

# Virtuelle Datenmodelle im SAP und die Verwendung in modernen Schnittstellen

Wissenswertes für SAP Kunden und (Non-SAP-)Dienstleister

Magdeburger Developer Days 2022

Jörg Müller (@mdjoerg)

17.05.2022



# Zur Person: Jörg Müller (@MDJoerg)



## Persönlicher Kontext

- Gebürtiger Magdeburger, Rückkehrer (2004)
- Heute wohnhaft im Sülzetal (bei Magdeburg)
- Verheiratet, 1 Kind
- Erster Wirtschaftsinformatiker an der „Otto-von-Guericke Universität“ (OvGU WIF93)
- SAP Erfahrung seit 1996
- Davon ca. 10 Jahre selbständig
- Regionales Engagement:
  - Organisator „SAP Stammtisch Magdeburg“
  - Kontakte zur regionalen IT Community
  - (Schul-)Digitalisierung

## Beruflicher Kontext

- „Innovation Manager“ bei BA Business Advice GmbH, Oldenburg, SAP Beratung, 70 MA
- Rollen: Berater, Entwickler, Vertrieb, Team- und Projektleitung, GF, IT Betreuer ...
- Schwerpunkte: Vertrieb und Marketing, Systemintegration/Schnittstellen, Datenanalyse
- Technische Projektleitung in SAP Einführungsprojekten (S/4 HANA)
- Mitarbeiterausbildung und fachliche Führung
- Lösungsentwicklung und Prototyping



SAP Stammtisch Magdeburg

<https://sapstammtisch.github.io/Magdeburg>

# An SAP andocken ...

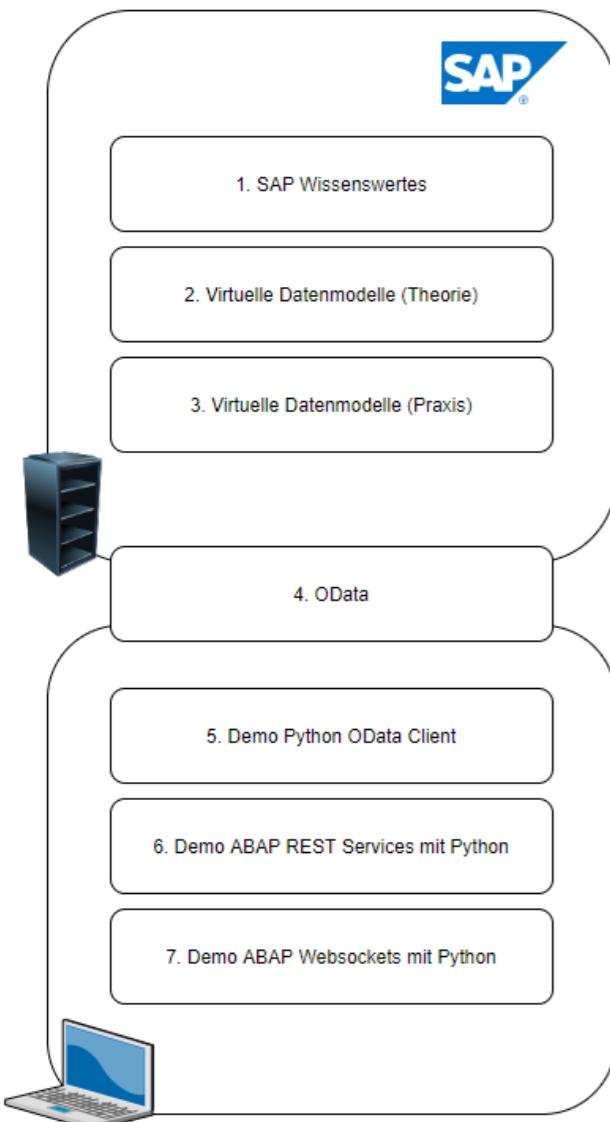


*In einer idealen SAP-Welt  
setzen alle SAP Kunden nur noch aktuellste Technologien ein, die sie natürlich beherrschen,  
alle benötigten Schnittstellen wären vorhanden und dokumentiert  
und man könnte verschiedene Systeme ohne Entwicklung  
auf Knopfdruck miteinander integrieren...*

*... In der Realität sind viele SAP Systeme > 10 Jahre in Betrieb,  
technisch nicht auf dem neuesten Stand,  
existieren Rahmenbedingungen, denen man nicht so einfach ausweichen kann,  
sind die wichtigsten Integrationstechnologien EDIFACT/IDocs, BAPI/RFC, SOAP und Batch-Input,...  
womit das Non-SAP-Umfeld eher nichts anfangen kann.*

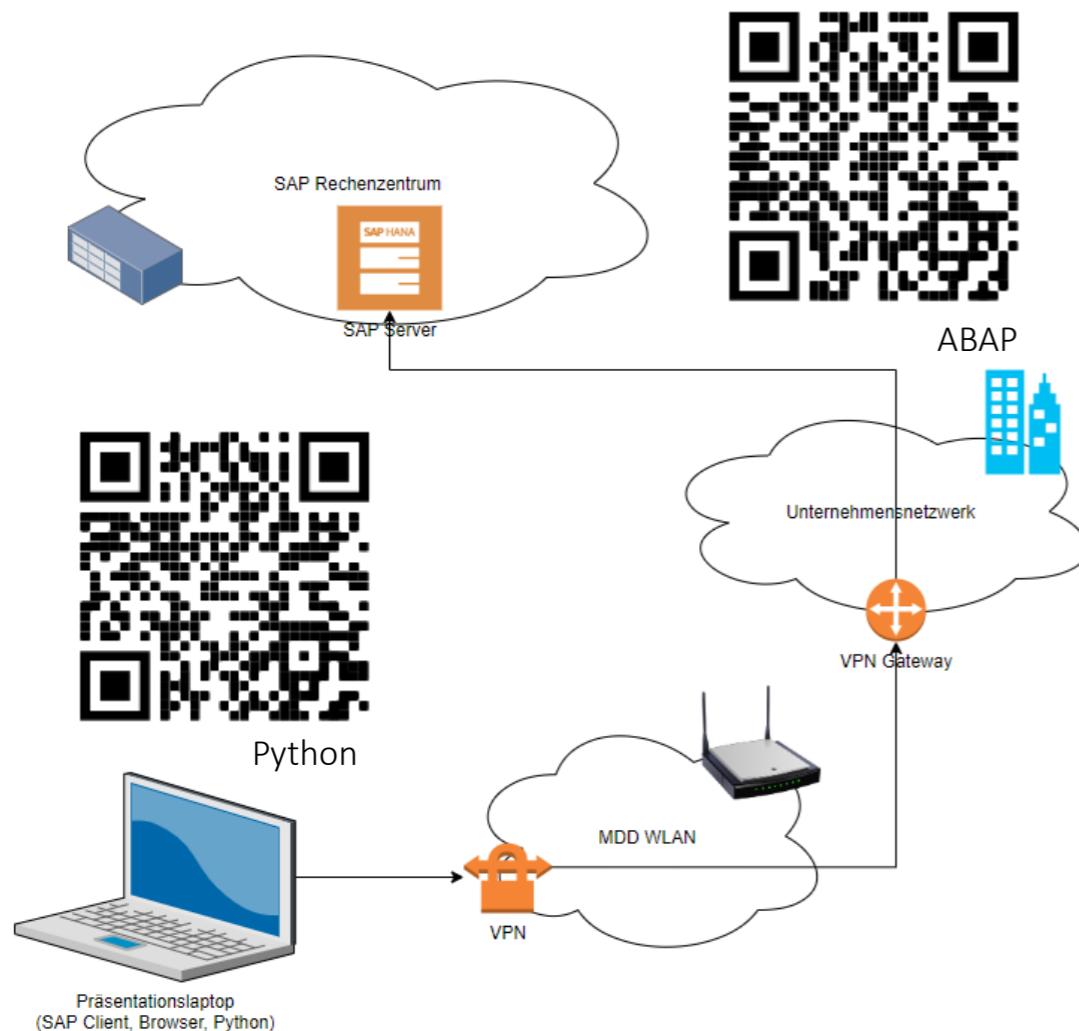
*Dieser Vortrag zeigt Kompromisse, wie selbst alte SAP Systeme Kontakt mit der neuen Welt aufnehmen können.*

