Enterprise Mobility Plattformen

Aktuelle Lösungen und Trends

White Paper

m-way consulting

Inhaltsverzeichnis

01 Einleitung

Das Thema Enterprise Mobility steht bei vielen Unternehmen für das aktuelle Jahr auf der Agenda. Zu groß sind die möglichen Vorteile für Unternehmen durch gesteigerte Produktivität und Zufriedenheit der Mitarbeiter, zu groß auch der Druck der Mitarbeiter und Kunden durch die zunehmende Consumerization und der generellen Verbreitung von mobilen Geräten und Anwendungen. Daher kann es sich kaum ein Unternehmen leisten, in diesem Bereich untätig zu bleiben. Eine geeignete mobile Strategie hilft den Unternehmen, sich im sehr dynamischen Umfeld von Enterprise Mobility einen Weg zu bahnen. Unter dem Begriff „Enterprise Mobility Management" (EMM) werden alle Aspekte einer mobilen Strategie zusammengefasst.

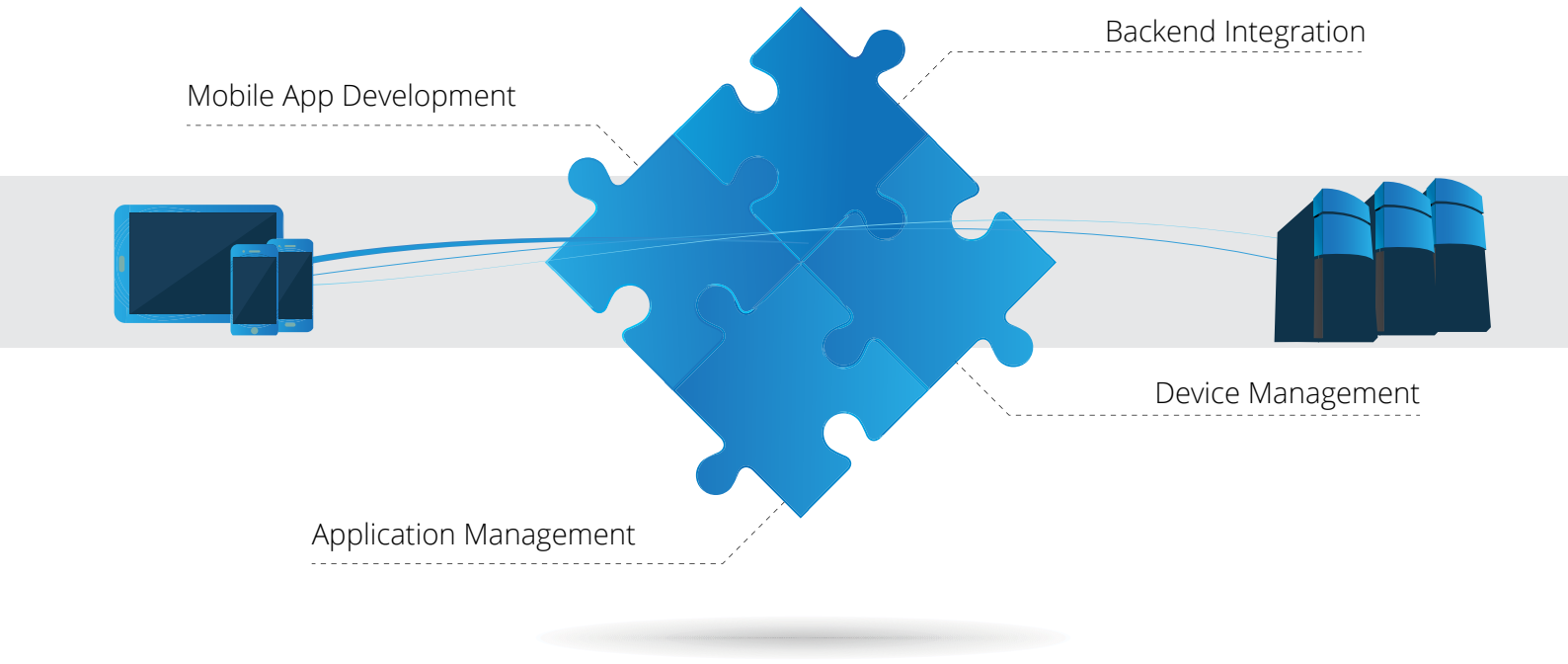
Doch die Umsetzung einer mobilen Strategie birgt trotz fortgeschrittener Technologien und Werkzeuge weiterhin viele Herausforderungen. Die Vielfalt an Anbietern von Mobility-Lösungen ist groß, ebenso deren Versprechungen. Die Auswahl einer geeigneten Plattform für die konkreten Anforderungen eines Unternehmens erfordert daher viel Zeit und Aufwand. Mit dem vorliegenden Dokument geben wir einen allgemeinen Überblick über die Aspekte von Mobility Plattformen und stellen einige Anbieter und deren Produkte vor. Dabei wird anhand einer exemplarischen Anwendung die Verwendung der verschiedenen Plattformen beschrieben, um einen Eindruck zur Komplexität und Funktionalität der jeweiligen Produkte zu erhalten. Zum Abschluss werfen wir einen Blick in die Zukunft und zeigen, welche Trends und Entwicklungen am Markt absehbar sind.

Die hier vorgestellten Produkte basieren auf den verfügbaren Versionen im Stand von November 2013.

02 Grundaspekte einer Mobility Plattform

Die zentrale Aufgabe einer Mobility Plattform ist die Unterstützung der Mobilisierung von Anwendungen. Während es im Consumer-Umfeld im Wesentlichen um die Entwicklung einer mobilen Anwendung geht, stehen im Enterprise-Umfeld die Anbindung an die bestehenden Systeme und Anwendungen im Vordergrund, sowie das Management einer Vielzahl von Anwendungen und Geräten. Die folgenden Grundaspekte einer Mobility Plattform beziehen sich daher auf Enterprise-Anwendungen.

Abbildung 1: Bestandteile einer Mobility Plattform



2.1 Mobile Application Development

Design und Entwicklung der mobilen Anwendung ist bei jeder Mobility Plattform ein wichtiger Bestandteil. Durch die oftmals heterogenen Geräte und Betriebssysteme in einem Unternehmen ist einer der wichtigsten Aspekte bei der Entwicklung der mobilen Anwendungen die Unterstützung von mehreren mobilen Plattformen. Die wenigsten Unternehmen sind bereit, für die gleiche Anwendung verschiedene Varianten implementieren zu lassen, etwa für iOS und für Android. Sowohl der initiale Aufwand als auch der Wartungsaufwand steigt mit jeder getrennt zu pflegenden Anwendung an. Daher sind insbesondere Web-Apps, also mobile Anwendung auf Basis von HTML5 und JavaScript, sowie deren native Abbildung als Hybride App für Unternehmen sehr interessant und werden von den meisten Plattformen angeboten. Alternativ dazu bieten einige Plattformen einen vollwertigen „cross-plattform“-Ansatz an, bei dem die App in einer Metasprache erstellt wird, aus der dann jede unterstützte Zielplattform generiert werden kann. Somit muss auch in diesem Fall nur eine App entwickelt und gewartet werden, durch die beliebig viele Apps generiert werden können.
Neben der Plattform-Frage werden bei der Entwicklung von mobilen Anwendungen folgende Aspekte durch viele Mobility Plattformen adressiert:

Offline-Fähigkeit

Eine mobile Anwendung, die einen ständigen Zugang zum Inter- oder Intranet erfordert, ist in vielen Situationen unbrauchbar. Daher ist die Offline-Fähigkeit von mobilen Anwendungen in vielen Bereichen unentbehrlich. Die Implementierung solch einer Offline-Fähigkeit ist zum Teil sehr aufwändig und komplex. Mobility Plattformen bieten hierzu vollständige Frameworks und Bibliotheken an, die in den eigenen Anwendungen genutzt werden können.

Security

Das Thema Security in mobilen Anwendungen ist vielfältig: angefangen von der Authentifizierung und Autorisierung der mobilen Anwendung bis hin zur Verschlüsselung von Daten auf dem Endgerät müssen Anwendungsentwickler viel Aufwand in die Absicherung der mobilen Anwendung stecken. Auch hier können Mobility Platformen mit Bibliotheken und Frameworks die Komplexität und den Aufwand reduzieren.

Benachrichtigungen

Die Möglichkeit, Benachrichtigungen an mobile Endgeräte zu schicken, bietet Unternehmen ganz neue Möglichkeiten bei der Interaktion mit Mitarbeitern und Anwendungen. Daher ist dies ein elementarer Bestandteil einer mobilen Anwendung. Die Implementierung ist jedoch für jede mobile Plattform unterschiedlich. Hier bieten Mobility Plattformen einheitliche Services an, um Benachrichtigungen einfach in die eigene Anwendung zu integrieren und plattform-übergreifend zu nutzen.

Application Designer

Die Implementierung der Oberflächen einer mobilen Anwendung ist zeitaufwändig und fehleranfällig. Insbesondere durch die Variabilität bei den Endgeräten hinsichtlich Bildschirmgröße und Auflösung muss das Layout einer Oberfläche so gewählt werden, dass es unter allen unterstützten Geräten optimal oder zumindest fehlerfrei angezeigt wird. Um hier die Entwickler zu unterstützen, bieten einige Mobility Plattformen eigene UI-Designer an, mit der die Oberfläche der Anwendung über einen grafischen Editor erstellt werden kann. Integrierte Simulatoren und Vorschau-Optionen erleichtern die schnelle Rückmeldung über das gewählte Layout auf verschiedenen Geräten und bei verschiedenen Displaygrößen.

