten beispielsweise in der KeyChain, korrekte Ableitung des kryptographischen Schlüssels, korrekte SSL-Verschlüsselung (evtl. Certificate Pinning), usw.
- Notwendigkeit gesetzter Berechtigungen

Des Weiteren sollten keine Services anderer Apps genutzt werden, um ein Permission Spreading siehe Abschnitt 4.1.1 zu verhindern. Der Nutzer sollte stets über Abhängigkeiten informiert werden.

Sicherheit
Permissions in angemessenem Verhältnis zur Funktionalität
Verschlüsselung von App-Daten (z.B. KeyChain)
Nutzung sicherer VPN-Verbindungen zum Unternehmen
Eigene Zugriffsauthentifizierung
Nutzen 2-Faktor-Authentifizierung
Verbindung mit SSL/TLS-Verschlüsselung zu allen genutzten Services
Verteilung über App-Store/ Enterprise App-Store/ Drittanbieter
Durchlaufen eines Review-Prozesses (Plattform/intern)
keine Nutzung der Services anderer Apps (Android Intents, iOS App Group)
Informieren der Mitarbeiter über Sicherheitsrichtlinien
Minimalanforderungen an Geräte
Risiko- und Sicherheitsanalyse

Tabelle 16 Kriterien des Teilaspekts Sicherheit zur Bewertung mobiler Enterprise Apps

Die letzten drei Kriterien dieses Bereichs beziehen sich auf das Unternehmen, in welchem die App verwendet wird. Mitarbeiter sollten stets über die Sicherheitsvorschriften des Unternehmens unterrichtet werden. Ihnen muss klar sein: welche Daten sie in welchem Bereich beispielsweise bei Nutzung einer BYOD-Lösung ablegen dürfen; wo sie auf Daten zugreifen können (beispielsweise bei einem Mobile Data Management mit Geofencing); wo Daten abgespeichert werden (Vertrag mit Cloud Service Anbieter oder intern); welche Online Services verwendet werden dürfen. Außerdem müssen sie über Sicherheitsrisiken und deren Entstehung unterrichtet werden. Dies schafft die Grundlage für eine sichere Umgebung der

App-Nutzung. Als Grundlage dient die eigene Sicherheits- und Risikoanalyse des Unternehmens. Hier werden Risiken und Strategien für deren Kontrolle festgehalten. Schließlich sollten Vorgaben für Minimalanforderungen der Geräte, wie minimal unterstützte Betriebssystemversionen existieren. Sicherheitslücken werden meist erst in neueren Versionen behoben. Ältere Versionen bekommen keine entsprechenden Updates vom Gerätehersteller. Eine Einschränkung ist daher sinnvoll und erleichtert das MDM.

b. Benachrichtigungen

Die Kriterien des Bereichs Benachrichtigungen entsprechen den in Tabelle 5 gezeigten Anforderungen an Notification Services. In diesem Bereich wird der für die App gewählte Notification Service bewertet.

Benachrichtigungen
Cloud unabhängig
Queuing
Rückantwort bei fehlgeschlagenen Nachrichten
Sichere Kommunikation
Statusabfrage bzw. Empfangsbestätigung
Zeitschranken

Tabelle 17 Kriterien des Teilaspekts Benachrichtigungen zur Bewertung mobiler Enterprise Apps

c. Mobile Cloud Computing – Cloud Storage

Wie in Abschnitt 4.3 beschrieben, bringt die Nutzung eines Cloud-Speichers Risiken mit sich. Ohne diesen Service sind Daten sicher und nicht unter der Kontrolle Dritter. Grundsätzlich wird empfohlen, sicherheitskritische Daten nicht in externen Cloud Speichern abzulegen. Eine Unternehmens-interne Lösung ist einem externen Anbieter grundsätzlich vorzuziehen. Sie ist so sicher, wie es die Sicherheitsrichtlinien inklusive Zugriffskontrolle und Backuplösung erlauben. Die meisten Unternehmen sind auf einen Online-Datenspeicher angewiesen. Die Anschaffung einer eigenen Infrastruktur lohnt sich ausschließlich für große Unternehmen, da sie mit erheblichen Kosten verbunden ist.
Wird kein Cloud-Speicher zum Backup bzw. zur Synchronisation der App-Daten genutzt, so befinden sich die Daten ausschließlich auf dem Endgerät. Dies wird ebenfalls als kritisch betrachtet, da ein Verlust der Daten durch den Verlust des Gerätes droht. Hier sollte zumindest ein manuelles Backup erfolgen. Die Mitarbeiter sollten über Sicherheitsrisiken beim Ablegen der Daten auf externen Geräten unterrichtet werden. Nachteile sind wiederum, dass sich die Daten nicht unter der Kontrolle des Unternehmens befinden.
Die Kriterien zum Datenschutz entsprechen den grundlegenden Aussagen des Abschnitts 4.3.1. Der Cloud-Storage-Anbieter sollte eine serverseitige Verschlüsselung anbieten. Dies schützt nicht vor dem Zugriff des Anbieters auf die Daten, jedoch vor dem Zugriff Dritter. Natürlich spielt die Kommunikation mit dem Cloud-Service über eine gesicherte Verbindung eine grundlegende Rolle. Anbieter stellen oft einen Enterprise Account für Großkunden zur Verfügung. Über diesen werden weitaus mehr Services angeboten, als für den Privatkunden. Von der Angabe genauer Speicherorte, über die Bereitstellung einer Backuplösung, Rechteverwaltung, Zugriffskontrolle und Dateiwiederherstellung, sind weitreichende Möglichkeiten gegeben, die einen guten Anbieter auszeichnen.

Mobile Cloud Computing – Cloud Storage
Nutzung Cloud Service intern/extern
Datenschutzrecht
Speicherung Personenbezogener Daten → Gilt das Datenschutzgesetz?
Transparenz über technische und organisatorische Sicherheitsmaßnahmen
Transparenz über Orte der Speicherung bzw. Unteranbieter
Datenverarbeitung in Drittstaaten
Cloud-Anbieter in USA geschäftlich tätig
Cloud Storage Lösung
Verschlüsselung der Daten vor dem Einbringen
Sichere Kommunikation
Serverseitige Verschlüsselung
Enterprise-Account
Rechteverwaltung
Dateiwiederherstellung

Tabelle 18 Kriterien des Teilaspekts MCC zur Bewertung mobiler Enterprise Apps

d. Mobile Device Management

Zu Beginn werden einige grundlegende Unternehmensentscheidungen bewertet. Zunächst stellt sich die Frage, ob ein MDM-System verwendet wird. Das Management von Devices, Applikationen und Inhalten, sollte in Unternehmen stets durch ein MDM bzw. EMM-System erfolgen. Die Vorteile von OTA-Management und Geräteüberwachung wurden erläutert. Ist dieses Kriterium nicht erfüllt, so wird der gesamte Abschnitt MDM als nicht erfüllt angesehen. Das Unternehmen sollte stets eine oder mehrere der möglichen Unternehmensstrategien aus Abschnitt 2.2.2 verfolgen. Hier muss eine Einigung erfolgen, da nur halbherzig umgesetzte Lösungen zu erhöhtem Verwaltungsaufwand und Sicherheitsrisiken führen. Werden mehrere Strategien verfolgt, so muss eine klare Abgrenzung existieren, welche Geräte auf welche Weise zu nutzen sind. Dies muss in den Sicherheitsrichtlinien und bei der Belehrung der Mitarbeiter berücksichtigt werden.
Die Realisierung der Trennung von privaten und geschäftlichen Daten ist von der gewählten Strategie und dem gewählten MDM-System bzw. der gewählten Plattform abhängig. Für einen COBO Ansatz genügt eventuell ein klassisches MDM. Für eine COPE- oder BYOD-Strategie ist eine Plattform mit integrierter BYOD-Lösung ideal. Containerlösungen von unterschiedlichen Anbieter müssen auf ihre Sicherheit hin überprüft und je nach Anwendungsfall bewertet werden.
Die weiteren Regeln wurden bereits in der Aufstellung der Kriterien für mobile BSs beschrieben. Hier wird jeweils bewertet, ob die Regel, falls sie im genutzten BS vorhanden ist, tatsächlich umgesetzt bzw. aktiviert wird.
Sollte es ein Backup von Gerätedaten geben, so muss darauf geachtet werden, dass der Verschlüsselungsschlüssel, mit welchem die Daten auf dem Geräte verschlüsselt wurden, nicht in der Cloud abgelegt wird. Dies ist ein häufiger Angriffspunkt. Wird dies nicht beachtet, sind die Daten so sicher wie der Authentifizierungsmechanismus des Cloud-Anbieters. Dies kann nicht vom MDM-Administrator überwacht werden und ist demnach keine akzeptable Lösung.
Als besonders wichtig sind die Kriterien einer vorhandenen Dateisystemverschlüsselung und einer Vorgabe von Authentifizierungsrichtlinien durch MDM-Regeln anzusehen. Hier erfolgt

eine doppelte Gewichtung. Durch die Vernachlässigung dieser Punkte entstehen häufig ausgenutzte Angriffspunkte. Beispielweise ist ein alphanummerisches Passwort mit vier Stellen heute nicht mehr zeitgemäß. Genauere Daten sind Abschnitt 4.1.3 zu entnehmen.

Mobile Device Management
Nutzung einer MDM-Lösung
Verfolgen einer Unternehmensstrategie
klassisches MDM/Container-App/integriertes BYOD
OTA-Management
Aktivierung Per-App VPN oder Always-On
Einschränkung von Apps
Backup von Gerätedaten
Verschlüsselungsschlüssel nicht in Backups enthalten
Backup auf externen Cloud Service unterbinden
Zugriff ausschließlich auf geschäftliche Daten
Ausschließliches Löschen von geschäftlichen Daten bei Remote Wipe
Dateisystemverschlüsselung aktiviert
Vorgabe von Richtlinien für Authentifizierung
Konfiguration WLAN und VPN-Verbindungen
Prüfen des Gerätezustands

Tabelle 19 Kriterien des Teilaspekts MDM zur Bewertung mobiler Enterprise Apps

e. Benutzerinterface und Trennung von der Anwendungslogik

Dieser Bereich umfasst die von der App tatsächlich umgesetzten Möglichkeiten zur Trennung von UI und Anwendungslogik, welche in Abschnitt 5.3 c beschrieben wurden. Hinzu kommt, ob eine tatsächliche Unabhängigkeit von UI und Anwendungslogik im Sinne einer unabhängigen Testbarkeit beider Komponenten erreicht wurde. Weiterhin sollte das Nutzerinterface einer Enterprise App mehrere Sprachen unterstützen. Eine vollständige Bewertung wird bereits bei zwei unterstützten Sprachen gegeben, wobei eine von diesen Englisch sein muss. Schließlich sollten mehrere Displayauflösungen unterstützt werden. Jede Plattform besitzt verschiedene Geräte mit unterschiedlich hohen Auflösungen. Eine Nutzung der App muss auf allen Geräten möglich sein und die Nutzererfahrung sollte konstant bleiben.

Benutzerinterface und Trennung von der Anwendungslogik
Nutzung von Pattern zur Separation von UI und Logik
Nutzung empfohlener Design Pattern der Plattform zur Separation von UI und Logik
Nutzung einer Beschreibungssprache zur Definition der UI
Nutzung zusätzlicher Features der Plattform zur Separation
Einhaltung der Designrichtlinien der Plattform
Unabhängigkeit von UI und Anwendungslogik
Unterstützung von Mehrsprachigkeit
Unterstützung eines Großteils der möglichen Displayauflösungen

Tabelle 20 Kriterien des Teilaspekts Benutzerinterface und Trennung von der Anwendungslogik zur Bewertung mobiler Enterprise Apps

5.6 Schlussfolgerung

Die Feststellung der Tauglichkeit mobiler Enterprise Apps für den Unternehmenseinsatz muss auf Grundlage verschiedener Teilaspekte erfolgen und ist nicht durch die reine Betrachtung der mobilen Applikation zu realisieren. Vielmehr muss die Umgebung, in welcher die App ausgeführt wird, mitbetrachtet werden. Die Umgebung betrifft das Unternehmen, welches Entscheidungen bezüglich der mobilen Plattform, des MDM, der genutzten Cloud Storage Lösungen, der Sicherheitsrichtlinien und der mobilen Strategien trifft.
Der hier beschriebene Abschnitt schlägt eine mögliche Realisierung vor, indem ein Kriterienkatalog, sowohl für die Bewertung mobiler Plattformen, als auch für die Applikation selbst und die Umgebung dieser zum Ausführungszeitpunkt erstellt wird.
Zunächst war das Ziel der Arbeit die Entwicklung eines Kriterienkatalogs zur Bewertung nativer mobiler Enterprise Apps aus der Sicht des Entwicklers. Die Anforderungen an mobile Enterprise Apps werden, wie im Laufe der Arbeit gezeigt wurde, nicht ausschließlich durch den Entwickler, sondern ebenfalls durch die mobilen Plattformen umgesetzt. Deren Einsatz ist von den Vorgaben der Unternehmen, von der gegebenen Hardware und anderen Faktoren abhängig. Damit ist der Entwickler nicht allein verantwortlich, wenn es um die Umsetzung von Kriterien geht, welche für die Tauglichkeit der App im Unternehmenseinsatz relevant sind. Sein Einfluss ist durch die Vorgaben der mobilen Plattform begrenzt. Aus diesem Grund ist das Einbeziehen mobiler Plattformen bei der Erstellung des Kriterienkatalogs unabdingbar. Dadurch wird eine vollständigere Aufstellung von Kriterien und genauere Bewertung möglich.
Außerdem ist zu bedenken, dass ebenfalls die IT-Abteilung der Unternehmen, welche für das Management der mobilen Infrastruktur verantwortlich ist, einen Einfluss auf die Nutzung der App besitzt. Vor allem beim MDM haben Entscheidungen wie die Dateisystemverschlüsselung, Geräte-Backups oder die Wahl der Authentifizierung einen Einfluss auf die Tauglichkeit im Unternehmenseinsatz.
Die Kriterien der mobilen Plattformen und der Enterprise App ähneln sich in einigen Teilbereichen stark. Eine Begründung liegt darin, dass die mobile Plattform Funktionen zur Verfügung stellt und der Entwickler entscheidet, ob er diese in der App nutzt. Beispiele sind Verfahren der Zugriffsauthentifizierung oder Verschlüsselung. Allerdings werden nicht alle Funktionen von allen mobilen Plattformen angeboten. Wird eine Funktion nicht zur Verfügung gestellt, so besteht eine Einschränkung des Entwicklers, die sich aus der Wahl der mobilen Plattform herleitet.
Im Desktop-Bereich kann die Entscheidung der zugrundeliegenden Plattform weitaus weniger relevant sein, als im mobilen Bereich. Die Hardware in Desktop-Systemen ist deutlich einheitlicher, kann konfiguriert und im Bedarfsfall leicht angepasst und erweitert werden. Der Programmcode ist dank reichhaltiger Compiler auf unterschiedlichsten Desktop-BSs ausführbar und leichter portierbar. Die BSs bieten meist äquivalente Funktionen an. Eine Begründung hierfür ist die zeitliche Entwicklung dieser Systeme. Da sie bereits deutlich länger als ihre mobilen Vertreter existieren, in Entwicklung sind und miteinander konkurrieren, konnten sich nützliche und vom Nutzer anerkannte Funktionen herauskristallisieren, die mit der Zeit auf allen Systemen umgesetzt wurden. Im mobilen Bereich werden immer neue Funktionen, UIs, Programmiersprachen, Technologien verbreitet, um dem Nutzer vereinfachte, schnellere Interaktionen zu erlauben und sich von Konkurrenten abzuheben. Die Hardware ist festverbaut und selten erweiterbar. Dies führt zu Inkompatibilitäten zwischen Systemen und zu einem größeren Einfluss der mobilen BSs auf die Funktionalität der zu entwickelnden Apps.

Das Ergebnis der Bewertung mobiler Plattformen zeigt, dass nach dem Aspekt Sicherheit iOS 8 die beste Bewertung erhält. Im Teilaspekt MDM erhält Blackberry 10 die Bestnote. Das Gesamtergebnis wird von iOS und Blackberry angeführt. Samsung Knox und Windows Phone liegen gleichauf und zeigen gute Ergebnisse. Android 5 fällt mit einer befriedigenden Bewertung weit zurück.

6 Verwendung des Katalogs in einem Softwareprojekt

Der in Abschnitt 5 vorgestellte Kriterienkatalog wird in einem realen Softwareprojekt angewendet. Für diesen Zweck wurde eine mobile Enterprise Applikation entwickelt, welche die Arbeitszeiterfassung in Projekten zur Aufgabe hat. Die Applikation wird vorgestellt und es wird die Umsetzung einzelner Kriterien des Katalogs gezeigt. Schließlich findet eine Bewertung der Applikation mit anschließender Auswertung statt.
Das gesamte Softwareprojekt ist dieser Arbeit auf DVD beigelegt (siehe Anlage E). Dort befinden sich außerdem Informationen zum Einrichten einer Testumgebung und Dokumente, die während des Entwicklungsprozesses entstanden.