# Was könnt Ihr heute erwarten?



- Der Vortrag wurde bewusst so zusammengestellt, dass auch Neugierige **ohne SAP Kenntnisse** etwas mitnehmen können
- Auf einer Entwicklerkonferenz wird entwickelt und Code gezeigt: damit Ihr Euch den **Code** im Nachgang noch einmal anschauen könnt, gibt es alles auch als **Github** Repository
- Ein Schwerpunkt des Vortrags liegt auf **Demos in Python**, als Kompromiss auf einer Entwicklerkonferenz mit Schwerpunkt auf Microsoft und Java ;-)
- Die vorgestellten (modernen) **Technologien** wurden so ausgewählt, dass gerade Non-SAP-Entwickler Anregungen für die oft **beim Kunden anzutreffenden Landschaften** mitnehmen können
- Vieles, was SAP noch so könnte und wo die SAP die Schwerpunkte sieht, wird nur kurz erwähnt oder weggelassen...

# Demolandschaft und Github Repositories



- Die **Demolandschaft** entspricht einer typischen Umgebung, wie sie häufig in SAP Projekten zu finden ist:
  - Der Endanwender ist per **VPN** mit dem Unternehmen verbunden
  - Dort wird es über eine **sichere Verbindung** zum **SAP System** weiter geroutet
  - Das **SAP System** steht in einem separaten **Rechenzentrum**
  - Es ist – wie hier – häufig unmöglich, direkt **vom SAP Rechenzentrum in das Netz des Kunden** zu gelangen  
→ Herausforderung für **Schnittstellen**, die vom SAP ausgehen
- **Github Repositories:**
  - [https://github.com/MDJoerg/mdd22\\_python](https://github.com/MDJoerg/mdd22_python)
  - [https://github.com/MDJoerg/mdd22\\_abap](https://github.com/MDJoerg/mdd22_abap)
- Diese **Folien**  
→ gibt es im Github Respository mdd22\_python nach dieser Veranstaltung

# SAP Wissen Historie & Eigenheiten

Was man wissen und beachten sollte...



# 50 Jahre SAP



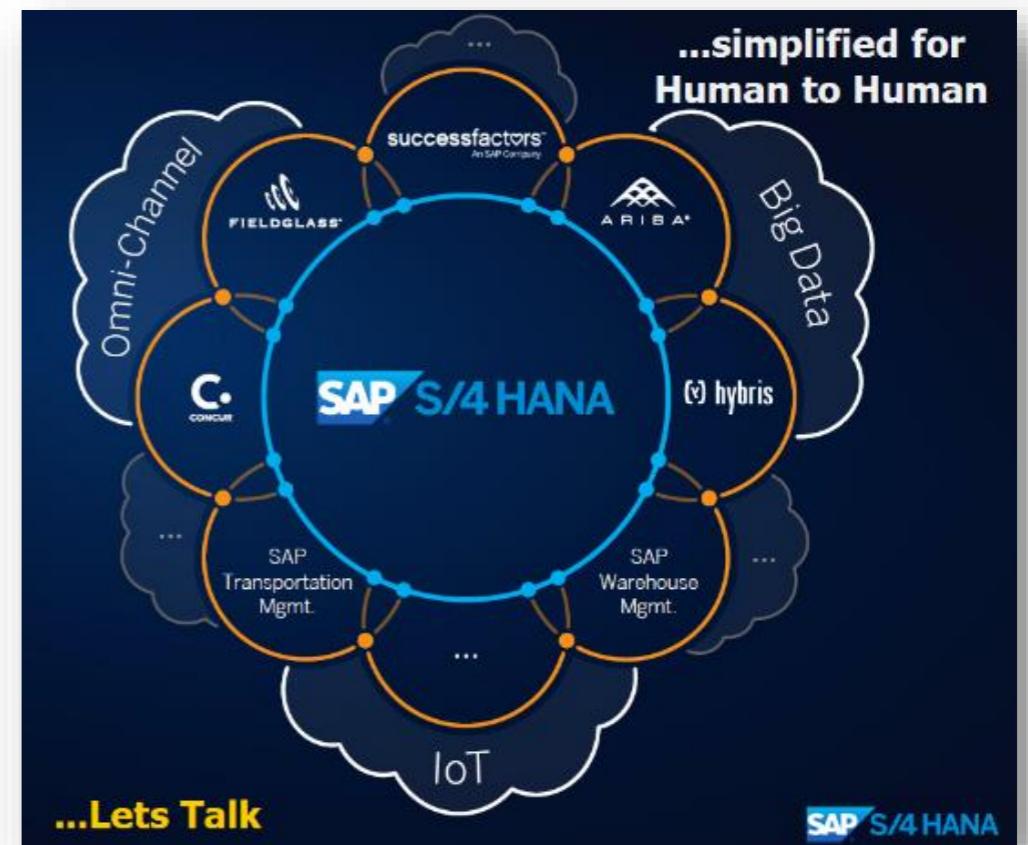
- SAP ist ein bekannter **deutscher Softwarehersteller** und im DAX derzeit mit ca. 8,5% Anteil gelistet
- Das Unternehmen wurde **1972** gegründet
- Der Sitz ist **Walldorf** (bei Heidelberg)
- SAP ist weltweit **größter Anbieter von Unternehmensanwendungen**
- Das bekannteste und wichtigste Produkt ist **SAP R/3 bzw. SAP ERP**
- Die derzeit aktuelle ERP Produktlinie ist „**SAP S/4 HANA**“ basierend auf der Datenbank „**SAP HANA**“
- Mit der **Business Technology Plattform (BTP)** tritt SAP inzwischen als **Cloud Anbieter** auf und bietet weitere cloudbasierte Services und Lösungen
- Die wichtigste Programmiersprache ist **ABAP**, weitere sind: Javascript, Java, Python, ...
- 50 Jahre sind eine lange Zeit und hinterlassen hier und da **historische Spuren...**
  
- Quellen:
  - <https://de.wikipedia.org/wiki/DAX>
  - <https://de.wikipedia.org/wiki/SAP>
  - <https://de.statista.com/themen/232/sap/#dossierKeyfigures>
  - <https://assets.cdn.sap.com/sapcom/docs/2017/04/16b2e4dd-b67c-0010-82c7-eda71af511fa.pdf>