2.2 Backend Integration

Bei der Erstellung einer mobilen Enterprise-Anwendung geht es in der Regel um die Mobilisierung einer bestehenden Anwendung oder eines Systems. Bestehende Funktionalität soll durch die mobile Anwendung wiederverwendet werden. Damit ist ein Kernaspekt bei der Entwicklung der mobilen App die Integration in die bestehenden Anwendungen und Systeme. Da viele Anwendungslandschaften in Unternehmen sehr inhomogen sind, müssen hier verschiedenste Schnittstellen, Sprachen und Protokolle beherrscht werden, um mit den Backendsystemen kommunizieren zu können. Gleichzeitig bieten die wenigsten Backendsysteme geeignete Schnittstellen für die mobile Nutzung an, da hierbei besonders auf eine optimierte und möglichst schlanke Nutzlast geachtet werden muss. Daher bieten nahezu alle Mobility Plattformen eine Middleware in Form eines Application Servers an, über den die Kommunikation mit den Backendsystemen erfolgen kann und die den mobilen Applikationen eine geeignete Schnittstelle auf diese Systeme zur Verfügung stellt.
Der Integrationsumfang dieser Mobility Plattformen ist jedoch unterschiedlich. Im einfachsten Fall ermöglicht die Plattform, eine Schnittstelle des Backendsystems aufzurufen und dessen Daten – in abgewandelter oder optimierter Form – an das mobile Endgerät zu übertragen. In vielen Fällen ist dies jedoch nicht ausreichend, da z. B. mehrere Backensysteme genutzt werden müssen oder die Daten durch eine komplexere Logik transformiert werden muss. Hierzu bieten einige Mobility Plattformen die Möglichkeit, Anwendungslogik zu integrieren, die auf den Application Servern der Mobility Plattform läuft. Diese wird oftmals deskriptiv erstellt, also etwa in Form eines Ablaufdiagramms oder einer XML-Beschreibung, in einigen Fällen kann hier auch programmatisch in den Programmfluss eingegriffen werden.
Letztliches Unterscheidungsmerkmal bei der Anbindung an bestehende Backendsysteme ist die Unterstützung verschiedener Systeme und Schnittstellen. Angefangen von plattformunabhängigen Formaten wie SOAP-XML oder REST, bis hin zu proprietären Schnittstellen wie SAP-RFC oder CICS, ist die Vielfalt bei den angebotenen Mobility Plattformen groß.
Neben der reinen Anbindung an Backendsysteme muss die Middleware weiterhin die serverseitige Abbildung der am Client genutzten Services anbieten:

Offline-Fähigkeit

Offline-fähige Anwendungen müssen zum Einen mit den notwendigen Daten für die Offline-Fähigkeit versorgt werden. Zum Anderen sollen sie alle Änderungen nach einer erneuten Verbindung beidseitig synchronisieren. Dies ist insbesondere bei der Anbindung von Backendsystemen komplex, hier muss auf Versionierung und Sperrkonzepte geachtet werden.

Benachrichtigungen

Mobile Anwendungen können auf Benachrichtigungen reagieren, diese müssen jedoch vom Server initiiert werden. Dies muss die Middleware leisten und in die verschiedenen Prozesse und Anwendungen integriert werden können.

Security

Authentifizierung und Autorisierung der Client-Anwendungen müssen von der Middleware verarbeitet werden können. Daneben sollten evtl. angebundene Identity Management Systeme wie etwa ein LDAP integriert werden können.

2.3 Application Management

„Apps, die von öffentlichen App Stores für mobile Endgeräte heruntergeladen werden, schränken die IT-Sicherheit ein und torpedieren die Anwendungs- und Einkaufsstrategie“, Ian Finley, Gartner

Die Entwicklung einer mobilen Anwendung und der dazugehörigen serverseitigen Anbindung an Backendsysteme ist nur der erste Schritt. Diese Anwendungen müssen genauso wie herkömmliche Anwendungen in einem definierten Prozess getestet, abgenommen und verteilt werden. Anschließend muss die Anwendung gewartet und verwaltet werden, Änderungen verteilt werden, die Nutzung überwacht werden, Fehler aufgenommen werden, etc. Daher ist ein Enterprise App Store für die Verteilung und Überwachung von mobilen Anwendungen notwendig. Zwar könnten mobile Anwendungen auch über öffentliche App Stores wie etwa von Apple oder Google vertrieben werden, dies hat jedoch zwei Nachteile: die Apps müssen den Richtlinien der App Stores entsprechen, was aufgrund von Architektur und Security oftmals zu Problemen führt. Zudem macht sich ein Unternehmen angreifbar, wenn externe Nutzer die App installieren können. Über einen eigenen Enterprise App Store können interne Anwendungen an die Mitarbeiter verteilt werden, bei notwendigen Updates diese automatisch aktualisiert werden, und die Nutzung und Verteilung der Anwendung überwacht werden. Durch vorgelagerte Prozesse kann der Test und die Abnahme neuer Anwendungen kontrolliert werden.
Das Application Management beinhaltet auch die Verwaltung des Betriebs einer Anwendung, sowohl die clientseitige Anwendung auf dem mobilen Endgerät, als auch die serverseitige Komponente auf der Middleware der Mobility Plattform. Wie verhalten sich Laufzeiten und Ressourcen-Verbrauch? Wie ist die Last auf die einzelnen Anwendungen verteilt? Wo entstehen potentiell Engpässe? Welche Anwendungen sind von einer Störung betroffen? Diese Fragen müssen über ein geeignetes Application Management beantwortet werden können.
Ein letzter Aspekt beim Application Management ist die Verwaltung der Benutzer und Berechtigungen. Welche Benutzer auf welche Anwendung zugreifen können und welche Funktionen sie innerhalb der Anwendung nutzen dürfen, muss vom Administrator jederzeit kontrolliert und geändert werden können. Oftmals werden bestehende Verzeichnisdienste wie ein LDAP-Server eingebunden, damit Benutzerinformationen nicht redundant im Unternehmen vorliegen.

2.4 Device Management

Neben der Verwaltung der mobilen Anwendungen müssen natürlich auch die mobilen Endgeräte im Unternehmen verwaltet werden können. Über ein Mobile Device Management können alle mobilen Endgeräte zentral verwaltet und gesteuert werden. Wichtige Sicherheits-Einstellungen können zentral auf alle Geräte übertragen und überwacht werden. Im Notfall, etwa bei Verlust oder Diebstahl, können Geräte aus der Ferne gesperrt oder gelöscht werden. Einige Mobility Plattformen haben ein entsprechendes Device Management in ihrem System integriert, andere setzen hier auf Drittanbieter, die angebunden werden können. Der Vorteil einer integrierten Lösung ist die Reduzierung der Systeme, so dass nur ein System für alle Aspekte von Enterprise Mobility verwaltet werden muss. Gleichzeitig bieten spezialisierte MDM-Produkte zum Teil einen größeren Funktionsumfang an. Oftmals entscheidet sich diese Frage durch die vorhanden Tatsachen: ein Mobile Device Management ist in vielen Unternehmen bereits vorhanden, eine Mobility Plattform jedoch eher selten. In diesen Fällen wird oftmals das bestehende MDM weiterverwendet und infolgedessen dieser Aspekt der Mobility Plattform vernachlässigt.

2.5 Feature-Matrix

Die folgende Matrix zeigt die grundlegenden Features von Mobility Plattformen und deren Abdeckung durch die hier vorgestellten Produkte.

Abbildung 2: Feature-Matrix Mobility Plattformen

	IBM Worklight	SAP SUP	Kony	Verivo Akula	Convertigo	FeedHenry
Entwicklung von nativen Apps	✓		✓	✓		✓
Entwicklung von web Apps	✓	✓	✓	✓	✓	✓
Entwicklung von hybriden Apps	✓	✓	✓	✓	✓	✓
Entwicklung von cross-plattform Apps			✓			
Integrierte Entwicklungsumgebung (local)	✓	✓	✓	✓	✓	
Integrierte Entwicklungsumgebung (cloud)						✓
Graphischer UI-Editor	✓	✓	✓	1		
Client Bibliotheken / APIs	✓	✓	✓	✓	✓	✓
Preview-Funktionalität	✓	✓	✓	✓	✓	✓
Client Debugging	✓	✓	✓	✓	✓	✓
Anbindung SAP		✓	✓	✓		
Anbindung WebServices	✓	✓	✓	✓	✓	✓
Anbindung REST-Services	✓	✓	✓	✓	✓	✓
Anbindung Datenbanken	✓	✓	✓	✓	✓	✓
Anbindung weitere Dienste	✓	✓	✓	✓	✓	✓
Graphischer Editor für Backend-Services	✓	✓	✓		✓	
Integrierte Entwicklungsumgebung (local)	✓	✓	✓	✓	✓	
Integrierte Entwicklungsumgebung (cloud)						✓
Serverseitige Datenverarbeitung	✓	✓	✓	✓	✓	✓
Serverseitige Anwendungslogik	✓			✓	✓	✓
Logging	✓	✓	✓	✓	✓	✓
Debugging						
Enterprise App Store	✓2	✓2	✓			✓
Benutzerverwaltung / Rechteverwaltung	✓	✓	✓			
Anbindung LDAP / AD	✓	✓	✓			✓
Reporting / Monitoring	✓	✓	✓			✓
Clustering	✓	✓	? 3		✓	? 3
Cloud Installation		✓	✓		✓	✓
On-Premise Installation	✓	✓	✓	✓	✓	
Device Management	✓2	✓2	✓			
iOS Unterstützung	✓	✓	✓			
Android Unterstützung	✓	✓	✓			
Blackberry 10 Unterstützung	✓		✓			
Windows Unterstützung	✓	✓	✓			

Application Development
Backend Integration
Application Management
Device Management

1 Graphischer Editor der genutzten Plattform (z.B. Xcode)
2 Über zusätzliches Produkt des gleichen Anbieters
3 Nicht angegeben

03 Marktübersicht

3.1 Einordnung

Der Markt der Mobility Plattformen ist unübersichtlich und inhomogen. Durch die fehlende eindeutige Definition des Begriffs „Mobility Plattform“ finden sich hier Anbieter mit sehr unterschiedlichen Schwerpunkten und Auslegungen dieses Begriffs. Grob kann man die Plattformen in die folgenden beiden Kategorien einteilen:

„One of the most interesting aspects of the MADP market is that traditional enterprise software, low-cost distruptors and open-source sales models are simultaneously having an impact on the market“
– Gartner, 2013

Mobile Application Development Platforms

Plattformen in dieser Kategorie setzen den Schwerpunkt auf die Entwicklung der mobilen Applikation. Durch umfangreiche Tools und Frameworks wird der Aufwand und das erforderliche Wissen zur Erstellung von mobilen Apps reduziert, gleichzeitig ermöglichen viele der Plattformen die Erstellung von cross-plattform Apps, also Anwendungen, die auf verschiedenen mobilen Endgeräten laufen können. Viele der Produkte bieten auch eine Anbindung an gängige Enterprise Backend-Systeme an. Hierbei wird in der Regel auch eine Middleware mitgeliefert, über die diese Anbindung erfolgt.