6.1 Fallstudie

Ein Großunternehmen möchte die Zeit, welche seine Mitarbeiter in verschiedenen Projekten tätig sind, nachvollziehbar für die Buchführung und das Rechnungswesen aufzeichnen.
Das Unternehmen bearbeitet seine Aufträge in Form von Projekten. In diesen existieren verschiedene Rollen mit unterschiedlichen Gehältern. Jeder Mitarbeiter kann an verschiedenen Projekten beteiligt sein und nimmt in einem Projekt, an welchem er beteiligt ist, mindestens eine Rolle ein. Die Arbeitszeiten der Mitarbeiter sind flexibel gestaltet.
Das Unternehmen besitzt eine mobile IT-Infrastruktur. Es existiert ein MDM-System. Die mobile Strategie des Unternehmens ist ein COPE-Ansatz. Hierbei setzt das Unternehmen auf Windows Phone Geräte. Jeder Mitarbeiter erhält ein solches Gerät und setzt dieses für Unternehmenszwecke ein. Ein privater Einsatz ist gestattet. Das Unternehmen entscheidet sich, die Aufzeichnung durch eine mobile Business App zu realisieren.
Ziel der App ist es, einen Mehrwert für das Unternehmen und den Auftraggeber der Projekte zu erhalten, indem das Aufzeichnen der benötigten Zeit für Aufgaben innerhalb von Projekten deutlich vereinfacht, der Verwaltungsaufwand für das Erstellen von Abrechnungen minimiert und eine bessere Nachvollziehbarkeit der geleisteten Arbeit für den Auftraggeber ermöglicht wird.

6.2 Anforderungen

Die Anforderungen an die umzusetzende Applikation sind dem Lastenheft (siehe Anlage A) zu entnehmen.

6.3 Plattformentscheidung

Die Umsetzung der mobilen Applikation erfolgt auf der Windows Phone 8.1 Plattform. Folgende Gründe führten zu dieser Entscheidung:

- vorhandene mobile Infrastruktur des Unternehmens baut auf Windows Endgeräten auf
- Unwirtschaftlichkeit eines Wechsels des Anbieters bzw. der Plattform
- einfaches Portieren der App für den Einsatz auf dem Windows Desktop oder Tablet

Vorteile und Nachteile dieser Entscheidung sind im Einzelnen aus dem Kriterienkatalog und der Bewertung aus Anlage C zu entnehmen. Im Folgenden sind die wichtigsten Kriterien aufgelistet.

Vorteile:

- sehr gute kryptographische Eigenschaften (FIPS 140-2 zertifiziert)
- gute MDM-Funktionalitäten
- ausgereifte Entwicklungsplattform mit Microsoft Visual Studio
- sehr gute Möglichkeit zur Strukturierung der App
- Möglichkeit eines Enterprise-App-Stores
- geringe Fragmentierung der Plattform
- bisher wenig Schadsoftware bekannt

Nachteile:

- keine integrierte BYOD-Lösung, Nutzung einer Container-App oder klassisches MDM
- mangelhaftes Berechtigungssystem

Die Bewertung des Windows Phone 8.1 BS von 0,86 geht in die Gesamtbewertung zu einem Anteil von 50% ein, siehe Abschnitt 6.6.2.

6.4 Architekturentscheidungen

Die Architektur der mobilen Applikation, inklusive der Kommunikation zu Back-End-Systemen, ist in Abbildung 11 verdeutlicht. Es wird eine 6 Schichtenarchitektur realisiert. Die ersten vier Schichten beinhalten die mobile Anwendung. Die letzten zwei Schichten beschreiben die Kommunikation mit dem Back-End.

Grundlegende Entscheidungen

Die Umsetzung der mobilen Applikation erfolgt auf der Windows Phone 8.1 Plattform, wie in Abschnitt 6.3 beschrieben. Die Plattform beruht auf dem .NET Framework. Dieses besteht aus einer Laufzeitumgebung, der Common Language Runtime (CLR), sowie einer umfangreichen Klassenbibliothek (siehe Abbildung 11 rechts oben).
Es erfolgt eine native Umsetzung der Applikation, um eine bestmögliche Unterstützung der Funktionalitäten der Plattform zur Verfügung zu stellen. Das Unternehmen sieht eine Weiterentwicklung der App, nach dem Release der Grundfunktionalitäten, die hier vorgestellt werden, vor. Aus diesen Gründen soll eine zukunftssichere, erweiterbare App entwickelt werden, die eine maximale Ausschöpfung des Potenzials der Plattform ermöglicht. Insbesondere soll die bestmögliche Performanz erreicht werden. Die langfristige Planung des Unternehmens sieht keinen Plattformwechsel vor, sodass die Notwendigkeit einer Multiplattformunterstützung nicht besteht.
Nach Tabelle 8 kann eine native Anwendung unter Windows Phone in den Sprachen C# oder C++ mit XAML oder in JS erfolgen. C++ bietet sich für grafikintensive Anwendungen in Zusammenarbeit mit DirectX an. JS wird in Hinblick auf eine Plattformunabhängigkeit eingesetzt. Keine dieser Anwendungsfälle trifft zu. Aus diesem Grund wird die klassische und weitverbreitete native Entwicklung unter C#/XAML realisiert.

Realisierung des MVVM-Patterns

Vorlage für den grundlegenden Aufbau der App bildet das von Microsoft empfohlene MVVM-Design Pattern. Abbildung 11 verdeutlicht das Pattern durch eine orangene Kennzeichnung. Es

ermöglicht die Separierung von UI und Businesslogik mit Hilfe der unter WPF bereitgestellten Hilfsmittel siehe Abschnitt 4.5.3, fördert Übersichtlichkeit und Erweiterbarkeit und sorgt für ein unabhängiges Testen der Teilkomponenten. Dementsprechend erfolgt eine Aufteilung in Views (Presentation Layer), die in XAML-Dateien mit zugehöriger Code-Behind-Datei realisiert werden. Die UI-Logik übernehmen die ViewModel-Klassen (View Model Layer). Sie dienen insbesondere der Kapselung der Model-Klassen (Business Model Layer), sodass ein ideales Mapping zwischen Model und View möglich wird. Dieses Mapping wird durch einen Binding-Mechanismus zwischen View und ViewModel ermöglicht.

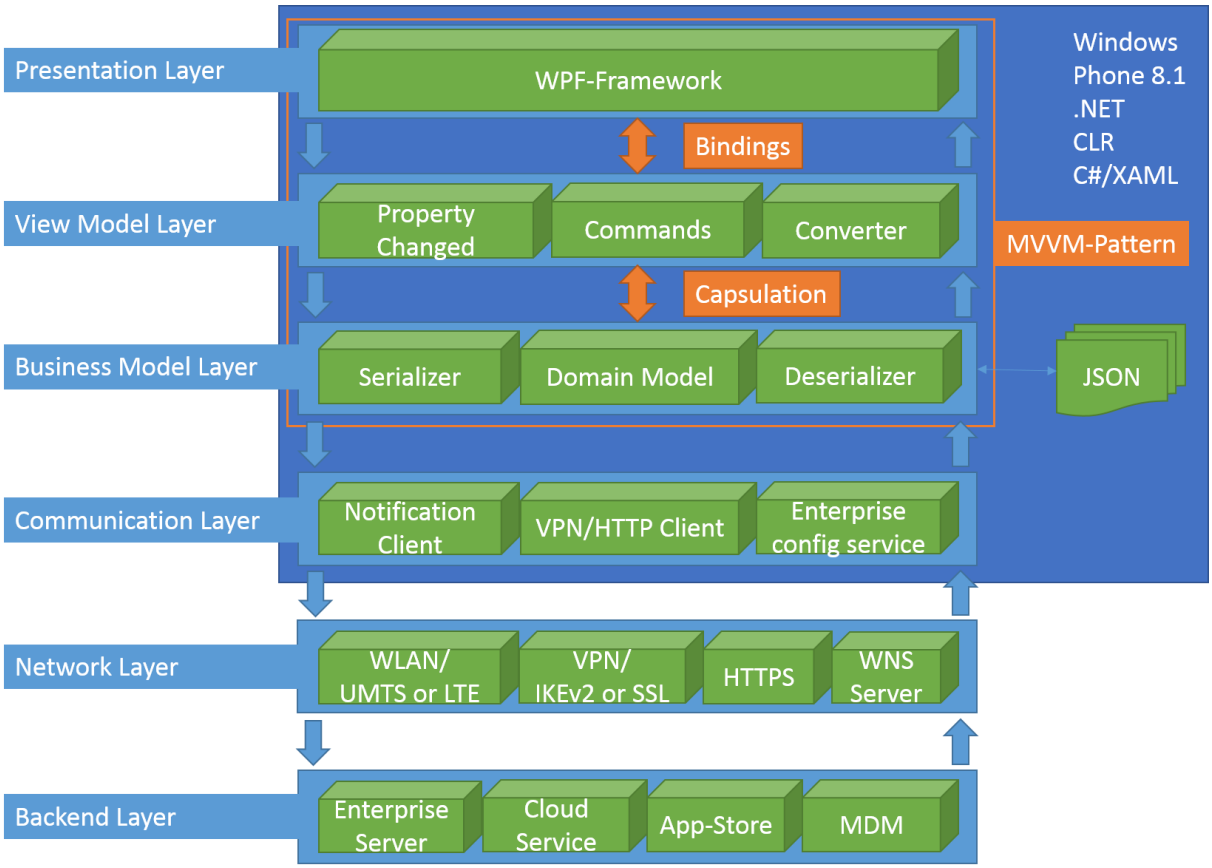

Abbildung 11 Schichtenarchitektur der mobilen Zeiterfassungs-App

Umsetzung der Schichtenarchitektur

Presentation Layer:
Das UI wird, entsprechend einer nativen Entwicklung, dem Look-and-Feel der Windows Phone Plattform gerecht. Einen Richtwert stellen die Microsoft Designprinzipien [17] sowie die Richtlinien für Windows-Runtime-Apps[18] dar. Das zugrundeliegende Grafik-Framework für die Gestaltung des UI ist das WPF-Framework. Dieses bildet eine Teilmenge des .NET-Frameworks. Insbesondere enthält es die auf XML basierende Sprache XAML zur Beschreibung von Anwendungsoberflächen. Diese trägt zur Trennung von UI und Logik bei.

[17] https://msdn.microsoft.com/de-de/library/windows/apps/hh781237.aspx
[18] https://msdn.microsoft.com/de-de/library/windows/apps/hh465424.aspx

View Model Layer:
Diese Schicht dient als Abstraktionsschicht zwischen Model und View. Sie kapselt Modelklassen und stellt Attribute in einer Form bereit, sodass die Präsentationsschicht ideal auf Modelinhalte binden kann. Der Binding Mechanismus sowie das dafür notwendige INotifyPropertyChanged-Interface und das ICommand-Interface, sind in Abschnitt 4.5.3 näher beschrieben. Converter sind Klassen, welche das Konvertieren der Werte von an die Präsentationsschicht gebundenen Attributen in beide Richtungen eines Bindings erlauben. Zum Beispiel kann ein Attribut, welches ein Datum beinhaltet vor der Anzeige in eine angepasste Form umgewandelt werden. Gleichzeitig kann bei einer Datumseingabe des Nutzers diese Eingabe in ein Datumsobjekt umgewandelt werden. In XAML kann für jedes Attribut-Binding ein Converter angegeben werden, sodass dasselbe Attribut des ViewModels auf unterschiedliche Arten dargestellt werden kann.

Business Model Layer:
Diese Schicht realisiert das Model mit den grundlegenden Anwendungsklassen, der Businesslogik und den Hilfsklassen, zum Serialisieren und Deserialisieren der Daten. Die vorliegende App serialisiert Objekte in das JSON-Format. Gründe sind eine kompakte und einfache Darstellung von Objektdaten, eine weite Verbreitung des Formats (sehr gute Verfügbarkeit von Parsern). Als Datenaustauschformat erzeugt JSON im Vergleich zu XML kleinere Datenmengen. Außerdem ist eine komplexe Auszeichnung, wie sie in XML möglich ist, nicht notwendig.

Communication Layer:
Die Kommunikationsschicht stellt Klassen für die Verbindung zum Back-End bereit. Sie dient der Entgegennahme von Notifications, dem Aufbauen von VPN-Verbindungen zum Unternehmensserver oder Verbindungen über HTTP zu Cloud-Services sowie der Autorisierung/Authentifizierung und Verschlüsselung von zu sendenden Daten. Der Enterprise Configuration Service ist für die Kommunikation mit einem MDM-Server verantwortlich. Über ihn werden Services, wie der Policy Manager, EnterpriseAppManager, RemoteLock oder VPN-Konfigurator, über das Windows Phone 8.1 MDM-Protokoll angesprochen.

Network Layer:
Diese Schicht dient der Übertragung der Daten von der mobilen Applikation zum Back-End. Zum einen sind Verbindungen drahtlos über WLAN oder das Mobilfunknetz möglich. Unter Windows Phone können VPN-Verbindungen mit IKEv2 oder SSL bzw. HTTP-Verbindungen mit SSL/TLS konfiguriert werden. Das Back-End ist nicht zu jedem Zeitpunkt online bzw. ist das mobile Endgerät durch Verbindungsengpässe nicht immer erreichbar. Aus diesem Grund existieren Server, die z.B. Notifications zwischenspeichern, um sie später weiterzuleiten.

Back-End Layer:
Das Back-End stellt diverse Services bereit, auf welche die App im Laufe ihres Lebenszykluses zugreift und umgekehrt. Dies kann der Enterprise-Server des Unternehmens, eine Cloud-Storage Lösungen, auf welchen der Nutzer seine Zeiterfassungsdaten speichert und der App-Store, aus welchem der Nutzer die App herunterlädt oder Updates umfängt sein. Außerdem existiert ein MDM-System, welches eine VPN-Verbindung zum Unternehmen vorgibt und Cloud-Backups untersagen kann.

6.5 Umsetzung der Kriterien anhand ausgewählter Beispiele

Im Folgenden wird die Umsetzung der Kriterien der Entwicklungsphase anhand der Beispiele Berechtigungen, Benachrichtigungen und Trennung von Benutzerinterface und Anwendungslogik an der umgesetzten App gezeigt.

6.5.1 Berechtigungen und Background-Tasks

Die Zeiterfassungs-App benötigt einige Berechtigungen, die unter Windows Phone 8.1 im Package.appmanifest.xml deklariert werden müssen. Folgende Berechtigungen sind erforderlich:

- Zugriff auf das Internet
- Anzeigen einer Kachel auf dem Sperrbildschirm

Setzen der Berechtigungen

Der Zugriff auf das Internet ist erforderlich, um beispielweise Reports an das Unternehmensnetzwerk zu versenden. Diese Funktion ist in der ersten Version der App nicht realisiert. Es erfolgt ein Export via Mail bzw. im PDF-Format. An der Umsetzung der Funktion besteht jedoch großes Interesse, sodass sie im nächsten Entwicklungsschritt umzusetzen ist. Die vorzunehmenden Einstellungen für diese Berechtigung werden aus diesem Grund hier erläutert. Die VPN-Verbindung und Konfiguration wird, wie in Abschnitt 4.1.5 beschrieben, über ein Per-App VPN umgesetzt und eine der dort beschriebenen Methoden zur sicheren Kommunikation mit dem Unternehmensnetzwerk angewendet.
Die Anzeige einer Kachel auf dem Sperrbildschirm dient der Information des Nutzers über laufende Tasks und Projekte. Zur Realisierung einer Kachel auf dem Sperrbildschirm muss eine Hintergrundaufgabe deklariert werden.
Um Einstellungen an der Manifest-Datei vorzunehmen, ist es unter Visual Studio 2013 möglich, den grafischen Editor (Manifest-Designer) zu verwenden oder direkt die XML-Datei zu bearbeiten. Zur Deklaration von Berechtigungen verfügt der Manifest-Designer über die Registerkarten Anforderungen, Funktionen und Deklarationen.
Unter Anforderungen werden für die App erforderliche Hardwareressourcen festgelegt. Möglich sind hier Gyroskop, Magnetometer, NFC, Frontkamera sowie Rückseitige Kamera. Zu beachten ist, dass nicht alle Ressourcen auf jedem Endgerät vorhanden sind. Die Zeiterfassungs-App benötigt auf dem aktuellen Stand der Entwicklung keine dieser Ressourcen.
Unter dem Reiter Funktionen besteht die Möglichkeit, Systemfeatures oder Hardware auszuwählen, welche die App verwenden darf. Hier wird die Berechtigung „Internet (Client und Server)“ bestätigt, um der App eingehende und ausgehende Zugriffe auf das Internet zu ermöglichen. Unter Windows Phone werden Berechtigungen als Capabilities bezeichnet. Die Deklaration in der Manifest-Datei zeigt Listing 13.

```
<Capabilities>
    <Capability Name="internetClientServer" />
</Capabilities>
```

Listing 13 Deklaration der Berechtigung „Internet (Client und Server)“ unter Windows Phone

Auf dem Registerreiter Deklarationen werden, von der App auszuführende, Aufgaben festgelegt. Beispielsweise können Dateitypen registriert werden, welche von der App geöffnet werden können. Des Weiteren besteht die Möglichkeit, Hintergrundaufgaben zu definieren. Hintergrundaufgaben führen App-Code auch bei angehaltener App aus. Im Falle der Zeiterfassungs-App soll eine Kachel auf dem Sperrbildschirm angezeigt werden. Die App befindet sich im Hintergrund, während der Sperrbildschirm angezeigt wird. Aus diesem Grund ist die Deklaration einer Hintergrundaufgabe notwendig. Zunächst wird unter „Verfügbare Deklarationen“ eine Hintergrundaufgabe hinzugefügt. Im nächsten Schritt muss die Art der Aufgabe angegeben werden. Zur Auswahl stehen z.B. das Wiedergeben von Audio, das Zugreifen auf Sensoren und Geräte, das Durchführen der Überwachung auf Geofence-Ereignisse, das Reagieren auf Systemereignisse, das Reagieren auf Push-Benachrichtigungen und das Reagieren auf einen Timer im Hintergrund. Durch das Definieren eines Timers kann eine Hintergrundaufgabe in periodischen Abständen durchgeführt werden. Im vorliegenden Fall besteht die Hintergrundaufgabe aus der Kachelaktualisierung. Das minimale Aktualisierungsintervall beträgt dabei 15 min. Um kürzere Intervalle zu realisieren, besteht zum einen die Möglichkeit Pushbenachrichtigungen zu definieren. Diese werden von einem Cloud Service mit Hilfe des WNS in entsprechend kurzen Intervallen an die App versendet. Zum Zweiten können ScheduledTileNotification definiert werden siehe Abschnitt 4.2.1. Die App empfängt diese Benachrichtigungen und aktualisiert die Kachel entsprechend. Im Moment ist ausschließlich eine Aktualisierung durch einen Timer realisiert. Außerdem wird die Kachel erneuert, während die App aktiv ist und eine entsprechende Änderung an Tasks und Projekten vorgenommen wird.