UMSATZ VON SAP WELTWEIT	27,84 Mrd. €
NETTOGEWINN VON SAP	5,26 Mrd. €
ANZAHL DER BESCHÄFTIGTEN VON SAP WELTWEIT	107.415



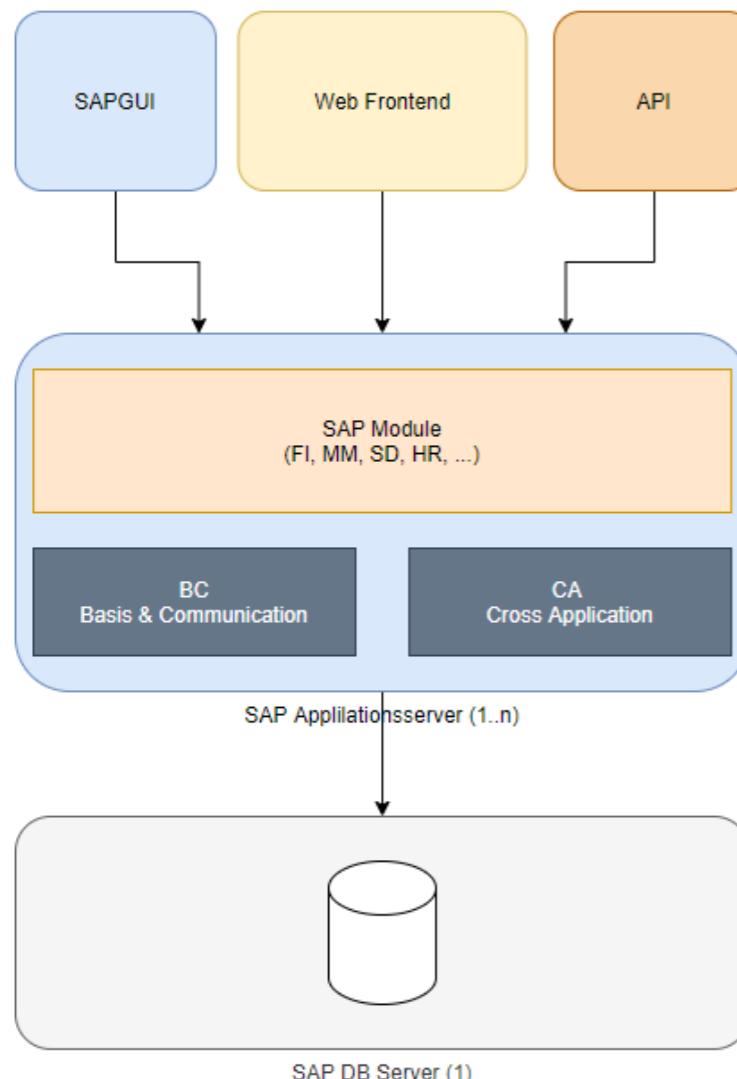
# SAP's Kernprodukt – Enterprise Ressource Planning (ERP)

BA

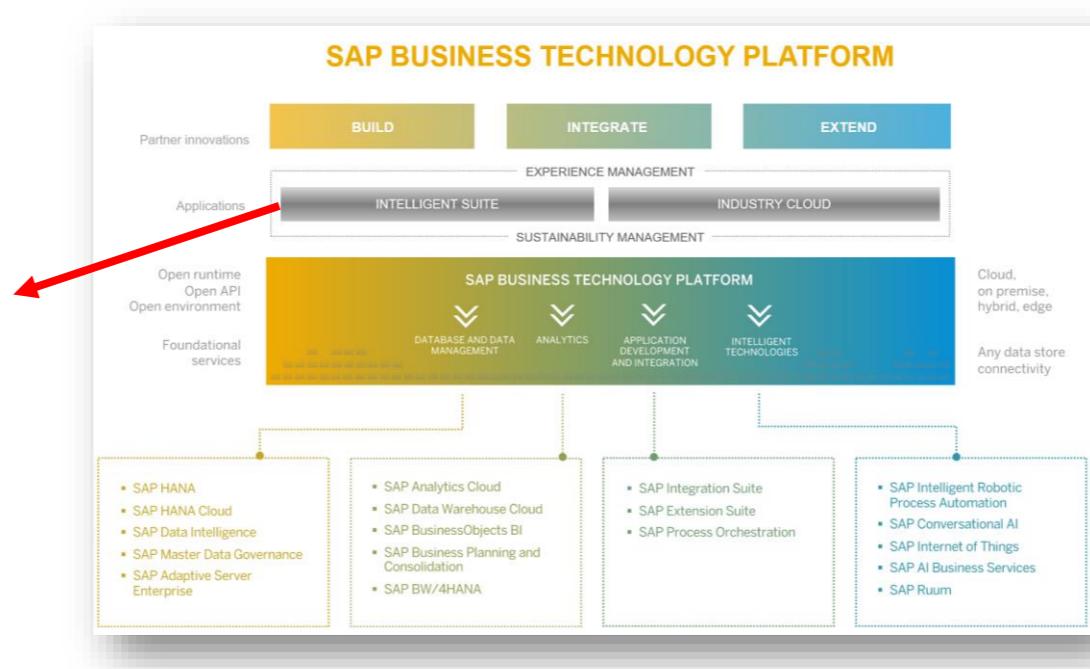


- Für S/4 HANA ist auch der Name „Digital Core“ geläufig → die Kunden sehen Ihre SAP ERP bzw. S/4 Systeme oft als „führendes System“
- Um diesen Digital Core wird dann „außen angebaut“ → am besten mit der SAP BTP
- Die meisten SAP Systeme in Europa stehen im „Keller“ der Kunden oder eines Dienstleisters

# SAP NetWeaver ABAP Systemarchitektur



SAP NetWeaver ABAP Architektur  
(z.B. als Basis für SAP S/4 HANA)



- Die SAP ERP und S/4 HANA sind Teil der „Intelligent Suite“ aus Sicht der SAP BTP
- Der SAP NetWeaver ABAP ist die Basis für die meisten SAP Lösungen, die nicht für die SAP Cloud konzipiert wurden
- Das „Herzstück“ sind 1 bis mehrere SAP Applikationsserver und genau ein Datenbank Server
- Im SAP Applikationsserver sind immer bestimmte Grundfunktionen (z.B. Data Dictionary, Entwicklungsumgebung, Administration, Kommunikation, Schnittstellen) enthalten
- Durch das Einspielen weiterer Softwarekomponenten (z.B. SAP ERP Module) entsteht erst eine SAP Lösung (meistens On-Premise = steht beim Kunden oder Dienstleister)
- Ein Zugriff auf SAP muss immer über den Applikationsserver erfolgen, obwohl der technische Zugriff direkt auf die Datenbank technisch möglich wäre!