Mobile Enterprise Application Platforms

Plattformen dieser Kategorie decken ein breiteres Spektrum von Enterprise Mobility ab. Auch hier ist die Entwicklung der mobilen Apps mit Hilfe von Tools und Frameworks ein wichtiger Bestandteil, aber zusätzlich werden zentrale Aspekte wie das Management der Apps und Geräte sowie die Einbindung in die Unternehmensinfrastruktur abgedeckt.

Diese Trennung wird von den Anbietern nicht klar benannt, zudem gibt es auch viele Anbieter, die sich zwischen diesen beiden Kategorien bewegen. Somit ist eine klare Zuordnung einer Mobility Plattform zu einer dieser beiden Kategorien nicht immer möglich.

Die bekannteste Bewertung und Definition von Mobility Plattformen stammt von Gartner, die mit ihrem „Magic Quadrant Mobile Application Development Platforms“ Anbieter und Produkte der ersten Kategorie analysieren. Gleichzeitig sind einige der dort untersuchten Produkte keine reinen Development Plattformen, sondern bewegen sich stark in die Richtung der Application Plattformen. Daher lässt sich der Gartner-Report nicht zur Definition der Anbieter heranziehen.

3.2 Anbieter & Produkte

Aus der Vielzahl an Anbietern und Produkten am Markt wurden für diese Studie sechs Produkte untersucht. Dabei sind sowohl die „Big Player“ wie SAP und IBM vertreten, aber auch kleinere und weniger bekannte Anbieter, die teilweise unterschiedliche Ansätze und Strategien verfolgen. In Kapitel 3.3 werden weitere Anbieter aufgelistet, die aber in dieser Studie nicht weiter betrachtet wurden.

Akula

Die Mobility Plattform „Akula“ der US-amerikanischen Firma Verivo ist seit 2012 auf dem Markt. Akula fokussiert sich dabei auf die Bereitstellung von Bibliotheken für die Entwicklung von Enterprise-Apps sowie eines Servers für die Kommunikation mit den Backend-Systemen. Dadurch sollen Entwickler sich auf die wesentlichen Aspekte einer mobilen Lösung und deren fachliche Anforderungen konzentrieren können.
Akula besteht aus drei Komponenten:

Akula Server

Der auf JEE 7 basierende Server läuft wahlweise auf einem Tomcat oder JBoss Server als On-Premise-Variante im Unternehmen. Der Server liefert Konnektoren für verschiedene Backend-Systeme wie Datenbanken, Web-Services oder SalesForce. Auf den Akula Server werden die entwickelten Server-Komponenten für die mobilen Anwendungen installiert.

Akula Server SDK

Das SDK für die Server-Komponenten bietet Klassen zur Nutzung der Akula-Komponenten für die Kommunikation mit den Backend-Systemen und den mobilen Clients. Die Implementierung erfolgt vollständig in Java.

Akula Client SDK

Akula bietet SDKs für native Apps unter Android und iOS an, zudem ein SDK für Web-Apps mit HTML5 und JavaScript, optional auch als Container in PhoneGap. Die Projekte selber müssen manuell in der jeweiligen IDE angelegt werden, anschließend können die SDKs als Bibliotheken eingebunden und genutzt werden.
Akula bietet keine eigenen Produkte für Application und Device Management an.

Convertigo

Convertigo ist eine französische Firma mit einer gleichnamigen Mobility Plattform. Gegründet wurde die Firma 2009, ihren Hauptsitz hat die Firma in Orsay in Frankreich und beschäftigt aktuell etwa 25 Mitarbeiter. Convertigo wird vollständig unter Open Source zur Verfügung gestellt, die Entwicklung wird jedoch zentral durch die Firma selber betrieben. Convertigo bietet eine Entwicklungsumgebung auf Eclipse-Basis sowie einen auf Apache Tomcat basierenden Server an.
Convertigo besteht aus zwei zentralen Komponenten:

Convertigo Studio

Die auf Eclipse basierende Entwicklungsumgebung ermöglicht die Entwicklung von Web- oder hybriden Anwendungen. Unterstützt werden dabei verschiedene Framworks wie etwa jQuery Mobile oder Sencha Touch, für die hybriden Container wird PhoneGap integriert. Die komplette Client-Entwicklung erfolgt in HTML, JavaScript und CSS. Es wird kein graphischer UI-Editor angeboten. Neben der Client-seitigen Entwicklung wird in Convertigo Studio auch die Backend-Anbindung definiert. Dies erfolgt über verschiedene Konnektoren, die gängige Schnittstellen wie HTTP und SQL unterstützen und sogar einen CICS-Connector für IBM Mainframe Systeme. Mittels der Konnektoren können Backend-Schnittstellen angesprochen werden und über eine definierte Abfolge an Schritten (genannt „Transactions“) die Verarbeitung der erhaltenen Daten gesteuert werden. Diese Anbindung erfolgt vollständig ohne Programmierung.

Convertigo Enterprise Mobility Server

Der Enterprise Mobility Server ist die Middleware zwischen Client und angebundener Backend-Systeme. Auf den Server werden die im Studio definierten Anbindungen und Verarbeitungen (Transactions) installiert und ausgeführt. Der Server kann sowohl im eigenen Rechenzentrum (on-premise) als auch als Cloud-Lösung betrieben werden.

FeedHenry

FeedHenry ist eine irische Firma und Anbieter ihrer gleichnamigen Mobility Plattform. Seit 2010 bietet FeedHenry ihr als Mobile Backend-as-a-Service (MBaaS) klassifiziertes Produkt an. Mobile Anwendungen können sowohl nativ, als web- oder hybrid-App entwickelt werden, die Anbindung an Backend-Server erfolgt über die Cloud und bietet Node.js als Umgebung an.
FeedHenry trennt die Entwicklung in Client- und Server-Anteile:

Client

Eine neue App wird immer über die FeedHenry-Cloud erstellt. Anschließend kann der Anwender verschiedene Varianten auswählen: die App kann als Projekt-Template für die gängigen Plattformen (iOS als XCode-Projekt, Android, HTML5) heruntergeladen und bearbeitet werden, oder aber im webbasierten Editor von FeedHenry, welcher eine HTML5-App mit einem optionalen Container bereitstellt. Hier kann der Entwickler direkt den Code editieren, bekommt jedoch keine Unterstützung durch einen graphischen Editor. Die Projekt-Templates für die anderen Plattformen enthalten FeedHenry-Bibliotheken für die gängigen Mobility-Services sowie den Zugang zum FeedHenry Cloud-Server, auf dem die serverseitigen Anteile der Anwendung laufen.

Server

Die serverseitige Logik der mobilen Anwendung wird auf Basis von Node.js im Web-Editor von FeedHenry implementiert. Dabei liefert FeedHenry Konnektoren zu verschiedenen Backend-Systemen wie SalesForce, SharePoint und verschiedenen Datenbanken und Serviceanbietern an. Die Implementierung wird durch die Bereitstellung von Code-Beispielen erleichtert, zudem können die Skripte direkt aus dem Editor ausgeführt und die Ausgabe in einer Echtzeit-Vorschau betrachtet werden.
Die entwickelten mobilen Anwendungen können über einen cloud-basierten App-Store verteilt werden, die Serverkomponenten werden ebenfalls auf dem Cloud-Server installiert. Über verschiedene Auswertungen können zudem die Nutzung der Apps überwacht und verfolgt werden.

KonyOne

Kony ist einer der führenden Anbieter von Mobility Plattformen und ist seit 2007 mit dem Produkt „KonyOne“ am Markt vertreten. Bisher hat Kony insbesondere den Consumer-Markt adressiert, seit einiger Zeit betreibt Kony aber einen verstärkten Ausbau des Enterprise-Feldes und ist mittlerweile zu einem nahezu vollwertigen Anbieter einer Mobile Enterprise Application Platform herangewachsen.
Mit etwa 800 Mitarbeitern und über 58 Millionen Dollar Venture-Kapital ist das Unternehmen breit aufgestellt. Nach eigenen Aussagen hat das Unternehmen bereits über 350 Kunden in 45 Ländern, darunter 70 der 500 größten Unternehmen der Welt (Fortune 500).
Die KonyOne Plattform besteht aus fünf Komponenten:

Kony Visualization Cloud

Für die ersten Phasen der Entwicklung mobiler Anwendungen bietet Kony mit der Visualization Cloud ein Werkzeug zur Erstellung von grafischen Prototypen. Diese können über eine Weboberfläche erstellt werden und können damit mit geringem Aufwand einen Eindruck der späteren Anwendung vermitteln. Für die Implementierung der App kann dieser Prototyp dann in die Kony Development Cloud übernommen werden und dient als Vorlage für die Umsetzung.

KonyOne Server

Die Anbindung an die Backendsysteme erfolgt über eine Middleware auf Basis von JBoss oder Tomcat, den KonyOne Server. Benötigte Schnittstellen werden über Service-Definitionen aus dem KonyOne Studio installiert und können dann von den mobilen Anwendungen genutzt werden.

KonyOne Studio / Kony Development Cloud

Das Herzstück der KonyOne Plattform ist die Entwicklungsumgebung KonyOne Studio. Diese auf Eclipse basierende IDE bietet umfangreiche Werkzeuge zur Erstellung von cross-plattform Apps über einen einfach zu bedienenden UI-Designer. Kony unterstützt dabei nahezu alle mobile Plattformen, die durch einen Generator aus dem Kony-eigenen Format erzeugt werden können.