Deklaration des Background-Tasks

Die Deklaration des Background-Tasks in der Manifest-Datei der App zeigt Listing 14.

```
<Extensions>
    <Extension Category="windows.backgroundTasks"
    EntryPoint="BackgroundTask.LockScreenTileBackgroundTask">
        <BackgroundTasks>
            <Task Type="timer" />
        </BackgroundTasks>
    </Extension>
</Extensions>
```

Listing 14 Deklaration eines Background-Tasks in der Manifest-Datei unter Windows Phone

Der Einstiegspunkt gibt die Klasse des Background-Tasks an, welche von IBackgroundTask abgeleitet werden muss und die Run-Methode implementiert.
Für die Ausführung des Background-Tasks sind folgende Schritte notwendig. Zunächst muss der Nutzer die App als sperrbildschirmfähig kennzeichnen. Das heißt, er muss sie unter den Systemeinstellungen als App, die auf dem Sperrbildschirm platziert wird, auswählen. Eine Hintergrundaufgabe kann ausschließlich erfolgreich registriert werden, wenn dies der Fall ist. Im Quellcode wird die „RequestAccessAsync“-Methode des „BackgroundExecutionManagers“, wie in Listing 15 zu sehen, aufgerufen.

```
var result = await BackgroundExecutionManager.RequestAccessAsync();
```

Listing 15 Erfragen der Berechtigung eines Background-Tasks unter Windows Phone

Damit wird eine Anforderung an das System gestellt, der App die Ausführung von Hintergrundaufgaben zu gestatten. Das Ergebnis der Abfrage „result“ ist vom Enum-Typ „BackgroundAccessStatus“. Die Anforderung kann abgelehnt werden, wenn die App vom Nutzer nicht als sperrbildschirmfähig ausgewählt bzw. noch keine Entscheidung getroffen wurde oder die maximale Anzahl von Background-Tasks, welche das System zulässt, überschritten wurde. Der Ergebniswert ist in diesem Fall „Denied“ oder „Unspecified“. Eine erfolgreiche Anforderung wird durch den Ergebniswert „AllowedMayUseActiveRealTimeConnectivity“ angezeigt. Nach einer erfolgreichen Überprüfung der Berechtigung wird die Hintergrundaufgabe registriert. Der Registrierungsprozess besteht aus den folgenden Schritten:

1. Überprüfung auf eine bestehende Registrierung und Rückgabe dieser, falls vorhanden
2. Definieren des Background-Tasks mit Hilfe einer Instanz der „BackgroundTaskBuilder“-Klasse; diese bekommt Name, Einstiegspunkt, Trigger und falls vorhanden, eine Bedingung der Ausführung des Tasks übergeben
3. Registrieren des Tasks durch Aufruf der Register-Methode des „BackgroundTaskBuilders“ und Rückgabe der Registrierung

Es können verschiedene Bedingungen angegeben werden, wann ein Task ausgeführt werden soll. Diese sind beispielsweise eine bestehende Internetverbindung, die Präsenz des Nutzers oder eine aktuell niedriger Workload von Hintergrundprozessen. Der Background-Task der Zeiterfassungs-App registriert einen „TimeTrigger“ der alle 15 min den Hintergrundprozess anstößt (siehe Listing 16). Eine Bedingung ist nicht erforderlich.

```
var registeredTask =
RegisterBackgroundTask("BackgroundTask.LockScreenTileBackgroundTask",
"LockScreenTileBackgroundTask", new TimeTrigger(15, false), null);
```

Listing 16 Registrieren des Background Tasks der Zeiterfassungs-App

Der zweite Parameter des „TimeTrigger“-Konstruktors gibt an, ob der Trigger einmalig ausgelöst werden soll. Eine mögliche Bedingung des Auslösens wird über den vierten Parameter der „RegisterBackgroundTask“-Methode angegeben.
Schließlich ist es möglich, die EventListener „OnCompleted“ und „Progress“ auf dem Task zu registrieren, um Aufgaben auszuführen, während der Task ausgeführt wird oder sobald er abgeschlossen ist. Die Zeiterfassungs-App implementiert den EventListener „OnCompleted“ und bringt alle Tiles auf den aktuellen Stand. Dies wird in Listing 17 dargestellt.

```
private void OnCompleted(IBackgroundTaskRegistration task,
BackgroundTaskCompletedEventArgs args)
{
    TileViewModel.UpdateAllTiles();
}
```

Listing 17 Updaten aller Tiles der Zeiterfassungs-App beim Komplettieren des Background-Tasks

6.5.2 Benachrichtigungen und Kacheln

Benachrichtigungen (Notifications) benötigt die Zeiterfassungs-App für das Erstellen und Updaten der Kacheln bzw. Tiles auf dem Startbildschirm und auf dem Sperrbildschirm. Eine Erläuterung zu Tiles unter Windows Phone erfolgt in Abschnitt 4.2.1. Unter Windows Phone

können Tile-Notifications sowohl von einem lokalen API-Aufruf, als auch von einem Cloud-Service ausgehen. Die Zeiterfassungs-App nutzt im Moment lokale Notifications.

Erstellen und Updaten der Tiles

Der Ablauf des Erstellens und Updatens eines Tiles wurde bereits in Abschnitt 4.2.1 gezeigt. Es folgen die durchzuführenden Schritte für die Zeiterfassungs-App.

1. Aktivieren der NotificationQueue
2. Setzen des XMLDocument mit dem TileTemplateType „TileSquare150x150Text01“
3. Einsetzen des Tile-Inhaltes in das XMLDocument
4. Erstellen der TileNotification. Diese enhält das XMLDocument als Inhalt
5. Updaten des Tiles durch den Aufruf der Update-Methode des „TileUpdateManagers“
6. Wiederholen der Schritte 2 bis 5 für den TileTemplateType „TileWide310x150Text01“

Unterschiede zu Abschnitt 4.2.1 bestehen im eingesetzten Text. Sind aktive Tasks vorhanden, so wird das Tile mit dem Projektnamen, dem Titel des aktiven Tasks, seiner Anfangszeit und seiner bisherigen Dauer geladen. Andernfalls wird ausschließlich der Titel des Projektes mit der spätesten Anfangszeit angezeigt. Außerdem wird im vorliegenden Fall ein Tile für die Größe 150x150 und eines mit der Größe 310x150 initialisiert. Das letztere ist ein breiteres Tile, welches der Größe zweier nebeneinander angeordneter Tiles der erstgenannten Größe entspricht (Wide). Unter Windows Phone kann der Nutzer auf dem Startbildschirm zwischen den beiden Größen wählen. Eine Unterstützung beider Größen ermöglicht eine bessere Nutzererfahrung.
Auf dem Sperrbildschirm ist es möglich, das breite Tile und/oder ein Badge, also ein Logo mit optionalem Counter, anzuzeigen. Der Entwickler hat die Aufgabe, in der Manifest-Datei unter der Registerkarte „Anwendung“ unter „Benachrichtigungen bei gesperrtem Bildschirm“ die Option „Text für Infoanzeiger und Kachel“ auszuwählen, um sowohl Tile als auch Badge für die Anzeige auf dem Sperrbildschirm freizuschalten. Daraufhin wird ein „LockScreen“-Tag als untergeordneter Knoten des „VisualElement“-Tags in der Manifest-Datei hinzugefügt (siehe Listing 18).

```
<m3:LockScreen Notification="badgeAndTileText"
BadgeLogo="Assets\Square58x58Logo.png" />
```

Listing 18 Lock-Screen-Tag in der Windows Phone Manifest-Datei

Das „Notification“-Attribut gibt an, dass sowohl ein Badge, als auch das breite Tile in der lokalen Notification enthalten sind. Außerdem wird das „DefaultTile“-Tag eingefügt. Dies ist in Listing 19 zu sehen.

```
<m3:DefaultTile Wide310x150Logo="Assets\WideLogo.png"
Square71x71Logo="Assets\Square71x71Logo.png">
```

Listing 19 DefaultTile-Tag in der Windows Phone Manifest-Datei

Hier muss ein breites Logo für das breite Tile des Sperrbildschirms angegeben werden. Des Weiteren ist die Registrierung einer Hintergrundaufgabe nötig, siehe Abschnitt 6.5.1.
Schließlich muss der Nutzer der Anzeige der App auf dem Sperrbildschirm zustimmen. Die Einstellung nimmt der Nutzer unter Einstellungen → Sperrbildschirm → Benachrichtigungen vor. Dort muss er die App für eine detaillierte Statusanzeige auswählen.

Zukünftige Einführung von Pushbenachrichtigungen

Eine Aktualisierung der Tiles in kürzeren Zeitabständen kleiner 15 min., um beispielsweise die momentane Dauer eines aktiven Tasks auf dem Tile aktuell zu halten, ist für einen späteren Entwicklungsschritt geplant. Weiterhin soll dem Unternehmen die Möglichkeit gegeben werden, neue Termine und Projekte direkt auf das Endgerät zu pushen. Eine Realisierung erfolgt über Pushbenachrichtigungen. Diese können beispielsweise über Microsoft Azure Mobile Services realisiert werden. Die notwendigen Komponenten sind

- ein mobile Service als BackEnd und Sender der Notifications,
- WNS als Kommunikationskomponente,
- Microsoft Azure Notification Hubs als Managementkomponente und
- die Mobile Applikation als Empfänger der Notifications.

Es erfolgt ein Hinzufügen des nötigen Codes für den mobilen Service und die App sowie die Registrierung der App für den Azure Notification Hub. Nach erfolgreicher Einrichtung werden Nachrichten durch den Azure Notification Hub vom BackEnd an die App mit Hilfe des WNS versendet. Ein detailliertes Vorgehen wird an dieser Stelle aus Platzgründen nicht gegeben. Es sei darauf hingewiesen, dass die Erzeugung des BackEnd- und App-Codes sowie die Registrierung für den Entwickler über Visual Studio automatisiert vorgenommen werden können und die Arbeit somit deutlich erleichtert wird.[19]

6.5.3 Trennung von Benutzerinterface und Anwendungslogik

Die Grundlagen zu einer Trennung von Benutzerinterface und Anwendungslogik unter Windows Phone 8.1 sind Abschnitt 4.5 zu entnehmen. Die Zeiterfassungs-App orientiert sich an den beschriebenen Richtlinien und versucht diese bestmöglich umzusetzen. Um dies zu zeigen, wird der Aufbau der Zeiterfassungs-App anhand des Klassendiagramms in Abbildung 12 veranschaulicht. Die Aufteilung gemäß des MVVM-Patterns wird deutlich.

Model

Das Model, in Form von Projekten und Zeitintervallen (Tasks), beinhaltet die Businessdaten bzw. Fachkonzeptklassen. Um keine Änderungen am Business Model vornehmen zu müssen und die Daten dennoch optimal für die Anzeige aufzubereiten, wird es gekapselt. Dafür sind die ViewModel-Klassen zu erstellen.

ViewModel

Das ProjektViewModel speichert zunächst das ihm zugeordnete Projekt und dem Projekt zugehörige Zeitintervalle. Zeitintervalle werden wiederum in TimeIntervalViewModels gekapselt. Das ProjectViewModel kapselt die einzelnen Attribute der Project-Klasse, wie UniqueId, Title, Description usw. Für jedes Attribut wird eine Getter- bzw. Setter-Methode erstellt. Außerdem kommen neue Attribute hinzu, welche für das spätere Binding in der View benötigt werden. Der ActiveTask beispielsweise hält den bereits gestarteten Task. Dieser besitzt ein Startdatum, jedoch kein Enddatum und wird aus den zu diesem Projekt gehörenden

[19] Das detaillierte Vorgehen wird unter https://azure.microsoft.com/en-us/documentation/articles/mobile-services-dotnet-backend-windows-universal-dotnet-get-started-push/ geschildert.

Zeitintervallen ermittelt. Das bereits verbrauchte Budget (SpentBudget) wird aus der bisher benötigten Zeit sowie den einzelnen Tagessätzen (DailyRates) berechnet. Jedes Zeitintervall besitzt genau einen Tagessatz. Außerdem werden projektspezifische Funktionen (Commands), wie das Editieren und das Löschen eines Projektes, definiert. Diese können von einem UI-Element direkt ausgelöst werden. Das beschriebene Prinzip zum Erstellen der ProjectViewModels wird ebenfalls auf die TimeIntervallViewModels angewandt.

Abbildung 12 Klassendiagramm der Zeiterfassungs-App

Die oben beschriebenen ViewModel-Klassen kapseln unmittelbar das Model und sind aus diesem Grund architektonisch näher am Model angesiedelt. Das AllProjectsViewModel und das AllTimeIntervalsViewModel dienen der Aufbewahrung der ViewModel-Klassen der unteren Ebene und sind direkt an die Pages angepasst, für deren Binding sie verantwortlich sind. Sie sind demnach eng mit der View verbunden. Das AllProjectsViewModel hält Attribute und Commands, die in der MainPage benötigt werden. Dazu gehören z.B. eine Collection, die alle Projekte in Form von ProjectViewModels hält sowie Commands, die Projekte hinzufügen oder neue Tasks erzeugen, starten, stoppen und pausieren. Außerdem ist die Klasse für das Laden und Speichern der App-Daten verantwortlich. In einer späteren Version der App ist es geplant, diese Funktionalität in eine separate Klasse auszulagern.
Das TimeIntervalsViewModel dient der detaillierten Darstellung von Projekten und Zeitintervallen, nachdem auf der MainPage ein spezifisches Projekt ausgewählt wurde. Aus diesem Grund wird hier das aktuell ausgewählte Projekt sowie das aktuell ausgewählte Zeitintervall, falls vorhanden, aufbewahrt. Ein detaillierter Navigationsverlauf befindet sich im Abschnitt der Beschreibung der View. Außerdem sind Commands für das Editieren von Projekten und Zeitintervallen implementiert. Dies geschieht in der EditProjectsPage bzw. EditTimeIntervalsPage.

View

Der Einstiegspunkt einer mobilen Applikation ist unter Windows Phone die Klasse „App“. Sie erbt von der Oberklasse „Application“. Diese ist für die Kapselung der Windows Phone Anwendung notwendig und gibt grundlegende Events, wie die OnLaunched-Methode, welche beim Starten der App aufgerufen wird, vor. Die App-Klasse realisiert Abläufe, die den Lebenszyklus der App betreffen. Sie implementiert beispielsweise die OnLaunched-Methode und die OnSuspending-Methode, welche ausgeführt wird, wenn die Ausführung der Anwendung angehalten wird. Außerdem werden Attribute deklariert, welche global und während des gesamten Lebenszykluses der App benötigt werden. Dies sind beispielsweise die einzelnen Pages und die beiden ViewModel-Klassen AllProjectsViewModel und AllTimeIntervalsViewModel. Die App-Klasse ist eine partielle Klasse und bildet mit der App.xaml eine Einheit. In dieser werden global benötigte Templates abgelegt.
Die View besteht aus der App-, MainPage-, TimeIntervalsPage-, EditProjectsPage- und EditTimeIntervalsPage-Klasse. Jede dieser Klassen stellt eine Page dar. Pages dienen der einfachen Navigation untereinander. Sie bestehen jeweils aus dem CodeBehind-File in C# und der zugehörigen XAML-Datei, in welcher die UI definiert ist. Sie implementieren unter anderem die Event-Listener „OnNavigatedTo“ sowie „OnNavigatedFrom“ im CodeBehind-File, um Aufgaben auszuführen, sobald die Page betreten bzw. verlassen wird. Bei einer Navigation auf die Page können beispielsweise Navigationsparameter übernommen werden, um den Zustand der Seite in Abhängigkeit der Navigationsquelle festzulegen. Beim Verlassen werden Ressourcen freigegeben. Im CodeBehind werden Event-Listener für UI-Elemente implementiert. Ein Beispiel ist die Elementauswahl innerhalb einer ListView wie die Methode „ItemView_ItemClick“ siehe Abbildung 13 auf der MainPage. Dies ist notwendig, da das ListView-Control das Binding eines Commands aus dem ViewModel nicht unterstützt. Außerdem wird im CodeBehind der Zustand der Seite gehalten und UI-Elemente entsprechend diesem angepasst. Mehr zu Pages und den einzelnen UI-Elementen ist Abschnitt 4.5.3 zu entnehmen.

