Kapitel 3
Architektur und Integration

In diesem Kapitel stellen wir Ihnen die architektonischen Grundkonzepte von SAP Gateway und die Integration mit anderen SAP-Schnittstellen aus einer Rundumsicht vor. Das Kapitel schafft mit dem Aufbau des notwendigen theoretischen Hintergrundwissens die Grundlagen für spätere Kapitel.

SAP Gateway hat ein klar umrissenes Ziel: SAP-Systeme sollen für Nicht-ABAP-Entwickler geöffnet werden. Ursprünglich war hierbei der Zielumfang auf Anwendungsfälle der sogenannten leichtgewichtigen Konsumierung (*Lightweight Consumption*) beschränkt. Um diese Anwendungsfälle zunächst einmal zu ermöglichen und dann auch zu vereinfachen, stellt SAP Gateway sowohl eine Plattform für die Bereitstellung (*Provisioning*) als auch für die Konsumierung (*Consumption*) von Services zur Verfügung. Zusätzlich werden verschiedene Werkzeuge für die Erstellung von Inhalten geliefert. Mittlerweile haben die Definition der Gateway-Prinzipien und ihre Umsetzung in weiten Teilen des SAP-Produktportfolios die Nutzungsmöglichkeiten von SAP Gateway stark erweitert. Zusätzliche Nutzungsszenarien, wie beispielsweise analytische Konsumierung (*Analytical Consumption*) oder auch Szenarien mit hoher Last, sind somit möglich geworden. Inzwischen hat SAP Gateway innerhalb des technischen SAP-Ökosystems einen festen Platz und findet weithin Verwendung. Vor diesem Hintergrund und mit der Herausbildung des ABAP RESTful Application Programming Models ist SAP Gateway wieder einmal weiterentwickelt worden, um es in Cloud-Szenarien und mit innovativen Technologien wie SAP S/4HANA und SAP Fiori einsetzen zu können.

In Abschnitt 3.1 erläutern wir die architektonischen Konzepte hinter den Gateway-Prinzipien. Einen Überblick über die Architektur von SAP Gateway geben wir Ihnen in Abschnitt 3.2. In Abschnitt 3.3 erhalten Sie detaillierte Informationen zu den verschiedenen Datenquellen, die mit Ihrem SAP-System integriert werden können. Abschließend werden wir in Abschnitt 3.4 das ABAP-Programmiermodell für SAP Fiori vorstellen, das für den Zugriff auf das Backend stark auf SAP Gateway baut, und anschließend in Abschnitt 3.5 einen Blick auf die nächste Programmiermodell-Evolution, das ABAP RESTful Application Programming Model, werfen.

3.1 Gateway-Prinzipien

Zeitlose Software

In der Anfangszeit der Enterprise-Software waren Systeme monolithisch und gegenüber der Außenwelt abgeschottet. Mit der Evolution dieser monolithischen Systeme hin zu Landschaften aus verschiedensten Systemen für unterschiedlichste Zwecke entstand eine Herausforderung: Wie sollten diese Systeme zu Ende-zu-Ende-Prozessen verbunden werden? Wie zu Ende-zu-Ende-Szenarien ohne größere Brüche? Heutzutage sind aus mehreren Systemen zusammengesetzte Landschaften nicht nur die tagtägliche Realität, sondern auch wünschenswert – beispielsweise um verschiedene Aspekte des Geschäftsbetriebs zu separieren. Die Einführung von Mittelschichten (*Middle Tier*) in Unternehmenslandschaften, um eine verbesserte Konnektivität zu erreichen, führte leider auch zu höheren Kosten, höherer Komplexität und Insellösungen sowie zu einer Reihe von siloartigen Ansätzen für das Bereitstellen der Daten für die Mittelschichten bzw. die Verbindung der verschiedenen Systeme untereinander.

Prinzipien zeitloser Software

Um die Komplexität von Landschaften aus verschiedenen Systemen zu reduzieren, hat SAP eine Reihe von Prinzipien erarbeitet, die vorgeben, wie Enterprise-Software-Systeme konstruiert werden sollten: die Prinzipien für zeitlose Software (*Timeless Software Principles*). Zeitlose Software ist ein systematischer, mit Schichten arbeitender Ansatz, der vorgibt, wie Aufzeichnungssysteme effizient zur Verfügung gestellt und wie sie mit dem Rest der Gesamtsystemlandschaft, den Entwicklungswerkzeugen, der Laufzeitumgebung und dem Lebenszyklus der Software verknüpft werden sollten.

Auf diesem Ansatz basierend, wurden die Aspekte, die für ein System lokal sind und das Verhalten eines gegebenen Systems oder einer Plattform adaptieren (*Platform Adaptation*), und die Anforderungen eines gegebenen Konsumenten und der notwendigen Infrastruktur voneinander abgegrenzt, um einen speziellen Konsumenten zu bedienen (*Consumer Adaptation*). Diese Trennung ist sinnvoll, weil in einer idealen Landschaft die Plattformadaptierung an ein bestimmtes Aufzeichnungssystem (*System of Record*) gekoppelt und nur einmal für jedes System notwendig ist. Die Konsumentenadaptierung für einen speziellen Kanal (beispielsweise mobil, Portal, Social Media, Benutzeroberfläche/UI, Analytics) wird nur einmal pro Landschaft benötigt und nicht für jedes einzelne Individualsystem.

Gateway-Prinzipien

Die Gateway-Prinzipien bestehen aus einer Reihe von architektonischen Konzepten, Methoden, Entwurfsmustern und Standards, die beschreiben, wie ein Enterprise-System seine Daten und Funktionen für die Konsumierung zur Verfügung stellen sollte. Sie zielen darauf ab, Zugang zu Aufzeichnungssystemen über verschiedene Kanäle und für diverse Konsumenten

zur Verfügung zu stellen. Die Gateway-Prinzipien dienen hierbei als Handlungsempfehlung für die Plattformadaptierung in Aufzeichnungssystemen und sollen Entwicklern die Erstellung von Applikationen auf diesen Plattformen ermöglichen, die den Prinzipien der zeitlosen Software folgen. Zudem sind sie auf verschiedene Gebiete anwendbar (beispielsweise leichtgewichtige, analytische oder auch soziale Konsumierung). Elementare Gateway-Prinzipien sind:

- **Offenheit**
  Services können von jedem Gerät und von jeder Plattform aus konsumiert werden und jede User Experience generieren.
- **Zeitlosigkeit**
  Services funktionieren mit nahezu jeder SAP-Backend-Version (SAP Business Suite oder SAP S/4HANA).
- **Einfachheit der Konsumierung**
  APIs (*Application Programming Interfaces*) sind sehr einfach zu konsumieren, und für die Konsumierung ist kein internes SAP-Wissen nötig.
- **Benutzerfokussierung**
  Der Haupttreiber für die Architektur sind Szenarien für die Benutzerinteraktion.
- **Arbeitsteilung**
  Nicht-SAP-Entwickler können Services ohne jedes ABAP-Wissen konsumieren; diese Entwickler können sich nahezu völlig unabhängig von der SAP-Gateway-Serviceentwicklung um die Client-Entwicklung kümmern.

SAP Gateway ist in seiner derzeitigen Form das Ergebnis der Anwendung der in den Gateway-Prinzipien festgelegten architektonischen Konzepte, der Entwurfsmuster (*Design Patterns*) und der Standards für die verschiedenen Anwendungstypen.

3.2 SAP-Gateway-Architektur

Aus einer Gesamtsicht betrachtet, besteht die Architektur von SAP Gateway aus drei Schichten: der SAP-Backend-Schicht, der SAP-Gateway-Server-Schicht und der Konsumentenschicht (siehe Abbildung 3.1). Jede dieser Schichten dient einem klar umrissenen Zweck und bündelt für die Erfüllung dieses Zwecks notwendige Komponenten.

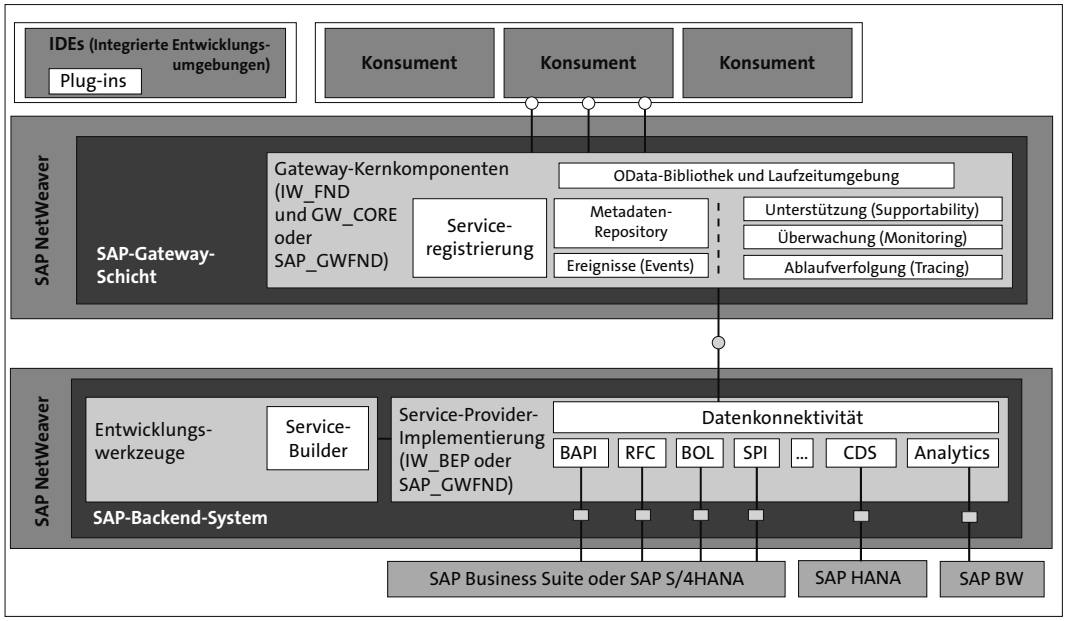

Abbildung 3.1 SAP Gateway – Architektur

Separation of Concerns

Mit diesem dreischichtigen Ansatz werden die Lebenszyklen von bestehenden SAP-Business-Suite- oder SAP-S/4HANA-Implementierungen und den betreffenden Client-Applikationen entkoppelt und die relevanten Belange getrennt (*Separation of Concerns*). Daraus resultierend werden die Abhängigkeiten zwischen den Systemen auf ein Minimum reduziert, sodass Upgrades, Updates und Deployments keine systemübergreifenden Auswirkungen haben (*Cross Impact*). Es ergibt sich eine hochflexible Landschaftsarchitektur, die vielfältige Applikationen und Szenarien ermöglicht.

Konsumentenschicht

In der Konsumentenschicht erfolgt die eigentliche Konsumierung der SAP-Gateway-Services, das heißt, hierauf greifen die Services zu. Konsumenten sind UI-zentrierte Clients, die OData-protokollkonforme Quellen konsumieren, die von SAP Gateway exponiert werden. Diese Exponierung kann entweder direkt oder indirekt erfolgen. Indirekt heißt hier: über zusätzliche, zwischengeschaltete Infrastrukturkomponenten (beispielsweise die SAP Mobile Services). Typische Konsumenten der SAP-Gateway-Services sind z. B. mobile Geräte oder SAP Fiori.

SAP-Gateway-Schicht

Es überrascht nicht, dass die SAP-Gateway-Schicht den größten Teil der SAP-Gateway-Funktionalitäten und -Komponenten beherbergt, inklusive der Kernkomponenten. Diese Schicht dient dabei folgenden Zwecken:

- Sie ist das Bindeglied zwischen dem Backend und den konsumentenbasierten Laufzeitkomponenten, den Metadatenkomponenten und der OData-Bibliothek (OData Library).
- Sie stellt Werkzeuge für die Entwicklung und Erstellung von SAP-Gateway-Services zur Verfügung.
- Sie bietet alles, was zum Betrieb einer SAP-Gateway-Landschaft benötigt wird, beispielsweise Logging, Tracing, Performanceanalyse-Werkzeuge und Globalisierungsunterstützung.

SAP-Backend-Schicht

Die SAP-Backend-Schicht ist die Schicht, in der die Daten der Geschäftsapplikationen angesiedelt sind. Zusätzlich liegt hier die Geschäftslogik von SAP S/4HANA oder der SAP Business Suite, auf die ein Großteil der Applikationen zugreift. In Bezug auf die zu SAP Gateway gehörigen Komponenten besteht die SAP-Backend-Schicht von SAP NetWeaver ABAP 7.40 SP02 an aus der Softwarekomponente `SAP_GWFND`, die bereits standardmäßig in SAP NetWeaver enthalten ist. In älteren Versionen enthielt das Add-on für Business Enablement und Event Provisioning `IW_BEP` die SAP-Gateway-relevante Funktionalität und musste zusätzlich deployt werden.