Kony Management Cloud

Die Bereiche Application Management und Device Management fasst Kony unter der Kony Management Cloud zusammen. Hier können die entwickelten Apps verwaltet werden, Geräte konfiguriert werden, und Auswertungen sowie Statistiken zur Nutzung der Anwendungen eingesehen werden.

Kony Apps Cloud

Als letzten Bestandteil bietet Kony über die Apps Cloud vorgefertigte mobile Anwendungen für bestimmte Unternehmensbereiche an. Diese können entweder direkt genutzt werden, oder werden für das Unternehmen spezifisch angepasst.

Sybase Unwired Platform

SAP ist mit der Übernahme der Sybase Unwired Platform (SUP) zu einem der großen Anbieter von Mobility Plattformen aufgestiegen. Seit 2008 ist die Sybase Unwired Platform bereits am Markt verfügbar. Die Sybase Unwired Platform besteht aus zwei Komponenten:

SAP mobile SDK

Die auf Eclipse basierende Entwicklungsumgebung ermöglicht die Entwicklung von mobilen Anwendungen und die Definition von angebundenen Backend-Systemen und -Schnittstellen. Dabei wird der Schwerpunkt auf die Anbindung der eigenen SAP-Systeme gesetzt, zusätzlich können auch Datenbanken oder Web-Services angebunden werden. Die Schnittstellen zwischen Client und Server werden über Mobile Business Objects (MBO) definiert, welche sowohl die verarbeiteten Daten als auch die Schnittstellen am Backend definieren. Diese können dann über einen graphischen UI-Editor direkt in die entsprechenden Elemente der Oberfläche verlinkt werden. SAP mobile SDK unterstützt Web-Apps in Kombination mit verschiedenen Frameworks wie SAPUI5 oder Sencha Touch sowie hybride Apps über PhoneGap.

SAP Unwired Server

Die serverseitige Anbindung der Backend-Systeme (insbesondere SAP-Systeme) erfolgt durch den SAP Unwired Server. Hier werden die vom SAP mobile SDK erstellten Server-Schnittstellen (MBO) installiert und führen die eigentliche Kommunikation mit dem Backend durch.

Neben der Sybase Unwired Platform bietet SAP über das Produkt Afaria ein Mobile Device Management an. Hier können die gewohnten Funktionen eines Device Management Systems genutzt werden. Es besteht die Möglichkeit, Afaria direkt in er SAP Mobile Platform zu integrieren, sodass diese beiden Produkte homogen zusammenarbeiten können. Es bleiben aber weiterhin zwei getrennte Anwendungen.

Ein Application Management bzw. einen eigenständigen Enterprise App Store bietet die Sybase Unwired Platform nicht an.

Worklight

IBM hat mit der Übernahme des israelischen Konzerns „Worklight“ und dessen Produkte Anfang 2012 sein Produktangebot um eine Mobility Plattform erweitert, welche fortan unter dem Namen IBM Worklight vertrieben wird. Damit können native, hybride und web-Anwendungen entwickelt und verwaltet werden.
IBM Worklight besteht aus verschiedenen Komponenten:

Worklight Studio

Die auf Eclipse basierende Entwicklungsumgebung ermöglicht die Entwicklung von nativen, hybriden und web-Anwendungen mit Unterstützung durch einen graphischen Editor. Damit können Oberflächen ohne großen Programmieraufwand zusammengestellt werden. Es unterstützt verschiedene Frameworks wie jQuery Mobile, Dojo Mobile oder Sencha Touch. Weiterhin wird im Worklight Studio die serverseitige Anbindung der mobilen Anwendungen an die Backend-Server über sogenannte „Adapter“ und „Procedures“ implementiert. Hier werden gängige Schnittstellen-Formate wie Web-Services über REST und SOAP unterstützt. Die Adapter werden auf den Worklight Server installiert und können dann von der Anwendung genutzt werden.

Worklight Server

Die Laufzeitumgebung für die serverseitigen Komponenten der mobilen Anwendung laufen auf dem Worklight Server. Hier werden die implementierten Anbindungen an die Backend-Systeme installiert und ausgeführt.
Das Mobile Device Management sowie das Application Management erfolgt bei IBM über getrennte Produkte. Für das Device Management bietet IBM den Endpoint Manager an. Dieser unterstützt alle gängigen Funktionen eines Device Managements an und kann mit dem Worklight Server zusammenarbeiten. Für das Application Management bietet IBM den Worklight Application Center an, welcher einen Enterprise App Store und die nötigen Management-Funktionen zur Verwaltung der eigenen mobilen Anwendungen anbietet.

3.3 Weitere Anbieter

Jeden Monat tauchen am Markt neue Anbieter von Mobility Plattformen auf, daher ist eine vollständige Übersicht aller Produkte und Hersteller kaum möglich. Die folgende Liste enthält daher nur eine unvollständige Auflistung weiterer Anbieter:

MADP nach Gartner Magic Quadrant 2013

- Adobe – Air / Edge / PhoneGap
- Antenna – AMPchroma
- Appcelerator – Titanium
- Apple – iOS / XCode
- Blackberry – Blackberry SDK
- ClickSoftware
- Dojo – Toolkit / Mobile / Dijit / Maqetta
- DSI – Mobile Platform
- Google – Android SDK
- jQuery – jQuery Mobile
- Microsoft – Windows 8 / Azure
- MicroStrategy – Mobile App Platform
- Motorola Solutions – RhoMobile Suite
- Netbiscuits – Tactile
- salesforce.com – SalesForce Touch Platform
- Sencha – Architect / Touch / Charts
- Usablenet – U-Experience
- Xamarin – Xamarin

Europäische Anbieter

- Appear Networks – IQ Mobility Platform
- commsult – mobileERP
- Mobile Data Collection – MDC
- M-Way Solutions – mCAP Mobility Platform
- VeliQ – MobiDM

Open Source / Community

- AeroGear
- OpenMEAP

04 Beispiel-Implementierung

Anhand einer Beispiel-Anwendung sollen die Funktionen und Arbeitsweisen der verschiedenen Plattformen vorgestellt werden. Dabei werden die grundlegenden Arbeitsschritte zur Erstellung der mobilen Anwendung vorgestellt und bestimmte Aspekte der jeweiligen Plattform hervorgehoben.

Als Referenz-Beispiel wird eine einfache mobile Anwendung erstellt, die Kontakte aus einem Backend-System ausliest, diese in einer Liste darstellt und die Möglichkeit zum Editieren der Kontakte bietet. Diese Basisanwendung wird dann je nach Funktionsumfang der jeweiligen Plattform erweitert um Authentifizierung, Offline-Fähigkeit und Push-Benachrichtigungen.

Die Anwendung soll sowohl auf iOS als auch Android-Geräten lauffähig sein. Daher wird als Grundumfang eine hybride Anwendung auf HTML5-Basis erstellt, die dann zusätzlich über einen nativen Container verfügbar sein soll. Optional werden zusätzlich auch native Anwendungen erstellt.

Als Backend-System wird ein Web-Service-kompatibles System wie etwa Microsoft Outlook über den Exchange Web-Service (EWS) angebunden. Optional wird hier auch ein SAP-System über RFC angebunden.

Folgende Schritte werden für jede Plattform durchgeführt und beschrieben:

1. **Aufsetzen des Projekts**
2. **Anbinden der Backends**
3. **Test**
4. **Verwaltung der Anwendung**
5. **Verteilen der Anwendung**
6. **Implementieren der Client-Logik**
7. **Implementieren der Oberflächen**

Verivo 200 Waltham, MA, US k.A.

Akula

Für dieses Beispiel werden die Akula SDKs in der Version 1.0.1 verwendet. Da Akula keine eigene IDE mitliefert, kann hier auf die bevorzugte Entwicklungsumgebung zurückgegriffen werden.

1. Aufsetzen des Projekts

Akula unterstützt die Zielplattformen iOS, Android und JavaScript (web oder hybrid), dafür muss jeweils ein Client SDK von der Akula Webseite heruntergeladen und installiert werden. Abhängig von der Zielplattform erfolgt dann die Entwicklung in der jeweils geeigneten IDE, also XCode für iOS, Eclipse für Android und JavaScript. Akula macht keine konkreten Vorgaben bezüglich der IDE, bietet damit aber auch keine direkte Integration in etwa Eclipse an. Weiterhin muss der Akula Server auf dem lokalen Rechner installiert werden.

Für die hybride App wird das Akula JavaScript SDK genutzt. Dieses ist eine Sammlung an JavaScript-Dateien, welche die Kommunikation mit dem Akula Server steuern, der Entwickler kann sich rein auf die Entwicklung der App konzentrieren. Die JavaScript-Bibliotheken müssen in das jeweilige Projekt eingebunden werden.

Für die Entwicklung der serverseitigen Logik wird ein eigenes Projekt angelegt, welches eine definierte Struktur haben muss. Diese muss vom Entwickler selber angelegt werden. Hier werden über XML-Dateien die Steuerflüsse definiert, und über Java-Klassen können individuelle Backend-Module implementiert werden.

2. Implementieren der Oberflächen

Die Oberflächen der mobilen Anwendung werden im Fall der hybriden Web-App in der jeweiligen IDE mit HTML, JavaScript und CSS implementiert. Dabei besteht keine Abhängigkeit zu den Akula Bibliotheken, hier kann somit grundsätzlich jedes UI-Framework und jeder Editor genutzt werden. Das gleiche gilt für native Apps, auch hier wird die App unabhängig von Akula in der Umgebung der Zielplattform entwickelt.