Navigation

Der Navigationsverlauf zwischen den einzelnen Pages ist in Abbildung 13 dargestellt. Es sei erwähnt, dass sowohl für das Anlegen neuer, als auch das Bearbeiten vorhandener Projekte bzw. Tasks, die EditProjectsPage bzw. EditTimeIntervalsPage verwendet wird. Hier wird lediglich der Zustand der Seite angepasst und die UI-Elemente werden entsprechend konfiguriert.

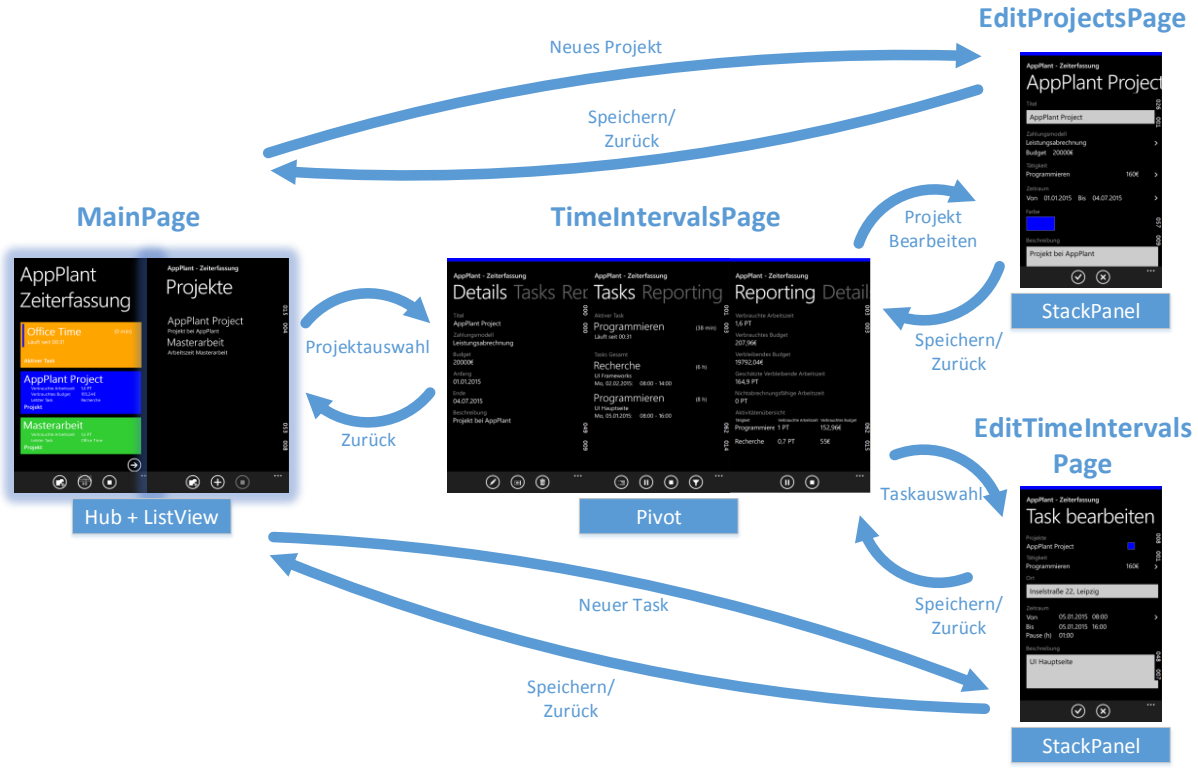

Abbildung 13 Navigationsfluss zwischen den einzelnen Pages der Zeiterfassungs-App

Die einzelnen Pages haben folgende Funktionen:

MainPage:

- definiert den Einstiegspunkt in der App-Klasse
- realisiert als Hub-Steuerelement bestehend aus:
  - Start: Anzeige von aktiven Tasks und aktuellen Projekten
  - seitliches Scrollen: Anzeige aller Projekte als Liste (ListView)

TimeIntervalsPage:

- Pivot-Steuerelement mit Unterseiten: Projektdetails, Tasks und Reporting
  - Projektdetails als Auflistung (StackPanel)
  - Tasks als ListView
  - Reporting als Auflistung (StackPanel)
- Filtern von Tasks in einem Zeitraum
- Übergang zu:
  - Erstellen von neuen Tasks für das aktuelle Projekt
  - Bearbeiten eines Vorhandenen Tasks
  - Bearbeiten von Projektdetails

EditProjectsPage/ EditTimeIntervalsPage:

- Anzeige/Bearbeiten der Projektdetails/ Taskdetails
- Löschen des Projektes/ Tasks
- Speichern von Änderungen
- Abbrechen

6.6 Beurteilung der Tauglichkeit für den Enterprise-Einsatz

Im Folgenden findet die Bewertung der mobilen Applikation anhand des Kriterienkatalogs aus Abschnitt 5.5 statt. Dies entspricht der Entwicklungs- und Einsatzphase. Die detaillierte Bewertung einzelner Kriterien dieser Phase ist Anlage D zu entnehmen. Schließlich wird durch die Aggregation mit der Bewertung der Vorentwicklungsphase aus Abschnitt 5.4 eine Einschätzung der Tauglichkeit für den Enterprise-Einsatz gegeben.

6.6.1 Bewertung der mobilen Enterprise Applikation

a. Sicherheit

Zunächst sind die gesetzten Berechtigungen der App darauf zu prüfen, ob sie in einem angemessenen Verhältnis zur bereitgestellten Funktionalität der App stehen. Eine Bewertung ist an dieser Stelle eine Ermessensfrage. Es ist hilfreich, für jede Berechtigung die folgenden Sachverhalte zu notieren:

- allgemeine Bedeutung:
    - Welche Funktion stellt die Berechtigung allgemein zur Verfügung? Diese Frage lässt sich meist über die Dokumentation beantworten.
- Bedeutung im aktuellen Kontext
    - Welche Funktionen stellt die Berechtigung für die aktuelle App zur Verfügung?
- Stehen diese Funktionen in Zusammenhang mit einer vom Nutzer erwarteten Funktion dieser App?
    - Erwartet der Nutzer die oben genannten Funktionen, wenn er diese App installiert? Wenn nein, dann muss eine Begründung erfolgen, warum diese Funktion dennoch für die Funktionalität der App benötigt wird oder hilfreich ist.
- Kontrolle über diese Funktion
    - Besitzt der Nutzer Kontrolle über diese Funktion, oder wird sie automatisiert ausgeführt? Kann der Nutzer diese Funktion deaktivieren? Eine automatisierte Funktion, die nicht deaktiviert werden kann, sollte für die Funktionalität der App stets unerlässlich und dem Nutzer bekannt sein.
- ausführliche und verständliche Formulierung für den Nutzer
    - Hier muss eine ausführliche Beschreibung für den Nutzer erfolgen. Warum sollte er dieser Berechtigung zustimmen? Ist die Berechtigung unerlässlich für die Funktion der App, oder stellt sie eine hilfreiche zusätzliche Funktionalität zur Verfügung?

Die Zeiterfassungs-App benötigt die Berechtigung für das Internet. Dadurch kann sie Abrechnungen über eine VPN-Verbindung an den Auftraggeber bzw. das eigene Unternehmen versenden. Da dies unter den Funktionen der App vermerkt ist, sollte der Nutzer diese Funktion erwarten. Der Nutzer hat die Möglichkeit, Abrechnungen zu versenden. Für die Funktion der App ist dies allerdings nicht zwangsläufig erforderlich. Es stellt also eine hilfreiche zusätzliche Funktionalität dar. Der Nutzer kann selbst entscheiden, wann er diese Funktion ausführt.
Zusätzlich wird die Berechtigung für den Background-Task benötigt. Dieser führt App-Code auch bei angehaltener App aus. Er wird für die Anzeige der Sperrbildschirmkachel benötigt, um dem Nutzer aktuelle Informationen über Tasks und Projekte anzuzeigen. Es handelt sich demnach um eine hilfreiche zusätzliche Funktionalität, die nicht unbedingt erforderlich ist. Unter Windows Phone 8.1 muss der Nutzer dieser Berechtigung separat über die Sperrbildschirmeigenschaften zustimmen. Möchte er eine Anzeige auf dem Sperrbildschirm, so stimmt er explizit zu. Er kann seine Entscheidung jeder Zeit widerrufen.
Die Berechtigungen sind plausibel begründbar. Entsprechende Funktionen werden stets mit Wissen des Nutzers ausgeführt. Das Kriterium ist vollständig erfüllt.
App-Daten können dank der KeyChain separat verschlüsselt werden. Dieses Kriterium erfüllt die App nicht. Es wird die vom BS genutzte Dateisystemverschlüsselung vorausgesetzt.
VPN-Verbindungen werden über gesicherte Verbindungen mittels IPsec-Standard aufgebaut.
Eine eigene Zugriffsauthentifizierung, bzw. eine 2-Faktor-Authentifzierung, ist nicht vorhanden. Die Sicherheit der Daten ist damit so groß wie das gewählte Authentifizierungsverfahren des BS. Entsprechende Vorgaben erfolgen über das MDM.
Die App nutzt im Moment keine weiteren Services von Drittanbietern, sodass der Punkt SSL/TLS-Verschlüsselung erfüllt ist.
Die App wird über den Enterprise App-Store des Unternehmens verteilt und ist damit sicher vor Manipulationen. Je nach Anstellung kann sich jeder Mitarbeiter die benötigten Apps installieren. Die App durchläuft vor der Bereitstellung den Review-Prozess des Unternehmens. Sie läuft vollkommen eigenständig. Die Nutzung von Services anderer Apps ist nicht geplant.
Das Unternehmen hat im Zuge des Einstiegs in die Nutzung einer mobilen Infrastruktur umfassende Sicherheitsrichtlinien erlassen. Dazu gehören eine Sicherheits- und Risikoanalyse sowie eine Sicherheitsbelehrung der Mitarbeiter. Außerdem werden für den Enterprise Einsatz genutzte Geräte auf Minimalanforderungen geprüft. Die Fragmentierung der Geräte unter Windows Phone ist deutlich geringer, als beispielsweise unter Android. Minimalvoraussetzung ist die Unterstützung des Windows Phone 8.1 BS.
In diesem Bereich erreicht die App eine Bewertung von 0,83. Abzüge entstehen vor allem durch die fehlende Authentifizierung und Verschlüsselung innerhalb der App. Diese werden auf das BS ausgelagert.

b. Benachrichtigungen

Für das Versenden von Notifications ist in einer späteren Version die Nutzung des WNS geplant siehe Abschnitt 6.5.2. Aus diesem Grund wird der WNS bereits hier in die Bewertung der mobilen Applikation aufgenommen und trägt zur Gesamtbewertung bei. Außerdem soll beispielhaft die Auswirkung einer Nutzung auf die Gesamtbewertung gezeigt werden. Eine Auflistung der Eigenschaften zeigt Tabelle 5. Damit ergibt sich eine Bewertung von $1/3$. Grund für diese Bewertung ist die fehlende Unterstützung für die Nachverfolgung von Nachrichten. Sowohl Empfangsbestätigungen, als auch Statusabfragen, sind nicht möglich. Außerdem ist der

Dienst Cloud-abhängig, was ein Sicherheitsrisiko darstellt. Der Dienst eignet sich nicht für das Versenden sicherheitskritischer Informationen.

c. Mobile Cloud Computing – Cloud Speicher

Im Moment nutzt die App keinen Cloud Speicher. Alle Daten sind auf dem Endgerät gespeichert. Es wird eine Funktion für ein Backup der App-Daten zur Verfügung gestellt. Dabei werden die Daten in einer JSON-Datei auf dem internen Dateisystem an einem benutzerdefinierten Ort abgelegt und können über eine Wiederherstellungsfunktion wieder in die App geladen werden. Es ist möglich, diese Daten manuell auf einem Desktop-PC zu sichern. Theoretisch besteht die Möglichkeit, dieses Backup in einem Speicherbereich abzulegen, welcher über einen Cloud-Service synchronisiert wird. Da ein externer Cloud-Speicher über das MDM-System ausgeschlossen wird, ist dies im aktuellen Anwendungsfall allerdings nicht möglich. Nachteile ergeben sich durch den Verlust von Daten, sollte das Gerät verloren gehen. Die Mitarbeiter sind stets selbst dafür verantwortlich, ein Backup durchzuführen.
Es besteht die Möglichkeit, eine negative Bewertung zu erwägen, da ein Datenverlust droht bzw. eine positive Bewertung zu bevorzugen, da das Risiko des Abflusses von Daten an Dritte verringert wird. Eine Entscheidung kann in diesem Fall nicht eindeutig argumentiert werden. Aus diesem Grund und da sich die Kriterien stets auf die Nutzung eines Cloud-Services beziehen, wird dieser Teilaspekt nicht bewertet.
Der Umstand, dass kein Backup stattfindet, wird in Abschnitt d. MDM berücksichtigt und als negativ bewertet.

d. Mobile Device Management

Das Unternehmen nutzt den Microsoft System Center 2012 R2 Configuration Manager und Windows Intune als MDM-Lösung. Laut Gartners Magic Quadrant for Client Management (Juni 2015) zählt der Configuration Manager von Microsoft zu den führenden Systemen für die automatisierte Systemadministration. Er wird für Unternehmen mit vorhandener Microsoft Infrastruktur empfohlen. Stärken sind eine hohe Skalierbarkeit und eine gut Unterstützung durch Softwareanbieter und Provider. Nachteil ist der schlechte Support von Applikationen und Geräten, die nicht von Microsoft stammen.
Das Management von Geräten wie PCs, Laptops, Tablets und Smartphone geschieht über Windows Intune. Der Configuration Manager sendet Konfigurationen an und empfängt Statusnachrichten von Intune, die dieses von den Geräten empfängt. Intune dient dabei als Gateway, das mit Geräten kommuniziert und Einstellungen speichert (vgl. MICROSOFT, 2014e).
Die Konfiguration der Geräte erfolgt wie in Abschnitt 4.4.1 a beschrieben. Es erfolgt ein klassisches MDM ohne zusätzliche Container-App.
Die Unternehmensstrategie ist ein COPE-Ansatz. Microsoft Geräte werden vom Unternehmen an die Mitarbeiter ausgegeben.
Der MDM-Ansatz bietet OTA-Management von Geräten. Es können WLAN- und VPN-Konfigurationen übertragen werden. Dabei wird das Per-App-VPN aktiviert, um stets eine sichere Verbindung zum Back-End zu gewährleisten.
Durch das Betreiben eines Enterprise App-Store wird die Menge verfügbarer Apps eingeschränkt.
Ein Backup von Gerätedaten findet im Moment nicht statt. Damit werden Sicherheitsrisiken, wie das Abfließen von Daten an externe Cloud-Services, unterbunden. Es wird angestrebt, eine

interne Cloud-Lösung umzusetzen. Bis zu deren Realisierung sollten Mitarbeiter ein manuelles Backup der Daten durchführen.
Die Sicherheitsrichtlinien des Unternehmens versichern den Mitarbeitern, ausschließlich auf geschäftliche Daten zuzugreifen. Workplace benachrichtigt den Nutzer jedoch über den potenziellen Zugriff des Unternehmens auf die persönlichen Daten (falls es die Sicherheitsrichtlinien dem Unternehmen erlauben). Durch Containerization könnte der persönliche und der geschäftliche Bereich besser voneinander getrennt werden. Ein Restrisiko besteht demnach für den Mitarbeiter weiterhin. Workplace erlaubt es, beim Entfernen des Accounts ausschließlich geschäftliche Daten zu löschen (Partial Wipe).
Die Sicherheitsrichtlinien des Unternehmens bestehen auf eine aktivierte Dateisystemverschlüsselung. Dafür wird die in Windows Phone 8.1 integrierte BitLocker-Technologie aktiviert. Außerdem werden Passwortvorgaben vorgegeben für Art, Komplexität, minimale Länge, maximale Anzahl falscher Passworteingaben vor einem Device Wipe und Auslaufdatum.
Intune erlaubt ein ständiges Überwachen von Geräten über Status- und Inventarisierungsnachrichten.
Dank der soliden MDM-Lösung von Microsoft, wird eine Bewertung von 0,87 erreicht. Kritikpunkte sind einzig das fehlende Backup. Außerdem sorgt eine Containerlösung für eine striktere Trennung von privaten und geschäftlichen Daten, als das klassische MDM.

e. Benutzer Interface und Trennung der Anwendungslogik

Die App wurde auf Grundlage des MVVM-Patterns realisiert. Dies wird von Windows Phone explizit unterstützt. Die Definition der UI erfolgt durch die XAML-Beschreibungssprache. Features wie TwoWay-Bindings, Converter, Commands, NotifyPropertyChanged-Interface unterstützen die Umsetzung. Es wurde auf die Einhaltung der Microsoft Designrichtlinien geachtet (siehe auch Abbildung 13). Es existiert eine Trennung von Daten (Model) und Logik (ViewModel). Eine komplette Unabhängigkeit von UI (View) und Anwendungslogik existiert allerdings nicht. Es wurde versucht, so wenig wie möglich Logik in den Code-Behind Dateien zu implementieren. Eine vollständige Trennung konnte nicht erreicht werden.
Es werden die Sprachen Deutsch und Englisch unterstützt. Außerdem werden alle aktuell auf der Windows Phone Plattform existierenden Auflösungen unterstützt. Es wird eine gleichbleibende Nutzererfahrung gewährleistet.
Mit 0,82 erreicht die App in diesem Bereich ein gutes Ergebnis. Lediglich die fehlende Trennung von UI und Logik ist ein Kritikpunkt.

