Oracle APEX und Microservices, passen sie zusammen?

Oleg Kiriltsev
MT AG
Ratingen

Schlüsselworte
Oracle APEX, Microservices, Datenbank

Einleitung

Ich möchte kurz auf die klassische Enterprise Anwendung eingehen. In der Regel werden solche Anwendungen in einem monolithischen Stil gebaut und haben drei grundlegende Bestandteile:

- GUI (HTML-Seiten zusammen mit CSS- und JavaScript-Dateien)
- Application Server
- Datenbank

Der Application Server ist für die Verarbeitung der Fachlogik, die Kommunikation mit der Datenbank und die Generierung der HTML-Seiten verantwortlich. Alle Serverkomponenten haben relativ starke Abhängigkeiten zwischen einander, und die jede Änderung (neue Anforderung oder Bugfix) führt dazu, dass das gesamte System neu gebaut und in Produktion deployed werden müsste.

Mit der Zeit und der Anzahl der Funktionalitäten ist es schwieriger die gute und verständliche modulare Struktur mit der minimalen Abhängigkeit zwischen den Anwendungskomponenten zu halten. Die Änderungen in einem Modul ziehen automatisch die Anpassungen in anderem(n) Modul(en) mit sich. D.h., dass niemand in Wirklichkeit genau abschätzen kann, welche Auswirkungen die Änderungen auf das gesamte System haben könnten. Alle betroffenen und zusammenhängenden Komponenten müssen technisch und fachlich neu getestet werden, bevor die neue Version der Anwendung in Produktion gebracht werden kann. All das führt dazu, dass die Release Zyklen größer werden. Oftmals vergehen einige Wochen/Monate, bevor die neue Fachanforderung im Produktivsystem erscheinen.
Eine der möglichen Lösungen für die oben beschriebenen Probleme / Unbequemlichkeiten ist der Microservices Architekturstil. Bevor ich aber zur Beschreibung der Microservices Architektur komme, möchte ich gerne die Oracle APEX Architektur kurz erläutern.

Was ist Oracle APEX?

Oracle APplication EXpress (Oracle APEX) ist ein Framework für die Entwicklung von datenzentrierten Web-Anwendungen. Die Anwendungsdefinition wird in der Oracle Datenbank, in den sogenannten Metadaten-Tabellen abgespeichert. Das Framework generiert automatisch die Web-Seiten mit Hilfe der DB-Engine (PL/SQL) und auf Basis der abgespeicherten Metadaten. Die Entwicklung kann komplett im Browser erfolgen und benötigt keine zusätzliche Client-Software.

In der Abbildung 1 wird das Beispiel für die Oracle APEX Architektur dargestellt. Aus meiner Sicht können ganz eindeutig die Bestandteile wie auch beim monolithischen Stil erkannt werden:

- Application Server
- DB

Der Application Server hat hier nicht die gleiche Funktion, wie bei den anderen Enterprise Anwendungen (z.B. Java EE Anwendung), sondern die DB spielt hier beide Rollen – Application Server und DB. Wie schon oben erwähnt wurde, generiert die Framework Engine die Web-Seiten auf Basis der gespeicherten Metadaten und läuft komplett in der DB, wo auch die Daten abgespeichert werden. Der Application Server ist im Beispiel (Abbildung 1) trotzdem vorhanden. Dort läuft der sogenannte ORDS (Oracle Rest Data Services), das ist eine Art Proxy, der unter anderem ein paar

optionale Funktionen anbietet, wie z.B. Bereitstellung von Web-Services. Weiteren Informationen über das Thema ORDS können unter http://www.oracle.com/ gefunden werden.

