Entwicklung Mobiler Apps – ein Überblick

Von Thomas Kestler, elevato GmbH – 06.07.2012

Einleitung

Die Entwicklung mobiler Apps wird für Projektleiter, Architekten und Entwickler zunehmend wichtiger. Klar, das Thema existiert ja nicht erst seit gestern, aber mehr und mehr Unternehmen planen die Entwicklung neuer oder Integration bestehender Apps in ihre Infrastruktur. Dabei geht es um Nutzung bestehender und neuer Services durch Mitarbeiter und Kunden/Partner, sowie die Integration von mobilen Endgeräten der o. g. Personengruppen (BYOD = Bring Your Own Device).

Im Folgenden möchte ich einen kurzen Überblick über Technologien und Architekturen geben. Dabei geht es mir insbesondere um den Blickwinkel des Unternehmenseinsatzes mobiler Anwendungen (also nicht Consumer Markt).

Die wichtigsten Plattformen

Die wichtigsten Plattformen sind iOS (iPhone, iPad), Android (Smartphones, Tablets), Windows Phone und Blackberry. Symbian und Windows Mobile sind auf dem absteigenden Ast. Bada von Samsung ist auch am verschwinden.

Apple iOS

Apple hat den Smartphone-Boom ausgelöst mit einem Gerät, das von Anfang an sehr benutzerfreundlich konzipiert war. Das Betriebssystem iOS hatte Anfangs noch starke Defizite (kein Multitasking), aber seit iOS V4 ist das Betriebssystem als ausgereift zu bezeichnen (aktuell ist derzeit V5.1).

Die Entwicklung von Apps erfolgt mit Objective- C. Diese Sprache ähnelt stark C und muss ggf. erst neu erlernt werden. Als Entwicklungsumgebung wird meist Xcode von Apple eingesetzt.

Apple überzeugte von Anfang an durch ansehnliche, benutzerfreundliche Geräte, aber verschreckt bis heute und seine Abschottung und Produktpolitik.

Android

Android wurde von Google entwickelt und hat den Markt rasant erobert, nicht zuletzt wegen der offenen Politik von Google. Android basiert auf dem Linux-Kernel und darauf aufsetzen einem Framework, das größtenteils in Java geschrieben ist. Als Java-VM wird Dalvik verwendet. Abgesehen von dem anderen Format der Klassendateien und Archive merkt der Entwickler davon aber nicht viel, er entwickelt wie gewohnt in Java.

Der Android-SDK in Verbindung mit Eclipse bietet eine hervorragende, kostenlose Entwicklungsumgebung samt Emulator.

Problematisch sind die schnellen Updatezyklen und die Behäbigkeit vieler Smartphone-Hersteller bei der Bereitstellung von Updates auf neuere Android-Versionen. Der Wechsel von 2.x auf 3.x/4.x brachte massive Änderungen in APIs und Konzepten mit sich. Auch sonst gibt sich Google recht ungeniert bei API-Änderungen, was die Pflege- und Portierungsaufwände erhöht.

Windows Phone 7
Windows Phone 7 löst das bisherige Windows Mobile ab, das sich am Markt nicht durchsetzen konnte. Die Konzepte von Windows Phone sind deutlich moderner und von der Konkurrenz inspiriert. Die Entwicklung von Apps erfolgt in C# oder VB.NET, am häufigsten mit dem komfortablen VisualStudio. Gerade für .NET-Entwickler gerät deshalb hier der Einstieg sehr einfach.

Blackberry
Blackberry war ein Pionier für mobiles Messaging und hat daher eine breite Kundenbasis. Technisch gesehen wurde Blackberry aber von den Smartphones rechts überholt. Das Betriebssystem Blackberry OS liegt aktuell in Versin 7.1 vor und hat eine lange Historie hinter sich. Das kommende Betriebssystem Blackberry OS 10 (BB10) soll zu iOS und Android aufschließen, aber derzeit ist noch gar kein Release-Datum bekannt und es gibt auch keine Geräte. Weiterhin bietet Blackberry das Playbook Tablet mit einem eigenen Betriebssystem (Tablet OS) an, welches auf QNX basiert.