Gesamtbewertung

Die Gesamtbewertung wird über das in Abschnitt 5.2 eingeführte Bewertungssystem durchgeführt. Die Bewertung der Teilaspekte ergibt sich aus Definition 4. Die Gesamtbewertung folgt aus Definition 5. Tabelle 21 zeigt Bewertung und Gewichtung der einzelnen Teilaspekte.
Die Teilaspekte Benachrichtigungen und Cloud Storage erhalten aus folgenden Gründen eine geringere Gewichtung. Die bisherige Lösung nutzt keine Cloud Storage Lösung und speichert Daten lediglich intern bzw. exportiert diese. Benachrichtigungen besitzen in der vorliegenden App eine zweitrangige Bedeutung, da sie ausschließlich für das Aktualisieren der Kachel für den Sperrbildschirm genutzt werden.

Teilaspekte	Bewertung	Gewichtung
Sicherheit	0,83	7/24
Benachrichtigungen	0,33	1/8
Cloud Storage	-	0
Mobile Device Management	0,87	7/24
UI und Trennung von der Anwendungslogik	0,82	7/24
Gesamt	**0,78 - Gut**	

Tabelle 21 Bewertung der Entwicklungs- und Einsatzphase nach Teilaspekten

6.6.2 Zusammenfassende Bewertung der Phasen

Die Bewertung der mobilen BSs der Vorentwicklungsphase und die Bewertung der mobilen Applikation der Entwicklungs- und Einsatzphase werden in Tabelle 22 zusammengefasst, um eine Einschätzung der Tauglichkeit der mobilen Applikation für den Enterprise-Einsatz zu ermöglichen.
Die Zusammenfassung erfolgt anhand der Formel für die Gesamtbewertung der einzelnen Phasen nach Definition 5. In diesem Fall werden Teilaspekte durch Phasen getauscht.

	Bewertung	Gewichtung
Vorentwicklungsphase	0,86	0,5
Entwicklungs- und Einsatzphase	0,78	0,5
Gesamt	**0,82 - Gut**	

Tabelle 22 Bewertung der einzelnen Phasen und Gesamtbewertung

6.6.3 Schlussfolgerung

Eine Gesamtbewertung von 0,82 bescheinigt der App unter der Berücksichtigung ihrer Umgebung eine gute Tauglichkeit für den Enterprise-Einsatz. Über Definition 6 ergibt sich eine Note von 2.0. Das heißt, die Sicherheitsmaßnahmen beteiligter Teilsysteme und die Struktur der App genügen den Anforderungen mittelgroßer Unternehmen siehe Anlage B. In der vorgestellten Fallstudie wird ein Großunternehmen betrachtet. Ein Einsatz ist durchaus denkbar. Es wird jedoch empfohlen, einige Verbesserungen vorzunehmen. Beispiele sind:

- eigene Authentifizierung innerhalb der App und/oder 2-Faktor-Authentifizierung an den Endgeräten
- Verschlüsselung entsprechender Authentifizierungsdaten
- Durchführen eines Backups von Gerätedaten
- verbessertes Design der App durch die Vermeidung von Code-Behind
- Umstieg auf einen zuverlässigen Notification Service
- Nutzung einer private Cloud falls nötig

7 Fazit und Zukunftsaussichten

Die vorliegende Arbeit beschäftigt sich mit der Erarbeitung von Kriterien, nach denen eine native mobile Enterprise Applikation auf ihre Tauglichkeit für den Unternehmenseinsatz geprüft werden kann. Nach einer grundlegenden Einführung in die Thematiken Mobilität, mobile Applikationen und Plattformen werden zentrale Aspekte der Entwicklung und des Einsatzes von Apps im Unternehmensumfeld untersucht. Vertiefend werden die Teilaspekte Sicherheit, Benachrichtigungen, Berechtigungen, Mobile Cloud Computing, Mobile Device Management sowie Benutzerinterface und Trennung von Anwendungslogik vorgestellt. Es folgt die Konstruktion eines Kriterienkatalogs. Dieser besteht in einer ersten Phase aus Kriterien zur Bewertung der mobilen Plattform, auf welcher die App lauffähig ist. Beispielhaft werden die Systeme Android 5, iOS 8, Windows Phone 8.1 und BlackBerry 10 bewertet. In einer zweiten Phase werden App-spezifische Kriterien erarbeitet. Durch die Zusammenarbeit mit der Firma appPlant ist eine mobile Enterprise Applikation zur Zeiterfassung von Arbeitszeiten. Die App wird beispielhaft auf die Umsetzung einzelner Kriterien überprüft, bewertet sowie auf ihre Tauglichkeit für den Unternehmenseinsatz geprüft.

In diesem Abschnitt werden die Ergebnisse der Arbeit aufgelistet und Kritik vorgebracht. Außerdem erfolgt das Einbringen von Verbesserungsvorschlägen sowie Ideen, die aufgrund der begrenzten Entwicklungszeit nicht umgesetzt werden konnten.

7.1 Ergebnisse

Die folgenden Ergebnisse wurden anhand der schriftlichen Arbeit und der Beispielumsetzung einer mobilen Enterprise Applikation erreicht. Die Ergebnisse sind nach den Schwerpunkten der Arbeit gegliedert.

Analyse des Ökosystems:

- Die Umgebung mobiler Enterprise Applikationen wurde analysiert und ein zusammenfassender Überblick in Form einer Visualisierung und Beschreibung erarbeitet.
- ➢ Das Resultat ist ein umfassendes Verständnis über native mobile Enterprise Apps und beteiligter Teilsysteme zur Entwicklung, zum Management und zur Verteilung dieser Apps.

Analyse und Umsetzung von Anforderungen:

- Die Anforderungen an mobile Enterprise Applikationen, inklusive eingesetzter Technologien, wurden recherchiert und beschrieben. Dabei wurden exemplarisch Umsetzungen dieser Anforderungen auf den mobilen Plattformen Android, iOS, Windows Phone und Blackberry aufgezeigt.
- ➢ Daraus folgt ein umfassendes Verständnis benötigter Mechanismen und Technologien mobiler Enterprise Applikationen. Außerdem wird der Zusammenhang zwischen diesen Mechanismen und den eingesetzten mobilen Plattformen deutlich.

Erstellen eines Kriterienkataloges:

- Aus den Anforderungen wurde ein Kriterienkatalog für die Bewertung mobiler Enterprise Applikationen abgeleitet.
- Eine Begründung für die Notwendigkeit der Aufteilung des Katalogs in zwei Phasen wurde gegeben. Grundlegend wurde festgestellt, dass die Möglichkeit der Umsetzung von Anforderungen nicht ausschließlich durch den Entwickler, sondern ebenfalls durch die Wahl der mobilen Plattform entschieden wird. Eine detaillierte Begründung ist den Abschnitten 5.1 und 5.6 zu entnehmen. Der Katalog bewertet zunächst die mobile Plattform in der sogenannten Vorentwicklungsphase und befasst sich in einem zweiten Schritt mit der Entwicklung und Umgebung der mobilen Applikation in der Entwicklungs- und Einsatzphase.
- ➢ Der Katalog dient der Analyse einer mobilen Applikation im Hinblick auf Anforderungen der Unternehmen. Durch die Aufteilung in Teilbereiche kann eine detaillierte und schwerpunkthafte Einschätzung erfolgen.

Erstellen eines Bewertungssystems:

- Ein Bewertungssystem für den Kriterienkatalog wurde eingeführt.
- Die Einführung umfasst die mathematisch genaue Beschreibung sowie Begründung des Systems.
- ➢ Das Resultat besteht darin, die Tauglichkeit einer mobilen App für den Enterprise-Bereich bewerten zu können.

Entwicklung:

- In einer Fallstudie wurde eine mobile Enterprise Applikation für die Zeiterfassung von Arbeitszeiten und Projekten unter Windows Phone 8.1 entwickelt und vorgestellt.
- Die App setzt Anforderungen aus dem Enterprise-Bereich, wie beispielweise das Einsetzen etablierter Pattern zur Trennung von UI und Anwendungslogik und das Nutzen von Notifications um. Außerdem wird eine der untersuchten mobilen Plattformen eingesetzt.
- ➢ Resultate sind
    - die Umsetzung einer mobilen Enterprise Applikation als Praxisanteil und das Kennenlernen des Entwicklungsprozesses sowie beteiligter Technologien aus der Sicht des Entwicklers,
    - die Möglichkeit der Evaluation der App durch den Kriterienkatalog und damit das Anwenden des Katalogs auf eine mobile Enterprise Applikation,
    - ein realer Einsatz der App im Unternehmensumfeld.

Fallstudie und Evaluation:

- Anhand der entwickelten App wurde die Anwendung des Kataloges und des Bewertungssystems für die Evaluierung einer mobilen Enterprise Applikation exemplarisch gezeigt.
- Die Zeiterfassungs-App wurde anhand des Kriterienkatalogs evaluiert und das Ergebnis der Bewertung einzelner Phasen vorgestellt.
- Es wurde gezeigt, dass das Aufdecken von Schwächen und Stärken durch die Bewertung einer App anhand des Kataloges möglich ist.

- Das Resultat ist die Unterstützung des Entwicklers bei seiner Arbeit, indem er gezieltes Feedback innerhalb einzelner Teilbereiche erhält. Durch die Beschreibung von Beispieltechnologien und -umsetzungen erhält er gleichzeitig Anhaltspunkte zur Verbesserung der App.
- Schließlich ist es auf Unternehmensseite möglich, die Gesamtbewertung bzw. die Bewertung einzelner Teilbereiche zu analysieren und die Tauglichkeit für den eigenen Einsatz einzuschätzen.

7.2 Verbesserungsvorschläge und Zukunftsaussichten

An dieser Stelle werden Kritik und Verbesserungsvorschläge eingebracht. Durch den iterativen Prozess der Entwicklung der Kriterien und der mobilen Applikation entstanden einige interessante Ideen zur Verbesserung und Weiterentwicklung erst gegen Ende der Arbeit. Diese werden hier aufgezeigt.

Kritik und Verbesserungen an den Anforderungen:

- Es konnten nicht alle Anforderungen in gleichem Umfang untersucht werden. Der Teilaspekt MDM wurde ausführlich behandelt. Anforderungen des MAM und MCM sind weniger stark in den Kriterien vertreten. Die Erweiterung um zusätzliche Aspekte trägt zu einem aussagekräftigeren Ergebnis bei. Beispiele sind:
  - Performanz und Energieeffizienz
  - Multi-Plattform-Unterstützung
  - Personalisierung
- Es stellte sich heraus, dass die Sicherheit einer mobilen Applikation stark abhängig von der Vermeidung einiger typischer Sicherheitslücken, wie z.B. Jailbreaks, fehlerhafte Ableitung kryptographischer Schlüssel und fehlerhafte Nutzung der APIs ist. Ursache der Fehler kann ein fehlendes Hintergrundwissen der Entwickler in diesem Bereich sein. Aus diesem Grund ergibt sich die Idee einer detaillierten Auseinandersetzung mit typischen Sicherheitslücken und Ursachen. Die Ergebnisse machen eine genauere Analyse im Bereich Sicherheit möglich.
- Der Teilaspekt MCC wurde ausschließlich im Hinblick auf Cloud-Speicher untersucht. Hier besteht die Möglichkeit, weitere Services einzubeziehen bzw. die Kriterien allgemeingültig zur Bewertung verschiedener Cloud Services zu formulieren.

Kritik und Verbesserungen an dem Kriterienkatalog und der Bewertung:

- Die Anzahl der Kriterien der einzelnen Teilaspekte des Katalogs unterscheidet sich teilweise deutlich. Um ein aussagekräftiges Ergebnis zu erzielen, sollten die Gewichte der Teilaspekte entsprechend der Summe der Gewichte der Kriterien sowie der Relevanz des Teilaspektes angepasst werden.

- Die Phase Entwicklung- und Einsatz beinhaltet Teilaspekte, wie Cloud-Speicher und Notification Services. Das Vorgehen der Bewertung bei einer Nutzung mehrerer oder keiner dieser Services wurde nicht restlos aufgezeigt. Mögliche Lösungen sind:
    - Kein Service: Wählen einer Gewichtung des Teilaspektes von 0.
    - Mehrere Services: Aufteilen des Teilaspektes und der entsprechenden Gewichtung.

Kritik und Verbesserungen an der Zeiterfassungs-App:

- Das MVVM-Pattern diente als Grundlage der App-Struktur. Hier ergaben sich einige Verbesserungsvorschläge, wie das Einführen separater Controller-Klassen, um Code-Behind zu vermeiden und eine bessere Trennung von View und View-Logik zu erreichen.
- Die Weiterentwicklung der App soll die Synchronisation mit Cloud Speicher Lösungen gestatten. Dabei wird eine private Cloud Lösung empfohlen.

Zukunftsaussichten

Ende 2015 stellen die Plattformhersteller Apple, Microsoft und Google neue Versionen ihrer mobilen BSs vor. Genaue Veröffentlichungstermine sind nicht bekannt. Apple wird mit iOS 9 auf die Steigerung von Stabilität, Leistung und Sicherheit der vorhandenen Features setzen. Grundlegende Neuerungen sind bis jetzt nicht bekannt. Windows 10 Mobile, als Nachfolger von Windows Phone 8.1, setzt auf das Zusammenwachsen von PC und Smartphone. Zukünftig soll ein einheitlicher Betriebssystemkern existieren. Das Prinzip der Universal Apps soll ausgebaut werden, sodass dieselbe App, sowohl auf dem Desktop, als auch auf dem Smartphone läuft und dabei lediglich die Views angepasst werden. Android 6 (M) heißt das neue Google System. Hier werden Neuerungen am Berechtigungssystem vorgenommen, sodass dem Nutzer erweiterte Kontrollmöglichkeiten zugestanden werden. Unter anderem wird der Nutzer befragt, sobald eine Berechtigung benötigt wird. Dies wird zu deutlichen Verbesserungen in der Bewertung des Android Systems führen. Zu Neuerungen des Blackberry BS sind im Moment keine aussagekräftigen Äußerungen bekannt.

Literaturverzeichnis

APPLE (2012a). Concepts in Objective-C Programming. Model-View-Controller. *https://developer.apple.com/library/ios/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html* (11.01.2016).

APPLE (2012b). Implementierung von iPhone und iPad. Virtuelle private Netzwerke (VPN). *https://www.apple.com/de/ipad/business/docs/iOS_6_VPN_DE.pdf* (11.01.2016).

APPLE (2014a). App Programming Guide for iOS. *https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Introduction/Introduction.html* (11.01.2016).

APPLE (2014b). CloudKit Quick Start. *https://developer.apple.com/library/ios/documentation/DataManagement/Conceptual/CloudKitQuickStart/Introduction/Introduction.html* (11.01.2016).

APPLE (2014c). Cryptographic Services Guide. *https://developer.apple.com/library/mac/documentation/Security/Conceptual/cryptoservices/Introduction/Introduction.html* (11.01.2016).

APPLE (2014d). iOS Enterprise Deployment Overview. *https://www.apple.com/iphone/business/docs/iOS_Enterprise_Deployment_Overview_EN_Feb14.pdf* (11.01.2016).

APPLE (2015a). Information Property List Key Reference. *https://developer.apple.com/library/ios/documentation/General/Reference/InfoPlistKeyReference/Introduction/Introduction.html* (11.01.2016).

APPLE (2015b). iOS Security. *https://www.apple.com/business/docs/iOS_Security_Guide.pdf* (11.01.2016).

APPLE (2015c). Local and Remote Notification Programming Guide. *https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html* (11.01.2016).

ARNS, T., BESSING, S., BUGGISCH, C. & GRACKLAUER, M. (2014). *Apps & Mobile Services – Tipps für Unternehmen:* Berlin.

BAL, S. N. (2013). Mobile web–enterprise application advantages. *International Journal of Computer Science and Mobile Computing, 2*(2), 36–40.

BALZERT, H. (2000). *Lehrbuch der Software-Technik. Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung.* Spektrum Akademischer Verlag: Heidelberg.

BAUMGARTEN, U., BERNHOFER, A. & DÖRFEL, R. (2012). Entwicklung mobiler Betriebssysteme im Lichte neuer Apps, veränderter Herausforderungen und der Virtualisierung. In: S. VERCLAS & C. LINNHOFF-POPIEN (Hg.): *Smart Mobile Apps:* Springer Berlin Heidelberg, 465–474.

BEDELL, C. (2010a). The benefits and different types of SSL VPNs. *http://searchenterprisewan.techtarget.com/tutorial/The-benefits-and-different-types-of-SSL-VPNs* (11.01.2016).

BEDELL, C. (2010b). VPN types: Protocols and network topologies of IPsec VPNs. *http://searchenterprisewan.techtarget.com/tutorial/VPN-types-Protocols-and-network-topologies-of-IPsec-VPNs* (11.01.2016).

BLACKBERRY (2014a). Enterprise Mobility Management. *http://de.blackberry.com/content/dam/blackBerry/pdf/business/german/Enterprise-Mobility-Management-ein-Leitfaden-CIOs-DE.pdf* (11.01.2016).

BLACKBERRY (2014b). How Push Service Works. *http://developer.blackberry.com/devzone/develop/platform_services/how_push_service_works.html* (11.01.2016).

BLACKBERRY (2014c). Protection For Every Enterprise. How Blackberry Security Works. *http://de.blackberry.com/content/dam/bbfoundation/pdf/case-study/na/en/Protection%20for%20Every%20Enterprise%20-%20How%20BlackBerry%20Security%20Works.pdf* (11.01.2016).