3. Anbinden der Backends

Die Anbindung der Backends erfolgt bei Akula in drei Schritten. Zunächst müssen über sogenannte „Modules“ der WebService angebunden werden. Hierzu liefert Akula eine Reihe vordefinierter Modules mit, unter Anderem ein HTTP Module. Über eine XML-Konfiguration kann dieses Module definiert werden, hier müssen Endpoint und SOAP-Template beschrieben werden. Im zweiten Schritt wird dieses Module durch eine „Route“ aufgerufen. Eine Route beschreibt eine Folge an Transaktionen, über die Daten von entfernten Systemen oder Datenbanken gelesen und transformiert werden können. Auch die Routes werden über eine XML-Datei konfiguriert, hier wird als einer der Transaktionen die XML-Konfiguration des HTTP Module referenziert. Im letzten Schritt wird nun ein von außen erreichbarer Endpoint definiert, über den die mobilen Anwendungen die Route und Modules aufrufen können. Diese Definition erfolgt ebenfalls über eine XML-Datei, die URL und

Operation der REST-Schnittstelle definiert.
Für komplexere Operationen können auch Java-Klassen implementiert werden, die dann als Modules in einer Route ausgeführt werden können. Dabei wird das Akula SDK genutzt, um von definierten Klassen zu erben, und hier die spezifische Funktionalität zu erweitern. Alle Elemente zusammen werden dann in einen sogenannten „App Scope“ gebaut, einem Archiv, welches sowohl die XML-Konfigurationen, die Java-Klassen und die Beschreibungen des App Scopes enthalten. Dieses Archiv kann dann auf dem Akula Server installiert werden.

4. Implementieren der Client-Logik

Die Client-Logik und die Anbindung an den Akula Server erfolgen über die mitgelieferten Akula Bibliotheken. Diese liefern Operationen für Authentifizierung und Autorisierung, Zugriff auf bereitgestellte Server Endpoints, und weitere Bibliotheken. Das Mapping der gelesenen Daten wiederum erfolgt in der Anwendung selber, muss also vollständig implementiert werden. Die interne Client-Logik muss ebenfalls manuell implementiert werden.

5. Test

Die mobile Anwendung kann über die vorhandenen Mittel der IDE bzw. im Browser getestet werden. Akula liefert hierzu keine weitergehende Unterstützung. Für das Testen der Server-Anteile können die definierten Endpoints direkt per Browser oder eines REST-Clients aufgerufen und das Ergebnis überprüft werden. Fehlermeldungen werden in eine Logfile im Akula Server geschrieben, über dieses kann eine Fehleranalyse durchgeführt werden.

6. Verteilen der Anwendung

Die mobile Anwendung wird mit den üblichen Mitteln gebaut. Im Falle einer hybriden App kann PhoneGap/Cordova genutzt werden, hierfür liefert Akula zusätzliche Hilfsmittel, um die entsprechende App zu bauen. Die weitere Verteilung der mobilen App erfolgt dann über Drittanbieter App Stores. Die Serveranteile können über das App-Scope Archiv auf einem beliebigen Server installiert werden. Abhängig vom Server und der URL muss jedoch der Client angepasst werden.

7. Verwaltung der Anwendung

Akula bietet keine weiteren Möglichkeiten zur Administration und Verwaltung der Anwendungen.

convertigo 25 Orsay, FR k.A.

Convertigo

Nachfolgend wird die Entwicklung der Beispielanwendung mit Convertigo erläutert. Hierbei kommen die beiden Hauptkomponenten, das Convertigo Studio und der Convertigo Enterprise Mobility Server, zum Einsatz. Der Server kann entweder lokal oder in der Cloud betrieben werden und verwendet einen Apache Tomcat. Die für dieses Szenario vorliegende Trial Version unterstützt allerdings nur eine lokale Installation.

1. Aufsetzen des Projekts

Zunächst muss das Convertigo Studio installiert werden. Da Convertigo OpenSource ist kann das Studio direkt über SourceForge bezogen werden. Über die installierte IDE ist es möglich ein neues Projekt anzulegen, welches zunächst im lokalen Workspace abgelegt wird. Der verwendet Workspace kann anschließend mit dem Convertigo Server abgeglichen werden, um so auch anderen Entwicklern zur Verfügung zu stehen. Alternativ kann es auch in ein SVN Repository überführt werden. Für das neu angelegte Projekt stehen verschiedene Templates zur Verfügung, über die eine grobe Struktur der App erzeugen, auf der dann die Entwicklung aufsetzen kann. Um eine hybride App zu entwickeln wird hier das auf jQuery Mobile basierende Template ausgewählt.

2. Implementieren der Oberflächen

Die Oberflächen der mobilen Anwendung müssen von Hand entwickelt werden. Zwar erleichtern die UI-Frameworks wie jQuery Mobile oder Sencha Touch die Entwicklung, allerdings ist kein grafischer UI-Editor in Convertigo Studio vorhanden. Die Entwicklung der Oberfläche erfolgt somit per HTML, JavaScript und CSS, eine native Entwicklung wird nicht unterstützt. Ein Live Preview der Oberfläche wird innerhalb der Entwicklungsumgebung nicht bereitgestellt, man kann die Anwendung jedoch über Eclipse als Web-Anwendung starten und im Browser testen. Das Layout der Anwendung kann dabei auf verschiedenen Geräten und Plattformen betrachtet werden.

3. Anbinden der Backends

Die Anbindung von Backend-Systemen erfolgt direkt über Eclipse. Hierbei bietet das Convertigo Studio einen Wizard an, welcher die Anbindung verschiedener Backend-System über Konnektoren ermöglicht. Nachdem der Konnektor für den WebService-Backend angelegt wurde, ist es nun im nächsten Schritt möglich, Operationen (Transaktionen) auf diesem Konnektor auszuführen. Bei einem WebService stehen hier entweder HTTP, JSON oder XML Transaktionen zur Verfügung. Um erhaltene Daten entsprechend aufzubereiten werden sogenannte Sequenzen eingesetzt. Sequenzen bestehen aus einem oder mehreren Schritten, mit welchen es möglich ist diverse Operationen auf den Daten durchzuführen. Diese Funktionalität ermöglicht es, Logik auf den Server auszulagern und somit ressourcen- und bandbreiteschonend

Daten für das Endgerät bereitzustellen und aufzuarbeiten. Sequenzen werden ebenfalls in Eclipse über einen Wizard definiert. Die korrekte Ausführung der Transaktionen und Sequenzen kann gegenüber dem Server mit zuvor definierten Testdaten überprüft werden.

4. Implementieren der Client-Logik

Die Client-Logik wird über das Convertigo Studio in HTML und JavaScript implementiert. Convertigo bietet mit dem Convertigo Templating Framework (CTF) die Möglichkeit, auf Transaktionen und Sequenzen des Convertigo Servers zuzugreifen und diese in den Client zu integrieren. Hierbei setzt CTF auf das Model-View-Controller Prinzip, wobei das Model die Sequenzen darstellen, der View in den HTML Templates zu finden ist und die Routingtabelle als Controller gilt. Das Mapping erfolgt direkt im Quellcode und wird über sogenannte C8O-calls realisiert. Hierdurch können direkt vom Client aus Transaktionen, Sequenzen oder Variablen angesprochen werden und die Daten in die Oberfläche gemappt werden.

5. Test

Convertigo bietet die Möglichkeit über das Webinterface (die Admin-Konsole) des Convertigo Servers die Applikation über einen Simulator zu testen. Bei dem Simulator handelt es sich letztendlich um einen View der Webapplikation, welcher die implementierte Oberfläche und Interaktion wiedergibt.

6. Verteilen der Anwendung

Wird ein Eclipse-Projekt im Workspace angelegt, welches mit dem Server automatisch synchronisiert wird, so kann die App über die Admin-Konsole für die verschiedenen Betriebssysteme gebaut werden. Nachdem der Build-Prozess erfolgreich abgeschlossen ist, wird ein QR-Code generiert welcher mit einem Link zum nativen Container der App versehen ist. Alternativ wird ebenfalls ein QR-Code mit einem Link zur Web-App Variante angezeigt. Die Links können verwendet werden um die App direkt dem Anwender zur Verfügung zu stellen oder aber um sie in einen bereits vorhanden Enterprise App Store eines Drittanbieters einzupflegen.

7. Verwaltung der Anwendung

Eine Verwaltung der Anwendung ist nur sehr rudimentär möglich. Es ist zwar möglich, die Projektdaten sowie die damit verbundenen Konnektoren zu überwachen, allerdings gibt es keine Informationen über die ausgerollten Apps, wie etwa Nutzungsstatistiken oder eingesetzte Versionen. Laut den Dokumenten von Convertigo, lassen sich jedoch auch In-App-Updates und Push Notifications realisieren. Zusätzlich soll es möglich sein für die Verwendung von gewissen Sequenzen Sicherheitsregeln zu definieren, welche Beispielsweise eine Authentifizierung erforderlich machen.

30 Waterford, IE k.A.

FeedHenry

Für dieses Beispiel nutzen wir die FeedHenry Hybrid Implementierung, für die eine web-basierte Entwicklungsumgebung bereitgestellt wird.

1. Aufsetzen des Projekts

Um mit der FeedHenry Plattform zu entwickeln, muss man sich zunächst für eine Entwicklungsart bzw. Zielplattform entscheiden. Bei einer hybriden App bietet Feedhery eine webbasierte Entwicklungsumgebung an, über die Apps vollständig im Browser entwickelt werden können. Für native Apps und Web Apps erzeugt FeedHenry lediglich den Projektrahmen, der von der Webseite heruntergeladen werden kann und im jeweiligen Editor (Eclipse, XCode) geöffnet und bearbeitet werden kann. Die Zugangsdaten zur Cloud-Umgebung müssen dabei von Hand in den Projekteigenschaften eingetragen werden. Die FeedHenry Bibliotheken sind in diesen Projekten bereits integriert und können direkt genutzt werden.

Für dieses Beispiel wird der hybride Ansatz von FeedHenry gewählt. Dazu wird zunächst eine App angelegt, anschließend wird „FeedHenry Hybrid“ als Entwicklungsart gewählt, und zuletzt wird der Editor gestartet.

2. Implementieren der Oberflächen

Die Oberflächen der FeedHenry Hybrid App werden im webbasierten Code-Editor bearbeitet. Ähnlich wie bei gewöhnlichen IDEs bietet die FeedHenry-Umgebung eine Baumansicht, in der die Dateien der App dargestellt werden, einen Code-Editor mit Syntax-Hervorhebung, sowie einer Vorschau der gerade entwickelten App. Der JavaScript-Editor bietet keine Code-Completion, zeigt jedoch Syntax-Fehler und Warnungen im Editor direkt an. Nach jedem Speichern einer Datei wird die Vorschau aktualisiert, somit kann die Auswirkung der Änderung direkt nachvollzogen werden. Die Vorschau kann auf verschiedene Geräte und Displaygrößen angepasst werden.