Blackberry bietet mehrere Entwicklungsumgebungen an, jedoch nicht jede für alle Betriebssysteme[1]. So gibt es Entwicklungsumgebungen für C/C++, Java und AIR. HTML5 wird dort auch als Entwicklungsumgebungen (Platform) aufgeführt. Außerdem gibt es eine Android-Runtime, mit der Java-basierte Android-Apps auf Blackberry (nur Tablet OS und BB10) ausgeführt werden können. Die Java-Plattform enhält JavaME und viele JSRs.

Neues Bedienparadigma
Für mobile Geräte haben sich andere Bedienparadigmen entwickelt als für Desktop-Applikationen. Popup-Dialog sind eher selten, meist erfolgt die Navigation über Stacked-Panes, die Ansichten werden also al Stapel aufeinander gelegt. Mit der BackTaste wird der Stapel wieder abgebaut.

Die Bedienung erfolgt per Tippen oder Gesten (Wischen, Ziehen) oder Lageveränderung (Drehen des Gerätes).
Anwendungen werden auch kaum explizit beendet, sondern laufen im Hintergrund weiter.

Neues Kommunikationsparadigma
Anders als bei Desktops hat das mobile Gerät nicht ständig Netzempfang. Apps müssen daher darauf vorbereitet sein, dass das Netz immer wieder mal weg sein kann. Push-Technologien (z. B. Apple Push oder Google C2DM) ermöglichen asynchrone Benachrichtigung von mobilen Endgeräten.

Technologien
Man kann derzeit grob vier Technologie-Ansätze für mobile Plattformen unterscheiden:

- Native Apps
- WebApps (HTML5)
- Hybrid (HTML5/CSS/JS) Adapter (z. B. PhoneGap)
- Runtime-Lösungen (Adobe AIR)

1 https://developer.blackberry.com/devzone/platforms

Native Apps

Native Apps werden speziell für eine Plattform entwickelt. Sollen mehrere Plattformen unterstützt werden, so muss die Entwicklung mehrfach erfolgen. Aktuell kann man von den drei o. g. Wichtigen Plattformen ausgehen, d.h. Man entwickelt und pflegt die Anwendung also dreimal.

Der Vorteil nativer Apps ist der direkte Zugriff auf die APIs des Betriebssystems auf Sensoren und Datenbanken. Insbesondere Features, die noch nicht Eingang in Standards wie HTML5 gefunden haben, lassen sich nur so ansprechen, z.B. NFC, USB oder Bluetooth. Die Performance ist bei lokalen Apps optimal. Der Nachteil liegt darin, dass für die verschiedenen Betriebssysteme jeweils eigene Entwicklungswerkzeuge, Programmiersprachen und API erlernt werden müssen.

WebApps (HTML5)

Obwohl HTML5 noch nicht verabschiedet ist, bieten die aktuellen Drafts bereits einen Leistungsumfang, mit dem sich viele mobile Anwendungsfälle umsetzen lassen. Einfache Location-based Services wie Umkreissuche sind so direkt mit HTML5 und Javascript umsetzbar und damit auf den meisten Plattformen nutzbar. Das wählen einer Telefonnummer geht über das Protokoll `tel:` und Mailversand über `mailto:`.

Mittlerweilen haben sich die APIs stabilisiert, es sind aber einige Funktionen bereits wieder entfallen oder durch andere ersetzt worden.

Mittlerweile bieten Frameworks wie *Oracle ADF Mobile* eine dedizierte Unterstützung für mobile Endgeräte (in Form von CSS und JS-Bibliotheken). Für Unternehmen, die ohnehin ADF für die interne Entwicklung nutzen, durchaus ein relevanter Aspekt. Darüber hinaus plant Oracle auch die Erstellung nativer ADF-Apps[2].