BLACKBERRY (2014d). The Transformative Power of Multi-Plattform mobile App Develpment. Ushering in the Next Generation of Enterprise Mobility. *http://us.blackberry.com/content/dam/blackBerry/pdf/business/english/BlackBerry_MADP_Whitepaper_1_270514_LR_view.pdf* (11.01.2016).

BLACKBERRY (2015). Enterprise solution comparison chart. Quick Reference. *https://help.blackberry.com/de/bes12/12.1/overview/1796383.html* (11.01.2016).

BÖHMER, W. (2005). *VPN - Virtual Private Networks. Kommunikationssicherheit in VPN- und IP-Netzen, über GPRS und WLAN.* Hanser: München.

BUDSZUS, J., BERTHOLD, O., FILIP, A. & POLENZ, S., ET AL. (2014). Orientierungshilfe – Cloud Computing. *https://www.datenschutz-bayern.de/technik/orient/oh_cloud.pdf* (11.01.2016).

CHEN, S., PANDE, A. & MOHAPATRA, P. (2011). Sensor-assisted facial recognition. In: A. CAMPBELL, D. KOTZ, L. COX & Z.M. MAO (Hg.): *the 12th annual international conference*, 109–122.

CHIA, P. H., YAMAMOTO, Y. & ASOKAN, N. (2012). Is This App Safe?: A Large Scale Study on Application Permissions and Risk Signals: *Proceedings of the 21st International Conference on World Wide Web*, New York, NY, USA: ACM, 311–320.

CUNNINGHAM, A. (2015). Google quietly backs away from encrypting new Lollipop devices by default. *http://arstechnica.com/gadgets/2015/03/google-quietly-backs-away-from-encrypting-new-lollipop-devices-by-default/* (11.01.2016).

DINH, H. T., LEE, C., NIYATO, D. & WANG, P. (2013). A survey of mobile cloud computing. Architecture, applications, and approaches. *Wireless Communications and Mobile Computing, 13*(18), 1587–1611.

DROPBOX. Using the Datastore API on iOS. *https://www.dropbox.com/developers/datastore/tutorial/ios* (11.01.2016).

FAAS, R. (2013). Containerization can create a BYOD disaster. *http://www.citeworld.com/article/2114835/consumerization/containerization-byod-disaster.html* (11.01.2016).

GARTNER (2015). Gartner Says Emerging Markets Drove Worldwide Smartphone Sales to 19 Percent Growth in First Quarter of 2015. *http://www.gartner.com/newsroom/id/3061917* (11.01.2016).

GOOGLE (2015a). Activities. *http://developer.android.com/guide/components/activities.html* (11.01.2016).

GOOGLE (2015b). Implementing an XMPP Connection Server. *https://developers.google.com/cloud-messaging/ccs* (11.01.2016).

GOOGLE (2015c). System Permissions. *http://developer.android.com/guide/topics/security/permissions.html* (11.01.2016).

GREER, D. (2007). Interactive Application Architecture Patterns. *http://aspiringcraftsman.com/2007/08/25/interactive-application-architecture/* (11.01.2016).

GRUMAN, G. (2015). Mobile security: iOS vs. Android vs. BlackBerry vs. Windows Phone. *http://www.cso.com.au/article/574311/mobile-security-ios-vs-android-vs-blackberry-vs-windows-phone/* (11.01.2016).

HILDEBRANDT, A., LUTHIGER, J., STAMM, C. & YEREAZTIAN, C. (2012). Eine Kategorisierung mobiler Applikationen. In: FACHHOCHSCHULE NORDWESTSCHWEIZ (Hg.): *IMVS Fokus Report,* 27–32.

HP (2013). Exploring The OWASP Mobile Top 10: M3 Insufficient Transport Layer Protection. *http://h30499.www3.hp.com/t5/Fortify-Application-Security/Exploring-The-OWASP-Mobile-Top-10-M3-Insufficient-Transport/ba-p/5966473#.VTzhgiHtmkq* (11.01.2016).

INFO-TECH RESEARCH GROUP (2014). The State of BlackBerry in 2014. *http://www.new.infotech.com/infographics/4792/print.pdf* (11.01.2016).

ISO (2013). Information technology — Security techniques — Information security management systems — Requirements, *35.040*(27001:2013). *https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-2:v1:en* (16.01.2016).

JOOS, T. (2015). Android-Praxis: VPN einrichten und nutzen. *http://www.tecchannel.de/kommunikation/handy_pda/2033962/smartphone_android_praxis_vpn_einrichten_und_nutzen/* (11.01.2016).

KERN, M. (2012). Eine neue Generation von Geschäftsanwendungen. In: S. VERCLAS & C. LINNHOFF-POPIEN (Hg.): *Smart Mobile Apps:* Springer Berlin Heidelberg, 95–106.

KERSTEN, H. & KLETT, G. (2012). *Mobile Device Management.* mitp.

KREUZHUBER, S., TEUFL, P. & ZEFFERER, T. (2014). *Sicherheitsanalyse Mobile Plattformen:* Wien.

MARGARET ROUSE (2014). Address space layout randomization. *http://searchsecurity.techtarget.com/definition/address-space-layout-randomization-ASLR* (11.01.2016).

MERZ, P. (2014). Sichere mobile Unternehmensanwendungen. *HMD Praxis der Wirtschaftsinformatik, 51*(1), 45–55.

MICROSOFT (2014a). Microsoft-Designprinzipien. *https://msdn.microsoft.com/library/windows/apps/hh781237.aspx* (11.01.2016).

MICROSOFT (2014b). Separate UI and app logic using the Model-View-ViewModel pattern. *https://msdn.microsoft.com/en-us/library/windows/apps/jj721615%28v=vs.105%29.aspx* (11.01.2016).

MICROSOFT (2014c). Tile and tile notification overview (Windows Runtime apps). *https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh779724.aspx* (11.01.2016).

MICROSOFT (2014d). Tiles, badges, and notifications (Windows Runtime apps). *https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh779725.aspx* (11.01.2016).

MICROSOFT (2014e). Try it out: enroll a Windows Phone 8.1 device. *https://technet.microsoft.com/en-us/windows/dn771709.aspx* (11.01.2016).

MICROSOFT (2014f). Übersicht über Windows-Pushbenachrichtigungsdienst (Windows Push Notification Service, WNS) (Windows-Runtime-Apps). *https://msdn.microsoft.com/de-de/library/windows/apps/hh913756.aspx* (11.01.2016).

MICROSOFT (2014g). Windows Phone 8.1 Mobile Device Management Overview. *https://www.microsoft.com/en-us/download/details.aspx?id=42508* (11.01.2016).

MICROSOFT (2014h). Windows Phone 8.1 Security Overview. *http://download.microsoft.com/download/B/9/A/B9A00269-28D5-4ACA-9E8E-E2E722B35A7D/Windows-Phone-8-1-Security-Overview.pdf* (11.01.2016).

MICROSOFT (2015a). Cloud Services Glossar: Infrastructure as a Service (IaaS). *http://www.microsoft.com/de-de/cloud/glossar/infrastructure_as_a_service.aspx* (11.01.2016).

MICROSOFT (2015b). Cloud Services Glossar: Platform as a Service (PaaS). *http://www.microsoft.com/de-de/cloud/glossar/platform-as-a-service.aspx* (11.01.2016).

MOBILE HELIX & VANSON BOURNE (2013). CIO Research. *http://www.marketwired.com/press-release/cio-concerns-over-cost-complexity-and-security-obstructing-enterprise-mobility-1816157.htm* (11.01.2016).

MOTWIN (2013). State of Mobile Application Development. A Survey Conducted by moTwin From Nov 15 – Dec 6, 2012. *http://www.itbusinessedge.com/slideshows/state-of-mobile-application-development-in-the-enterprise-10.html* (11.01.2016).

NAHAVANDIPOOR, V. (2014). *IOS 8 Swift Programming Cookbook: Solutions & Examples for IOS Apps.* O'Reilly Media.

NOWSECURE (2014). Secure Mobile Development. *https://www.nowsecure.com/resources/secure-mobile-development/* (11.01.2016).

ORTHACKER, C., TEUFL, P., KRAXBERGER, S. & LACKNER, G., ET AL. (2012). Android Security Permissions – Can We Trust Them? In: R. PRASAD, K. FARKAS, A. SCHMIDT & A. LIOY, ET AL. (Hg.): *Security and Privacy in Mobile Information and Communication Systems:* Springer Berlin Heidelberg, 40–51.

PETZOLD, M. (2011). Cloud Computing und Datenschutz. Eine Einführung. *http://www.it-recht-kanzlei.de/cloud-computing-wolke-daten.html* (11.01.2016).

PHIFER, L. (2006). Mobile VPN: Closing the gap. *http://searchmobilecomputing.techtarget.com/tip/Mobile-VPN-Closing-the-gap* (11.01.2016).

REENSKAUG, T. (2007). *The original MVC reports:* Oslo.

REIMER, L. (2013). BlackBerry Balance Enables a True Work and Personal Experience on BlackBerry 10. *http://bizblog.blackberry.com/2013/01/blackberry-10-balance/* (11.01.2016).

ROUSE, M. (2010). Software as a Service (SaaS). *http://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service* (11.01.2016).

SAMSUNG (2015). In-Depth Look at Capabilities. Samsung KNOX and Android for Work. *https://www.samsungknox.com/de/system/files/whitepaper/files/Samsung%20KNOX%20and%20Android%20for%20Work_2.pdf* (11.01.2016).

SANAEI, Z., ABOLFAZLI, S., GANI, A. & BUYYA, R. (2014). Heterogeneity in Mobile Cloud Computing. Taxonomy and Open Challenges. *IEEE Communications Surveys & Tutorials, 16*(1), 369–392.

SEAGATE (2010). FIPS 140-2-Standard und Selbstverschlüsselung. FAQ. *http://www.seagate.com/files/docs/pdf/de-DE/whitepaper/fips-140-2-faq-mb605.1-1007de.pdf* (11.01.2016).

SEIBEL, J. (2014). Comparing Android and iOS MDM Protocol Design Philosophies. *http://www.apperian.com/comparing-ios-android-mdm-protocol-design-philosophies/* (11.01.2016).

SHERMAN, M., CLARK, G., YANG, Y. & SUGRIM, S., ET AL. (2011). User-generated free-form gestures for authentication. In: A. CAMPBELL, D. KOTZ, L. COX & Z.M. MAO (Hg.): *the 12th annual international conference,* 176–189.

SIRTL, H. & KOCH, F. (2012). Smart Apps aus der Wolke. In: S. VERCLAS & C. LINNHOFF-POPIEN (Hg.): *Smart Mobile Apps:* Springer Berlin Heidelberg, 369–383.

SMITH, J. E. & RAVI NAIR (2005). The architecture of virtual machines. *Computer, 38*(5), 32–38.

TEUFL, P., FITZEK, A., HEIN, D. & MARSALEK, A., ET AL. (2014). Android encryption systems: *2014 International Conference on Privacy and Security in Mobile Systems (PRISMS),* 1–8.

TEUFL, P., ZEFFERER, T., STROMBERGER, C. & HECHENBLAIKNER, C. (2013). *iOS Encryption Systems:* Graz.

U.S.DEPARTMENT OF COMMERCE/NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (2002). FIPS PUB 140-2, Security Requirements for Cryptographic Modules: Gaithersburg. *http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf* (11.01.2016).

VALDES, R., VAN L. BAKER, MARSHALL, R. & WONG, J. (2014). *Magic Quadrant for Mobile Application Development Platforms*.

VALK, E. van der (2009a). Implementing the Model View ViewModel pattern. *http://blogs.msdn.com/b/erwinvandervalk/archive/2009/08/18/implementing-the-model-view-viewmodel-pattern.aspx* (11.01.2016).

VALK, E. van der (2009b). the difference between model-view-viewmodel and other separated presentation patterns. *http://blogs.msdn.com/b/erwinvandervalk/archive/2009/08/14/the-difference-between-model-view-viewmodel-and-other-separated-presentation-patterns.aspx* (11.01.2016).

VERCLAS, S. & LINNHOFF-POPIEN, C. (2012a). Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse. In: S. VERCLAS & C. LINNHOFF-POPIEN (Hg.): *Smart Mobile Apps:* Springer Berlin Heidelberg, 3–15.

VERCLAS, S. & LINNHOFF-POPIEN, C. (Hg.) (2012b). *Smart Mobile Apps.* Springer Berlin Heidelberg.

VIJAYAN, J., MANEA, A. & LAI, E. (2015). *The Definitive Guide to Enterprise Mobile Security. Strategies and Tactics for Business and IT Decision-Makers:* Waterloo, Kanada.

WEIß, F. & SÖLLNER, M. (2012). Technologische und marktseitige Unsicherheit bei der Neuentwicklung von Mobile Enterprise Services. In: S. VERCLAS & C. LINNHOFF-POPIEN (Hg.): *Smart Mobile Apps:* Springer Berlin Heidelberg, 177–190.

WIEMANN, V. (2013). *Mobile Enterprise Application Platforms:* Bielefeld.

WILDT, P. & MEISTER, R. (2012). Das Smartphone als sichere Burg. In: S. VERCLAS & C. LINNHOFF-POPIEN (Hg.): *Smart Mobile Apps:* Springer Berlin Heidelberg, 209–223.

WILLNECKER, F., ISMAILOVIĆ, D. & MAISON, W. (2012). Architekturen mobiler Multiplattform-Apps. In: S. VERCLAS & C. LINNHOFF-POPIEN (Hg.): *Smart Mobile Apps:* Springer Berlin Heidelberg, 403–417.

ZDNET (2015). Mobile Device Management: Funktionen im Überblick. *http://www.zdnet.de/88193558/mobile-device-management-funktionen-im-ueberblick/* (11.01.2016).

Rechtsgrundlagenverzeichnis

BDSG. Bundesdatenschutzgesetz in der Fassung der Bekanntmachung vom 14. Januar 2003 (BGBl. I S. 66), das zuletzt durch Artikel 1 des Gesetzes vom 25. Februar 2015 (BGBl. I S. 162) geändert worden ist

Richtlinie 95/46/EG. Richtlinie 95/46/EG des Europäischen Parlaments und des Rates vom 24. Oktober 1995 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr

Anlagenverzeichnis

A. Lastenheft

(Anforderungsspezifikation)

1. Zielbestimmung

Die mobile Applikation (App) befähigt den Nutzer, seine Arbeitszeit innerhalb von Projekten durch das Eintragen von Zeitintervallen (Tasks) festzuhalten. Projekte erhalten ein Abrechnungsmodell (Zahlungsmodell) und zugeordnete Zeitfenster eine Tätigkeit, welche die Aufgabe und den Verdienst festhält. Auf dieser Grundlage werden Zeitintervalle für eine spätere Abrechnung beim Auftraggeber genutzt und zu diesem über verschiedene Verfahren übertragen bzw. in geeigneten Formaten für eine Übertragung zur Verfügung gestellt.
Ziel ist es, einen Mehrwert für den Nutzer und den Auftraggeber zu erhalten, indem das Aufzeichnen der benötigten Zeit für Aufgaben innerhalb von Projekten deutlich vereinfacht und eine Nachvollziehbarkeit der geleisteten Arbeit für spätere Abrechnungen ermöglicht wird.
Diese Vereinfachung wird durch die Nutzung der Applikation auf mobilen Endgeräten, durch ein einfaches und effizient bedienbares Benutzerinterface und durch eine komfortable Datenhaltung mit Exportmöglichkeiten erreicht.

2. Produkteinsatz

Wer benutzt die App?

- Business Kunden, Unternehmen, Firmen, Einrichtungen mit zahlreichen Projekten und Mitarbeitern, die an späteren Abrechnungen interessiert sind
- Studenten, Schüler zur Zeitplanung für Kurse und Unterricht
- Personen, die eine Zeitplanung für alltägliche Aufgaben benötigen

Use Case

Beispiel: Projekte zwischen Unternehmen

Auftragnehmer:
- als Nutzer bezeichnet,
- interagiert direkt mit der App

Auftraggeber:
- indirekt;
- profitiert durch detailliertere, nachvollziehbarere Reports der Arbeitszeiten und
- verringerten Verwaltungsaufwand der Auftragnehmer => mehr effektive Arbeitszeit => weniger Kosten

Die App richtet sich unter anderem an Unternehmen, die ihren Mitarbeitern die Möglichkeit geben wollen schnell, einfach und mobil ihre Arbeitszeiten aufzuzeichnen und diese dem Auftraggeber als Report zuzusenden.
Beginnt der Benutzer ein neues Projekt im Unternehmen, so erstellt er dafür ein neues Projektobjekt in der App. Er ordnet dem Projekt, je nach Projektbeschaffenheit, ein Zahlungsmodell zu. Dieses gibt an, wie eingetragene Arbeitszeiten abgerechnet werden. Beginnt der Benutzer an diesem Projekt zu arbeiten, so erstellt er innerhalb des Projektobjekts ein Zeitintervall bzw. einen Task. Diesem ordnet er eine Tätigkeit zu, welche eine Beschreibung und den Verdienst während dieser Zeit angibt. Pausiert er seine aktuelle Arbeit, so kann er das aktuelle Zeitintervall ebenfalls pausieren. Diese Pause wird aufgezeichnet und kann später nachverfolgt und von der Arbeitszeit abgezogen werden. Beendet er seine Arbeit, so stoppt er das vorher begonnen Zeitintervall. Dieses ist damit beendet und steht für spätere Reports zur Verfügung. Möchte der Nutzer alle Zeitintervalle eines Monats zur Abrechnung seiner geleisteten Arbeitszeit an den Auftraggeber schicken, so filtert er in der App alle Zeitintervalle für diesen Monat und schickt diese als Report per Mail an den Auftraggeber oder lässt sich ein PDF mit den Intervallen erstellen. Möchte der Nutzer seine Daten sichern oder auf verschiedenen Geräten zur Verfügung stellen, so Synchronisiert er diese mit einem externen Dienst.