Einen UI-Designer bietet FeedHenry nicht, die Oberfläche muss daher komplett über HTML und JavaScript implementiert werden.

3. Anbinden der Backends

Zum Anbinden der Backends bietet FeedHenry diverse „Plugins“ an. Diese Plugins sind auf NodeJS basierende Implementierungen für die jeweiligen Zielplattformen wie Amazon, SAP oder SharePoint. Um Plugins zu nutzen müssen die Konfigurationen für das Backend angepasst werden und zunächst das benötigte Plugin importiert werden. Anschließend kann das Plugin im Cloud-Javascript genutzt werden, um etwa einen Aufruf auf einen Service durchzuführen. Zugangsdaten und IP-Adressen können über Umgebungsvariablen getrennt vom Quellcode definiert werden. Das Cloud-Javascript wird im gleichen Editor editiert wie auch die Client-Javascripts, die Dateien liegen jedoch in einem anderen Verzeichnis.

Die im Cloud-Javascript definierten Operationen können als reguläre REST-Services auf JSON-Basis aufgerufen werden, unter anderem auch aus der FeedHenry Client-App.

4. Implementieren der Client-Logik

Um im Client die bereitgestellten Operationen der FeedHenry Cloud aufzurufen, muss die Operation mit dem Namen über eine FeedHenry JavaScript-Bibliothek aufgerufen werden. Im Erfolgsfall wird ein Callback definiert, der die Daten im JSON-Format enthält, im Fehlerfall wird der Fehlercode und eine mögliche Fehlermeldung an den Client übermittelt.
Die zentrale Client-Logik wird regulär über Javascript und HTML entwickelt, hier bietet FeedHenry keine zusätzliche Unterstützung, etwa für das Mapping der Daten von der Cloud.

5. Test

Das Testen der entwickelten Applikation erfolgt zunächst über die Live Preview, die auch als eigenes Browser-Fenster gestartet werden kann, um mit den Browser-Mitteln auch den Code zu debuggen.
Die Server/Cloud-Anteile können über einfache REST-Aufrufe getestet werden. Über entsprechende Logs kann die Server-Anwendung in FeedHenry analysiert und Fehler identifiziert werden.

6. Verteilen der Anwendung

Sobald die Entwicklung abgeschlossen ist, kann die App für die benötigte Zielplattform gebaut werden. Diese kann in FeedHenry ausgewählt werden. Vor dem Bauen der App bietet FeedHenry die Möglichkeit an, für das MDM von Airwatch entsprechende SDKs zu integrieren, oder es sogar direkt in den Airwatch AppStore zu deployen. Anschließend wird der Quellcode vom Server gebaut und aufbereitet. Nach Abschluss dieses Vorgangs kann die App entweder direkt heruntergeladen werden, oder – sofern die Option gewählt wurde – aus dem öffentlichen Airwatch AppStore installiert werden. Alternativ bietet FeedHenry auch einen eigenen web-basierten AppStore für iOS und Android an.
Die Cloud-Anteile können über sogenannte Deployment Targets auf eine weitere Zielplattform installiert werden, z.B. um von der Testumgebung auf eine Produktivumgebung zu migrieren.

7. Verwaltung der Anwendung

FeedHenry bietet einige grundlegende Mittel zum Analysieren der Nutzung einer Anwendung. Dabei kann je Anwendung die Anzahl der Installationen, der Anwendungsstarts, der Cloudanfragen und der angemeldeten Benutzer erfasst werden. Zudem können Anfragen an die FeedHenry-Cloud getrackt werden, um etwa Laufzeit der Server-Requests zu erfassen.
Eine Benutzer- und Rechteverwaltung sowie ein Device Management bietet FeedHenry nicht an, hier muss auf Drittanbieter zurückgegriffen werden. Für das MDM und den EAS von Airwatch bietet FeedHenry eine direkte Integration an.

kony 1100 Orlando, US $ 51,1 Mill. US Dollar

KonyOne

Für dieses Beispiel betrachten wir das KonyOne Studio mit der Kony Development Cloud sowie der KonyOne Server. Sowohl die Kony Visualization Cloud als auch die Kony Management Cloud befinden sich noch im Beta-Stadium und sind nicht frei verfügbar.

1. Aufsetzen des Projekts

Das Projekt für die mobile Anwendung wird entweder im installierten KonyOne Studio oder aber in der Kony Development Cloud erstellt. Beide werden miteinander synchronisiert, so dass der Start beliebig gelegt werden kann. Da Kony für jede Zielplattform Apps erzeugen kann, muss das Projekt nicht für eine bestimmte Plattform erzeugt werden. Daher sind mit der Erstellung des Projekts die vorbereitenden Maßnahmen abgeschlossen.

2. Implementieren der Oberflächen

Die Implementierung der Oberflächen erfolgt vollständig in KonyOne Studio. Für jede einzelne Oberfläche muss hier eine „Form“ im gewünschten Formfaktor (Mobile, Tablet oder Desktop) erstellt werden. Die einzelnen Forms werden in einem graphischen Editor geöffnet, über den die Oberfläche erstellt werden kann. Dabei können die üblichen UI-Elemente wie Buttons und Texteingaben per Drag&Drop aus der Palette in die UI eingefügt werden. Das Layout ist dabei relativ, d. h. alle Elemente werden relativ zueinander definiert, ebenso die Größen der Elemente. Damit kann die Oberfläche auf Geräten mit unterschiedlichen Bildschirmgrößen angezeigt werden.
Alle UI-Elemente können über den Eigenschaften-Editor bearbeitet werden, um deren Aussehen und Verhalten zu beeinflussen. Damit kann etwa die Farbe geändert werden, die Umrandung oder der Text. Um für ein einheitliches Corporate Design und alle weiteren Apps immer das gleiche Design zu nutzen, können diese Eigenschaften auch in ein „Theme“ ausgelagert werden. Dieses Theme kann dann auf die App angewendet werden und überträgt alle Eigenschaften auf die jeweiligen Oberflächen-Elemente.
Es können neben allgemeinen Eigenschaften auch die jeweiligen spezifischen Eigenschaften des Betriebssystems angepasst werden, also etwa ein eigenes Aussehen unter iOS oder Android.
Da der graphische Editor nur schemenhaft die Oberfläche darstellt, kann die „echte“ Oberfläche über einen Preview erstellt werden. Über die „Live Preview“ Funktion ist dies auch in Echtzeit möglich, d.h. Änderungen an der Oberfläche im Editor werden nach dem Speichern direkt im Preview sichtbar.

3. Anbinden der Backends

Die Anbindung der Backends erfolgt ebenfalls im KonyOne Studio. In einer eigenen Ansicht werden alle verfügbaren Konnektoren angezeigt und für den gewünschten Konnektor kann dann eine neue Instanz erzeugt werden. Für die

Anbindung eines WSDL-basierten Web-Services wie etwa Exchange EWS wird der „WSDL Service“ genutzt. Alternativ könnte auch ein SAP-System oder ein RESTful Service angebunden werden. Sowohl bei WSDL als auch bei SAP können nach dem Anlegen des Konnektors die verfügbaren Operationen angezeigt werden. Die gewünschte Operation kann dann als „Service“ der mobilen Anwendung hinzugefügt werden, damit diese dort verfügbar ist.

Damit ein Service korrekt benutzt werden kann, muss dieser entsprechend konfiguriert werden. Dazu gehört die Definition der Ein- und Ausgabeschnittstelle. Hier muss per Editor definiert werden, welche Parameter als Eingabeparameter von der mobilen Anwendung gesetzt werden und welche Parameter vom Ergebnis des Services an die mobile Anwendung zurückgeliefert werden. Dabei wird der Entwickler von KonyOne Studio unterstützt, indem der Service testweise ausgeführt werden kann, um die Parameter zu ermitteln. Die Ausgabe kann dann über Ausdrücke in die gewünschte Struktur übertragen werden. Die fehlerfreie Ausführung wird im Anschluss direkt mit Testdaten gegen den echten Server getestet.

Die definierten Services werden dann auf den KonyOne Server installiert. Ab diesem Zeitpunkt können sie auch von der mobilen Anwendung aus ausgeführt werden. Ein Debugging der installierten Services ist nicht möglich.

4. Implementieren der Client-Logik

Um die definierten Services in der mobilen Anwendung nutzen zu können, müssen zunächst entsprechende Aktionen (Kony: Events) eingerichtet werden. Eine Aktion kann etwa durch den Klick auf einen Button ausgelöst werden. Diese Aktion wird dann als eine Folge von Schritten definiert, die in einer dedizierten Reihenfolge sind. Für die Schritte stehen verschiedene Operationen zur Auswahl, die wichtigsten sind das Aufrufen eines Services (synchron oder asynchron), das Zuordnen von Daten zu UI-Elementen, Bausteine für Verzweigungen sowie der Wechsel zwischen Oberflächen. Für die Anwendung wird zunächst eine Aktion zum Aufrufen des zuvor definierten Services benötigt. Innerhalb dieser Aktion können die Eingabeparameter von bestehenden UI-Elementen zugeordnet werden, etwa dem Wert eine Klappliste. Als nächste Aktion muss das Ergebnis des Services über eine Zuordnung auf die Elementen der UI abgebildet werden. Im letzten Schritt kann dann etwa noch ein Wechsel auf die nächste Oberfläche erfolgen, auf der die Daten angezeigt werden.

5. Test

Zum Testen der Anwendung muss diese zuvor gebaut werden. Bei Kony bedeutet dies, dass aus den Metadaten der Anwendung über verschiedene Generatoren die Zielplattform generiert wird, also Objective-C-Code für iOS, Java für Android, und HTML5 und JavaScript für Web-Apps. Die Auswahl der Zielplattformen kann bei jedem Generiervorgang neu angepasst werden. Abhängig von der Plattform kann die generierte Anwendung direkt in einem auf dem Testrechner installierten Simulator gestartet werden. Für iOS muss ein Umweg über XCode gegangen werden, in dem das generierte Projekt geöffnet wird und der dort enthaltene Simulator gestartet wird.