Hybrid

PhoneGap ist der bekannteste Repräsentant einer Hybrid-Technologie. Bei PhoneGap erfolgt die Entwicklung der eigentlichen Anwendung in HTML5/CSS/JS und über eine plattformspezifsches Browser-Plugin werden native Services als Javascript-API eingeblendet. Das Versprechen lautet auch hier, die Anwendung mit einer Code-Basis in HTML5+CSS+JS zu entwickeln und auf unterschiedliche Geräte zu verteilen. Die Auslieferung erfolgt als native App, die nur ein Wrapper dafür ist, die eigentliche App in Form von HTML5 auszuführen.

Die Bedienkonzepte (Navigation) wurden teilweise für mobile Nutzung erweitert, teilweise konnten bestehende APIs auf native APIs gemappt werden. Bei Flex gibt es z. B. subtile Unterschiede zwischen Flex (Desktop) und Flex (Mobile).

Runtime

Bei diesem Technologie-Ansatz erfolgt die Ausführung der App in einem eigenen Container (Runtime) wie z. B. AIR Runtime von Adobe. Als Beipsiel kann Adobe FlashBuilder dienen, mit dem man eine (mobile) Flex-Anwendung (ActionScript als Programmiersprache ist Javascript sehr ähnlich[3]) erstellt, die dann auf iOS, Android

2 Oracle ADF Mobile Next: Natives Frontend und der JSF-Renderer kommuniziert eben damit. Vgl. CaptainCasa.

3 ActionScript 3 ist ein Dialekt von ECMA-262 und Javascript sehr ähnlich. ActionScript 3 ermöglicht Typisierung und klassenbasierte Objektorientierung.

und Blackbarry Tablet OS in der AIR Runtime ablaufen kann. Für die Nutzung auf Andorid muss der Benutzer einmalig die AIR Runtime als eigene App installieren. Bei iOS wird die Flex-App mit der AIR Runtime zu einer App verpackt. Blackberry unterstützt AIR Runtime im Lieferumfang des Tablet OS (und später BB10).

App-Kategorien

Welche Technologie zum Einsatz kommen soll, hängt wesentlich von den Anforderungen an die App ab. So würde ich ein Spiel mit hohen Anforderungen an die Grafik sicherlich nativ entwickeln. Ich schlage willkürlich folgende App-Kategorien vor:

Informations-App

Hier steht das Abrufen von Informationen im Vordergrund. Nutzerkreis: öffentlich. Lokale Datenspeicherung kaum nötig, Online-Lösung.
Beispiel: Reiseauskunft der Bahn.

Unternehmens-App

Ähnlich wie bei einer Intranet-Anwendung auf dem Desktop erfolgen hier Zugriffe auf Intranet-Services (Kundendatenbank, etc). Personenkreis: Mitarbeiter, Partner. Lokale Datenspeicherung kann nötig sein. Beispiel: Tarifrechner für Versicherungsmakler, Zugriff auf BI-Daten[4],

Spezial-App

Hier soll das Endgerät ausgereizt werden, z.B. als BDE-Terminal[5]. Hier werden ggf. Schnittstellen wie USB[6], Bluetooth, NFC zum Einsatz kommen. Lokale Datenspeicherung nötig. Beispiel: App für Hausarztbesuch.

Spiele

Spiele stellen eine eigene Kategorie dar, sollen hier aber nicht näher betrachtet werden.

Entwicklungswerkzeuge

Die Wahl der Plattform und Entwicklungsumgebung hat Auswirkung auf die zur Verfügung stehenden Entwicklungswerkzeuge. Die zu verwendenden Entwicklungs-werkzeuge müssen in die bestehende Infrastruktur des Unternehmens passen.