Abhängig vom Landschafts-Set-up und dem Nutzungsszenario kann es verschiedene Ausprägungen des Deployments geben. Wie bereits in Kapitel 1, »Einführung in SAP Gateway«, erwähnt, gibt es drei Hauptarten des Deployments: das Deployment auf einem separaten SAP-Gateway-System (Hub-Deployment), ein eingebettetes Deployment auf dem SAP-S/4HANA-System (Embedded Deployment) und das Cloud-Deployment mit SAP OData Provisioning auf der SAP Business Technology Platform (SAP BTP)

SAP OData Provisioning zukünftig in der SAP Integration Suite

Zum Entstehungszeitpunkt dieses Buches (April 2024) ist geplant, OData Provisioning in die SAP Integration Suite zu integrieren.

Zusätzlich zu diesen drei Hauptoptionen ist ein »gemischtes« Deployment möglich. Die spezifische Ausprägung der Landschaft und das Nutzungsszenario entscheiden über die Wahl der Deployment-Variante. (Dies werden wir in Kapitel 4, »Deployment-Optionen, Installation und Konfiguration«, im Detail erläutern.)

Detaillierte Architektur

Wie bereits beschrieben, liegt der Hauptteil der Funktionalität und der Installation von SAP Gateway in der Mittelschicht (*Middle Tier*). Darüber hinaus ist es wichtig zu wissen, welche SAP-Gateway-Komponenten in anderen Schichten zu finden sind. Daher sehen wir uns die drei Schichten und ihre

Hauptkomponenten im Detail an (siehe Abbildung 3.2). Zudem erklären wir die Add-on-Struktur von SAP Gateway.

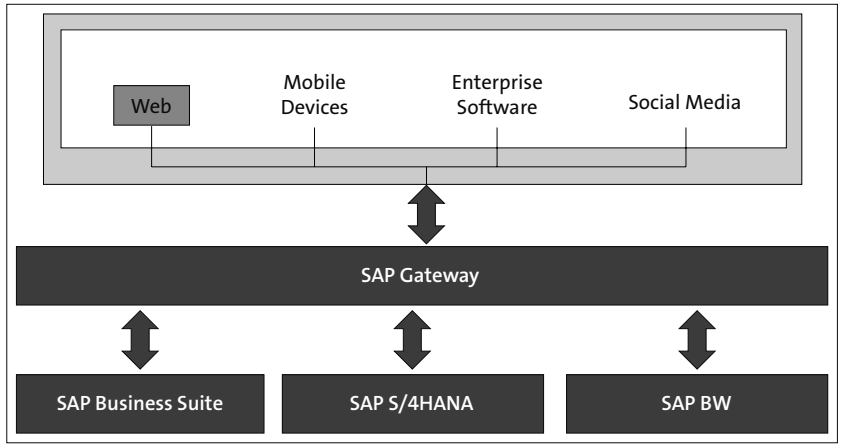

Abbildung 3.2 Hauptbestandteile von SAP Gateway

3.2.1 Konsumentenschicht

In der Konsumentenschicht können zwei Arten von Komponenten gefunden werden: SAP-Gateway-Konsumenten und integrierte Entwicklungsumgebungen (*Integrated Development Environment*, IDE). In diesem Abschnitt besprechen wir beide Arten von Komponenten.

SAP-Gateway-Konsumenten

SAP-Gateway-Konsumenten sind Komponenten, die SAP-Gateway-Services konsumieren. SAP Gateway ist primär dafür entwickelt worden, UI-zentrierten Clients (beispielsweise mobilen, nativen oder Webapplikationen wie SAPUI5/HTML5) den Zugriff auf Geschäftsdaten zu ermöglichen. Clients können auf SAP Gateway entweder direkt oder indirekt über zusätzliche Infrastrukturkomponenten zugreifen. SAP Mobile Services ist z. B. eine dieser zusätzlichen Infrastrukturkomponenten für mobile Szenarien.

REST-API-Ressourcen

SAP-Gateway-Konsumenten greifen auf REST-API-Ressourcen zu, die von SAP Gateway per HTTP(S) mithilfe des OData-Protokolls exponiert werden. In anderen Worten: Es gibt gar nicht so viel über SAP-Gateway-Konsumenten zu sagen – solange diese HTTP(S) konsumieren können, können sie auch SAP-Gateway-Services konsumieren. Kann zudem auch XML, JSON oder Atom geparst werden, können die empfangenen Daten auch prozessiert werden. SAP gibt tatsächlich keine Empfehlungen oder Einschränkungen bezüglich der Client-Technologie oder gar bezüglich der zu verwendenden

Programmiersprache, mit der SAP-Gateway-Services konsumiert werden sollten. Alles, was sichergestellt werden muss, ist, dass die Client-Technologie Möglichkeiten zur HTTP(S)-Kommunikation bietet und XML- oder JSON-Dokumente konstruieren und parsen kann. Darüber hinaus bietet SAP vollkommene Freiheit bezüglich der Client-Technologie. Die Entwicklungstätigkeiten können mit der Verwendung einer OData-Client-Bibliothek noch weiter vereinfacht werden. Die OData-Client-Bibliothek nimmt Ihnen als Entwickler dabei beispielsweise Serialisierung, Deserialisierung und die Validierung der OData-Payloads ab.

Integrierte Entwicklungsumgebungen (IDE)

Integrierte Entwicklungsumgebungen werden zur Entwicklung von SAP-Gateway-Konsumenten verwendet. Diese Entwicklungsumgebungen sind üblicherweise wohlbekannte Standardentwicklungsumgebungen, wie beispielsweise SAP Business Application Studio, Eclipse oder das Microsoft Visual Studio. Abhängig von der Zielumgebung bietet SAP Gateway spezifische Funktionalitäten und Verbesserungen, die eine einfache und schnelle Erstellung von Applikationen zur Konsumierung von SAP-Gateway-Services erlauben.

3.2.2 SAP-Gateway-Schicht

Kernkomponenten

Die SAP-Gateway-Schicht ist das Herz von SAP Gateway (wie Sie in Abbildung 3.2 gesehen haben). Die Herzkammern sind dabei die Kernkomponenten. Diese Kernkomponenten sind über verschiedene Versionen von SAP Gateway stabil, solange wir die spezifischen, einzelnen Komponenten betrachten. Die Paketierung in den unterschiedlichen SAP-Gateway-Versionen hat sich allerdings weiterentwickelt. Dies bedeutet, dass vor SAP NetWeaver 7.40 `GW_CORE` und `GW_FND` und ab SAP NetWeaver 7.40 `SAP_GWFND` die gleichen spezifischen Kernkomponenten enthalten – allerdings in einer geänderten Paketierung. In den neueren Versionen von SAP NetWeaver (ab 7.40) sind die SAP-Gateway-Kernkomponenten bereits Teil des SAP-NetWeaver-Stacks. Die Installation von zusätzlichen Softwarekomponenten für SAP Gateway ist für einen Großteil der Anwendungsfälle nicht mehr notwendig.

Die Kernkomponenten beinhalten einige zentrale Komponenten wie die Laufzeitkomponente, die OData-Bibliothek und die Metadatenkomponente (siehe Abbildung 3.3):

- Die Laufzeitkomponente beinhaltet eine OData-spezifische Laufzeitumgebung, die effizient OData-Requests prozessiert, und die benötigte Funktionalität, um OData-Services zu exponieren.
- Die OData-Bibliothek beinhaltet SAP-spezifische Metadaten, die bei der Konsumierung von SAP-Geschäftsdaten hilfreich sind. Dies sind beispielsweise Beschreibungen von Feldern, die vom ABAP-Dictionary (DDIC) empfangen werden können.
- Die Metadatenkomponente verwaltet die Metadaten im SAP-Gateway-Server. Die SAP-Gateway-Metadaten beschreiben primär OData-Modelle, die als OData-Service-Dokumente und OData-Service-Metadatendokumente exponiert werden. Im Speziellen exponiert die Metadatenkomponente die standardisierte Beschreibung von OData-Services durch das Zusammensetzen von OData-Service-Dokumenten und Servicemetadatendokumenten. Diese Daten werden dann im Metadaten-Cache vorgehalten.

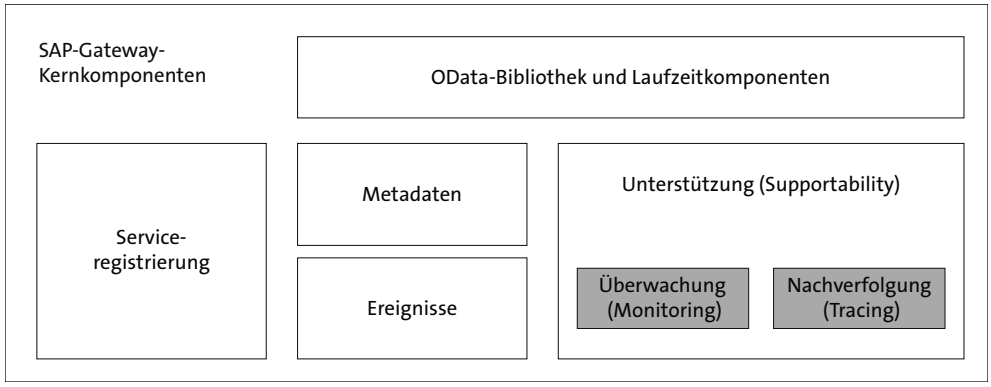

Abbildung 3.3 Kernkomponenten von SAP Gateway

Zusätzliche Kernkomponenten

Zusätzlich zu den zentralen Kernkomponenten gibt es eine Reihe anderer Komponenten, die in die Kategorie der Kernkomponenten fallen. Die Serviceregistrierung (*Service Registry*) ist der Datenspeicher, der die Verbindung zwischen einem OData-Service und der tatsächlichen Implementierung dieses Service hält. Dies kann im SAP-Backend-System oder aber lokal in SAP Gateway stattfinden. Laufzeitservices (*Runtime Services*) für Unterstützung (*Supportability*), Nachverfolgung (*Tracing*) und Überwachung (*Monitoring*) stellen eine sichere Verbindung zwischen den Konsumenten und den SAP-Systemen bereit und bieten auch die Möglichkeit, Nachrichten von SAP-Gateway-Komponenten zum SAP-Backend-System und zurück zu überwachen und zu verfolgen. Zudem werden Events (Ereignisse) unterstützt, um Push-Szenarien zu ermöglichen (beispielsweise Workflow-Szena-

rien, in denen eine Komponente auf Events in SAP-Systemen lauscht und dann Datenbeschreibungen dieses Events liefert, die wiederum an die Konsumenten geliefert werden können).

Generic Channel

In den ursprünglichen Releases von SAP Gateway wurden alle Entwicklungsaktivitäten im sogenannten *Generic Channel* ausgeführt. Dieser Kanal setzte auf einem generischen Framework auf, das eine Reihe von Exponierungsmodellen (*Exposure Models*) unterstützte. Mit SAP Gateway 2.0 SP3 wurde dann der *OData-Channel* eingeführt. Dieser ist auf die OData-Anforderungen und -Spezifikationen zugeschnitten und optimiert. Heute ist der OData-Channel die empfohlene und bevorzugte Option. Der Generic Channel sollte nicht mehr für Neuentwicklungen verwendet werden. Aus diesem Grund fokussiert dieses Buch den OData-Channel, bei dem die Entwicklung üblicherweise in der SAP-Backend-Schicht erfolgt. (Wir werden in Kapitel 5, »Einführung in die Erstellung von OData-Services mit SAP Gateway«, auf den OData-Channel eingehen.)

3.2.3 SAP-Backend-Schicht

Geschäftslogik

Die SAP-Backend-Schicht hält, wie zu erwarten, die Daten und die Geschäftslogik von SAP S/4HANA oder der SAP Business Suite. Diese Schicht besteht aus den Entwicklungswerkzeugen und der Service-Provider-Implementierung. Von SAP NetWeaver ABAP 7.40 SP02 an basieren sowohl die Entwicklungswerkzeuge als auch die Service-Provider-Implementierung auf `SAP_GWFND`. Dieses wird standardmäßig mit SAP NetWeaver in Versionen neuer als 7.40 SP02 ausgeliefert. `SAP_GWFND` ersetzt eine Reihe von Add-ons, die in früheren SAP-NetWeaver-Versionen zusätzlich installiert werden mussten. Diese ersetzten Add-ons beinhalten das Add-on für das SAP Business Suite Enablement und das Event Provisioning (`IW_BEP`), `IW_HDB`, `GW_CORE` und `IW_FND`.

Entwicklungswerkzeuge

Service Builder

Eine Reihe von in der SAP-Business-Suite-Schicht angesiedelten Entwicklungswerkzeugen unterstützt die Erstellung von SAP-Gateway-Services. Diese Entwicklungswerkzeuge sind insofern sehr wichtig, als es ohne sie nicht viel zu konsumieren gäbe – zumindest nicht viel, was für die spezifischen Anwendungsszenarien geeignet wäre (zumeist werden Services für die spezifische Nutzung angepasst, wenn nicht sogar komplett neu generiert oder entwickelt, um perfekt auf die Szenarien zugeschnitten zu sein).