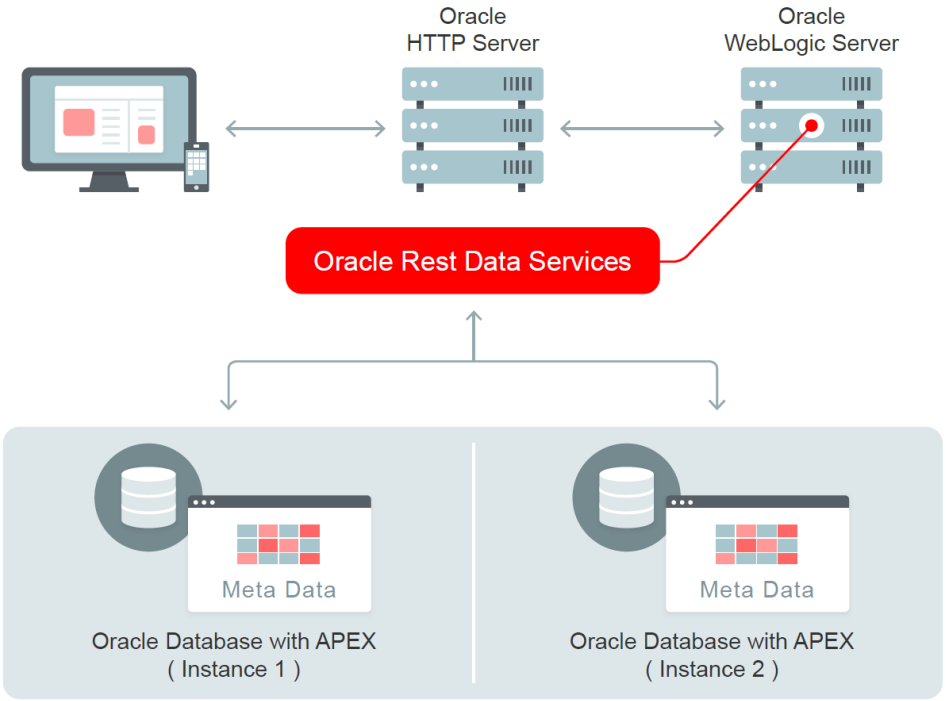

Abbildung 1. Beispiel für Oracle APEX Architektur (Quelle: www.oracle.com)

Die mit Oracle APEX entwickelte Anwendung besteht aus den gleichen Bestandteilen wie auch die anderen Enterprise Anwendungen, obwohl die Komponenten unterschiedliche Funktionen haben.

Was ist ein Microservice-Architektur Stil?

Das ist eine Architektur, in der ein Microservice ein wichtiger Bestandteil ist. Dabei besteht ein System aus vielen Microservices, die untereinander kommunizieren. Jeder Microservice führt eine Funktion aus, und hat keine Abhängigkeit von anderen Microservices.

Nachstehend sind ein paar Definitionen / Eigenschaften, was unter einem Microservice zu verstehen ist:
Jon Eaves von RealEstate.com.au aus Australien beschreibt ein Microservice als etwas was in 2 Wochen umgeschrieben/neu entwickelt werden könnte.
Noch eine Definition von einem Microservice ist: small enough and no smaller (klein genug aber nicht kleiner).
Eine wichtige Eigenschaft / Charakteristik von einem Microservice ist die Unabhängigkeit voneinander. D.h. bei einem Microservice ist es möglich, ihn zu ändern und eine neue Version in Betrieb zu nehmen, ohne die weiteren Änderungen bei den Consumers oder im System zu machen.

Diese Arbeit beschreibt nicht alle Eigenschaften und Vorteile von dieser Architektur, aber auf die zwei Punkte möchte ich hier ausführlicher eingehen:

- Heterogenität der Technologien
- Einfachheit von Deployment (der einzelnen Services)

Die Abbildung 2 zeigt, was man unter Heterogenität der Technologien vorstellen kann. Dadurch, dass die Microservices unabhängig voneinander sind, kann sich das Entwicklungsteam entscheiden, welche Technologie für die Lösung am besten passt und in diesem Fall eingesetzt wird. Das betrifft nicht nur die Programmiersprache, sondern auch wie und wo z.B. die Daten abgespeichert werden. Die „spezifischen" Daten, wie Benutzernachrichten oder Bilder, können in der dokumentorientierten oder dateiorientierten Datenbank abgespeichert werden.

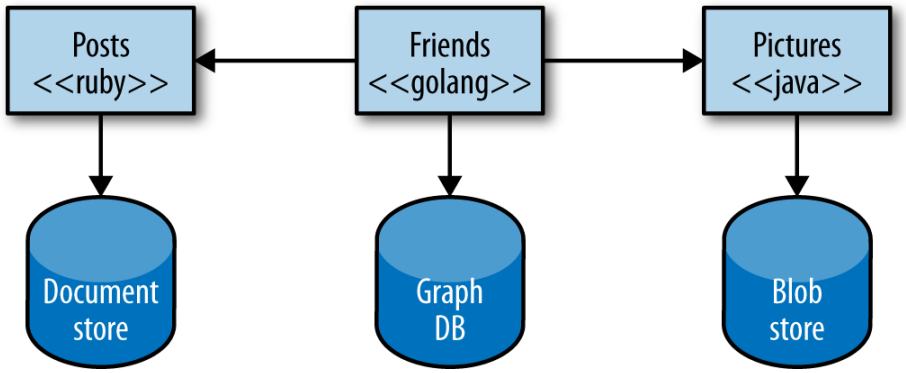

Abbildung 2. Unterschiedliche Technologien in der Microservices Architektur (Quelle: Building Microservices, Sam Newman, 2015)

Wichtiger Aspekt der Microservices Architektur ist die Unabhängigkeit. Die Implementierung der neuen Funktionalität kann unabhängig von den anderen Komponenten erfolgen. Sie müssen nicht mehr warten bis alle Systemteile zueinander passen, damit der Bugfix oder die neue Funktion in Produktion deployed werden kann. Bei einer monolithischen Architektur muss immer die gesamte Anwendung neu deployed werden, was einen großen Aufwand und höheres Risiko mit sich bringt. Wenn ein Fehler trotzdem auftritt, kann das Problem besser isoliert und behoben oder die „alte" Service-Version ausgerollt werden.