Folgende Aspekte sind relevant:

IDE

Legt man sich auf eine bestimmte IDE fest oder kann die Entwicklung in mehreren IDEs erfolgen? Beispiel: viele Java-Entwickler sind Eclipse gewohnt und kommen daher mit dem Android-SDK schnell zurecht, Eclipse läuft unter Windows, LINUX und Mac OSX sehr gut. Können alle Entwicklungsschritte innerhalb der IDE erfolgen?

4 BI=Business Intelligence
5 BDE=Betriebsdatenerfassung, z. B. Ablesegerät
6 USB-Hostmode nicht für alle Geräte möglich

Debugger
Welche Möglichkeiten zum Debugging bestehen? Ist der Debugger ausgereift genug? Ist er in die IDE integriert?

UI-Designer / Theming
Auch wenn die Oberflächen mobiler Anwendungen tendenziell weniger Elemente haben als Desktop-Screens, so ist ein guter UI-Designer sehr wichtig.

Emulator
Der Entwickler muss für eine Vielzahl von Gerätevarianten entwickeln, da ist es hilfreich, wenn er diese im Emulator simulieren kann. Insbesondere dann, wenn die Hardware noch nicht verfügbar ist. Früher oder später stellt sich auch die Frage der Testautomatiseirung.

Deployment
Die erstellte App muss zunächst zu Testzwecken auf das Gerät des Entwicklers gebracht werden. Danach muss Sie für einen größeren Personenkreis (Tester) verfügbar gemacht werden, aber noch nicht öffentlich. Danach muss die App in die öffentlichen oder firmeneigenen App-Stores deployt werden. Wenn mehrere Abteilungen eines Unternehmens Apps entwickeln, muss sichergestellt sein, dass alle Apps mit dem gleichen Zertifikat veröffentlicht wurden (Glaubwürdigkeit, Öffentlichkeitswirkung). Updates für bestehende Apps müssen in den Stores eingepflegt werden können (der Update-Mechanismus erfolgt meist durch die Market-App der Plattform).

Der Zugang zu den Stores über Developer-Accounts ist gerade für Unternehmen eine Herausforderung, da die Accounts zentral verwaltet werden sollen (Release-Management). Die Auslieferung Download von einer Website sollte nicht in Betracht gezogen werden (Thema Glaubwürdigkeit).

Datenschutz und Datensicherheit
Datenschutz befasst sich mit dem Schutz persönlicher Daten und unterliegt gesetzlichen Auflagen. Für mobile Anwendungen bestehen aber kaum andere Voraussetzungen als für Unternehmensanwendungen, außer der unbefugte Zugriff auf dem Smartphone oder Tablet hier durch USB-Debugging[7] oder einfache Entsperrpasswörter oder -muster tendenziell leichter erfolgen könnte.

Die Datensicherheit wird ebenfalls durch möglichen Verlust/Diebstahl des Gerätes, Abhören der Kommunikation über Mobilfunknetz oder WLAN, sowie Hacking/USB-Debugging tangiert[8].

Beide Bereiche erfordern im Vergleich zu Desktop-Anwendungen erhöhte Aufmerksamkeit, da Hacker und Kriminelle die mobilen Geräte als lohnendes Angriffsziel identifiziert haben und Betriebssysteme und Konzepte noch relativ jung sind. In letzter Zeit wurde zahlreiche Hacks bekannt, die Zugriffe einer App auf Daten einer anderen App ermöglichten (obwohl das Sandbox-Konzept genau das

7 Selbst bei unternehmenseigenen Geräten nicht auszuschließen. Man sollte also potenziell von dieser Gefahr ausgehen.
8 Siehe BSI: https://www.bsi.bund.de/ContentBSI/Themen/Mobilsecurity/mobilsecurity.html leider schon etwas veraltet.

verhindern soll). Außerdem gehen viele Anwender noch immer recht sorglos mit der Zustimmung der angeforderten Berechtigungen um. Alleine die Tatsache, dass das Rooting der meisten Smartphones über Exploits erfolgt, sollte alarmieren. Die meisten Smartphones kann man binnen einer Minute rooten.

