Anbindung mobiler Anwendungen an komplexe IT-Systeme

Patrick Runow[1),3)], Michael Clasen[2)], Hendrik Lock[3)]

[1)] Fakultät Technik und Wirtschaft
Hochschule Karlsruhe
Moltkestr. 30
76133 Karlsruhe
patrickrunow@yahoo.de

[2)] Hochschule Hannover
michael.clasen@fh-hannover.de

[3)] SAP AG, Walldorf
hendrik.lock@sap.com

Abstract: Die Entwicklung von einfachen mobilen Unternehmensanwendungen (Apps) stellt neue Anforderungen an die Datenbereitstellung und Integration von geschäftsprozessunterstützenden Applikationen wie ERP-Systemen. Dieser Beitrag gibt Einblick in eine Realisierung durch REST-basierte Services unter Verwendung des OData Protokolls.

1 Einleitung

Leichtgewichtige mobile Anwendungen für Smartphones (sogenannte Apps) werden immer populärer. Im Jahr 2010 wurden 10,9 Milliarden Apps für Smartphones heruntergeladen und in den letzten 3 Jahren mehr als 300.000 entwickelt [IDC10]. Neben dem Marktsegment der Freizeit-orientierten Anwendungen, wird auch das Segment geschäftsunterstützender mobiler Anwendungen immer wichtiger. Auch SAP hat sich im mobilen Markt mit einer neuen Infrastruktur und neuen mobilen Geschäftsanwendungen positioniert.

Zur Abbildung von Geschäftsprozessen auf mobilen Endgeräten müssen diese häufig auf Daten aus ERP-Systemen zugreifen. Diese Daten liegen hier in der Regel in einer normalisierten Form in häufig sehr komplexen relationalen Datenbanken vor. Dem Entwickler einer App kann jedoch nicht zugemutet werden, sich in die Datenbankstrukturen eines ERP-Systems einzuarbeiten und die benötigten Daten aus diversen Datenquellen wie z.B. Tabellen, Objekten und Reports herauszusuchen. Er soll sich stattdessen auf die leichte Bedienbarkeit und ein übersichtliches Design der App konzentrieren. Höhere Qualitäts- und Sicherheitsanforderungen erhöhen die Komplexität betrieblicher Apps weiter.

Im Folgenden wird eine anwendungszentrierte Abstraktion von den bestehenden Datenstrukturen eines SAP-Systems vorgestellt. Hierzu wird für jede App ein sog. Consumption (Daten) Modell entwickelt, welches als Kontrakt zwischen App- und Backend-Entwickler dient. Der App-Entwickler definiert durch diesen Kontrakt die Daten, die er für die App benötigt und der Backend-Entwickler stellt diese dann über eine Schnittstelle zur Verfügung. Die technische Umsetzung basiert auf dem REST-Architekturstil, sowie offenen Protokollen und Formaten wie OData.

2 Consumption Model

Das von SAP spezifizierte Consumption Model dient als konzeptionelle Schnittstelle zwischen der mobilen Anwendung und dem Serversystem. Für Client und Server ist das Consumption Model ein Kontrakt. Dieser Kontrakt definiert, welche Daten und Funktionalitäten angeboten werden, welche Beziehungen zwischen Daten und Funktionalitäten existieren und über welche Eigenschaften die Daten, Funktionen und Beziehungen verfügen. Ziel eines Consumption Models ist es, eine anwendungsspezifische Abstraktion von den nach technischen Gesichtspunkten gebildeten Strukturen und Funktionalitäten zu bilden. Ein Consumption Model ist somit vergleichbar mit einem Entity-Relationship-Modell bei der Datenmodellierung.