3. Produktübersicht

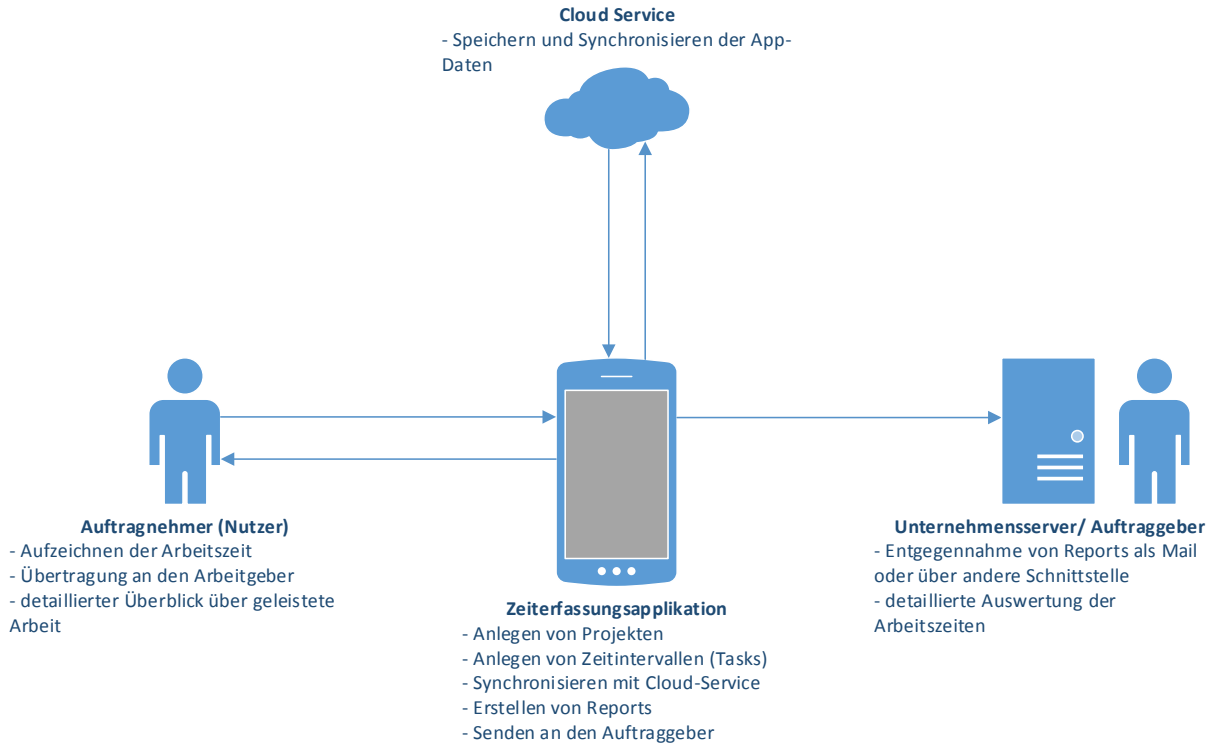

Abbildung 14 Kollaborationsdiagramm der Zeiterfassungsapplikation

4. Produktfunktionen

Es wird unterschieden nach:

Nutzer	Ein Nutzer ist eine Person mit Smartphone, die die Zeiterfassungs-App auf diesem installiert hat.
Zeiterfassungs-App	Die Zeiterfassungs-App ist eine native mobile Applikation für ein spezifisches mobiles Betriebssystem. Ihre Merkmale sind in diesem Dokument beschrieben.
Cloud Service	Ein Cloud Service ist in diesem Kontext ein Online Dienst, der Daten über eine wohl definierte Schnittstelle entgegennimmt, diese speichert und auf Anfrage über eine wohl definierte Schnittstelle zurückgibt.
Auftraggeber	Der Auftraggeber kann ein Unternehmen, eine Organisation, eine Einrichtung, eine Privatperson oder Ähnliches sein, der eine spezifische Aufgabe in Form von Projekten oder Ähnlichem an den Nutzer übergibt.

4.1 Muss-Funktionen

4.1.1 Projekte

/LF10/	Geschäftsprozess:	Erstellen eines neuen Projektes
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Erstellen eines neuen Projekteintrags mit Titel, Beschreibung, Zahlungsmodell, Budget;
/LF20/	Geschäftsprozess:	Darstellung aller Projekte
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Anzeige aller bisher erstellten Projekte;
/LF30/	Geschäftsprozess:	Darstellung der Details zu einem Projekt
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Darstellung der Parameter Titel, Beschreibung, Zahlungsmodell, Budget eines Projektes;
/LF40/	Geschäftsprozess:	Bearbeiten eines Projektes
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Bearbeiten der Parameter eines Projektes;
/LF50/	Geschäftsprozess:	Schließen eines Projektes
	Akteur:	Nutzer
	Beschreibung:	Das Schließen eines Projektes, sodass dieses nicht mehr bearbeitet werden kann, sollte je nach Zahlungsmodell oder auf den Willen des Nutzers hin möglich sein;

/LF60/	Geschäftsprozess:	Berechnung der Reporting-Details eines Projektes
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Berechnung und Anzeige des bisher verbrauchten Budgets, der bisher benötigten Zeit pro Projekt und pro Tätigkeit;

4.1.2 Zeitintervalle (Tasks)

/LF70/	Geschäftsprozess:	Schnelles Beginnen eines neuen Zeitintervalls
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Erstellen eines neuen Zeitintervalls mit einer Anfangszeit, jedoch ohne Endzeit;
/LF80/	Geschäftsprozess:	Beenden eines Zeitintervalls
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Beenden eines Zeitintervalls, das begonnen wurde durch Eintragen einer Endzeit;
/LF90/	Geschäftsprozess:	Pausieren eines Zeitintervalls
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Pausieren eines Zeitintervalls, um Pausen innerhalb eines Zeitintervalls aufzuzeichnen;
/LF100/	Geschäftsprozess:	Erstellen eines neuen Zeitintervalls
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Erstellen eines Zeitintervalls mit Angabe von Titel (Task), Beschreibung, Tätigkeit, Anfangszeit und Endzeit;
/LF110/	Geschäftsprozess:	Hinzufügen einer neuen Tätigkeit
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Erstellen einer Tätigkeit mit Beschreibung und Verdienst und Hinzufügen zum aktuellen Zeitintervall;
/LF120/	Geschäftsprozess:	Bearbeitung eines Zeitintervalls
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Ändern von Werten eines zuvor erstellten Zeitintervalls;
/LF130/	Geschäftsprozess:	Darstellung aller Zeitintervalle eines Projektes
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Zusammenfassende Anzeige aller Zeitintervalle eines Projektes;

/LF140/	Geschäftsprozess:	Detailansicht eines Zeitintervalls
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Detailansicht eines Zeitintervalls mit Titel, Anfangszeit, Endzeit, Pausendauer, Tagessatz, Beschreibung;
/LF150/	Geschäftsprozess:	Filtern der Zeitintervalle
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Filterung der Anzeige von Zeitintervallen nach bestimmten Kriterien;

4.1.3 Reports

/LF160/	Geschäftsprozess:	Erstellen eines Reports
	Akteur:	Zeiterfassungs-App, Nutzer
	Beschreibung:	Zusammenfassen verschiedener Zeitintervalle zu einem Report;
/LF170/	Geschäftsprozess:	Export eines Reports
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Export als PDF und/oder senden per Mail;

4.1.4 Datenhaltung

/LF180/	Geschäftsprozess:	Speichern der Daten auf dem Smartphone
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Persistentes Abspeichern der vom Nutzer während der App-Nutzung entstandenen Daten;

4.1.5 Ansicht

/LF190/	Geschäftsprozess:	Anzeige von Informationen außerhalb der App
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Anzeige aktuell laufender Zeitintervalle bzw. aktueller Projekte auf dem Sperrbildschirm oder ähnlichem, um eine schnelle Information des Nutzers zu bewirken;
/LF200/	Geschäftsprozess:	Anzeige einer Startseite
	Akteur:	Zeiterfassungs-App
	Beschreibung:	Die Startseite sollte die wichtigsten Informationen für den Nutzer auf einen Blick bereithalten;

4.2 Kann-Funktionen

/LF210/	Geschäftsprozess:	Speichern der Daten in der Cloud
	Akteur:	Zeiterfassungs-App, Cloud Service
	Beschreibung:	Übertragen der vom Nutzer eingetragenen Daten zu einem Cloud Service und persistente Speicherung der Daten durch diesen;
/LF220/	Geschäftsprozess:	Synchronisieren der Daten mit der Cloud
	Akteur:	Zeiterfassungs-App, Cloud Service
	Beschreibung:	Abgleich der aktuell auf dem Smartphone gehaltenen App-Daten und derer im Cloud Service, sodass auf beiden Medien die aktuellsten Daten vorliegen;
/LF230/	Geschäftsprozess:	Senden von Reports an das Unternehmen
	Akteur:	Zeiterfassungs-App, Unternehmensserver
	Beschreibung	Senden von erstellten Reports an das Unternehmen, zum Nachweis der eigenen Arbeitszeit pro Projekt;
/LF240/	Geschäftsprozess:	Login des Nutzers
	Akteur:	Zeiterfassungs-App, Cloud Service
	Beschreibung:	Einloggen des Nutzers beim Starten der App oder beim Auswählen eines Projektes.

5. Produktdaten

Projektdaten

/LD10/ pro Projekt: Titel, Beschreibung, Zahlungsmodell, Budget, Währung

Tätigkeit

/LD20/ Tagessatz, Beschreibung

Zeitintervalldaten

/LD30/ pro Zeitintervall: Titel (Task), Beschreibung, Tätigkeit, Anfangszeit, Endzeit, Pausendauer

Account-Daten (Kann-Funktion)

/LD40/ Nutzername und Passwort zum Synchronisieren mit dem Cloud Service bzw. für einen internen Login

/LD50/ VPN-Daten und Konfiguration für die Verbindung zum Unternehmensserver

6. Produktleistungen

/LL10/ Beim schnellen Erstellen eines Zeitintervalls /LF70/ wird dieses hervorgehoben vor den bereits beendeten Zeitintervallen angezeigt.

/LL20/ Pausen werden ausschließlich durch das Festhalten der Pausendauer gespeichert /LF90/.

/LL30/ Jedes Zeitintervall verfügt über eine Tätigkeit. In einem Projekt können mehrere Tätigkeiten hinzugefügt werden. Bei der Erstellung eines Zeitintervalls wird eine Tätigkeit ausgewählt.

/LL40/ Das Filtern der Zeitintervalle /LF150/ sollte nach folgenden Kriterien erfolgen: aktuelle Woche, aktueller Monat, letzte Woche, letzter Monat, Alle.

/LL50/ Das Erstellen eines Reports /LF160/ sollte durch Filtern nach bestimmten Kriterien erfolgen können (siehe /LL40/).

/LL60/ Der Export von Reports sollte per Mail und/oder als PDF möglich sein /LF170/.

/LL70/ Das Abspeichern der Daten auf dem Smartphone /LF180/ sollte stets eine persistente Speicherung bei unterschiedlichsten App-Zuständen garantieren. Dies muss mit dem App-Lebenszyklus des jeweiligen mobilen Betriebssystems und den jeweiligen Empfehlungen des Entwicklers abgeglichen werden.

/LL80/ Eine Offline Datenhaltung bzw. Nutzung der App sollte möglich sein.

/LL90/ Das App-Design und die Interaktionen sollten mit den Designrichtlinien des jeweiligen mobilen Betriebssystems übereinstimmen.

/LL100/ Die Produktdaten sind mit Hilfe aktueller Sicherheitstechnologien zu sichern.

/LL110/ Die Kommunikation mit jeglichen Services ist mit Hilfe aktueller Sicherheitsmaßnahmen abzusichern.

/LL120/ Die Speicherung von Daten ist lediglich auf dem Gerät oder in der privaten Cloud Lösung des Unternehmens vorzunehmen.

/LL130/ Zahlungsmodelle eines Projektes sind

- Leistungsabrechnung: Es existiert ein Budget. Die Arbeitsleistung wird nach Tagessätzen abgerechnet.
- Leistungsabrechnung ohne Budget: Es existiert kein Budget. Die Arbeitsleistung wird nach Tagessätzen abgerechnet.
- Fixpreis: Es existiert ein begrenztes Budget. Die Arbeitszeit wird nicht nach Tagessätzen abgerechnet.
- Arbeitszeiterfassung: Die eingetragenen Zeitintervalle werden nicht abgerechnet.

7. Qualitätsanforderungen

Produktqualität	sehr gut	gut	normal	nicht relevant
Funktionalität		x		
Zuverlässigkeit			x	
Benutzbarkeit	x			
Effizienz		x		
Änderbarkeit			x	
Übertragbarkeit			x	

Tabelle 23 Qualitätsanforderungen der Zeiterfassungsapplikation

8. Ergänzungen

Die Zeiterfassungs-App wird nativ für mehrere Plattformen (vorerst BlackBerry und Windows Phone 8.1) entwickelt. Sie soll Anforderungen aus dem Business-Bereich erfüllen.

B. Bewertungsschlüssel und Eignung

Relativer Anteil von	bis	Notenwert	Bezeichnung	Branche
0,95	1,00	1	Sehr gut	Höchste Anforderungen wie Finanzdienstleister oder Produktion bzw. große Unternehmen oder Regierungseinrichtungen
0,90	0,94	1,3	Sehr gut	
0,85	0,89	1,7	Gut	Mittlere Anforderungen wie In-Car, Smart-Home, Gesundheitswesen bzw. mittel große Unternehmen
0,80	0,84	2	Gut	
0,75	0,79	2,3	Gut	
0,70	0,74	2,7	Befriedigend	Geringe Anforderungen wie Lehranstalten, Medien bzw. kleine Unternehmen
0,65	0,69	3	Befriedigend	
0,60	0,64	3,3	Befriedigend	
0,55	0,59	3,7	Ausreichend	Nicht für Enterprise-Einsatz geeignet
0,50	0,54	4	Ausreichend	
0	0,49	5	Nicht bestanden	

C. Bewertung der mobilen Betriebssysteme

Kriterien	Android 5 (enthält Andoid for Work)	Android 5 + Samsung Knox 2.4	iOS 8	Blackberry 10.3	Windows Phone 8.1	Gewicht (G)	Faktor (α)
1. Sicherheit							
ASLR	ja	ja	ja	ja	ja	1	
DEP	ja	ja	ja	ja	ja	1	
Sandboxing	ja	ja	ja	ja	ja, AppContainer	1	
Verifizierung der geladenen Softwarekomponenten während des Bootprozesses (Secure Boot)	ja	ja	ja	ja	ja, UEFI, Trusted Boot, Chain-of-Trust	1	
App-Permissions							
Zeitpunkt der Abfrage von Permissions	Bei Installation der App	Bei Installation der App	Bei Installation (basic-Permissions wie Internetzugriff) und wenn benötigt (z.B. GPS, Kontakte, Kalender)	Bei erstmaligem Start der App	Wenn benötigt (nur bei sensiblen oder vertraulichen Infos, wie Anmeldedaten oder Position); sonst keine Abfrage, sondern Info auf Seite des App Stores	1	
Abwählen einzelner Permissions bei Abfrage	Nein	Nein	Ja	Ja	Nein	1	
Angabe von Gründen, wieso Permission benötigt wird	Nein, allgemeine Beschreibung	Nein, allgemeine Beschreibung	Ja, kontextbezogene Beschreibung	Nein, allgemeine Beschreibung	Nein	1	
Nachträgliches Abwählen/ Zustimmen einzelner Permissions	Nein, nur über Drittanbieter-Apps	Nein, nur über Drittanbieter-Apps	Ja	Ja (Settings > Security and Privacy > Application Permissions)	Nein (vsl. WP 8.1 GDR 2: Abwählen einzelner Permissions in den Settings)	2	
Softwareverschlüsselung	ja	ja	ja	ja	ja	1	
Hardwareverschlüsselung	geräteabhängig	geräteabhängig	ja, A7 Crytpo-Prozessor	ja	ja, TPM Crypto-Prozessor	1	