Der Service Builder (Transaktion SEGW) ist das Hauptwerkzeug für die Serviceerstellung und bietet Entwicklern Werkzeuge zur Vereinfachung dieser Serviceerstellung im gesamten Entwicklungslebenszyklus eines Service. Der Service Builder bietet eine OData-konforme Modellierungsumgebung für die Erstellung und Wartung von SAP-Gateway-Services. Alle Entwicklungsartefakte, die Entwickler zur Serviceerstellung benötigen, werden visuell dargestellt, und Assistenten ermöglichen die Erstellung von aggregierten Objekten auf der Grundlage von Remote Function Calls (RFC), dem Business Object Repository (BOR) und Business Application Programming Interfaces (BAPIs). Zusätzlich unterstützt der Service Builder Entwickler bei der codebasierten Entwicklung von SAP-Gateway-Services (für weitere Details bezüglich des Service Builders siehe Kapitel 5, »Einführung in die Erstellung von OData-Services mit SAP Gateway«).

Das zweite Werkzeug in Bezug auf die Erstellung von auf *Core Data Services Views* (CDS Views) basierenden OData-Services sind die auf Eclipse basierenden ABAP Development Tools (*ABAP in Eclipse*). Zusätzlich zur Entwicklung von CDS Views können die ABAP Development Tools auch für codebasierte Implementierungen genutzt werden. Wird die Transaktion SEGW aus den ABAP Development Tools gestartet, so kann anstelle der Transaktion SE24, die bei einem Start von SEGW aus dem klassischen SAP GUI ausgeführt würde, der Eclipse-Editor benutzt werden.

Core Data Services

Weitergehende Informationen hinsichtlich der CDS Views finden Sie hier: *http://s-prs.de/v9808011*

3.2.4 Evolution der Add-on-Struktur

Add-ons

Wie bereits im letzten Abschnitt erläutert, bestand SAP Gateway in Versionen vor SAP NetWeaver 7.40 SP02 im Grundsatz aus einer Reihe von Add-ons auf der ABAP-Technologieplattform. Mit SAP NetWeaver 7.40 SP02 ist dann das Deployment von SAP Gateway deutlich verbessert worden. Die Add-on-Strukturen vor und nach SAP NetWeaver 7.40 SP02 unterscheiden sich also, wobei die neue Add-on-Struktur vereinfacht und optimiert worden ist. Die meisten der ursprünglichen Add-ons sind im SAP-NetWeaver-Standard aufgegangen. Es verbleiben allerdings auch jetzt noch einzelne Add-ons für spezifische Anwendungsfälle.

Nachdem wir uns die Architektur und die Hauptbestandteile von SAP Gateway angesehen haben, ist es nun an der Zeit, einen Blick auf die Add-on-Struktur zu werfen.

Werfen wir also einen Blick auf die Evolution der SAP-Gateway-Add-ons/-Komponenten aus einer zeitlichen Perspektive. Unter SAP Gateway 2.0 gab es drei verpflichtende Add-ons, die auf Systemen mit SAP NetWeaver 7.31 oder niedriger installiert werden mussten: GW_CORE, IW_FND und IW_BEP (siehe Abbildung 3.4).

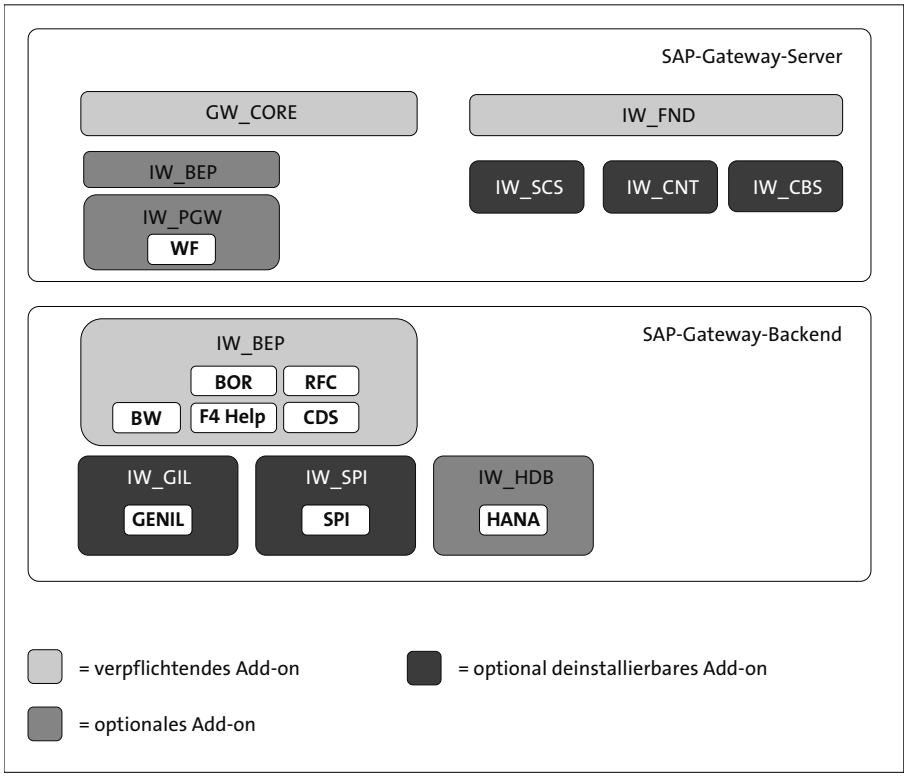

Abbildung 3.4 Struktur der SAP-Gateway-2.0-Add-ons

Mit SAP NetWeaver 7.40 SP02 gab es dann eine Optimierung des Deployments (siehe Abbildung 3.5). Diese Deployment-Optimierungen mündeten primär in zwei neue Softwarekomponenten: IW_FNDGC und SAP_GWFND. SAP_GWFND kombiniert vier alte Add-ons – IW_HDB, IW_BEP, IW_FND und GW_CORE.

Vereinfachte Add-on-Struktur

Mit SAP NetWeaver 7.51 wurde die Ad-on-Struktur dann noch weiter vereinfacht (siehe Abbildung 3.6). Das Add-on IW_PGW wurde ebenfalls Teil der Softwarekomponente SAP_GWFND.

IW_FNDGC wird dabei nur für die Unterstützung von Legacy-Funktionalitäten benötigt. Diese sind für die Kunden relevant, die von SAP Gateway 2.0 auf SAP NetWeaver 7.40 und höher migrieren. Eine Reihe von älteren Add-ons wird zudem nicht mehr empfohlen (IW_CBS, IW_CNT und IW_SCS). Die vier Add-ons IW_HDB, IW_BEP, IW_FND und GW_CORE wurden deinstallierbar gemacht.

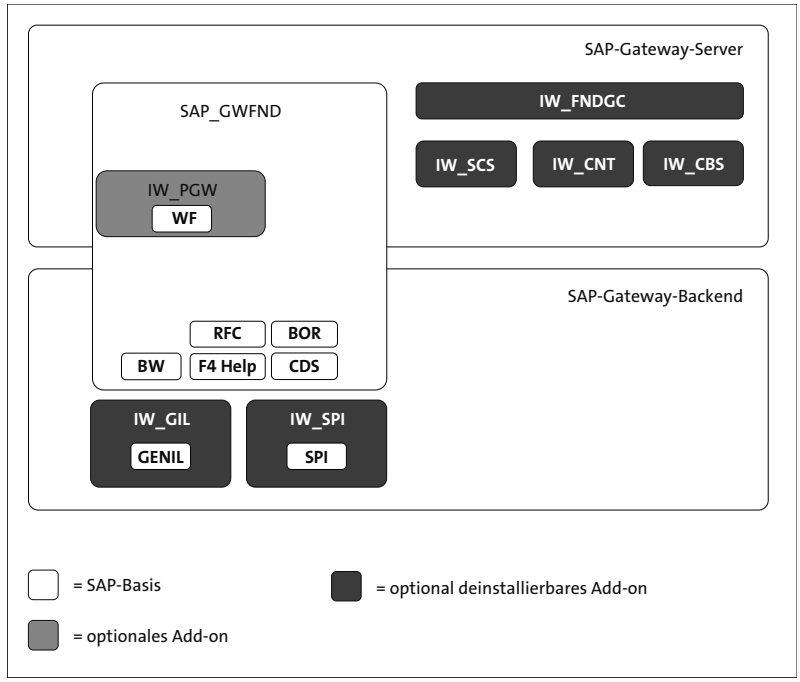

Abbildung 3.5 Neue Add-on Struktur in SAP NetWeaver 7.40

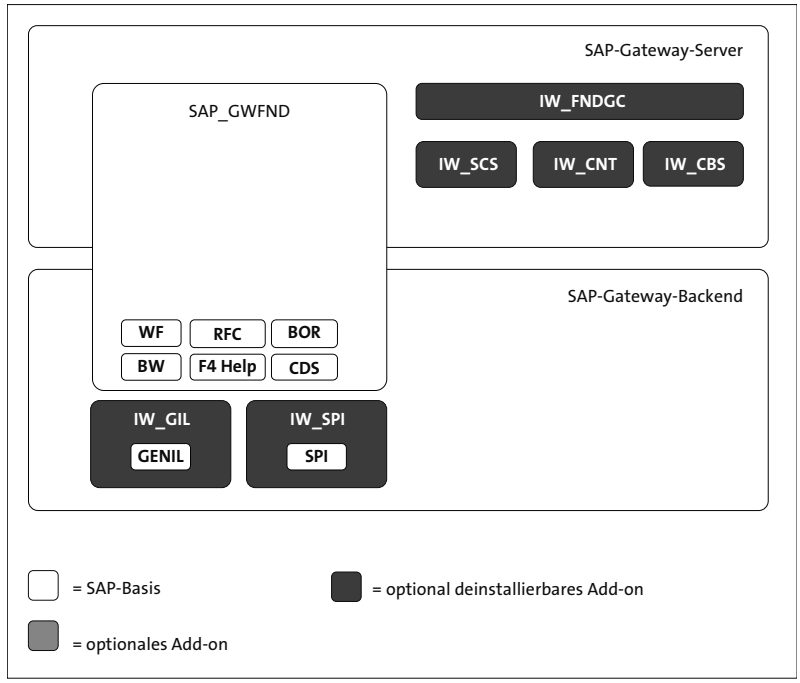

Abbildung 3.6 Add-on-Struktur in SAP NetWeaver 7.51

Wie gezeigt, existieren für SAP-NetWeaver-Versionen vor 7.40 SP02 zwei Add-ons für das SAP-Gateway-Schicht-Framework (GW_CORE und IW_FND) und vier Add-ons für die SAP-Backend-Schicht (IW_BEP, IW_HDB, IW_GIL und IW_SPI).

Funktionen

Von SAP Gateway 7.40 SP02 an vereint die Komponente SAP_GWFND die vier alten Add-ons (IW_HDB, IW_BEP, IW_FND und GW_CORE). SAP_GWFND ist Teil des SAP-NetWeaver-Standards. Ab SP02 kann SAP Gateway ohne die Installation von zusätzlichen Softwarekomponenten genutzt werden.

SAP_GWFND beinhaltet dabei den funktionalen Umfang der kombinierten Komponenten. Somit findet sich hier die für das OData-Protokoll benötigte Funktionalität und zudem das Framework des SAP-Gateway-Servers. Dieses beinhaltet die Laufzeitkomponenten, die Metadatenkomponenten und gemeinsam genutzte Services wie Monitoring, Supportability und Sicherheit.

Im Ergebnis kann somit jeder SAP-NetWeaver-Server (ab Version 7.40 SP02) die Rolle eines SAP-Gateway-Hubs oder SAP-Gateway-Backends übernehmen.

Optionale Add-ons

Zudem existieren noch einige optionale Add-ons. Diese können zusätzlich zu den mit SAP NetWeaver 7.40 SP2 gelieferten installiert werden, wenn spezielle Funktionalitäten genutzt werden sollen. Diese optionalen Add-ons sind:

- **IW_PGW (Process Gateway)**
  Sollte der Einsatz von SAP-Fiori-Approval-Anwendungen, wie beispielsweise *My Inbox*, geplant sein, so sollte das Add-on IW_PGW (üblicherweise als Process Gateway bezeichnet) installiert werden. IW_PGW ermöglicht die Exponierung von SAP Business Process Management (SAP BPM) und Process Observer Tasks für SAP BPM und SAP Business Workflow (auch als Unified Inbox bezeichnet). Dieses Add-on muss auf dem SAP-Gateway-Hub-Server installiert werden und erwartet eine Installation des Add-ons IW_BEP auf eben diesem Hub. Sollte der Hub auf SAP NetWeaver 7.40 oder 7.50 basieren, so werden die benötigten Softwarekomponenten durch SAP_GWFND zur Verfügung gestellt. Basiert der Hub auf SAP NetWeaver 7.51 (oder höher), so wird das Add-on IW_PGW nicht mehr benötigt, da es Teil der Softwarekomponente SAP_GWFND geworden ist.
- **IW_GIL (GenIL)**
  Das Add-on IW_GIL stellt einen generischen OData-Adapter für auf dem *Generic Interaction Layer* basierenden Content zur Verfügung. Beachten Sie, dass IW_GIL nur bis SAP S/4HANA Release 2022 unterstützt wird.