Ein Debuggen der Anwendung ist über einen Debug-Modus möglich, dieser wird über einen speziellen App-Container auf dem Simulator integriert.

6. Verteilen der Anwendung

Aktuell ist die Kony Management Cloud noch nicht verfügbar, daher kann die Verwaltung der Anwendungen nicht beschrieben werden.

7. Verwaltung der Anwendung

Aktuell ist die Kony Management Cloud noch nicht verfügbar, daher kann die Verwaltung der Anwendungen nicht beschrieben werden.

SAP 65.667 Walldorf, DE € 16,22 Mrd. Euro

Sybase Unwired Platform

In diesem Beispiel nutzen wir die Sybase Unwired Platform in der Version 2.3. Der Nachfolger dieser Version, die SAP Mobile Platform 3.0, lag zum Zeitpunkt der Evaluierung noch nicht vor und konnte daher hier nicht betrachtet werden.

1. Aufsetzen des Projekts

Für das Arbeiten mit der SAP Unwired Platform muss das SDK installiert werden, welches sowohl die IDE in Form eines modifizierten Eclipse als auch den Sybase Server sowie das Sybase Control Center zur Konfiguration und Administration des Servers auf einem Windows-Rechner einrichtet.
Zum Aufsetzen des Projekts wird in Eclipse ein entsprechendes Mobile Application Project erzeugt, in dem sowohl die Client- als auch die Serveranteile abgelegt werden. Durch einen Wizard werden die erforderliche Projektstruktur und Einstellungen selbstständig angelegt.

2. Implementieren der Oberflächen

Die SUP IDE bietet einen graphischen UI-Editor, mit dem die Oberflächen der mobilen Anwendung erstellt werden können. Per Drag&Drop können hier die Oberflächen mit den üblichen UI-Elementen gestaltet werden. Dabei stellen die Mobile Business Objects (MBO) einen zentralen Bestandteil dar, durch sie können weite Teile der Oberfläche automatisch generiert werden. Im Anschluss besteht die Möglichkeit, die generierten Anteile manuell an die jeweiligen Bedürfnisse anzupassen. Die MBO müssen für diesen Schritt bereits existieren, dieser Workflow wird im folgenden Abschnitt beschrieben.
Die erstellten Oberflächen können jederzeit über einen Wizard in eine Web-App überführt werden. Hierbei werden die erforderlichen HTML, Javascript und CSS-Dateien automatisch generiert. Diese erstellt Anwendung kann dann über den Browser getestet werden. Die automatisch generierten Dateien können bearbeitet werden, die Änderungen werden jedoch bei einer erneuten Generierung überschrieben.

3. Anbinden der Backends

Zur Anbindung eines SOAP-Web-Services bietet SUP einen Wizard an, über den die erforderlichen Parameter des Services wie URL und WSDL angegeben werden können. Aus den Web-Services erzeugt SUP dann ein Mobile Business Object, welches die Datentypen und Operationen des Services repräsentiert. Aus diesem MBO kann dann im letzten Schritt über die SUP die JavaScript-Bibliotheken generiert werden, welche dann in der mobilen Anwendung genutzt werden können, um diese Services auszurufen und die Daten zu verarbeiten.

4. Implementieren der Client-Logik

Die Implementierung der Client-Logik erfolgt anhand der definierten MBOs. Über diese werden sowohl die Daten vom Server abgerufen als auch direkt an die jeweiligen UI-Elemente gemappt. Weitere Client-Logik kann individuell mit HTML und Javascript implementiert werden.

5. Test

SUP bietet die Möglichkeit über die Mobile Workflow Forms Editors HTML-View die Benutzeroberflächen über einen Simulator zu testen. Die Anwendung kann aber auch über einen normalen Browser und dessen integrierte Debugging-Funktionen getestet werden.

6. Verteilen der Anwendung

Die Sybase Unwired Plattform bietet keinen eigenen Enterprise App Store zur Verteilung der Anwendung. Über das MDM „Afaria“ von SAP können Anwendungen jedoch gezielt an registrierte Geräte des Device Managements verteilt werden. Dazu muss die Afaria-Installation im SUP konfiguriert werden. Dazu muss die SAP Afaria App auf dem jeweiligen Gerät installiert sein und das Gerät provisioniert sein. Der Benutzer kann dann über die Afaria App die für ihn bereitgestellte Applikationen installieren.

7. Verwaltung der Anwendung

SUP bietet ein Benutzer- und Rechtemanagement an, über das für erstellte Anwendungen und MBOs die jeweiligen Berechtigungen vergeben werden können. Benutzer können etwa aus einen LDAP angebunden werden, und die Berechtigung für die Benutzer einer Anwendung oder eines MBO vergeben werden.

434.246

Armonk, US

104,5 Mrd. US Dollar

Worklight

Für dieses Beispiel betrachten wir die frei verfügbare Developer Edition Version 6.0 von IBM Worklight, die über den Eclipse Marketplace bezogen werden kann. Die Developer Edition beinhaltet das Worklight Studio, den Worklight Server, Worklight Runtime Components sowie eine Worklight Console.

1. Aufsetzen des Projekts

Das Worklight Studio basiert auf Eclipse als integrierte Entwicklungsumgebung. D. h., dass alle von Eclipse unterstützten Betriebssysteme genutzt werden können (Windows, Linux, Mac OSX). Die Kompilierung von iOS-Applikationen ist allerdings nur auf Mac OSX-Systemen möglich, dort ist zudem eine direkte Integration von Worklight- und Xcode-Projekten möglich. Um mit der Entwicklung mobiler Apps zu beginnen, muss zunächst Eclipse installiert werden. Danach kann die Developer Edition von IBM Worklight über den Eclipse Marketplace als Plugin hinzugefügt werden. Um beispielsweise Apps für Android entwickeln zu können, müssen zusätzlich noch das Android SDK sowie die Android Development Tools im Eclipse integriert werden.

Im Worklight Studio wird zu Beginn der Entwicklung ein neues Worklight Projekt angelegt. Dort muss festgelegt werden, ob eine hybride oder native Applikation erstellt werden soll. Die benötigten Zielplattformen (Android, iOS, etc.) können später über das Hinzufügen von sogenannten „Environments“ definiert werden. Nach der Erstellung des Projekts werden alle erforderlichen Bestandteile für die Worklight-Anwendung angelegt. Das beinhaltet sowohl den Client- als auch den Server-Anteil.

2. Implementieren der Oberflächen

Die Implementierung der Oberflächen erfolgt vollständig im Worklight Studio. Hierfür wird ein Paletten-Editor angeboten, über den vordefinierte UI-Elemente direkt per Drag&Drop in die App eingefügt werden können. Nach dem Anlegen eines Elements können die Attribute und Funktionen über entsprechende Parameter-Einstellungen geändert werden. Eine schemenhafte Echtzeit-Vorschau stellt das Layout der App vorab dar. Die endgültige Oberfläche kann über den mitgelieferten Mobile Browser Simulator betrachtet werden. Das Aussehen der Elemente kann mittels CSS-Klassen definiert werden.

3. Anbinden von Backend-Systemen

Die Anbindung der Backend-Systeme erfolgt ebenfalls im Worklight Studio. Mithilfe der sogenannten Adapter können zahlreiche Backendsysteme angesprochen werden. Diese Adapter werden genutzt, um Daten von den Backends abzurufen und bestimmte Aktionen bzw. Transformationen der gelesenen Daten durchzuführen. Verfügbare Adapter sind SQL-, HTTP- (REST oder SOAP), Cast Iron sowie JMS-Adapter. In diesem Beispiel werden die Daten per SOAP-Schnittstelle von einem Exchange-Server abgerufen. Der Adapter wird per XML konfiguriert. Um die Metadaten des Adapters im XML einzutragen, kann entweder der Source- oder Design-Editor genutzt

werden. Dabei wird das Protokoll, die Zieldomain und der Port eingetragen. Nachdem die Schnittstelle im Adapter definiert wurde, können mittels sogenannter Prozeduren verschiedene Transaktionen mit den gelesenen Daten durchgeführt werden, bevor sie in der Anwendung weiterverarbeitet werden. Die Prozeduren werden über JavaScript im Adapter implementiert. Diese Prozeduren ermöglichen z. B. eine Sortierung der Daten oder das Mapping der Datenfelder auf einen anderen Datentypen. Nach der Implementierung des Adapters wird dieser auf dem Server installiert. Ab diesem Zeitpunkt kann dieser auch von der mobilen Anwendung aus aufgerufen werden. Ein Testen der installierten Adapter ist möglich, indem die Prozeduren des Adapters im Editor mit Eingabeparametern aufgerufen werden können. Durch das zurückgelieferte Ergebnis kann überprüft werden, ob der Adapter funktioniert und die gewünschten Daten liefert. Ein echtes Debuggen der Adapter ist jedoch nicht möglich.

4. Implementieren der Client-Logik

Die Client-Logik wird weitestgehend manuell implementiert. Sofern ein Mobiles Framework wie JQuery Mobile oder Sencha Touch eingesetzt wird, kann dies bei der Layout-Erstellung helfen. Das Mapping der Daten zwischen Adapter und Client muss allerdings manuell festgelegt werden. Server-Aktionen können direkt in der Client-Logik aufgerufen werden. Das Error-Handling erfolgt auf Plattform-Basis (Android, iOS) und ermöglicht auch die Erstellung von Logfiles für genaue Fehlerkontrollen. IBM stellt mit den Worklight Runtime Components gängige APIs für die native und hybride App-Entwicklung zur Verfügung. Mithilfe dieser Libraries können native Device-Funktionalitäten wie bspw. der GPS-Zugriff angesprochen werden. Des Weiteren ist es möglich auf den Local Storage des Clients zuzugreifen und somit eine Offline-Fähigkeit für die Anwendung zu implementieren.