Kriterien	Android 5 (enthält Andoid for Work)	Android 5 + Samsung Knox 2.4	iOS 8	Blackberry 10.3	Windows Phone 8.1	Gewicht (G)	Faktor (α)
Dateisystemverschlüs-selung	ja, nicht standardmäßig aktiviert	ja, nicht standardmäßig aktiviert	ja	ja (persönlicher und geschäftlicher Bereich getrennt)	ja, nur über MDM	1	
Applikationsspezifische Verschlüsselung	ja	ja	ja	n.a.	ja	1	
FIPS 140-2 Zertifizierung	nein	ja, Level 1	ja, Level 1	ja, Level 1	ja, Level 1	1	
S/MIME	nein	ja	ja	ja	ja	1	
SSL/TLS	ja	ja	ja	ja	ja	1	
nativer VPN-Client	ja	ja	ja	ja	ja	1	
iPsec/SSL VPN	ja/nur proprietär	ja/ja	ja	ja	ja/ja, Unterstützung für IKEv2	1	
Per-App VPN	ja (in gesichertem Bereich)	ja (in gesichertem Bereich)	ja	nein	ja (auto-triggered-VPN)	1	
Always-On VPN	ja	ja	ja	nein	ja	1	
Zugriffsauthentifizierung über PIN, Passwort	ja	ja	ja	ja	ja	1	
Biometrische Verfahren	ja, Gesichtserkennung	ja, Gesichtserkennung	ja, Fingerabdruck	nein	nein	0,5	
Single Sign On	nein	ja	ja	ja	ja	1	
Wi-Fi Authentifizierung	EAP-TLS, EAP-TTLS, EAP-PEAP, EAP-SIM, WPA, WPA2, WEP	EAP-TLS, EAP-TTLS, EAP-PEAP, EAP-SIM, WPA, WPA2, WEP	EAP-TLS, EAP-TTLS, EAP-FAST, EAP-SIM, PEAPv0, PEAPv1, and LEAP, WPA2 Enterprise	EAP-FAST, EAP-TLS, EAP-TTLS, and PEAP, WEP, WPA-Personal, WPA2-Personal, WPA-Enterprise, and WPA2-Enterprise	WPA, WPA2, WEP, EAP-TLS, EAP-TTLS	1	
Review-Prozess der Apps im App-Store	ja	ja	ja	ja	ja	1	
Zwischenergebnis	16 von 24,5	20 von 24,5	24,5 von 24,5	20 von 23,5	21 von 24,5	24,5	3/7
gemäß (5.5)	0,65	0,82	1	0,85	0,86		

Kriterien	Android 5 (enthält Andoid for Work)	Android 5 + Samsung Knox 2.4	iOS 8	Blackberry 10.3	Windows Phone 8.1	Gewicht (G)	Faktor (α)
2. Mobile Device Management und Bring Your Own Device							
MDM-Client	ja	ja	ja	ja	ja	1	
Anzahl MDM-Regeln	Wenige Regeln (ca. 10) meist Sicherheitsfeatures, herstellerspezifische Erweiterungen	durch Samsung Knox erweiterte Features (über 600 Richtlinien, über 1400 APIs)	Große Anzahl an Regeln (über 100), die sich mit diversen Aspekten des Systems befassen	Große Anzahl an Regeln für Work Space, Eingeschränkte Regeln für privaten Bereich	Geringe Anzahl an Regeln; Obermenge der unter Exchange ActiveSync (EAS) verfügbaren Regeln	1	
Entziehen der MDM-Kontrolle durch den Benutzer	ja, Deinstallation möglich, damit verbundene Einschränkungen sind vom Agenten festgelegt	ja, Deinstallation möglich, Löschen aller Daten des Knox-Containers	ja, Vorherige Spezifikation, ob Konfigurationsprofil entfernt werden darf	nein	ja, jedoch Deinstallation aller Unternehmens-applikationen und Unternehmenskontos	1	
Integrierte BYOD-Lösung und Funktionsumfang	ja	ja, Erweiterung um zahlreiche APIs	nein	ja, große Anzahl an Regeln s.o.	nein	1	
Möglichkeit von Containerization	ja	ja	ja	ja	ja	1	
OTA-Management (Regeln und Konfiguration, Firmware-Updates)	ja	ja	ja, Konfigurationsprofil, MDM, and Exchange ActiveSync	ja	ja, Updates über MDM oder WP Update Service	1	
Einschränkung von Apps (Whitelist/Blacklist), Enterprise App Store	ja, im sicheren Bereich/ ja, Verwalten eines privaten Kanals über Admin-Konsole	ja/ja	nein, keine MDM-Regel, Möglichkeit: Liste von Apps abrufen + Restriktion durchführen / ja	ja, Application Restriction Rule IT/ja	ja, über App Allow/Deny-Listenfunktion oder Assigned Access / ja	1	
Remote Wipe	ja	ja	ja	ja, remote und lokal	ja +Device Retirement (ausschließliches Entfernen von Firmendaten) + Device Wipe Treshold	1	
Remote Locate	ja, via Android Device Manager	ja, via Android Device Manager	ja, über "Find my iPhone" Service	ja	ja, Microsoft Dienst "Mein Handy finden"	1	
Remote Lock	ja (Lock, Ring, Wipe)	ja (Lock, Ring, Wipe)	ja, über "Find my iPhone" Service	ja, über Web Service	ja (Lock, Ring, Reset Passwort)	1	

Kriterien	Android 5 (enthält Andoid for Work)	Android 5 + Samsung Knox 2.4	iOS 8	Blackberry 10.3	Windows Phone 8.1	Gewicht (G)	Faktor (α)
Passwort und Authentifizierungs-kontrolle	ja	ja	ja	ja	ja	1	
Lokale Verschlüsselung (native Unterstützung, über Software)	ja	ja	ja	ja	ja Bitlocker + removable Storage	1	
Konfiguration der Kommunikation wie VPN	ja	ja	ja	ja	auto-triggered VPN	1	
Konfiguration der Kommunikation wie WLAN	Liste erlaubter APs	ja	ja	ja	ja, - Wi-Fi Profile, Untersagen Wi-Fi Hotspot, kein Hinzufügen zusätzlicher Profile, Untersagen von Traffic über Wi-Fi-Verbindung	1	
Backup/Restore von Gerätedaten	ja, Google Chrome Lesezeichen, WLAN-Passwörter, Wörterbücher und Geräteeinstellungen über Google-Zugang	ja, Google Chrome Lesezeichen, WLAN-Passwörter, Wörterbücher und Geräteeinstellungen über Google-Zugang	ja, über iCloud	ja, auto-Backup, BlackBerry über Desktop Software oder BlackBerry Web Desktop Manager	keine vollständige Backup-Lösung, Synchronisieren von Kontakten, Nachrichten, Fotos, Systemeinstellungen etc. mit Microsoft-Konto	0,5	
Backups auf externe Cloud-Services unterbinden	nein, nicht nativ	ja, Unterbinden der Synchronisation mit Google Account	Unterbinden von iCloud- und iTunes-Backups	Regel: Cloud Storage Access durch Work Space; Unterbinden von z.B. Box oder Dropbox	nein, über Black-/Whitelisting von entsprechenden Apps möglich oder Assigned Access	1	
Zustand von Geräten auslesbar	nein, nicht nativ	ja	ja	ja	Remote Inventory, z.B. installierte Apps, Telefonnummer, OS Version, MAC-Adresse, …	1	
Aktivierung Per-App VPN, Always-On	ja /n.a	ja/ja	n.a/ja	nicht verfügbar	ja/ja	1	

Kriterien	Android 5 (enthält Andoid for Work)	Android 5 + Samsung Knox 2.4	iOS 8	Blackberry 10.3	Windows Phone 8.1	Gewicht (G)	Faktor (α)
Sicheres Löschen von Daten (Daten nicht mehr auf Datenträger)	nein, Verschlüsseln + anschließendes Zurücksetzen möglich	nein, Verschlüsseln + anschließendes Zurücksetzen möglich	nein, jedoch Löschen des Block-Schlüssels = Dateien nicht zu entschlüsseln	ja, Security Wipe, außerdem Daten der Speicherkarte nicht mehr zugänglich	nein, Verschlüsseln + Zurücksetzen möglich	1	
Assigned Access	nein, nicht nativ	ja	n.a.	n.a.	ja	1	
Zwischenergebnis	13,5 von 19	18 von 19,5	15 von 18	17,5 von 17,5	16,5 von 19,5	19,5	3/7
gemäß (5.5)	0,71	0,92	0,83	1	0,85		
3. Benutzer Interface und Trennung der Anwendungslogik							
Möglichkeiten der Plattformen	nativ, hybrid, web	nativ, hybrid, web	nativ, hybrid, web	nativ, hybrid, web	nativ, hybrid, web	1	
Trennung von Layout, Design und Logik möglich	ja	ja	ja (.nib, .xib)	ja	ja	1	
Beschreibungssprache für die UI	ja, deklarative Beschreibungssprache XML, programmatisch	ja, deklarative Beschreibungssprache XML, programmatisch	ja, deklarative Beschreibungssprache XML, programmatisch	ja, QML als vollständige deklarative Programmiersprache, programmatisch	ja, deklarative Beschreibungssprache XAML, programmatisch	1	
UI Entwicklung über Designer (nur Preview/ Manipulation)	ja	ja	ja	Preview möglich	ja	1	
Möglichkeit Separation-Presentation-Pattern zu implementieren	ja	ja	ja	ja	ja	1	
Sonstige Features zur Separation, Hilfen für Entwickler bei der Umsetzung bestimmter Pattern	Activities, Fragments	Activities, Fragments	Controller-Superklassen, Notifications	Slots, Signals	Commands, Notifications	1	
Zwischenergebnis	5 von 6	5 von 6	5,5 von 6	5 von 6	5,5 von 6	6	1/6
gemäß (5.5)	0,83	0,83	0,92	0,83	0,92		

Kriterien	**Android 5 (enthält Andoid for Work)**	**Android 5 + Samsung Knox 2.4**	**iOS 8**	**Blackberry 10.3**	**Windows Phone 8.1**	**Gewicht (G)**	**Faktor (α)**
Gesamtbewertung							
gemäß (5.6)	0,71	0,86	0,92	0,91	0,86		
gemäß (5.7)	2,7 - Befriedigend	1,7 - Gut	1,3 - Sehr Gut	1,3 - Sehr Gut	1,7 - Gut		

D. Bewertung der mobilen Applikation

Kriterien	Bewertung (S)	Kommentar	Gewicht (G)	Faktor (α)
1. Sicherheit				
Permissions in angemessenem Verhältnis zur Funktionalität	ja	siehe Tabelle	2,0	
Verschlüsselung von App-Daten (z.B. KeyChain)	nein		0,5	
Nutzung sicherer VPN-Verbindungen zum Unternehmen	ja	iPsec/ SSL VPN	1,0	
Eigene Zugriffsauthentifizierung	nein		0,5	
Nutzen 2-Faktor-Authentifizierung	nein		1,0	
Verbindung mit SSL/TLS-Verschlüsselung zu allen genutzten Services	ja	empfohlen TLS 1.0+	1,0	
Verteilung über App-Store/ Enterprise App-Store/ Drittanbieter	Enterprise App-Store		1,0	
Durchlaufen eines Review-Prozesses (Plattform/intern)	ja, intern	Korrekte Verschlüsselungsparameter (z.B. iOS Schutzklasse, Ableitung des kryptographischen Schlüssels), Korrekte Plattformfunktionen	1,0	
keine Nutzung der Services anderer Apps (Android Intents, iOS App Group)	ja		0,5	
Informieren der Mitarbeiter über Sicherheitspolicies	ja		1,0	
Minimalanforderungen an Geräte	ja		1,0	
Risiko- und Sicherheitsanalyse	ja	Welche Daten in BYOD gespeichert?, sicherheitskritische Daten vermeiden	1,0	
Zwischenergebnis	9,5 von 11,5	gemäß (5.5): 0,83	11,5	7/24
2. Notifications				
Notification Service	WNS			
Cloud unabhängig	nein		1,0	
Queuing	ja		1,0	
Rückantwort bei fehlgeschlagenen Nachrichten	nein		1,0	
Sichere Kommunikation	ja		1,0	
Statusabfrage bzw. Empfangsbestätigung	nein		1,0	
Zeitschranken	nein		1,0	
Zwischenergebnis	2 von 6	gemäß (5.5): 0,33	6,0	1/8

Kriterien	Bewertung (S)	Kommentar	Gewicht (G)	Faktor (α)
3. Mobile Cloud Computing - Cloud Speicher				
Nutzung Cloud Service intern/extern	nein/nein			
Datenschutzrecht				
Speicherung Personenbezogener Daten → Gilt das Datenschutzgesetz?		nein - besser	1,0	
Transparenz über technische und organisatorische Sicherheitsmaßnahmen			1,0	
Transparenz über Orte der Speicherung bzw. Unteranbieter			1,0	
Datenverarbeitung in Drittstaaten		nein - besser	0,5	
Cloud-Anbieter in USA geschäftlich tätig		nein - besser	0,5	
Cloud Storage Lösungen				
Verschlüsselung der Daten vor dem Einbringen			1,0	
Sichere Kommunikation		TLS 1.0+	1,0	
Serverseitige Verschlüsselung			1,0	
Enterprise-Account		Gewichtung 0 bei interner Cloud	1,0	
Rechteverwaltung			1,0	
Dateiwiederherstellung			1,0	
Zwischenergebnis	0 von 0	momentan kein Cloud-Service genutzt	10,0	0
4. Mobile Device Management				
Nutzung einer MDM-Lösung	ja		1,0	
Verfolgen einer Unternehmensstrategie	COPE	BYOD, COBO, COPE	1,0	
klassisches MDM/Container-App/integriertes BYOD	klassisches MDM		1,0	
OTA-Management	ja		1,0	
Aktivierung Per-App VPN oder Always-On	ja		1,0	
Einschränkung von Apps	ja	Blacklisting/Whitelisting, Enterprise App-Store	1,0	
Backup von Gerätedaten	nein	Backup ausschließlich manuell und lokal auf internen Gerätespeicher (von dort manueller Export per Mail oder auf PC) möglich, nicht automatisiert	1,0	
Verschlüsselungsschlüssel nicht in Backups enthalten	nicht verfügbar		1,0	
Backup auf externen Cloud Service unterbinden	ja		1,0	

Kriterien	Bewertung (S)	Kommentar	Gewicht (G)	Faktor (α)
Zugriff ausschließlich auf geschäftliche Daten	ja	theoretisch Zugriff auf persönliche Daten möglich	1,0	
Ausschließliches Löschen von geschäftlichen Daten bei Remote Wipe	ja		1,0	
Dateisystemverschlüsselung aktiviert	ja		2,0	
Vorgabe von Richtlinien für Authentifizierung	ja	PIN, Passwort, Länge, Komplexität	2,0	
Konfiguration WLAN und VPN-Verbindungen	ja		1,0	
Prüfen des Gerätezustands	ja	Jailbreak, Password, Apps	1,0	
Zwischenergebnis	14 von 16	gemäß (5.5): 0,87	17,0	7/24
5. Benutzer Interface und Trennung der Anwendungslogik				
Nutzung von Pattern zur Separation von UI und Logik	ja	Übersichtliche Struktur der App	2,0	
Nutzung empfohlener Design Pattern der Plattform zur Separation von UI und Logik	ja		1,0	
Nutzung einer Beschreibungssprache zur Definition der UI	ja	Übersichtlichkeit und Wiederverwendbarkeit	1,0	
Nutzung zusätzlicher Features der Plattform zur Separation	ja		1,0	
Einhaltung der Designrichtlinien der Plattform	ja	Gewährleistung des nativen Look-and-Feel der Plattform	2,0	
Unabhängigkeit von UI und Anwendungslogik	nein	unabhängiges Testen der beiden Komponenten möglich	2,0	
Unterstützung von Mehrsprachigkeit	ja	mind. 2 Sprachen	1,0	
Unterstützung eines Großteils der möglich Displayauflösungen	ja		1,0	
Zwischenergebnis	9 von 11	gemäß (5.5): 0,82	11,0	7/24
Gesamtbewertung gemäß (5.6): $\varphi = \frac{9,5}{11,5} * \frac{7}{24} + \frac{2}{6} * \frac{1}{8} + 0 + \frac{14}{16} * \frac{7}{24} + \frac{9}{11} * \frac{7}{24} =$	0,78	gemäß (5.7): 2,3 - Gut		$\sum_{i=1}^{5} \propto_i = 1$

E. DVD (Inhalt)

/readme.txt	Informationen über CD, Masterarbeit und Testumgebung;
/app_development/	Zeiterfassungs-App als Visual Studio 2013 Solution;
BackgroundTask/	Projekt des notwendigen Background Tasks für die Registrierung eines LockScreen-Tiles;
TimeTracking/	Projekt der Zeiterfassungs-App;
Help/	Code-Dokumentation als Windows Help Datei und Website;
TimeTrackingApp.sln	Ausführbare Visual Studio Solution-Datei;
/test_software/	Software zum Aufsetzen der Testumgebung: Visual Studio 2013 Community web installer, Sprachpaket englisch und Emulator;
/master_thesis/	Dateien der Masterarbeit;
bibliography/	Quellen in Form eines Citavi Projektes (PDFs Quellen falls möglich);
images/	Abbildungen;
Masterarbeit_Buchwald_16.pdf	PDF der Masterarbeit;
/process/	Dokumente, erstellt während des Entstehungsprozesses der Arbeit;
presentation/	Abschlusspräsentation der Masterarbeit und Handout;
Glossar.pdf	Erklärung von Konzepten und Begriffen der mobilen Zeiterfassungs-App;
Konzept.pdf	App-Konzeptualisierung für Windows Phone nach Microsoft;
Kriterienkatalog_Entwicklun…pdf	Kriterienkatalog der Entwicklung- und Einsatzphase und Bewertung mobiler BSs;
Kriterienkatalog_Vorentwick....pdf	Kriterienkatalog der Vorentwicklungsphase und Bewertung mobiler Betriebssysteme;
Lastenheft.pdf	Anforderungsspezifikation der mobilen Zeiterfassungs-App;
.	Zwischenpräsentationen zur App-Entwicklung unter Windows Phone 8.1 und Kriterienanalyse, Abbildungen der App-Architektur und Umgebung mobiler Apps.

Erklärung

"Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann. Gedruckte und elektronische Version dieser Arbeit sind identisch.“

Leipzig, 29. Februar 2016

Björn Buchwald