- **IW_SPI (Service Provider Infrastructure)**
  Das Add-on `IW_SPI` stellt einen generischen OData-Adapter für auf der *Service Provider Infrastructure* (SPI) basierenden Content zur Verfügung. Grundlegende Informationen zu SPI finden Sie unter dieser URL: *http://s-prs.de/v9808012*.
- **IW_HDB (SAP HANA)**
  Das Add-on `IW_HDB` liefert einen Adapter für Geschäftsinhalte für SAP Gateway mit SAP HANA, der eine OData-Exponierung von SAP HANA Views mittels des Protokolls *ABAP Database Connectivity* (ADBC) ermöglicht.

Mit dem Übergang zu SAP S/4HANA wurde die Entscheidung getroffen, in SAP S/4HANA nicht alle SAP-Gateway-Add-ons zu unterstützen. Dies liegt daran, dass nicht alle Add-ons in SAP S/4HANA verwendet werden, weshalb sie auch folgerichtig nicht zur Erzeugung von OData-Services eingesetzt werden können. Daher wurden diese Add-ons deinstallierbar gemacht, sodass sie vor einer Migration von einem SAP-Business-Suite-System zu einem SAP-S/4HANA-System deinstalliert werden können.

Migration auf SAP S/4HANA

Wir empfehlen, SAP-Gateway-Add-ons, die nicht benötigt und/oder nicht in SAP S/4HANA unterstützt werden, vor einer Migration auf SAP S/4HANA zu deinstallieren.

Tabelle 3.1 zeigt eine Liste von Add-ons mit den zugehörigen SAP-Hinweisen, in denen Sie weitere Informationen finden.

Add-on	Unterstützt in SAP S/4HANA	Deinstallierbar mit Transaktion SAINT	SAP-Hinweis
IW_FNDGC	ja	ja	2319096
IW_GIL	ja	ja	2319114
IW_SCS	nein	ja	2319097
IW_CNT	nein	ja	2312680
IW_CBS	nein	ja	2312680
IW_SPI	nein	ja	2313222

Tabelle 3.1 Unterstützung optionaler SAP-Gateway-Add-ons in SAP S/4HANA

3.3 Integration mit anderen SAP-Technologien

Wiederverwendung von Datenquellen

Wie schon in der Einführung zu diesem Kapitel erwähnt, hat SAP Gateway das klare Ziel, klassische SAP-Systeme für Nicht-ABAP-Entwickler zu öffnen. Diese klassischen SAP-Systeme bieten bereits eine Reihe von Möglichkeiten, um Daten über unterschiedliche Kanäle der Außenwelt zur Verfügung zu stellen. Daher ist es logisch, dass man, um eine möglichst große Menge von Services für eine Wiederverwendung zur Verfügung zu stellen, die bereits vorhandenen Komponenten als Basis nimmt. SAP Gateway stellt deshalb Werkzeuge bereit, die auf eine einfache Weise Services aus bereits existierenden Datenquellen der SAP-Backends generieren. Diese bauen auf den generischen Datenzugriffsvarianten auf. Das heißt, dass Entwickler Datenquellen – wie beispielsweise RFC/BOR, SAP BW InfoCubes, multidimensionale Ausdrücke (MDX), SAP BW Easy Query, CDS oder auf dem GenIL- und auf dem Service-Provider-Infrastructure-Framework basierenden Content – für die Erstellung von SAP-Gateway-Services verwenden können.

SAP-Standardkonzepte oder Add-ons

In manchen Fällen werden auch SAP-Standardtechnologien verwendet (beispielsweise RFC oder BOR, CDS). In anderen Fällen ermöglicht SAP Gateway die Verwendung von generischen Datenzugriffsmöglichkeiten per Add-on (beispielsweise Service Provider Infrastructure). Diese Add-ons sind entweder lokale Add-ons, die auf dem betreffenden Business-Suite-System (oder, wo zutreffend, SAP S/4HANA) deployt werden (beispielsweise `IW_GIL`), oder fernaktivierte Add-ons (beispielsweise `IW_SPI`). Wir werden uns diese verschiedenen Varianten nun genauer ansehen.

3.3.1 Remote Function Call

Schnittstellen

Ein *Remote Function Call* (RFC) ist ein SAP-Standardkonzept, um einen Funktionsbaustein in einem vom aufrufenden System verschiedenen System zu rufen. Diese Remote Function kann dabei allerdings durchaus auch im selben System aufgerufen werden (als ein lokaler Call). Die Fähigkeit zum Aufrufen von Remote Functions wird vom RFC-Interface-System bereitgestellt. RFCs erlauben Remote Calls zwischen zwei SAP-Systemen oder zwischen einem SAP-System und einem Nicht-SAP-System. RFCs bestehen aus zwei Arten von Schnittstellen (Interfaces): einer Schnittstelle zum Aufrufen von ABAP-Programmen und einer Schnittstelle zum Aufrufen von Nicht-ABAP-Programmen. Weil SAP Gateway auf das Öffnen von ABAP-Systemen abzielt, ist die relevante Schnittstellenart für SAP Gateway in diesem Kontext die der ABAP-Schnittstellen.

Gibt es ein verfügbares RFC-Function-Modul, kann ein SAP-Gateway-Service einfach mithilfe der SAP-Gateway-Entwicklungswerkzeuge exponiert werden.

3.3.2 Business Object Repository

BAPIs

SAP-Geschäftsobjekte und deren BAPIs werden im *Business Object Repository* (BOR) in einer auf der Hierarchie der betreffenden SAP-Backend-Application-Area basierenden Struktur verwaltet. Jeder einzelne SAP-Geschäftsobjekttyp und die zugehörigen Methoden sind im BOR identifiziert und beschrieben. Diese Beschreibung beinhaltet den SAP-Geschäftsobjekttyp, die zugehörigen SAP-Schnittstellentypen und die Komponenten wie Methoden, Attribute und Ereignisse. *Business Application Programming Interfaces* (BAPIs) sind definiert als die Methoden der SAP-Geschäftsobjekttypen. Nach außen legen SAP-Geschäftsobjekte nur ihre Schnittstellen offen.

Das BOR beinhaltet wie gesagt alle SAP-Geschäftsobjekttypen, sodass es eine perfekte Basis für die Generierung von SAP-Gateway-Services darstellt. Diese können einfach mithilfe der SAP-Gateway-Entwicklungswerkzeuge erstellt werden.

3.3.3 Service Provider Infrastructure

Die Service Provider Infrastructure (SPI) ist eine Applikations- und UI-unabhängige Schicht für die Exponierung von Geschäftsdaten, die in der gesamten SAP Business Suite Verwendung finden. Beispiele sind die Bill of Material (BOM, Materialliste) im SAP Product Lifecycle Management (SAP PLM), die Kaufaufträge im Materialmanagement (MM) und die Sustainability Enhancements (Verbesserung der Zukunftsfähigkeit) für Verkäufer im Finanzwesen (FI).

User Interface von SAP Business Suite entkoppeln

Das Hauptziel der SPI besteht darin, das User Interface komplett von der SAP Business Suite zu entkoppeln. Es gibt keine Abhängigkeit von Web Dynpro oder von einer anderen UI-Technologie. Stattdessen fungiert die SPI als Backbone für unterschiedliche Feeder-Technologien, wie beispielsweise SAP Interactive Forms by Adobe.

Durch die häufige Verwendung in allen Bereichen der SAP Business Suite wird die SPI zu einer hervorragenden Quelle für SAP-Gateway-Services. Ein Gateway-Service kann hier einfach unter Verwendung des Service Builders aus einem SPI-Objekt erstellt werden. Möglich ist dies in SAP-Business-Suite-Systemen – in SAP-S/4HANA-Systemen funktioniert es nicht. Hintergrund ist, dass das benötigte Add-on `IW_SPI` nicht durch SAP S/4HANA unterstützt wird (siehe Tabelle 3.1).

3.3.4 SAP BW InfoCubes

Das der Business Intelligence, dem Business Planning, analytischen Services und dem Data Warehousing dienende *SAP Business Warehouse* (SAP BW) ist eine exzellente Quelle qualitativ hochwertiger analytischer Daten.

Relationale Tabellen

InfoCubes sind dabei die zentralen Objekte im SAP BW. Ein InfoCube ist eine Menge von relationalen Tabellen, die nach einem Sternschema angeordnet sind. Eine große Faktentabelle (*Fact Table*) ist von mehreren Dimensionstabellen (*Dimension Tables*) umgeben.

3.3.5 Multidimensionale Ausdrücke

Zugriff auf eine OLAP-Datenbank

Multidimensionale Ausdrücke, kurz MDX für *Multidimensional Expressions*, ist eine Sprache für die Abfrage und Manipulation von multidimensionalen Daten in einer OLAP-Datenbank (Online Analytical Processing). MDX ist stark von Microsoft gefördert worden und ein allgemein akzeptierter Industriestandard, der von einer Reihe von Anbietern unterstützt wird – auch von SAP für SAP BW.

SAP Gateway bietet durch die Verwendung von MDX einen einfachen Weg, um BW-Funktionalitäten zu exponieren. Diese Exponierung erfolgt mithilfe des *SAP Gateway Analytics Service Generators*, der Teil des Service Builders ist. Wie der Name schon andeutet, handelt es sich hier um einen Generierungsprozess, eine codebasierte Programmierung ist nicht erforderlich. MDX ist die bevorzugte Wahl für ältere Releases von SAP BW. Es wird seit SAP BW 7.0 unterstützt. Für neuere Releases wird die Verwendung von SAP Business Warehouse Easy Query empfohlen, das seit SAP BW 7.30 SP08 oder SAP BW 7.31 SP05 verfügbar ist.

3.3.6 SAP Business Warehouse Easy Query und OData Query

Exponierung von BW-Daten

Leichtgewichtige Konsumierung ermöglicht den Zugriff auf analytische Abfragen von SAP BW in einfachen Szenarien, z. B. von mobilen Plattformen aus. Die Integration von SAP BW mit SAP Gateway ermöglicht es, Abfragen als Easy Queries oder als OData Queries zu stellen. Easy Query und OData Query sind dabei zwei sehr ähnliche Technologien. Easy Query wurde zuerst eingeführt, OData Query folgte dann später nach.

Easy Query

SAP Business Warehouse (BW) Easy Query ist deutlich weniger komplex als MDX und mit Blick auf Einfachheit und eine einfache Konsumierung entworfen worden. Tatsächlich muss nur auf Query-Ebene ein Kennzeichen im *BEx Query Designer* gesetzt werden. Alles Weitere nach dem Setzen des Kennzeichens erledigt dann das SAP-BW-System. Basierend auf als SAP BW

Easy Query markierten Abfragen und mit SAP Gateway für den tatsächlichen Zugriff können so OData-Services exponiert werden. Easy Queries können via SOAP, RFC und REST (OData) bereitgestellt werden.

OData Queries

OData Queries sind ein modernerer Ansatz und ermöglichen eine Integration direkt über SAP Gateway. OData-Abfragen bieten dabei bessere Leistung und mehr Funktionen. Außerdem ist die Konfiguration bei OData Queries einfacher. OData Queries werden, wie der Name schon sagt, über REST (OData) ausgeführt. Es ist wichtig zu beachten, dass der RFC- und SOAP-Zugriff im Vergleich zum Zugriff über SAP Gateway sehr eingeschränkt ist. Auch RFC/SOAP-Optionen wie Filterung, Sortierung oder Paginierung stehen nicht zur Verfügung.

Letztlich sind beide Optionen geeignet. OData Queries sind aber vorzuziehen, da sie einfacher und schneller sind.

3.3.7 Generic Interaction Layer

API

Der Generic Interaction Layer (GenIL) ist eine vereinheitlichende Schnittstelle zwischen dem Client und integrierten Applikationen. Ein Client (UI) kann GenIL verwenden, um auf alle APIs der verlinkten Applikationen zuzugreifen, ohne die API-Details oder die unterliegenden Datenstrukturen zu kennen. Das bedeutet, dass der Client nicht speziell für die Interaktion mit jeder individuellen Applikation programmiert werden muss. Stattdessen muss er nur für die GenIL-Applikation programmiert werden.

Teil des Business Object Layers

Der Business Object Layer (BOL) speichert die Daten eines Geschäftsobjekts zusammen mit einer definierten Menge von Attributen und Beziehungen während der Laufzeit. Als Teil des BOLs verbindet GenIL den Business Object Layer mit der unterliegenden Geschäftslogik und den unterliegenden Datenbanktabellen.

GenIL wird häufig für das SAP Customer Relationship Management Web Client UI (SAP CRM) verwendet. Zudem wird GenIL auch in anderen Applikationen der SAP Business Suite eingesetzt, wie beispielsweise SAP ERP Financials und SAP ERP HCM (Human Capital Management). Zudem wird GenIL in SAP-S/4HANA-Applikationen wie beispielsweise der CLAIM-Applikation verwendet.

SAP Gateway erlaubt die Generierung von OData-Services, die GenIL verwenden. Somit kann man von SAP bereitgestellte GenIL-Objekte genauso für die Generierung von OData-Services nutzen wie auch speziell erstellte kundenspezifische GenIL-Objekte. Die Knoten, Relationen und Abfragen in den GenIL-Modellen werden dabei in die entsprechenden Entitäten in einem OData-Modell transformiert.