Wie lässt sich dieser Stil auf die Architektur der Oracle APEX-Anwendungen übertragen? Ein paar Voraussetzungen bringen die Oracle DB und APEX mit, und zwar die DB-Schemas und Web-Services. Die Aufteilung der Daten in Schemas kann nicht nur organisatorisch, sondern auch

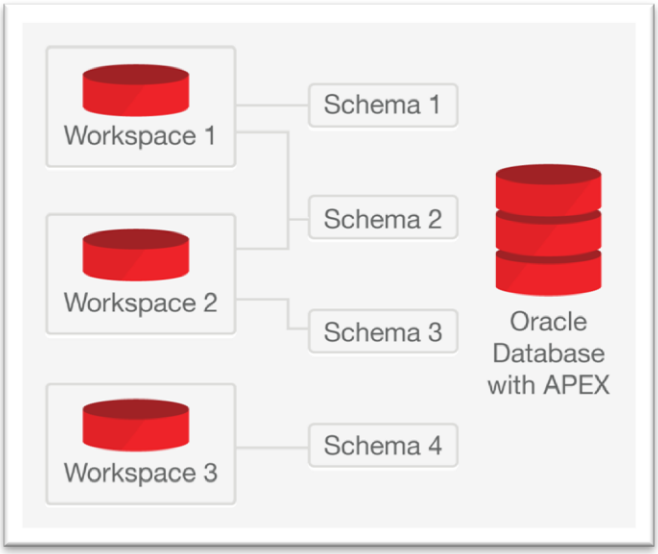

Abbildung 3. Oracle DB und unterschiedliche DB-Schemas (Quelle: www.oracle.com)

strukturell erfolgen: Machen Sie ein DB-Schema für Kundendaten (Name, Vorname, Adresse etc.), ein zweites Schema für die Auftrags- oder Rechnungsdaten oder für irgendwelche anderen zusammenhängenden Daten. Legen Sie alles nicht in einem Schema an, damit mit der Zeit eine „große Matschkugel“ (Big Ball of Mud, siehe https://de.wikipedia.org) entsteht, wo keiner weiß, was da los ist. Die Kommunikation zwischen den Schemas muss so erfolgen, dass ein Schema nur die öffentliche API (Application Programm Interface) vom anderen Schema kennt und nicht mehr. Die Funktionsaufrufe müssen auch keine Kompilierungsabhängigkeiten haben, z.B. Aufrufe über Web-Services. So kann das Schema mit Kundendaten geändert oder komplett durch andere Technologie ersetzt werden, ohne die anderen Systemteile was davon mitbekommen. Das Verwenden von Web-Services bietet Oracle DB und Oracle APEX „out the box“ mit Hilfe von mehreren PL/SQL-Packages. Die ORDS (Oracle Rest Data Services) lässt sich in wenigen Schritten die Web-Services von Oracle DB heraus bereitstellen.
Oracle DB und Oracle APEX bieten die Möglichkeiten in diesem Stil zu entwickeln. Sie müssen sich nur entscheiden, ob Sie es brauchen und nutzen wollen, und ob diese Möglichkeiten für das System Vorteile bringen.

Zusammenfassung
Es gibt sehr viele Punkte, die ein Softwareentwickler / Softwarearchitekt im Kauf nehmen sollte, wenn man sich für den Microservices Architekturstil entscheidet. Besonders in den ersten Projekten werden viele Probleme und Schwierigkeiten auftreten, um die Prozesse sauber zu gestalten und am Laufen zu halten. Alle Entscheidungen in diesem Bereich werden mehr intuitiv als bewusst getroffen, weil zum aktuellen Zeitpunkt keine richtige Standards und Regel existieren.
Aber bitte glauben Sie nicht, dass die Microservices eine Lösung für alle Probleme sind. Viel wichtiger für den Projekterfolg sind die schon längst bekannten Faktoren. Das sind z.B. das Team, die Kommunikation im Projekt und innerhalb des Teams und die klar definierten Fachanforderungen. Im technischen Bereich spielen die Softwarearchitektur, gut strukturierte und getestete Code weiterhin eine der wichtigsten Rollen.
Die Microservices erfordern das Denken in kleinen Komponenten – ein Service ist ein fachlicher Prozess. Die Systemteile müssen klar voneinander getrennt werden und auch möglichst keine Abhängigkeiten aufweisen. Das erlaubt einen hochqualitativen und wartbaren Code zu entwickeln. Wenn die Systemkomponenten tatsächlich voneinander unabhängig sind, dann können die unterschiedlichen Tools und Frameworks bei der Entwicklung ausgewählt werden, die am besten für die Lösung der Aufgaben passen. Versuchen Sie die Vorteile aus den Ideen und Konzepten mitzunehmen, unabhängig davon, ob Ihre Softwarearchitektur Microservices Architektur genannt werden kann oder nicht.

Kontaktadresse:
Oleg Kiriltsev
MT AG
Balcke-Dürr-Allee 9
D-40882 Ratingen

Telefon:	+49 (0) 2102 309610
Fax:	+49 (0) 2102 30961-101
E-Mail	oleg.kiriltsev@mt-ag.com
Twitter	@OKiriltsev
Internet:	www.mt-ag.com