In SAP-Systemen liegen die Daten als Objekte in einer normalisierten Form vor. Normalisierung stellt zwar die Konsistenz der Daten sicher, führt aber auch zu einer Fragmentierung der Informationen. Ein Geschäftsprozess nutzt Daten aus vielen unterschiedlichen Objekten. Bei der Abbildung eines Geschäftsprozesses auf eine mobile Anwendung werden die Daten projiziert, zusammengeführt und denormalisiert. Das Resultat wird durch ein konkretes Consumption Model ausgedrückt.Während ein Consumption Model eine anwendungszentrierte Sicht auf einen Service darstellt, wird der Service technisch durch das OData-Protokoll in XML abgebildet und arbeitet nach den Grundsätzen einer REST-konformen Architektur.

3 REST-Architekturstil

Representational State Transfer (Abk.: REST) bezeichnet einen Softwarearchitekturstil für verteilte, Hypermedia Informationssysteme und wurde durch Roy Fielding geprägt [Fil00]. Die größte Implementierung einer REST-konformen Architektur ist das World Wide Web selbst.

REST-Architekturen bestehen aus Komponenten, Konnektoren und Datenelementen. Sowohl die Komponente Server, als auch die Client-Komponente verfügt über einen Konnektor, der die Kommunikation durch Übermittlung von Datenelementen ermöglicht, ohne die eigentlichen Daten zu verändern. REST führt hierbei den Begriff der Ressource ein. Eine Ressource ist ein von außen und global adressierbares Datenelement und damit die konzeptuelle Verbindung zu einer Menge von Entitäten. Ressourcen werden allein vom Server verwaltet. Client und Server tauschen gegenseitig Repräsentatio-

nen von Ressourcen aus. Eine Repräsentation enthält den aktuellen Zustand einer Ressource und liegt typischerweise in Form eines Dokumentes (HTML, XML, JSON u.v.m.) vor. Da eine Ressource vom momentanen Zustand der Entität abstrahiert, führen zwei zeitlich versetzte Zugriffe oftmals zu unterschiedlichen Resultaten [Fil00].

Im Vergleich zu anderen Architekturstilen, sind REST-Architekturen besonders durch den Grundsatz der ausschließlich zustandslosen Kommunikation geprägt. Das heißt insbesondere, dass jede Anfrage ein in sich transaktional abgeschlossener, atomarer Vorgang auf dem Server ist. Und es bedeutet, dass die Verwaltung des Anwendungszustandes nicht auf dem Server, sondern nur auf dem Client erfolgen soll [Fi00]. Die Zustandslosigkeit stellt neue Anforderungen an bisher gängige Umsetzungsvarianten. Da in einer REST-Architektur alle Anfragen an einen Server unabhängig voneinander behandelt werden, wird Datenkonsistenz im Allgemeinen dadurch erreicht, dass sämtliche Informationen einer Transaktion in einer einzigen Serveranfrage gesendet werden.

Durch die zustandslose Kommunikation lassen sich auf einfache Weise hochskalierbare und hochverfügbare Serversysteme entwerfen. Bei dem Parallelbetrieb von mehreren Serversystemen kann jede Clientanfrage von allen Servern beantwortet werden. Der Mehraufwand, den ein clientseitiger Entwickler durch die manuelle Implementierung der Sitzungsverwaltung hat, wird schnell durch die Flexibilität und Transparenz wettgemacht. Ein Entwickler muss auf diese Weise nicht mehr sicherstellen, dass serverseitig gespeicherte Sitzungsdaten bei der nächsten Verwendung einer Anwendung noch verfügbar sind.

4. Protokolle und Formate

Das OData-Protokoll wird verwendet, um die anwendungsspezifische Abstraktion in Form eines Consumption Modells technisch umzusetzen. OData ermöglicht es, Daten und Geschäftslogik REST-konform über das Internet zu exponieren. Dabei basiert das OData-Protokoll auf einer Reihe von etablierten Formaten und Protokollen, die in der Abbildung dargestellt werden. OData selbst ist ein von Microsoft eingesetzter offener Standard; eine Einführung findet sich unter [ODP11].