3.3.8 SAP Business Process Management

Controlling und Automatisierung von Geschäftsprozessen

SAP Business Process Management (SAP BPM) erlaubt das Controlling und die Automatisierung von Geschäftsprozessen. Hierbei wird ein Unternehmen aus der Prozessperspektive betrachtet – im Gegensatz zu einer Organisations-Chart-Perspektive –, und Prozesse können Ende-zu-Ende modelliert werden. Dies ist sehr nützlich für die Analyse und Modellierung von Geschäftsprozessen. SAP BPM unterstützt den gesamten Lebenszyklus von Geschäftsprozessen – von der Planung über die Implementierung und das Monitoring bis hin zur Optimierung.

3.3.9 SAP Business Workflow

Geschäftsprozesse

SAP Business Workflow erlaubt die Erstellung und Ausführung von Geschäftsprozessen in SAP-Applikationssystemen: Workflow-Prozesse werden als Inhalte in der SAP Business Suite angeboten. Es ist möglich, diese von SAP ausgelieferten Workflows zu erweitern und kundenspezifische Versionen zu erstellen.

Um es Anwendern von SAP Business Workflow zu ermöglichen, Workflow-Items auf jedem Gerät oder jeder Plattform zu bearbeiten, kann SAP Gateway sogenannte SAP-Business-Workflow-Tasks als OData-RESTful-Services exponieren. Dies erlaubt eine Reihe von neuen Geschäftsszenarien, wie beispielsweise Workflow-Inboxen auf mobilen Geräten.

3.4 ABAP-Programmiermodell für SAP Fiori

SAP Fiori

SAP Fiori ist 2013 eingeführt worden und hat sich seitdem aus einer Sammlung von Applikationen zur standardmäßigen Benutzeroberfläche für SAP-Software entwickelt. Inzwischen wird es über verschiedenste SAP-Lösungen hinweg verwendet – hier reden wir beispielsweise auch über SAP S/4HANA. SAP Fiori bietet eine konsistente und vollumfängliche Nutzererfahrung für SAP-Software. SAP Fiori 2.0 wurde im Oktober 2016 veröffentlicht. SAP Fiori 3.0 folgte im April 2019 und stellt derzeit die neueste Evolutionsstufe hinsichtlich der Nutzererfahrung für SAP S/4HANA und die SAP Business Suite dar. SAP Fiori 3.0 fokussiert noch stärker die Nutzer und die Art, wie Nutzer von SAP-Software arbeiten. Es liefert nicht zuletzt eine über Cloud- und On-Premise-Systeme harmonisierte Nutzererfahrung.

Ausgehend von den sich ändernden Benutzererwartungen und der Einführung von SAP Fiori als der neuen Benutzeroberfläche wurde eine Neudefinition des ABAP-Entwicklungsansatzes für Lösungen wie SAP S/4HANA nötig.

Ende-zu-Ende-Programmiermodell

Um die gestiegenen Benutzererwartungen zu erfüllen und die stark verbesserte technologische Basis zu nutzen, hat SAP ein einfach zu konsumierendes Ende-zu-Ende-ABAP-Programmiermodell für die Erstellung von SAP-HANA-optimierten, browserbasierten Applikationen geliefert. Dieses Programmiermodell liefert dem neuesten Stand der Technik entsprechende Benutzererfahrungen sowohl auf On-Premise-Systemen als auch ganz speziell mit SAP S/4HANA in der Cloud. Dabei behält das neue Programmiermodell den Lebenszyklus und die Supportability-Features der Vorgängermodelle. Gleichzeitig unterstützt es neue Innovationen mit seiner Fähigkeit, datenintensive Logik nach SAP HANA zu verlagern und Logik zur Benutzeraktion im Browser auszuführen.

3.4.1 Architektur und Technologie

Architektur des ABAP-Programmiermodells für SAP Fiori

Das ABAP-Programmiermodell für SAP Fiori definiert die Architektur für effiziente Ende-zu-Ende-Entwicklung von SAP-HANA-optimierten SAP-Fiori-Apps. Es unterstützt die Entwicklung aller Arten von SAP-Fiori-Apps, beispielsweise transaktionale, Such-, analytische und Planungs-Apps.

Aus technologischer Sicht basiert das Programmiermodell auf bewährten Technologien und Frameworks wie CDS für die Datenmodellierung und den Datenzugriff, das OData-Protokoll für die Exponierung der Services, ABAP-basierte Applikationsservices für die Custom-Logik, SAPUI5-basierte Benutzeroberflächen und das Business Object Processing Framework (BOPF) für die transaktionale Verarbeitung.

Aus einer datenfokussierten Ende-zu-Ende-Perspektive liefert das auf SAP HANA basierende CDS-Paradigma Unterstützung sowohl für die transaktionalen als auch die analytischen Aspekte der unterliegenden Geschäftsobjekte.

Im Frontend bietet das SAPUI5-basierte SAP Fiori ein responsives Design für sowohl auf mobilen Endgeräten als auch auf traditionellen Desktops laufende Applikationen.

Im Detail sieht der Ende-zu-Ende-Stack des neuen ABAP-Programmiermodells für SAP Fiori aus, wie in Abbildung 3.7 dargestellt.

Drei Hauptschichten des Stacks

Die im Folgenden beschriebenen drei Hauptschichten bilden den Stack des ABAP-Programmiermodells für SAP Fiori:

1. In der SAPUI5-Client-Schicht ❶ ist entweder freie oder Template-basierte UI-Entwicklung mit SAPUI5 oder SAP Fiori Elements möglich. SAP Fiori Elements liefert Templates für häufige Patterns bei der SAP-Fiori-Applikationsentwicklung. SAPUI5 ist eine Client-seitige UI-Technologie von SAP.

2. Die SAP-NetWeaver-Schicht ❷ beinhaltet Folgendes:
   - die OData-Protokoll-Infrastruktur mit SAP Gateway für eine direkte OData-Service-Exponierung
   - eine effektive und effiziente Applikations-Entwicklungsumgebung mit modernen ABAP-Entwicklungswerkzeugen
   - Transaktions-, Fehlerbehandlungs-, Ereignis- und Draft-Handling-Services mit BOPF
   - eine gemeinsame Datenmodellierungsinfrastruktur für alle Services mit CDS
3. Mit der SAP-HANA-Schicht ❸ werden schließlich spezielle Vorteile von *SAP HANA* genutzt. SAP HANA ist die In-Memory-Plattform von SAP, um Realtime-Applikationen zu entwickeln und zu deployen. Mit der Nutzung von ABAP-gemanagten datenbasierten Prozessen und CDS-Tabellenfunktionen werden die auf absolute Spitzenleistung ausgelegten SAP-HANA-Technologien im Rahmen des ABAP-Programmiermodells eingesetzt.

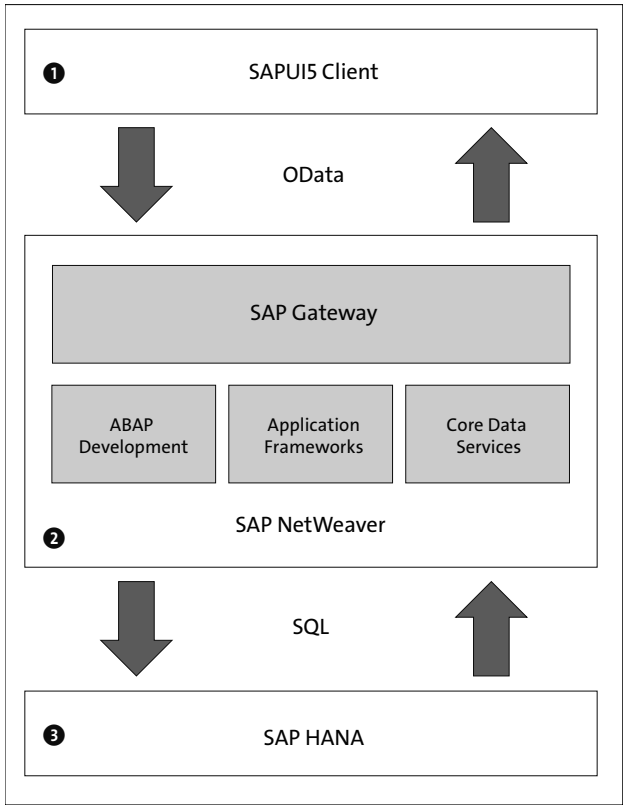

Abbildung 3.7 ABAP-Programmiermodell für SAP Fiori

Sehen wir uns nun die unterschiedlichen Schichten und ihre Komponenten im Detail an, damit Sie sie besser verstehen und lernen, wie sie zusammen funktionieren.

3.4.2 SAPUI5 und SAP Fiori Elements

Wie schon im vorherigen Abschnitt erläutert, gibt es bezüglich der SAP-Fiori-Entwicklung zwei Möglichkeiten, aus denen die am besten passende gewählt werden kann:

- freie UI-Entwicklung mit SAPUI5
- Template-basierte UI-Entwicklung mit SAPUI5 oder SAP Fiori Elements

In diesem Abschnitt werden wir beide Möglichkeiten betrachten.

SAPUI5

SAPUI5 ist eine auf JavaScript, CSS und HTML5 basierende Benutzeroberflächentechnologie auf Client-Seite. Mit SAPUI5 entwickelte Apps laufen im Browser auf allen möglichen Endgeräten (mobilen, Tablets, Desktop-PCs). Um mit SAPUI5 Apps zu entwickeln, können Sie entweder App-Templates verwenden (Template-basierter Ansatz) oder aber die SAPUI5-App vollständig selbst entwickeln (Freestyle-Ansatz).

App-Templates

App-Templates sind Best Practices für die App-Entwicklung. Sie können als Startpunkt genutzt werden, um den SAP-Fiori-Designrichtlinien entsprechende Apps zu entwickeln. Die Templates enthalten generische App-Funktionalität und auch gut zu erweiternde Tests, die schnell kundenspezifisch angepasst werden können. Zum Beispiel finden sich die folgenden, von SAP zur Verfügung gestellten Templates auf dem SAP Repository im GitHub (siehe *http://s-prs.de/v9808013*):

- **Das Basis-Template (openui5-basic-template-app)**
  Das Basis-Template zielt auf Entwickler ab, die ihre SAPUI5-App von Grund auf entwickeln möchten. Mit diesem Template haben Sie einen leeren Canvas (sozusagen eine »Leinwand«), um sofort mit dem Codieren anfangen zu können. Die grundlegende Dateistruktur entspricht SAP-Best-Practices.
- **Das SAP-Fiori-Worklist-Applikations-Template**
  Dieses Template implementiert einen typischen Worklist-Floorplan – eines der Entwicklungsmuster, die SAP in den SAP Fiori Design Guidelines spezifiziert hat.

- **Das SAP-Fiori-Master-Detail-Applikations-Template**
  Mit diesem Template wird ein typisches Split-Screen-Layout implementiert – eines der Entwicklungsmuster, die SAP in den SAP Fiori Design Guidelines spezifiziert hat.

Entwickler mit den nötigen Fähigkeiten und dem Wunsch nach der größtmöglichen Freiheit und Flexibilität können auch dem Freestyle-Ansatz folgen und eine App von Grund auf und vollständig selbst entwickeln.

SAP Fiori Elements

SAP Fiori Elements liefert Templates für Benutzeroberflächendesigns und die Entwicklung von SAP-Fiori-Apps, die häufig verwendeten Entwicklungsmustern entsprechen.

Vorteile

SAP Fiori Elements stellen ein konsistentes Design und die Übereinstimmung mit dem aktuellen SAP Design Guidelines sicher. Dank vordefinierter Floorplans, Views und Controller ist die Durchgängigkeit in Bezug auf das UI in einzelnen Apps und auch für einander ähnliche Apps gewährleistet.

Zudem reduzieren Sie den zur Erstellung von SAP-Fiori-Apps benötigten Frontend-Code, sodass Sie sich auf die Geschäftslogik konzentrieren können – die Entwicklung der Geschäftslogik wird also von der der UI abgekoppelt.

Da Sie mit SAP Fiori Elements UIs nicht wieder und wieder erstellen müssen, ist die Entwicklung mit ihnen sehr effizient. Inzwischen decken SAP Fiori Elements ca. 80 % aller typischerweise benötigten Apps ab.

Die daraus resultierenden Apps verwenden vordefinierte Views und Controller, die zentral bereitgestellt werden. Somit werden keine applikationsspezifischen View-Instanzen benötigt. Die SAPUI5-Laufzeit interpretiert die Metadaten und Annotationen des unterliegenden OData-Service und nutzt die korrespondierenden Views beim Starten der SAP-Fiori-App.

SAP Business Application Studio

Das zentrale Werkzeug für die Entwicklung von Apps mit SAP Fiori Elements ist das SAP Business Application Studio. Es bietet Assistenten für die Unterstützung des Entwicklungsprozesses an und ist der empfohlene Startpunkt für die Entwicklung mit SAP Fiori Elements.

Was Annotationen angeht, so wird empfohlen, die Annotationen in den CDS Views hinzuzufügen, auf denen der zu nutzende OData-Service basiert.