# Persona: „SAP Entwickler“

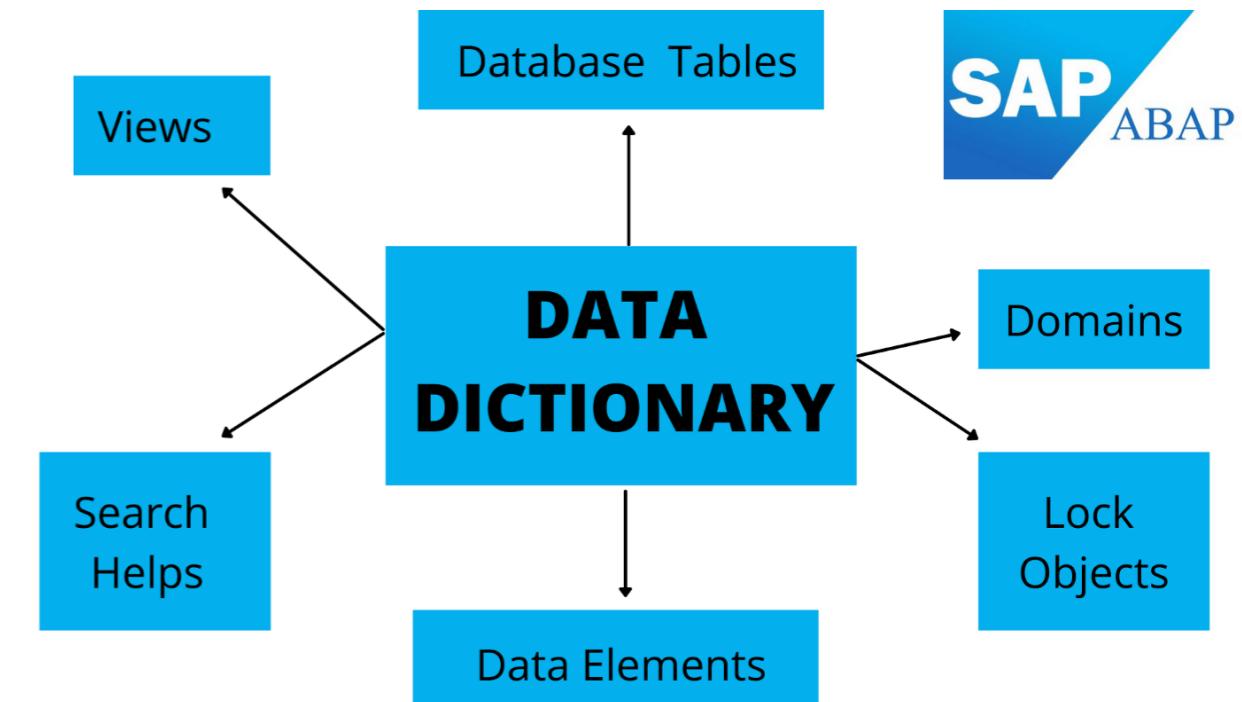


- Im SAP Beratungsgeschäft werden die **Rollen „Berater“ und „Entwickler“** unterschieden
- Zusätzlich betreuen SAP Basis **Administratoren** die „Technik“ und ggf. die „Infrastruktur“ die Plattform/Netzwerk
- **Berater sind in der Kommunikation mit dem Endanwender** arbeiten dann ggf. zusammen mit dem Entwickler an einer Lösung
- Die klassische **SAP Software** ist hochgradig **konfigurierbar** → viele Funktionen werden vom Berater „freigeschaltet“ und konfiguriert
- Aufgabe des Entwickler ist oft die **Erweiterung oder Anpassung der SAP Geschäftsprozesse** und keine Lösungsentwicklung mit einem Neustart (Rahmenbedingungen und Kontext sind vorgegeben)
- Viele hybride „beratende Entwickler“ bzw. „entwickelnde Berater“ sind anzutreffen und **fachliche Kompetenzen** sind wichtiger als Skills der Entwicklung (und Informatik)
- Die Entwicklung zum Cloud-Anbieter mit vielen neuen technischen Rahmenbedingungen ist eine große Herausforderung für die „Transformation“ der SAP Berater und Entwickler (z.B. „DevOps“)
- Die **Tools aus dem Non-SAP-Bereich** werden immer mehr auch im SAP Umfeld eingesetzt, allerdings ist eine wahrnehmbare Durchdringung noch lange nicht gegeben
- Die „**neue Welt**“ ist an vielen Stellen bereits nutzbar, wenn man weiß wie und wo...

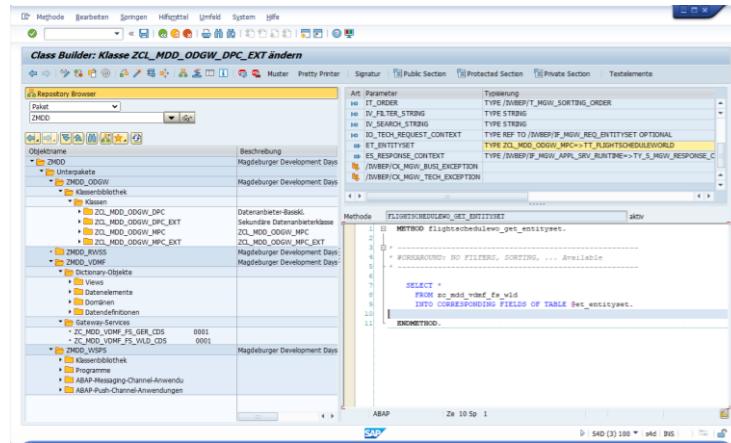
# SAP Entwickler arbeiten nicht direkt auf der Datenbank



- Das klassische SAP (R/3 und ERP) waren für **verschiedenen Plattformen** (z.B. Linux, Unix, AIX, Windows, ...) und **Datenbanken** (z.B. MS SQL, Oracle, DB/2, MaxDB, ...) ausgelegt
- Die **ABAP Programmierung** sollte aber **datenbankunabhängig** erfolgen, so dass auch ein Wechsel der Datenbank ermöglicht wird (z.B. notwendig bei der Informix Datenbank)
- Deshalb bietet der SAP Applikationsserver eine **Abstraktionsschicht** auf die tatsächliche Datenbank und für Entwickler entsprechende DB Tools (**Data Dictionary**)
- Diese Abstraktionsschicht ermöglicht auch die **Vererbung von Datenbankfragmenten** und nimmt den Entwicklern Arbeit bei der Generierung von physischen Datenbankobjekten ab (z.B. kein ALTER TABLE)
- Die Sprache ABAP bietet mit **OpenSQL** datenbanknahe Befehle, die nahe am Standard SQL sind
- Weitere Data Dictionary Objekte erleichtern die tägliche Arbeit bei der Entwicklung von **Geschäftsanwendungen**
- Der direkte Zugriff auf Daten über den Datenbankserver (z.B. über JDBC Treiber) ist ungewöhnlich und in den meisten Fällen sogar untersagt!