5. Test

Es gibt verschiedene Möglichkeiten, die Applikation zu testen. Während der Entwicklung hybrider Apps und Web-Apps wird eine Vorschau der Anwendung angezeigt, die bereits einen ersten Eindruck der App geben soll. Nachdem die App das erste Mal gebaut wurde, kann sie über eine in IBM Worklight integrierte Preview in Form des Mobile Browser Simulators gestartet werden. Der Mobile Browser Simulator ermöglicht die Anzeige für verschiedene Displaygrößen und Endgeräte, sowohl im Hoch- als auch im Querformat. Der Simulator greift dabei wahlweiße auf das Common Environment oder aber direkt auf das Android bzw. iOS Environment zurück. Er kann sowohl direkt in der Eclipse-Umgebung innerhalb des Worklight-Projekts gestartet werden oder über die Worklight Console, in der die auf dem Server veröffentlichen Anwendungen angezeigt werden. Die Worklight Console stellt eine Art Administrations-Oberfläche dar. Daneben kann die App auch direkt mittels der entsprechenden SDKs getestet werden, bei Android direkt über Eclipse, bei iOS wird die Anwendung zunächst als Xcode-Projekt geöffnet, wo sie dann mit dem dortigen Emulator getestet werden kann.

Ein Debuggen der Anwendung ist mit gängigen Methoden möglich (Eclipse, Desktop Browser, iOS Remote Web Inspector, Weinre, Wireshark). Zusätzlich bietet IBM einen Worklight Debugger, der als Objekt für die Ausgabe von Log-Nachrichten und Fehlermeldungen im Worklight-Projekt genutzt werden kann.

Die implementierten Adapter (REST, Webservices, etc.) können direkt im Worklight Studio getestet werden. Dazu wird die Methode (Prozedur) mit den gewünschten Parametern aufgerufen. Im Anschluss können die abgerufenen Daten untersucht werden.

6. Verteilen der Anwendung

Die Anwendung kann über das in IBM Worklight integrierte Application Center verteilt werden. In der IBM Worklight Developer Edition wird dieser allerdings nicht zur Verfügung gestellt. Prinzipiell können Anwendungen direkt in das Application Center aufgenommen werden, die Informationen wie Versionsnummer, App-Icon und Package-Name werden dabei direkt aus dem Worklight-Projekt ausgelesen. Im Anschluss kann über die Application Center Console die Nutzergruppe definiert werden, die Zugriff auf die App erhalten soll. Das können entweder einzelne Nutzer, Nutzergruppen oder über LDAP integrierte Nutzer mit deren bestehenden Rechteklassen sein. Um die App auf dem mobilen Endgerät installieren zu können muss der Anwender zunächst den Application Center Mobile Client installieren. Dieser mobile Enterprise Appstore zeigt lediglich die Apps an, die für den Nutzer freigeschaltet sind. Zudem wird angezeigt, sofern ein Update für bereits installierte Anwendungen zu Verfügung steht.

7. Verwaltung der Anwendung

Die Verwaltung der Anwendung erfolgt ebenfalls über das IBM Worklight Application Center. Über die Application Console können zudem Nutzungsstatistiken eingesehen und Updates der Anwendungen eingespielt werden. Die Verwaltung der mobilen Endgeräte ist ausgelagert und nicht Teil von IBM Worklight. Hierfür wird der IBM Endpoint Manager verwendet, welcher mit IBM Worklight zusammenarbeitet. So kann im Endpoint Manager direkt eine bei IBM Worklight veröffentlichte App für die entsprechend berechtigten Endgeräte freigegeben werden.

05 Trends

Wenn man die Entwicklung der Mobility Plattformen der letzten Monate und Jahre betrachtet, so ist zum einen ganz klar zu erkennen, dass das Thema „Mobility“ im allgemeinen und „Enterprise Mobility“ im speziellen im Fokus der meisten großen IT-Firmen steht, wie etwa die Untersuchung von CA zeigt: 84 % aller befragten Unternehmen erwarten durch Enterprise Mobility erhöhte Investitionsausgaben, noch vor Themen wie Virtualisierung und Big Data. Die „Big-Player“ wie SAP und IBM haben daher durch strategische Zukäufe ihr Produktportfolio in diesem Bereich ausgebaut und einige der Pioniere in diesem Segment haben mittlerweile einen soliden Kundestamm und ausreichend Kapitaldeckung. Jedoch befindet sich dieses Marktsegment immer noch in den Anfängen, die Marktdurchdringung von Mobility Plattformen ist weiterhin sehr gering, das flächendeckende Bewusstsein für die Notwendigkeit und den Nutzen solch einer Plattform bleibt noch aus.
Dennoch entwickelt sich der Markt der Anbieter laufend weiter. Dabei sind die folgenden Trends und Ausrichtungen zu erkennen:

Konsolidierung

Eine Konsolidierung im Markt ist schon seit längerem zu beobachten. Kleinere Unternehmen mit innovativen Mobility Produkten werden von großen Unternehmen aufgekauft, um in diesem zukunftsträchtigen Markt mit geeigneten Produkten Erfolge erzielen zu können. SAP hat Sybase und dessen Sybase Unwired Platform übernommen, IBM hat die Firma Worklight und Fiberlink übernommen, und zuletzt wurde das Unternehmen Antenna und dessen Mobility Plattform durch Pegasystems übernommen. Diese Konsolidierung wird sicher noch weitere Anbieter treffen, es wird sich zeigen, wie viele verschiedene Anbieter den wachsenden Markt am Ende noch bedienen werden.

Cloud

Der Trend zur Cloud ist auch im Bereich Mobility Plattform stark zu spüren. Während es schon zu Beginn einzelne Anbieter gab, die ihre gesamte Plattform als Cloud-Lösung angeboten haben, steigen nun immer mehr der großen Unternehmen wie zuletzt Kony und SAP auf die Cloud um bzw. ergänzen ihre Angebote um entsprechende Cloud-Lösungen. Für Kunden bietet dies oftmals einen stark vereinfachten Einstieg in eine Mobility Plattform, da die komplette Installation und Konfiguration eines Servers entfällt und durch flexible Lizenzmodelle auch der initiale Kostenaufwand deutlich reduziert werden kann. Damit wird es Unternehmen möglich, mit vergleichsweise geringem Aufwand und Kosten eine erste mobile Lösung zu entwickeln.

Vereinfachung

Die Entwicklung einer mobilen Anwendung mitsamt der Anbindung an Backend-Systeme ist ein komplexes Unterfangen. Um diese Komplexität vor dem Entwickler zu verbergen, machen Mobility Plattformen immer größere Anstrengungen und vereinfachen so die Entwicklung mobiler Lösungen. Während die UI-Entwicklung bereits seit längerem durch graphische Editoren und cross-plattform-Ansätze erleichtert wurde, wird nun zunehmend die Komplexität der Anbindung an Backend-Systeme und das gesamte Management der mobilen Lösungen stärker vereinfacht. Damit sollen auch technisch unerfahrene Entwickler mobile Lösungen entwickeln, warten und verwalten können.

Der Markt der Mobility Plattformen wird sich vermutlich in zwei Segmente aufteilen: die großen Anbieter wie IBM, SAP und Kony werden immer umfangreichere und vollständigere Lösungen anbieten, die einen möglichst breiten Nutzerkreis ansprechen sollen – dies macht sich dann auch im Preis bemerkbar. Dann wird es weiterhin einige kleinere Anbieter geben, die sich auf bestimmte Bereiche und Anforderungen spezialisieren. Diese bieten dann oftmals keine vollständige Lösung an, können dafür aber bestimmte Aufgaben besser adressieren als die großen Anbieter.

Über M-Way Consulting

M-Way Consulting unterstützt Unternehmen durch professionelle Beratung bei strategischen Entscheidungen sowie der Konzeption von individuellen Mobility Lösungen.

Wir bieten eine individuelle Beratung rund um Enterprise Mobility an und unterstützen Ihr Unternehmen bei der Auswahl der richtigen mobilen Strategie. Dabei greifen wir auf ein bewährtes Vorgehensmodell zurück. Anhand des Enterprise Mobility Prozesses konzipieren wir die einzelnen Phasen Ihres Projekts. Der iterative Ansatz erlaubt dabei jederzeit individuelle Anpassungen. Zusätzlich bieten wir Unternehmen über den Enterprise Mobility Check ein standardisiertes Analyseverfahren, mit welchem alle benötigten Voraussetzungen aufgenommen werden, um eine Mobile Strategie entwickeln zu können. Anhand unserer strategischen Beratung können Sie ihre Produkte und Prozesse auf den mobilen Bereich erweitern und infolgedessen zusätzliche Märkte und Zielgruppen erreichen. Durch unsere unabhängigen und objektiven Analysen der unterschiedlichen Mobility Plattformen finden wir für die ideale Lösung für Ihr Unternehmen und Ihre Infrastruktur. Um Ihre Mitarbeiter und Geschäftsprozesse für das mobile Unternehmen vorzubereiten, teilen wir unser Wissen mit Ihrem Unternehmen. Diese Neuausrichtung wird den Erfolg Ihres Unternehmens nachhaltig steigern und sichert Ihnen den entscheidenden Wettbewerbsvorteil im Mobile Business.

Wenn Sie weitere Fragen haben, kontaktieren Sie uns gerne.

Mail: info@mwayconsulting.com
Web: www.mwayconsulting.com

Veröffentlicht Februar 2014.

Copyright

Autor
Mirko Bleyh, Consultant

Editor
Christian Feser, Managing Director

Editor
Tobias Vetter, Consultant

Layout and Design
Candogan Ögüt, Designer

Quellen

Verivo:	www.verivo.com
Convertigo:	www.convertigo.com
FeedHenry:	www.feedhenry.com
IBM Worklight:	www.ibm.com/software/products/de/worklight/
Kony:	www.kony.com
Sybase Unwired Platform:	www.sybase.de/products/mobileenterprise/sybaseunwiredplatform

m-way consulting

Stresemannstr. 79 | 70191 Stuttgart | Germany
Phone: +49 711 252 548 00 | Fax: +49 711 252 548 09
www.mwayconsulting.com