3.4.3 SAP Gateway und das ABAP-Programmiermodell für SAP Fiori

Bezüglich der SAP-Fiori-Apps sind OData-Services obligatorische Komponenten. Sie sind das Mittel der Wahl, wenn es darum geht, Geschäftsdaten aus dem Backend zu holen oder im Backend zu aktualisieren. Wenig überraschend wird SAP Gateway verwendet, um OData-Services für diese Provisionierung zu erstellen, zu aktivieren und zu betreiben. Im Folgenden werfen wir einen Blick auf die Evolution der OData-Serviceimplementierungen und geben eine Einschätzung zur aktuellen Bedeutung im Rahmen des ABAP-Programmiermodells für SAP Fiori.

Evolution der OData-Serviceimplementierungen

SAP NetWeaver 7.0 SP28 oder neuer

Die ersten OData-Service-Implementierungen nutzten den Service Builder, um das OData-Modell und codebasierte Implementierungen der Methoden der Daten-Provider-Erweiterungsklasse zu erstellen. SAP-Gateway-Add-ons verwendend, ist diese Form der Serviceimplementierung für alle SAP-Backends verfügbar, die auf SAP NetWeaver 7.0 SP18 oder neuer basieren.

SAP NetWeaver Application Server (AS) für ABAP 7.50 SP01

Das ABAP Programming Model for SAP Fiori wurde zuerst mit dem SAP NetWeaver Application Server (AS) für ABAP 7.50 SP01 verfügbar gemacht. Nachfolgend sahen wir eine fortwährend wachsende Adaptierung der angebotenen Techniken, bis mit dem ABAP RESTful Application Programming Model die Nachfolgegeneration eingeführt wurde. Grundlage des ABAP Programming Models for SAP Fiori ist der Ansatz der referenzierten Datenquellen (*Referenced Data Sources*, RDS) oder alternativ die direkte OData-Exponierung mit der Annotation `odata.publish=true` für einfachere Szenarien.

Implementierung von OData-Services mit dem ABAP-Programmiermodell für SAP Fiori

Für die Implementierung eines OData-Service im Rahmen des ABAP-Programmiermodells für SAP Fiori gibt es drei verschiedene Möglichkeiten (siehe Abbildung 3.8):

- codebasierte Implementierung mithilfe des SAP Gateway Service Builders und somit Freestyle-Entwicklung mit weitgehender Flexibilität und Kontrolle bis ins Detail ❶
- OData-Serviceentwicklung mit SAP Gateway unter Nutzung von CDS über referenzierte Datenquellen ❷
- Über die Annotation `@OData.publish:true` steht out of the box eine OData-Unterstützung zur Verfügung, für die die Version des Protokolls gleichgültig ist ❸

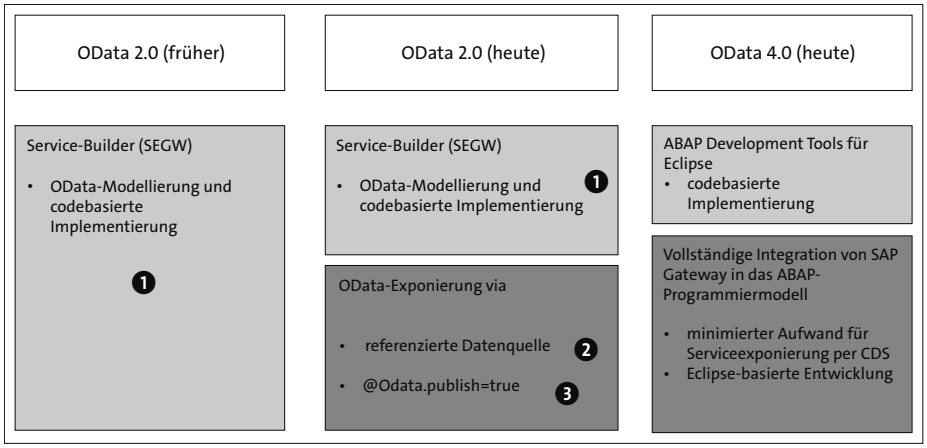

Abbildung 3.8 Möglichkeiten der Implementierung eines OData-Service mit dem ABAP-Programmiermodell für SAP Fiori

Wahl der Implementierungsoption

Welche Implementierungsoption gewählt wird, hängt von Folgendem ab:

- **SAP-Basisrelease des SAP-Backends**
  Zunächst einmal beeinflusst das zugrunde liegende Basisrelease des SAP-S/4HANA- oder SAP-Business-Suite-Systems, welche Implementierungsoptionen infrage kommen. Die Optionen reichen hier von der codebasierten Implementierung bis hin zur Nutzung der ABAP-Programmiermodelle. Die klare Empfehlung ist die Nutzung von CDS, durchaus auch in älteren Releases.
- **OData-Protokoll**
  Die Entscheidung für eine Implementierungsoption hängt auch von der gewünschten oder benötigten OData-Version, also OData 2.0 oder OData 4.0, ab. Beispielsweise gibt es Bibliotheken wie *Olingo OData Client for JavaScript*, die nur das OData-4.0-Protokoll unterstützen. Das ABAP-Programmiermodell unterstützt sowohl OData 2.0 als auch OData 4.0. Die Unterstützung für OData 4.0 wurde im Jahr 2021 eingeführt. Dies gilt sowohl für die ABAP-Umgebung der SAP Business Technology Platform als auch on-premise in SAP S/4HANA 2020 FPS01.

Empfehlungen

Generell können die folgenden Empfehlungen für spezifische Szenarien gegeben werden und als Entscheidungshilfe dienen:

- Verwenden Sie CDS für die Datenmodellierung.
- Verwenden Sie CDS und die BOPF-Integration, und lernen Sie BOPF-Konzepte wie Validierungen und Aktionen näher kennen.
- Verwenden Sie zur OData-Exponierung `@OData.publish:true` oder den Service Builder mit referenzierten Datenquellen für OData 2.0.

- Vermeiden Sie DPC-/MPC-spezifischen Code (DPC – Daten-Provider-Klasse, MPC – Modell-Provider-Klasse), wo immer möglich.

Codebasierte Implementierung von OData-Services

Der Service Builder wird verwendet, um das OData-Modell zu entwerfen und um eine codebasierte Implementierung der Methoden der Daten-Provider-Erweiterungsklasse zu erstellen, die als Adapter zwischen der Geschäftslogik und den Protokollspezifika dient. Die Implementierung der Daten-Provider-Erweiterungsklasse erfordert tiefgehendes Wissen bezüglich des OData-2.0-Protokolls – und Sie müssen den nötigen Code der Daten-Provider-Erweiterungsklasse von Grund auf entwickeln.

Entwicklung von OData-Services mit CDS via RDS

Als das neue Programmiermodell erstmals verfügbar gemacht wurde, unterstützte es zunächst nur lesenden Zugriff (SAP NetWeaver AS ABAP 7.50 SP01). Obwohl mit späteren Versionen (SAP NetWeaver AS ABAP 7.50 SP05) stapeleingabeähnliche SAP-Fiori-Apps erstellt werden konnten, stehen Legacy-APIs mit schreibendem Zugriff bis heute nicht zur Verfügung. Hier kommt der Service Builder ins Spiel, der referenzierte Datenquellen (RDS) nutzt.

Bei diesem Ansatz basiert das Datenmodell auf CDS Views, und der lesende Zugriff wird out of the box von den unterliegenden CDS Views bereitgestellt. Der Service Builder generiert eine Modell-Provider-Erweiterungsklasse und eine Daten-Provider-Erweiterungsklasse, die zur Implementierung von Erweiterungen mithilfe von Code genutzt werden können, während man gleichzeitig die generische Unterstützung für den lesenden Zugriff nutzt. Durch die Implementierung der `CREATE`-, `UPDATE`- und `DELETE`-Methoden ist es beispielsweise möglich, ein BAPI für das Update eines Geschäftsobjekts zu rufen.

@OData.publish:true

Sollten Sie das neue ABAP-Programmiermodell für die Erstellung hochmoderner, für SAP HANA optimierter SAP-Fiori-Apps in SAP S/4HANA nutzen, so erhalten Sie out of the box Unterstützung von OData 2.0. Durch einfaches Hinzufügen der Annotation `@OData.publish:true` zum CDS Consumption View wird ein OData-2.0-Service im SAP-Business-Suite- oder im SAP-S/4HANA-Backend generiert, der im SAP-Gateway-Server publiziert werden kann.

Unabhängigkeit von der Protokollversion

Sowohl die Geschäftsobjekt- als auch die CDS-Implementierung sind von der Protokollversion unabhängig, während die Unterstützung des OData-Protokolls vom SAP-Gateway-Framework bereitgestellt wird. Tiefgehendes Wissen hinsichtlich des OData-Protokolls ist somit nicht erforderlich. Einzig ein minimaler Implementierungsaufwand ist nötig, um `CREATE`-, `UPDATE`- und `DELETE`-Funktionalität für den Service zu erstellen.

Best Practices für das ABAP-Programmiermodell für SAP Fiori

Bezüglich des ABAP-Programmiermodells gibt es eine Reihe von Empfehlungen, was man tun und was man lassen sollte.

Idealerweise sollten Sie Folgendes nicht tun:

- manuelle Implementierung von nur lesenden OData-Aufrufen auf die Datenbank
- Geschäftslogik mit technischen Aspekten mischen
- Geschäftslogik mit protokollspezifischen APIs mischen

Folgende Technologien und Ansätze sind dagegen empfehlenswert:

- CDS für Datenbankartefakte
- CDS-Metadatenerweiterung für UI-Annotationen
- DCL (Data Control Language) für Berechtigungsprüfungen, die auf `READ`-/`QUERY`-Instanzen basieren. DCL wird in CDS verwendet, um instanzenbezogene Berechtigungsmodelle zu definieren.
- eigenständige BOPF-Objekte (Business Object Processing Framework)
- Integration von BOPF und CDS
- SAP-Gateway-Integration von CDS und BOPF
- OData-Exponierung von CDS/BOPF mit Blick auf zukünftige Entwicklungen
- Floorplan-Manager-Integration von CDS und BOPF

3.4.4 SAP HANA

SAP HANA ist eine In-Memory-Plattform, die eine ACID-konforme Datenbank mit fortgeschrittener Datenprozessierung, Applikationsservices und flexiblen Datenintegrationsservices verbindet. In dieser Hinsicht ist SAP HANA wesentlich mehr als ein System zum Management von Datenbanken. Es ist vielmehr eine verständliche Plattform für die Entwicklung und Ausführung von nativen datenintensiven Applikationen, die effizient in SAP HANA laufen und dabei Vorteile aus der In-Memory-Architektur und der parallelen Ausführung ziehen.

ABAP für SAP HANA

ABAP für SAP HANA bezeichnet hierbei alle Entwicklungen, die das Potenzial von SAP HANA in ABAP-basierten Applikationen nutzen. In diesem Kontext ist SAP HANA die primäre Datenbank, die unter der ABAP-Plattform liegt. Diese System-Deployment-Option wird von SAP NetWeaver AS for ABAP 7.40 an unterstützt.

Die Einführung der SAP-HANA-Plattform in das ABAP-Universum hat zu einer Veränderung des Programmierparadigmas geführt. In der ABAP-Plattform sind eine Reihe von Technologien für den Code Pushdown verfügbar, die fortgeschrittene, native Möglichkeiten von SAP HANA auf unterschiedlichste Weise nutzen. Beim Code Pushdown werden rechenintensive Operationen nicht wie bisher auf dem Applikationsserver, sondern bereits direkt auf der SAP-HANA-Datenbank ausgeführt. Das erspart den aufwendigen Prozess, die benötigten Daten aus der Datenbank auf den Applikationsserver zu transportieren.

Die Features und die Performance von SAP HANA bilden die Grundlage des SAP-Fiori-Programmiermodells. Dabei führen Features wie *ABAP Managed Database Procedures* (AMDP) und CDS-Tabellenfunktionen zu einer im gesamten Stack spürbaren Verbesserung der Performance. Da die Möglichkeiten von SAP HANA meist automatisch genutzt werden, müssen Sie als Entwickler nicht viel Zeit auf sie verwenden.

AMDP und CDS-Tabellenfunktion

Sehen wir uns die Besonderheiten der AMDP und der CDS-Tabellenfunktionen genauer an:

- **ABAP Managed Database Procedures**
  AMDP bieten ein klassenbasiertes Framework für die Erstellung und den Aufruf von SAP-HANA-Datenbankprozeduren und -funktionen aus der ABAP-Umgebung. Die ABAP-Plattform ist hier allein verantwortlich für das Lifecycle Management (Erstellung, Update, Löschen und Transport) von derartigen SAP-HANA-Entitäten.
- **CDS-Tabellenfunktionen**
  CDS-Tabellenfunktionen ermöglichen die Integration von ABAP Managed Database Procedures und Functions in CDS-basierten Datenmodellen. Dadurch kann auf die Möglichkeiten von SAP HANA direkt aus den CDS zugegriffen werden.