# SAP Entwicklungstools



SAP Web IDE



SAP Business Application Studio

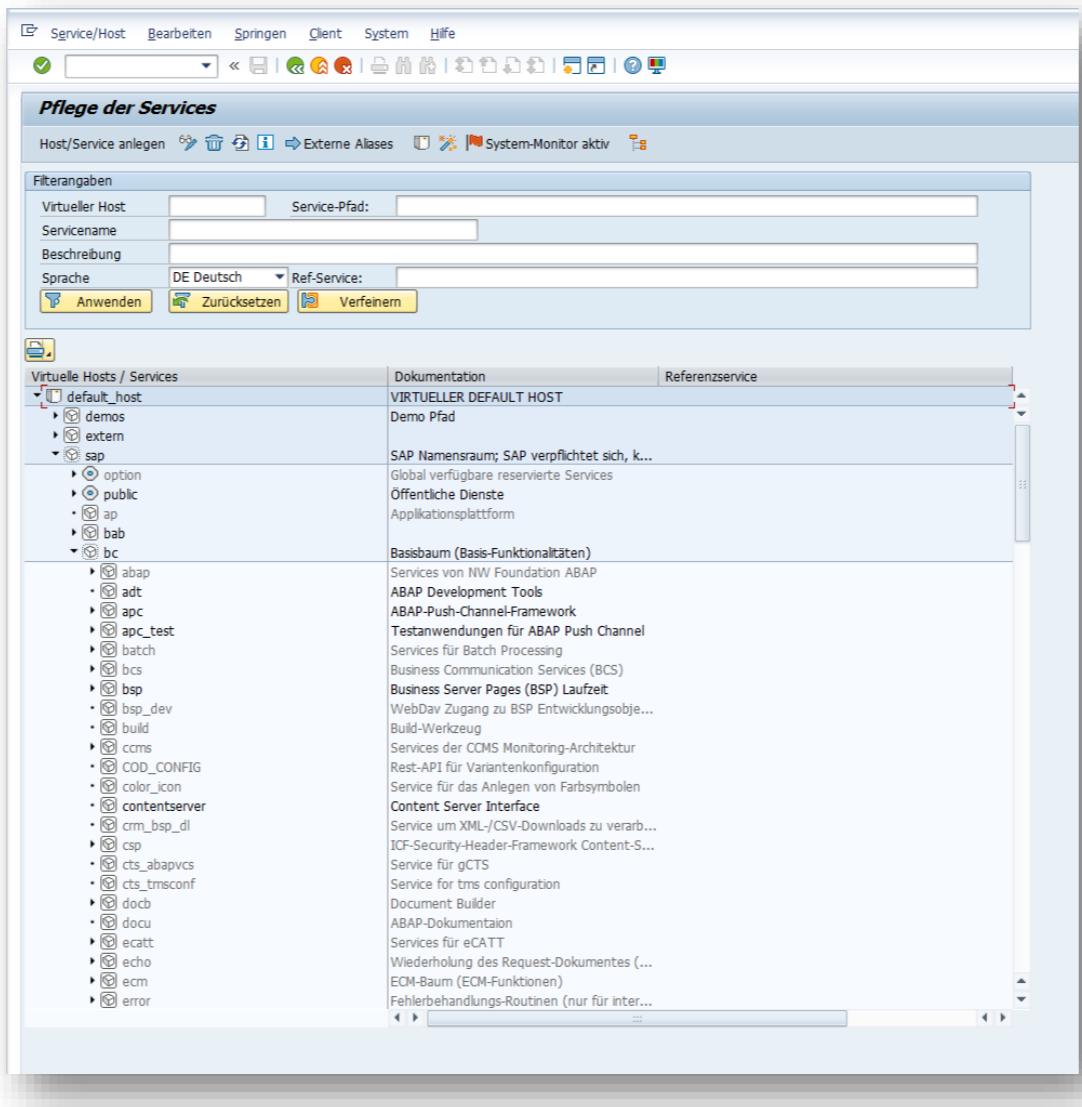
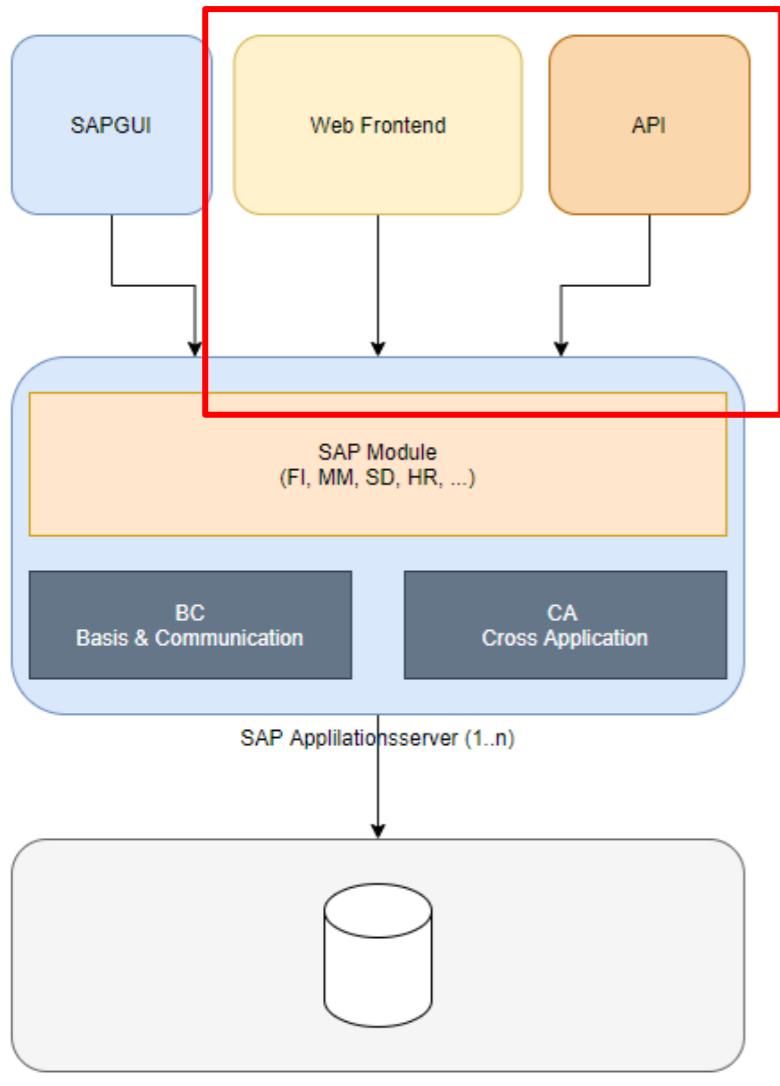


Visual Studio Code



- Im klassischen SAP waren alle Entwicklungstools Bestandteil des Applikationsservers und über SAPGUI erreichbar (Programmiersprache ABAP)
- In den letzten Jahren kamen die „ABAP Development Tools for Eclipse“ (ADT) dazu → für einige Entwicklungsobjekte inzwischen sogar erforderlich
- Vor allem die **Frontend-Entwicklung** (Javascript, FIORI/SAPUI5) verwendet die Cloud-basierten Tools „**WebIDE**“ (veraltet) und „**Business Application Studio**“ (**BAS**)
- Visual Studio Code (**VSC**) mit entsprechenden Plugins wird bei „modernen“ SAP Entwicklern immer beliebter und ebenfalls für die Frontend-Entwicklung eingesetzt
- Weitere Tools wie **abapGit** bringen moderne Entwicklungstools immer mehr auch ins SAP Backend und den Austausch von Code in Gang (siehe <https://abapgit.org> , <https://dotabap.org> )
- **Weitere Tools** und Programmiersprachen sind vor allem bei Schnittstellen und webbasierten Lösungen möglich
- Python bekommt gerade etwas „Rückenwind“ durch Schulungen bei openSAP/openHPI ...