Durch das Atom Format können einzelne Informationseinheiten zusammen mit ihren Metadaten durch Entries repräsentiert werden. Eine semantisch zusammengehörige Menge von Entries wird als Feed bezeichnet [No05]. Die für den Transport notwendige Abbildung des Formats auf grundlegende HTTP Methoden wird durch das Atom Publishing Protokoll erreicht [Gr07].

Die beiden Atom-Spezifikationen werden bei elektronischen Nachrichtensystemen genutzt, erlauben jedoch nur die Abbildung einfachster semantischer Beziehungen und Funktionalitäten. Die Integration von weiteren Datenbankfunktionalitäten wird durch das von Microsoft spezifizierte Open Data Protokoll (OData) [Mi11] erzielt.

OData basiert auf den Atom Spezifikationen und definiert im Kern die Abbildung von Datenbankenschemata sowie die Abfrage und Verwaltung der Daten. Auf diesen, in

Schichten aufeinander aufbauenden Formaten und Protokollen, basiert die Kommunikation zwischen der mobilen Anwendung und dem Serversystem. Daten, die in einer Datenbank gespeichert sind, können somit per URI adressiert und im XML-Format ausgelesen oder geändert werden. OData und seine URI Definition erlauben es sehr allgemeine Datenbankabfragen zu formulieren, weshalb OData stelleweise auch als *ODBC des Webs* bezeichnet wird.

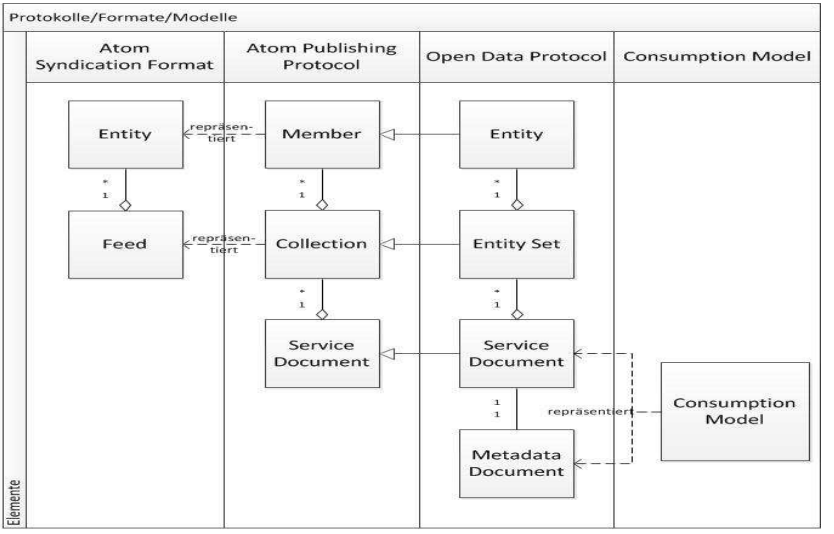

Abbildung 1: Zusammenhänge zwischen den Protokollen, Formaten und Datenelementen

Durch die Verwendung von in Schichten aufeinander aufbauenden offenen Formaten und Protokollen können sowohl existierende Implementierungen (z.B. Parser) wiederverwendet werden, als auch vorhandenes Wissen genutzt werden.

Literaturverzeichnis

[Fi00] Fielding, R: Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine, Dissertation, 2000.

[Gr07] Gregorio, J.; Google; de hOra, B.; NewBay Software: The Atom Publishing Protocol Network Working Group, 2007; S. 8-12.

[IDC10] International Data Corporation: IDC Forecasts Worldwide Mobile Applications Revenues to Experience More Than 60% Compound Annual Growth Through 2014 . [Online] http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22617910. [Cited: 06.11.2011]

[Mi11] Microsoft Corporation: [MS-ODATA]: Open Data Protocol (OData) Specification, 2011, Rev 10.1; S. 8

[No05] Nottingham, M.; Sayre, R.: The Atom Syndication Format. Network Working Group, 2005

[ODP11] Microsoft Corporation: Open Data Protocol, www.odata.org, 2011.