3.5 ABAP RESTful Application Programming Model

Funktionen

Der evolutionäre Nachfolger des ABAP-Programmiermodells für SAP Fiori ist das ABAP RESTful Application Programming Model. Das ABAP RESTful Application Programming Model ist primär darauf ausgerichtet, für den

Unternehmenseinsatz geeignete SAP-Fiori-UI-Services, Web-UIs, analytische Services und Datenintegrationsservices sowohl in der Cloud als auch vor Ort zu erstellen. Die resultierenden Services und UIs basieren auf OData und sind für SAP HANA optimiert. Um den Erstellungsprozess zu optimieren, bietet das ABAP RESTful Application Programming Model einen effizienten und standardisierten Weg zu diesen Services und UIs, der viele Vorteile bietet und Entwicklern mit Anleitung und bewährten Verfahren hilft.

Wir werden das ABAP RESTful Application Programming Model in den nun folgenden Abschnitten einführen und erläutern.

3.5.1 Einführung

Evolution der ABAP-Programmiermodelle

Das ABAP RESTful Application Programming Model ist die neueste Entwicklungsstufe des ABAP Programming Models. Diese Entwicklung in Bezug auf die Programmiermodelle (siehe Abbildung 3.9) begann mit der klassischen ABAP-Programmierung, wo ab ABAP-Release 7.4 deutliche Optimierungen durchgeführt wurden, um die ABAP-Plattform für die SAP-HANA-In-Memory-Datenbank fit zu machen. Diese Optimierungen beinhalteten die Einführung von ABAP Core Data Services (CDS), dem Business Object Processing Framework (BOPF) und sie machten SAP Gateway zu einem integralen Bestandteil von AS ABAP. Der nächste Schritt in diesem evolutionären Prozess war das ABAP-Programmiermodell für SAP Fiori, das mit ABAP-Release 7.50 eingeführt wurde (siehe voriger Abschnitt). Mit dem ABAP-Release 7.54 wurde der neueste Entwicklungsschritt, das ABAP RESTful Application Programming Model, zur Verfügung gestellt.

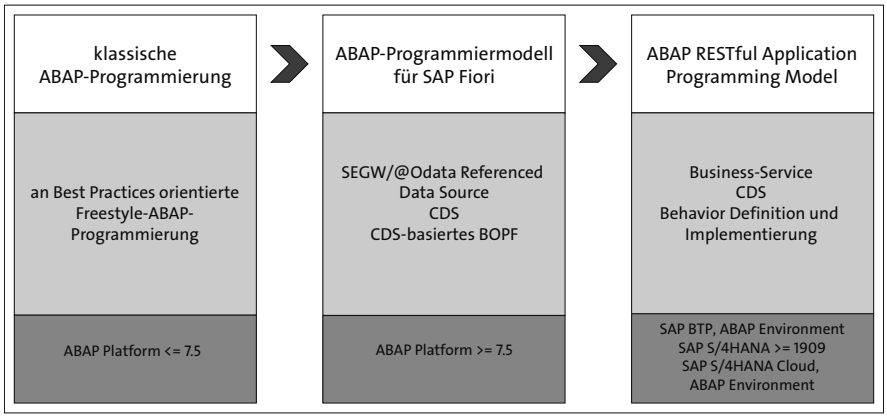

Abbildung 3.9 Evolution des ABAP-Programmiermodells

Das ABAP RESTful Application Programming Model definiert die Architektur für eine effiziente Ende-zu-Ende-Entwicklung von SAP-HANA-optimier-

ten OData-Services in der ABAP-Umgebung. Es basiert auf Technologien und Frameworks wie CDS zur Definition semantisch reicher Datenmodelle, einer Service-Modell-Infrastruktur zur Erstellung von OData-Diensten und ABAP-basierten Anwendungsdiensten für benutzerdefinierte Logik und SAPUI5-basierte Benutzeroberflächen.

Umgebungen

Das ABAP RESTful Application Programming Model ist generell in drei Umgebungen verfügbar, nämlich in der SAP Business Technology Platform, ABAP-Umgebung, in SAP S/4HANA ab der Version SAP S/4HANA 1909 und in der SAP S/4HANA Cloud, ABAP-Umgebung. Der Funktionsumfang in der SAP BTP, ABAP-Umgebung wird vierteljährlich an festgelegten Zeitpunkten erweitert, während neue Funktionen in SAP S/4HANA jährlich mit neuen Versionen geliefert werden.

Tools

ABAP Development Tools

Die Schlüsselwerkzeuge für das ABAP RESTful Application Programming Model sind die ABAP Development Tools in Eclipse (ADT), ABAP und CDS.

Die ADT in Eclipse sind eine integrierte Entwicklungsumgebung für ABAP-Entwickler. ADT basieren auf dem Eclipse-Framework, das in der Softwareindustrie weitverbreitet ist. Es handelt sich um eine leistungsstarke Werkzeugpalette, die darauf ausgelegt ist, den Entwicklungsprozess von ABAP-basierten Anwendungen und Diensten im SAP-Ökosystem zu optimieren und zu verbessern. Sie können ADT für das Schreiben von ABAP-Code, das Erstellen von RESTful-Services und das Verwalten Ihrer ABAP-Artefakte verwenden. Die ADT in Eclipse werden für alle möglichen Entwicklungsaufgaben eingesetzt und ermöglichen eine einfache Einarbeitung der Entwickler und einen Ende-zu-Ende-Entwicklungsfluss. Im Einzelnen bieten die ADT einen funktionsreichen Codeeditor, einen leistungsstarken Debugger, eine einfache Codenavigation, eine Versionskontrolle, ABAP-Git-Integration, einen Transportorganisator und am wichtigsten: eine nahtlose Integration mit SAP-Systemen, die es Entwicklern ermöglicht, direkt mit verschiedenen SAP-Objekten und Daten zu arbeiten.

CDS

ABAP ist die Grundlage für die Erstellung der Logik für die RESTful-Services. CDS sind eine Schlüsselkomponente des ABAP RESTful Application Programming Models. CDS sind ein Datenmodellierungs-Framework und eine domänenspezifische Sprache, die zur Definition und Verwaltung von Datenmodellen verwendet wird. CDS bieten Möglichkeiten, Datenstrukturen zu erstellen, Beziehungen zu definieren und Metadaten für Datenbanktabellen und Views anzureichern. Genauer gesagt bieten CDS eine deklarative Vorgehensweise, Datenstrukturen und Beziehungen zu definieren. Als Teil des ABAP RESTful Application Programming Models verwenden Entwickler

CDS Views, um Datenmodelle und Entitäten zu definieren, die als RESTful-Services freigegeben werden können.

CDS bieten eine Reihe von Vorteilen:

- **Datenmodellierung**
  CDS ermöglichen es, Datenmodelle auf deklarative Weise mit einer prägnanten Syntax zu definieren.
- **Wiederverwendbarkeit**
  CDS Views können in verschiedenen Anwendungen und Diensten wiederverwendet werden, was die Konsistenz beim Datenzugriff fördert und mögliche Redundanzen reduziert.
- **Leistungsoptimierung**
  CDS Views können für einen effizienten Datenbankzugriff optimiert werden. Das Framework bietet Funktionen wie Buffering, Indizierung und Abfrageoptimierung zur Leistungsverbesserung.
- **Datenannotationen**
  Mit CDS können Sie Annotationen verwenden, um die Metadaten von Datenmodellen zu bereichern. Annotationen liefern zusätzliche Informationen über die Daten, wie Labels, Beschreibungen und Formatierungsregeln.
- **Sicherheit**
  CDS Views können mit den Sicherheits- und Autorisierungsmechanismen von SAP integriert werden, um den Zugriff auf Daten auf einer granularen Ebene zu kontrollieren.
- **Integration mit SAP HANA**
  CDS Views sind eng mit SAP HANA integriert, was eine In-Memory-Verarbeitung und Optimierung von Abfragen beim Ausführen auf SAP-HANA-Datenbanken ermöglicht.

Darüber hinaus gibt es zum ABAP RESTful Application Programming Model eine Reihe von Entwicklungsleitfäden und Best Practices, die die Einarbeitung der Entwickler erleichtern. Leistungsfähige Frameworks helfen dabei, diese Best Practices hinsichtlich der technischen Implementierungsaufgaben zu erfüllen und Geschäftslogik in Code-Exits auf protokollagnostischen Schichten hinzuzufügen.

Best Practices

Einige der Best Practices für die Entwicklung mit dem ABAP RESTful Application Programming Model sehen wie folgt aus:

- Verwenden Sie CDS Views zur Definition des Datenmodells und nutzen Sie CDS-Funktionen wie Datenaggregation, Filterung und Calculated Fields für eine effiziente Datenabfrage.

- Verwenden Sie Annotationen klug, nicht zu viele und nicht zu wenige, um zusätzliche Metadaten für Ihr Datenmodell bereitzustellen. Vermeiden Sie gleichzeitig eine zu extreme Verwendung, da übermäßige Annotationen die Leistung beeinträchtigen können.
- Verwenden Sie konsistente Namen für Entitäten, Attribute und Operationen, um die Lesbarkeit Ihres Codes und Ihrer APIs zu verbessern.

Implementierungstypen und -dienste

Implementierungstypen

Das ABAP RESTful Application Programming Model unterstützt zwei Implementierungstypen, Greenfield- und Brownfield-Entwicklung. Die Entwicklung von OData-Diensten von Grund auf, sogenannte *Greenfield-Entwicklung*, wird über den sogenannten *Managed-Implementierungstyp* unterstützt. Die Entwicklung eines OData-Service basierend auf vorhandenem Code, *Brownfield-Entwicklung*, wird über den sogenannten *Unmanaged-Implementierungstyp* unterstützt.

Kategorien der Services

Das ABAP RESTful Application Programming Model ermöglicht die Entwicklung verschiedener Arten von Services, die wie folgt kategorisiert werden können:

- OData-basierte Services für die UI-Entwicklung zum Erstellen von ansprechenden, rollenbasierten und responsiven SAP-Fiori-Apps
- OData-basierte Services zur Bereitstellung als Web-APIs
- InA-basierte, analytische Services zur Erstellung von analytischen Apps in der SAP Analytics Cloud
- SQL-basierte Services zur Datenintegration

Vorteile des ABAP RESTful Application Programming Models

Im Vergleich zu seinem Vorgänger, dem ABAP-Programmiermodell für SAP Fiori, oder speziell der codebasierten Implementierung mit SEGW bietet das ABAP RESTful Application Programming Model ein Programmiermodell, das eine Reihe von Vorteilen gegenüber diesen beiden alternativen Optionen bietet.

Zunächst einmal ist das ABAP RESTful Application Programming Model eine sichere Option für eine strategische langfristige Lösung für die ABAP-Entwicklung. Investitionen in dieses Programmiermodell werden sich als langfristig nutzbar und langfristig unterstützt erweisen.

Wissensaufbau zum ABAP RESTful Application Programming Model

Der Aufbau von Wissen über das ABAP RESTful Application Programming Model ist ebenfalls eine gute Investition, da es eine Ende-zu-Ende-Entwicklungserfahrung mit einer standardisierten Architektur und Entwicklungsfluss bietet. Dies ermöglicht es, den Ansatz immer wieder für weitere Entwicklungen zu wiederholen – und dabei immer wieder den Vorteil der ursprünglichen Investition in den Wissenssaufbau zu nutzen. Wir empfehlen Ihnen dafür das Buch »ABAP RESTful Application Programming Model« (*www.sap-press.de/5869*).

Was dieser standardisierte Prozess ebenfalls mit sich bringt, ist eine hohe Entwicklungseffizienz, weil der Prozess hoch optimiert ist und weil Entwickler lernen, dem Prozess immer wieder zu folgen. Eine Vielzahl von Entwicklungsleitfäden und äußerst vorteilhaften Best-Practice-Empfehlungen helfen, den Entwicklungsprozess weiter zu optimieren. Die eingebaute Testbarkeit, Dokumentierbarkeit und Unterstützbarkeit, die mit dem ABAP RESTful Application Programming Model einhergehen, sind weitere wichtige Faktoren, die zu berücksichtigen sind.

Standardisierung und Flexibilität

Bisher haben wir über die Vorteile gesprochen, die der standardisierte Prozess mit sich bringt. Zur Ergänzung des standardisierten Prozesses bietet das ABAP RESTful Application Programming Model auch eine gewisse Flexibilität, die Abweichungen für nicht standardisierte Implementierungen ermöglicht. Es vereint Standardisierung und Flexibilität.

Die Benutzererfahrung, die das ABAP RESTful Application Programming Model bietet, basiert auf SAP Fiori und SAP HANA und ist, so wie die resultierende Software, von Grund auf benutzerfreundlich.

Cloud Native

Da das ABAP RESTful Application Programming Model in gewissem Maße Cloud Native ist, bietet es einige Cloud-Qualitäten wie sehr gute Skalierbarkeit, Out-of-the-Box-Erweiterbarkeit und Vertikalisierung.

Lassen Sie uns nun speziell einige sehr konkrete Vorteile betrachten, die das ABAP RESTful Application Programming Model gegenüber seinem Vorgänger bietet:

- Es eignet sich auch für Brownfield-Ansätze und nicht nur für Greenfield-Ansätze wie das ABAP-Programmierungsmodell für SAP Fiori.
- Das ABAP RESTful Application Programming Model ist viel besser für die Fehleranalyse geeignet als das ABAP-Programmiermodell für SAP Fiori. Dies resultiert aus der Architektur des ABAP-Programmiermodells für SAP Fiori, in der viele Artefakte in verschiedenen Schichten generiert werden.