# Moderne SAP Server sind wie Webserver ...

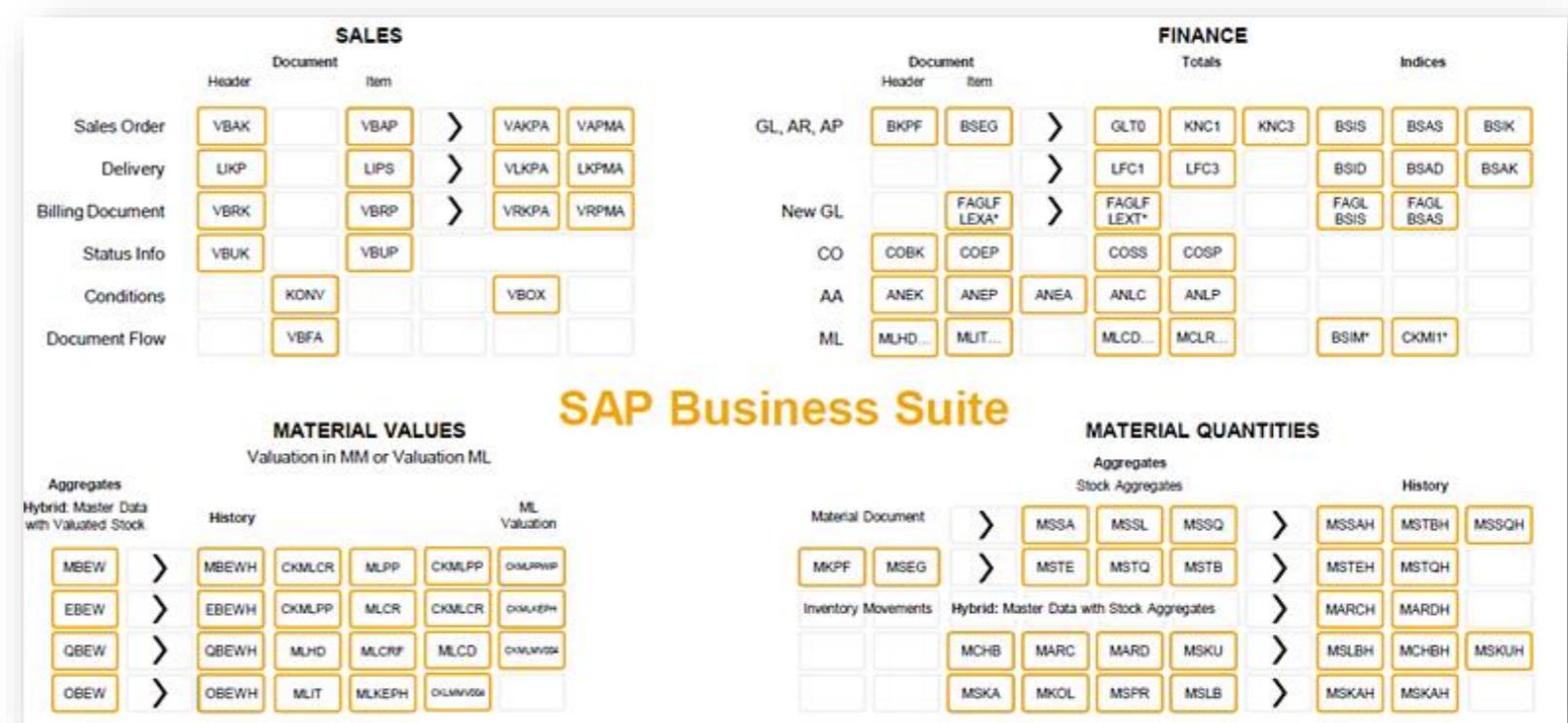


- Der SAP NetWeaver Stack enthält seit vielen Jahren den „Internet Communication Manager“ Layer (ICM), mit dem ein ABAP Server auf verschiedene Weise mit dem Internet kommunizieren kann
- Über die in Pfaden pflegbaren Services (**Transaktion SICF**) lassen sich internetbasierende Technologien umsetzen
- Webbasierte Oberflächen** und sonstige internetbasierende **Schnittstellen** findet man genau hier über entsprechende Implementierungen (z.B. FIORI Apps, WebDynpro, OData, SOAP, REST, ...)

# SAP Wissen

## Datenmodelle

Datenmodelle schaffen Fakten...



# Das Wichtigste zuerst...



*SAP Datenmodelle sind häufig nicht so,  
wie ein Anwender seine Welt sieht  
oder wie ein Entwickler  
es sich wünschen würde.*



# SAP Tabellen und –felder

- Häufig haben SAP Tabellen und deren Felder Namen, die aus **Abkürzungen von deutschen Begriffen** bestehen
- Beispiele
  - Tabelle „VBAK“ = Verkaufsbeleg Auftragskopf
  - Feld „VBAK-AUGRU“ = Auftragsgrund
- Grund: **50 jährige Historie** der SAP Software – bei der Erstellung dieser Objekte vor teilweise > 40 Jahren konnte man den internationalen Erfolg der SAP Software nicht ahnen
- Es gibt allerdings **Namenskonventionen und Längenbeschränkungen**
- Viele Tabellen haben deutlich **mehr Spalten**, als die Kunden tatsächlich nutzen und vermeintlich notwendig wären  
→ SAP hat den Anspruch international jede mögliche Anforderung „irgendwie“ umzusetzen. Es wird immer mal wieder „angebaut“
- Im SAP liebt man **Codes** (z.B. „0100“) deren Sinn sich oft nicht automatisch erschließt
- CHAR-Felder sind fast immer **UpperCase**, außer sie sind explizit anders gekennzeichnet
- ID Felder von Objekten (z.B. „Auftrag“) sind meistens als **alphanumerisch CHAR(10)** definiert und haben – auf der Datenbank – **führende Nullen** (Ausnahme: Feldwert fängt mit Buchstaben an)

**Dictionary: Tabelle anzeigen**

Transp. Tabelle: **VBAK** aktiv

Kurzbeschreibung: Verkaufsbeleg: Kopfdaten

Eigenschaften   Auslieferung und Pflege   **Felder**   Eingabehilfe/-prüfung   Währungs-/Mengenfelder   Indizes