Komfort versus Datensparsamkeit

Je weniger Daten lokal im mobilen Endgerät gespeichert sind, desto besser. Allerdings leidet dann der Komfort, da das ständige Nachladen über das Funknetz je nach Netzqualität den Arbeitsfluss und die „Experience“ erheblich stören kann. App-Entwicklern steht meist eine lokale Datenbank zum Ablegen von Anwendungsdaten zur Verfügung. Per Konzept (Sandbox) soll diese nur für diese App zugänglich sein. In der Praxis sind die Hürden hier allerdings nicht so hoch wie erhofft. Die SQLite-Datenbanken von Android enthalten Daten im Klartext und können leicht ausgelesen werden, wenn man als Angreifer die Sandbox einmal durchdrungen hat. Somit sollten sensible Daten in solchen Datenbanken von der App verschlüsselt abgelegt werden.

Kommunikation

Die Kommunikation sensibler Daten sollte prinzipiell immer über SSL oder noch besser VPN erfolgen. Die Mobilfunknetze sind nicht übermäßig sicher und ein Hotspot-WLAN schon gar nicht. Der Knackpunkt hier ist, wer die präsentierten SSL-Zertifikate akzeptiert: Die Browser haben ziemlich viele Root-CAs im Bauch, denen Sie blind vertrauen. Falls die Anwendung die SSL-Verbindung selbst aufbaut und nur der eigenen Root-CA vertraut, wird das Risiko deutlich verringert (solange die eigene Root-CA nicht kompromittiert wird, was in der Vergangenheit erschreckend häufig passierte).

Sofern Intranet-Services für mobile Anwendungen bereitgestellt werden sollen, muss der Zugriff entsprechend sicher autorisiert sein. Ein eigenes VPN für mobile Nutzung solcher Services scheint sinnvoll. Blackberry, iOS und Android unterstützen VPNs recht gut. Sofern https zum Einsatz kommen soll, sollte das Service-Gateway in der DMZ gegen DDoS-Attacken geschützt werden.

Zusammenfassung

Wir stehen beim Thema Mobility noch am Anfang und es wird sich weiter rasant entwickeln. Unternehmen sollten Strategien für die Integration und Bereitstellung mobiler Services erarbeiten.

Eine Standardisierung ist am ehesten im Bereich HTML5 zu beobachten, aber auch hier laufen die Standards der Technik hinterher[9].

Native Apps bieten vollen Leistungsumfang und für Entwickler sicher auch den meisten Spaß. Allerdings ist es für Unternehmen kam darstellbar, jede App drei oder viermal zu entwickeln.

Flex ist ein komplettes Ökosystem zur Entwicklung mobiler Apps, allerdings stellt sich die Frage nach der zukünftigen Produktpolitik von Adobe.

9 Mal ganz abgesehen von dem Standardisierungsprozess von HTML5, der in der Kritik steht.

PhoneGap / Cordova sind ein interessanter Ansatz und viele Apps basieren bereits heute darauf. Wie tragfähig dieser Ansatz in Zukunft sein wird und wie das „Produkt“ nach Übergabe durch Adobe an die Apache Foundation weiter entwickelt wird bleibt abzuwarten.

Die weitere Entwicklung von Blackberry ist unklar. Im Business-Bereich hat Blackberry sicher eine große Kundenbasis, im Consumer-Bereich spielen sie keine Rolle. Da aber BYOD immer wichtiger zu werden scheint, sollten Unternehmen auch für interne Apps plattformübergreifende Technologieansätze einsetzen.

Unternehmen sollten identifizieren, welche App-Kategorien für sie relevant werden können und dafür jeweils Referenzarchitekturen entwickeln.

Thomas Kestler ist Geschäftsführer der elevato GmbH. http://www.elevato.de