- Das ABAP-Programmiermodell für SAP Fiori kann nicht mit den OData-Funktionen des ABAP RESTful Application Programming Models konkurrieren. Es fehlt an Unterstützung für OData 4.0 und infolgedessen an asynchronen OData-V4.0Anfragen.

Ready-to-use-Beispiele für Best Practices des ABAP RESTful Application Programming Models

Für das ABAP RESTful Application Programming Model gibt es eine Reihe von Best Practices, die die Einarbeitung für Entwickler erleichtern. Diese Best Practices werden als fertige Beispiele auf Basis des ABAP-Flight-Reference-Szenarios bereitgestellt, einer überarbeiteten und angepassten Version des bekannten SAP-NetWeaver-Flight-Datenmodells, das für Schulungs- und Demozwecke geeignet ist. Das ABAP-Flight-Reference-Szenario ist standardmäßig verfügbar und kann in der kostenlosen Testversion der SAP BTP, ABAP-Umgebung verwendet werden.

Weitere Informationen finden Sie hier: *http://s-prs.de/v9808014*

Entwicklungsablauf

Drei Phasen

Der Hauptentwicklungsablauf kann in drei Phasen gegliedert werden, (siehe Abbildung 3.10): Datenmodellierung und Verhaltensdefinition, Business-Service-Exponierung und schließlich Konsumierung des Service.

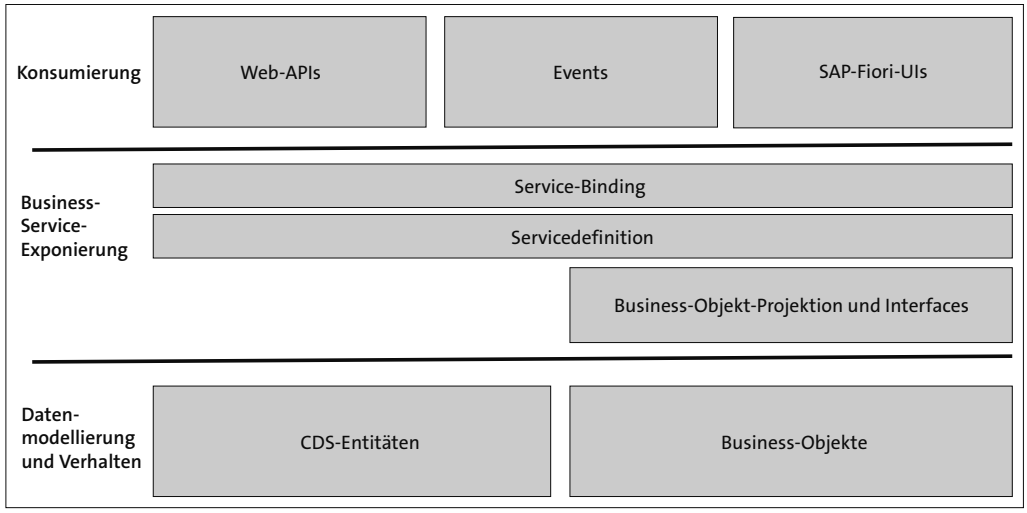

Abbildung 3.10 Phasen und ihre Artefakte als Teil des Entwicklungsablaufs im ABAP RESTful Application Programming Model

Dieser Ablauf hat den Vorteil, dass eine klare Struktur geboten wird und der Entwicklungsprozess gleichzeitig sehr flexibel gestaltet ist. Wenn eine Änderung benötigt wird, z. B. ein Service mit einem zusätzlichen Feld, das freigegeben werden muss, müssen Sie nicht von vorn beginnen. Sie können einfach auf bereits vorhandenen Artefakten aufbauen. Anders ausgedrückt können Sie einen anderen Serviceentwicklungsprozess, basierend auf bereits erstellten Artefakten, starten.

Werfen wir nun einen genaueren Blick auf die Artefakte, die auf dem Weg zur Bereitstellung eines OData-Dienstes erstellt werden, und auf die Phasen, in denen sie erstellt werden (siehe Abbildung 3.11).

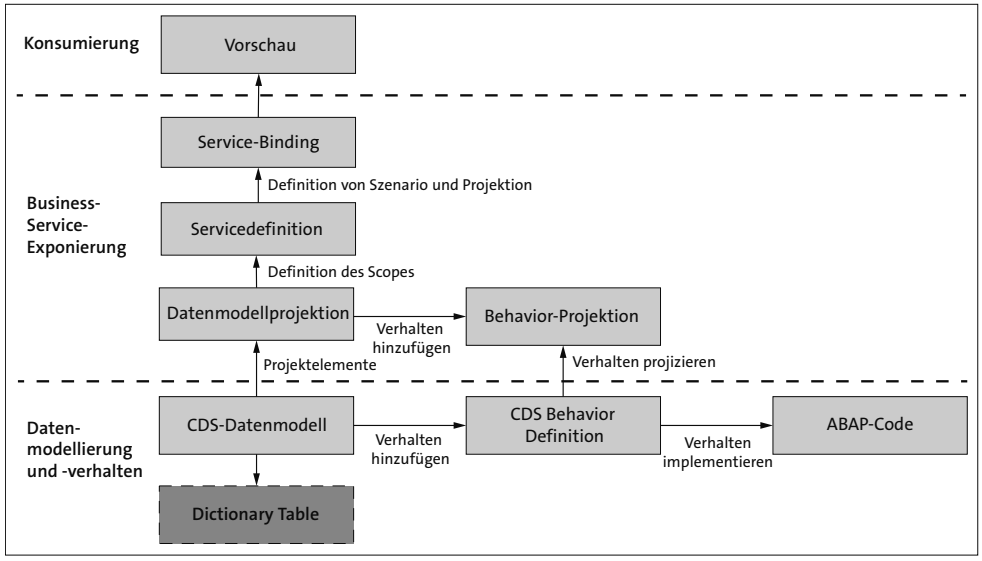

Abbildung 3.11 Detaillierte Ansicht der Artefakte während des Entwicklungsprozesses (Quelle: SAP)

Schauen wir uns jetzt die Phasen und die Artefakte, die in den Phasen erstellt und bearbeitet werden, genauer an. Wir werden dies entlang des Entwicklungsablaufs tun. Das heißt, wir starten mit der Daten- und Verhaltensmodellierung. Hierzu wird CDS genutzt. Dann beschreiben wir mit der CDS Behavior Definition, was mit dem Datenmodell gemacht werden kann. Wir exponieren einen auf der Daten- und Verhaltensmodellierung basierenden Business-Service und konsumieren diesen letztendlich.

3.5.2 Datenmodellierung und Verhalten

Das Datenmodell umfasst die Beschreibung der verschiedenen Entitäten, die ein Business-Scenario beinhaltet, sowie ihrer Beziehungen. Die Defini-

tion des Datenmodells findet in der Datenmodellierungs- und Verhaltensdefinitionsschicht statt, die sich mit den Daten und der entsprechenden Geschäftslogik befasst.

Datenmodell

Das ABAP RESTful Application Programming Model verwendet CDS, um das Datenmodell zu definieren und zu organisieren. CDS sind die grundlegenden Bausteine einer Anwendung. Jede Entität wird durch eine CDS-Entität repräsentiert. Für das Datenmodell gib es eine physische Darstellung in der Datenbank in Form von Datenbankansichten, die automatisch auf Basis eines CDS-Datenmodells erstellt werden. Je nach Anwendungsfall unterstützen Datenmodelle den transaktionalen Zugriff oder den Abfragezugriff auf die Datenbank.

Im Detail sehen die Artefakte als Teil dieser Schicht von unten nach oben so aus:

- Die Datenbankebene wird durch die Definition von Dictionary Tables definiert.
- Die Dictionary Tables bilden die Grundlage des Datenmodells.
- Das semantische Datenmodell wird dann durch CDS definiert. CDS sind Ansichten auf den Dictionary Tables. Daten, die in der Datenbank gespeichert sind, können in dieser CDS-zentrierten Phase verwendet und manipuliert werden.

CDS Behavior

Der zweite Teil dieser Phase konzentriert sich auf das CDS Behavior. Das Verhalten beschreibt, was mit dem Datenmodell gemacht werden kann. In Transaktionsszenarien definiert das Business Object Behavior, welche Operationen und welche Merkmale zu einem Geschäftsobjekt gehören. In schreibgeschützten Szenarien wird das Verhalten des Datenmodells durch die Abfragefähigkeiten definiert, z. B., ob die Daten filterbar sind. Das CDS-Verhalten wird in zwei Teilen beschrieben:

- Die Verhaltensdefinition bestimmt die `Create`-, `Update`- und `Delete`-Funktionalität.
- Die Verhaltensimplementierung liefert dann die Implementierung des Verhaltens. Im verwalteten Ansatz (Greenfield-Implementierung) wird die Implementierung von `Create`, `Update` und `Delete` automatisch durchgeführt.

3.5.3 Business-Service-Exponierung

Im Kontext des ABAP RESTful Application Programming Models ist ein Business-Service ein RESTful-Service, der von einem Verbraucher aufgerufen werden kann. Er wird definiert, indem sein Datenmodell zusammen mit

dem zugehörigen Verhalten freigegeben wird. Der Service besteht aus einer Servicedefinition und einer Servicebindung:

CDS-Projektion

- Die CDS-Projektion ist eine Teilmenge der Felder des zugrunde liegenden Datenmodells, die für die Anwendung relevant sind.
- Die Servicedefinition legt dar, welche Daten als Business-Service freigegeben werden.
- Service-Bindings ermöglichen es, Servicedefinitionen an ein Client-Server-Kommunikationsprotokoll wie OData zu binden.

3.5.4 Konsumierung

Der Service kann nun als UI-Service freigegeben werden, der von einer SAP-Fiori-Elements-App genutzt wird. Alternativ kann er als Web-API freigegeben werden. Ein OData-Service, der als Web-API freigegeben wird, besitzt keinerlei UI-spezifische Informationen in den Metadaten. Es ist die öffentliche Schnittstelle für jeden OData-Client, über die auf den OData-Service zugegriffen werden kann.

ABAP-RESTful-Application-Programming-Model-Events ermöglichen die asynchrone Nutzung von ABAP-RESTful-Application-Programming-Model-Geschäftsobjekten. Dies ist eine neue Entwicklung mit viel Potenzial. Diese Events erlauben die Erstellung von benutzerdefinierten Ereignissen, entweder von Grund auf oder indem vorhandene SAP-Standardereignisse erweitert werden, für asynchrone Echtzeiterweiterungs- und -integrationsszenarien.

3.6 Zusammenfassung

Die klassische Drei-Schichten-Architektur von SAP Gateway folgt den Gateway-Prinzipien sowie den Prinzipien zeitloser Software. Sie bietet die für vielfältige Einsatzgebiete notwendige Flexibilität von mobilen Szenarien über Desktop-Applikationen bis hin zu Internetanwendungsfällen. Zugleich bietet sie auch die für geschäftskritische Anwendungsfälle notwendige Stabilität und Verlässlichkeit. Die Hauptschicht der Architektur ist die SAP-Gateway-Schicht, in der die meisten Funktionalitäten und Werkzeuge zu finden sind. Teile von SAP Gateway können sich zudem in der SAP-Backend-Schicht befinden.

SAP Gateway stellt eine Menge Werkzeuge bereit, die die einfache Erstellung von SAP-Gateway-OData-Services ermöglichen. Dafür können auch bereits vorhandene, über andere SAP-Technologien zugreifbare Inhalte (Content)

verwendet werden. Dies beinhaltet eine Reihe von SAP-Standardtechnologien wie RFC, BOR, SPI, CDS und GenIL.

Nutzbar für die Entwicklung für SAP S/4HANA ab SAP ABAP Stack 7.50 SP01 steht das ABAP-Programmiermodell für SAP Fiori für eine vollkommen neue Ende-zu-Ende-Herangehensweise und ermöglicht Softwareentwicklung vom Frontend bis zum Backend. Dabei folgt das ABAP-Programmiermodell für SAP Fiori innovativen Konzepten mit einem ganz besonderen Fokus auf eine moderne Nutzererfahrung und State-of-the-Art-Cloud-Qualitäten.

Das ABAP-Programmiermodell für SAP Fiori baut auf Technologien wie beispielsweise SAPUI5 und SAP Fiori Elements, SAP Gateway, CDS, BOPF und SAP HANA auf und ist nicht nur für die Nutzung on premise entwickelt worden, sondern hat einen sehr starken Cloud-Fokus.

Das ABAP RESTful Application Programming Model baut auf Technologien wie SAPUI5 und SAP Fiori Elements, SAP Gateway, CDS, BOPF und SAP HANA auf und wurde nicht nur für die On-Premise-Nutzung entwickelt, sondern mit einem speziellen Fokus auf die Cloud.

Im folgenden Kapitel 4 werden wir Teil I des Buches mit einer Diskussion der Deployment-Optionen von SAP Gateway, ihrer Installation und Konfiguration abschließen.