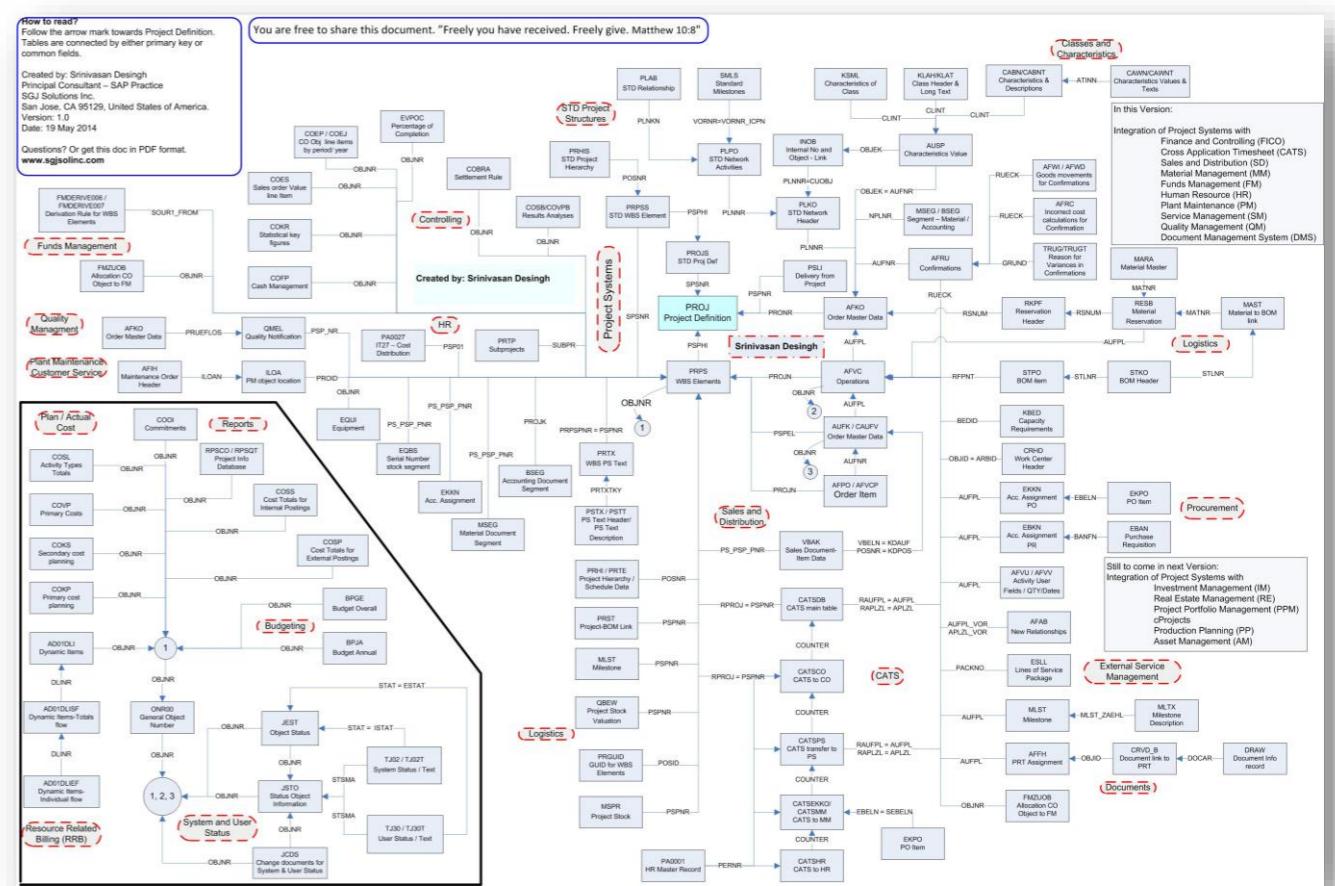
Suchhilfe   Eingebauter Typ

Feld	Key	Inl...	Datenelement	Datentyp	Länge	DezS...	KoordSyst...	Kurzbeschreibung
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0		0 Mandant
VBELN	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	VBELN_VA	CHAR	10	0		0 Verkaufsbeleg
ERDAT	<input type="checkbox"/>	<input type="checkbox"/>	ERDAT	DATS	8	0		0 Datum, an dem der Satz hinzugefügt wurde
ERZET	<input type="checkbox"/>	<input type="checkbox"/>	ERZET	TIMS	6	0		0 Erfassungszeit
ERNAM	<input type="checkbox"/>	<input type="checkbox"/>	ERNAM	CHAR	12	0		0 Name des Sachbearbeiters, der das Objekt hinzugefügt hat
ANGDT	<input type="checkbox"/>	<input type="checkbox"/>	ANGDT_V	DATS	8	0		0 Angebot/Anfrage gültig von
BNDDT	<input type="checkbox"/>	<input type="checkbox"/>	BNDDT	DATS	8	0		0 Befristet des Angebots für Leistungserbringung (gül. bis)
AUDAT	<input type="checkbox"/>	<input type="checkbox"/>	AUDAT	DATS	8	0		0 Belegdatum (Ein- bzw. Ausgangsdatum)
VBTYP	<input type="checkbox"/>	<input type="checkbox"/>	VBTYP	CHAR	4	0		0 Vertriebsbelegtyp
TRVOG	<input type="checkbox"/>	<input type="checkbox"/>	TRVOG	CHAR	1	0		0 Gruppe Transaktionsvorgang
AUART	<input type="checkbox"/>	<input type="checkbox"/>	AUART	CHAR	4	0		0 Verkaufsbelegart
AUGRU	<input type="checkbox"/>	<input type="checkbox"/>	AUGRU	CHAR	3	0		0 Auftragsgrund (Grund für Geschäftsvorgang)
GWLDI	<input type="checkbox"/>	<input type="checkbox"/>	GWLDI	DATS	8	0		0 Garantedatum
SUBMI	<input type="checkbox"/>	<input type="checkbox"/>	SUBMI_SD	CHAR	10	0		0 Submissionsnummer (SD)
LIFSK	<input type="checkbox"/>	<input type="checkbox"/>	LIFSK	CHAR	2	0		0 Lieferasperre (Belegkopf)
FAKSK	<input type="checkbox"/>	<input type="checkbox"/>	FAKSK	CHAR	2	0		0 Fakturasperre in Vertriebsbeleg
NETWR	<input type="checkbox"/>	<input type="checkbox"/>	NETWR_AK	CURR	15	2		0 Nettowert des Kundenauftrags in Belegwährung
WAERK	<input type="checkbox"/>	<input type="checkbox"/>	WAERK	CUKY	5	0		0 Währung des Vertriebsbelegs
VKORG	<input type="checkbox"/>	<input type="checkbox"/>	VKORG	CHAR	4	0		0 Verkaufsorganisation
VIWEG	<input type="checkbox"/>	<input type="checkbox"/>	VIWEG	CHAR	2	0		0 Vertriebsweg
SPART	<input type="checkbox"/>	<input type="checkbox"/>	SPART	CHAR	2	0		0 Sparte
VKGRL	<input type="checkbox"/>	<input type="checkbox"/>	VKGRL	CHAR	3	0		0 Verkäufergruppe
VKBUR	<input type="checkbox"/>	<input type="checkbox"/>	VKBUR	CHAR	4	0		0 Verkaufsbüro
GSBER	<input type="checkbox"/>	<input type="checkbox"/>	GSBER	CHAR	4	0		0 Geschäftsbereich
GSKST	<input type="checkbox"/>	<input type="checkbox"/>	GSKST	CHAR	4	0		0 Geschäftsbereich aus Kostenstelle
GUEBG	<input type="checkbox"/>	<input type="checkbox"/>	GUEBG	DATS	8	0		0 Gültigkeitsbeginn ( Rahmenvertrag / Sortiment )

# SAP Datenmodelle sind manchmal „unerwartet aufwendig“

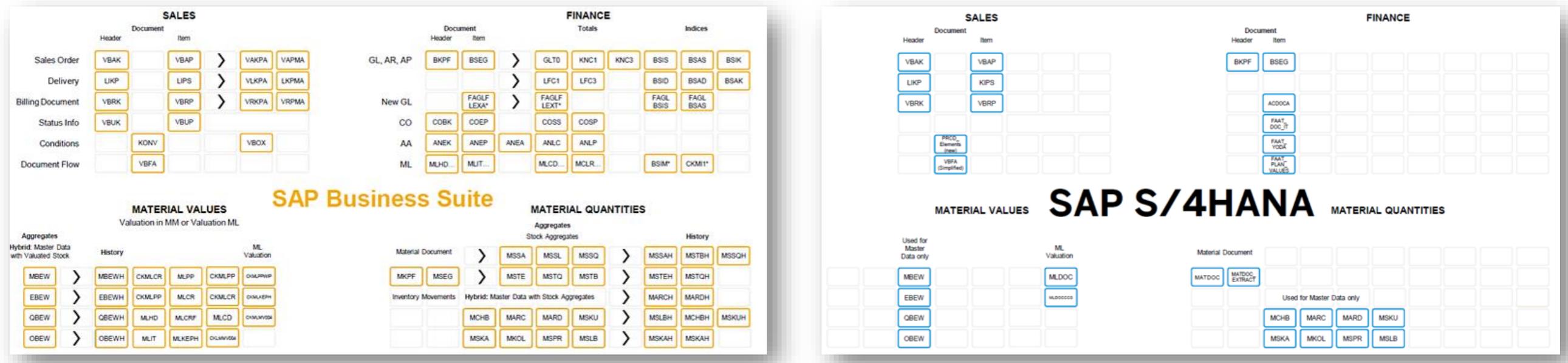


- Die Abbildung der Datenmodelle sind häufig **nicht so** anzutreffen, wie man es in der **Informatik** gelehrt bekommt
- Gründe sind:
  - 50 Jahre Historie
  - Wiederverwendung von Teilkomponenten
  - Separate Entwicklung von Modulen und deren spätere Integration
  - Datensparsamkeit vor 1990
- **Mehrere semantische Objekte** in einer Tabelle  
z.B. VBAK Verkaufsbeleg Auftragskopf enthält:
  - Anfrage
  - Angebot
  - Auftrag
  - Vertrag
  - ...
- Bewusste redundante Datenhaltung und Aggregate für die Performance
- Die **ABAP Geschäftslogik** und das sogenannte **SAP Customizing** entscheiden über den Inhalt und die Verwendung der Tabellendaten (weniger Datenbankgetriebene Entwicklung)
- Kunden haben oft Ihre „**Sonderlocken**“ einprogrammiert  
z.B. Vor dem Jahr 2010 bedeutet das Feld X Y, danach Z



<https://blogs.sap.com/2014/06/06/sap-ps-tables/>

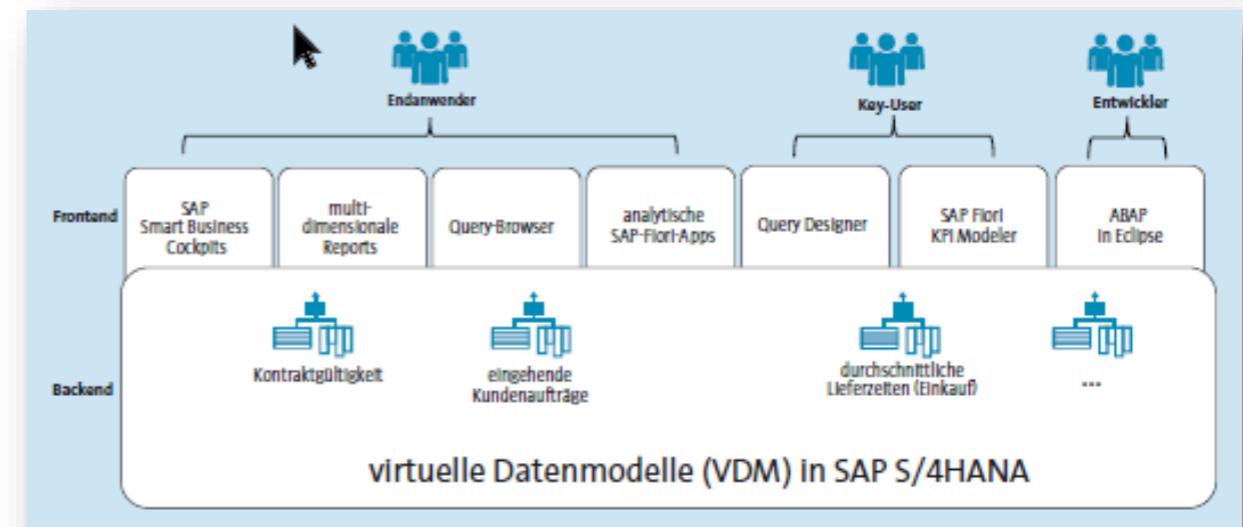
# SAP hat mit SAP S/4 HANA umgebaut...



- SAP S/4 HANA ist der Nachfolger der klassischen SAP R/3 ERP Produktlinie, aber rechtlich ein neues Produkt
- Das **Datenmodell** wurde an vielen Stellen **vereinfacht und modernisiert**
- An einigen Stellen wurde auch **Funktionen eingestellt oder durch andere ersetzt**
- Kunden können Ihre bisherigen ERP Systeme nicht einfach upgraden, sondern müssen **migrieren**
- An vielen Stellen sind allerdings noch **alte Programme** im Einsatz, die auf die alten ERP Datenmodelle aufsetzen
- Diese Herausforderung hat SAP mit den **virtuellen Datenmodellen** gelöst, die durch die **Core Data Services (CDS)** umgesetzt wurden...

# Virtuelle Datenmodelle (VDM Theorie)

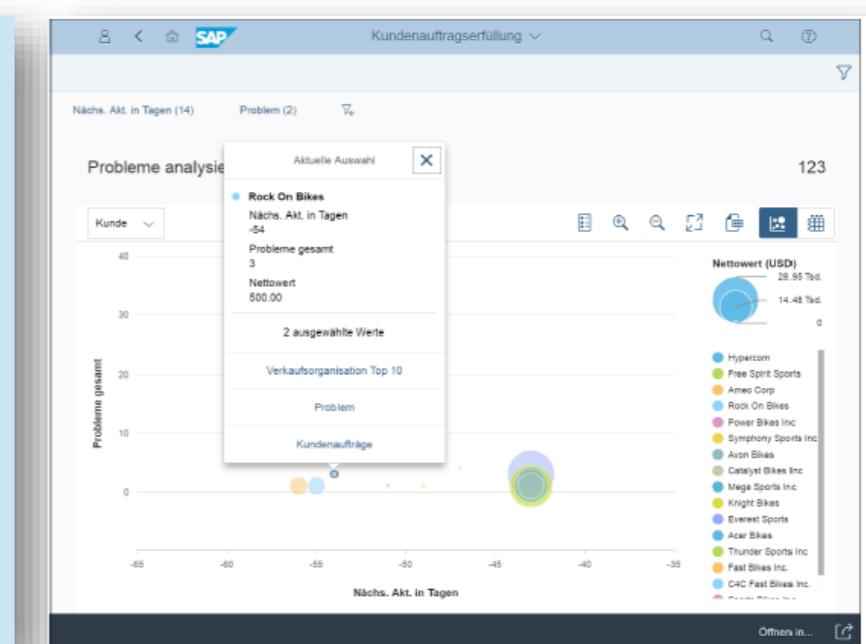
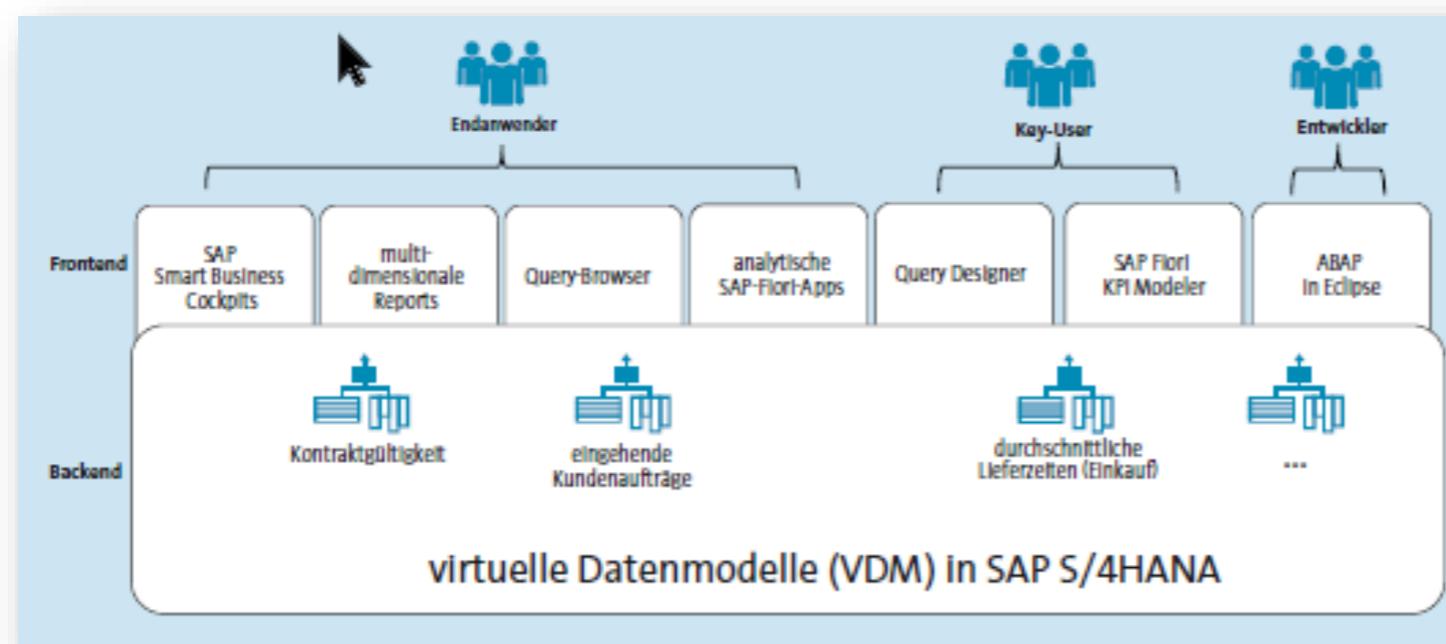
Eine kurze Einführung



# Virtuelle Datenmodelle (VDM) - Motivation SAP



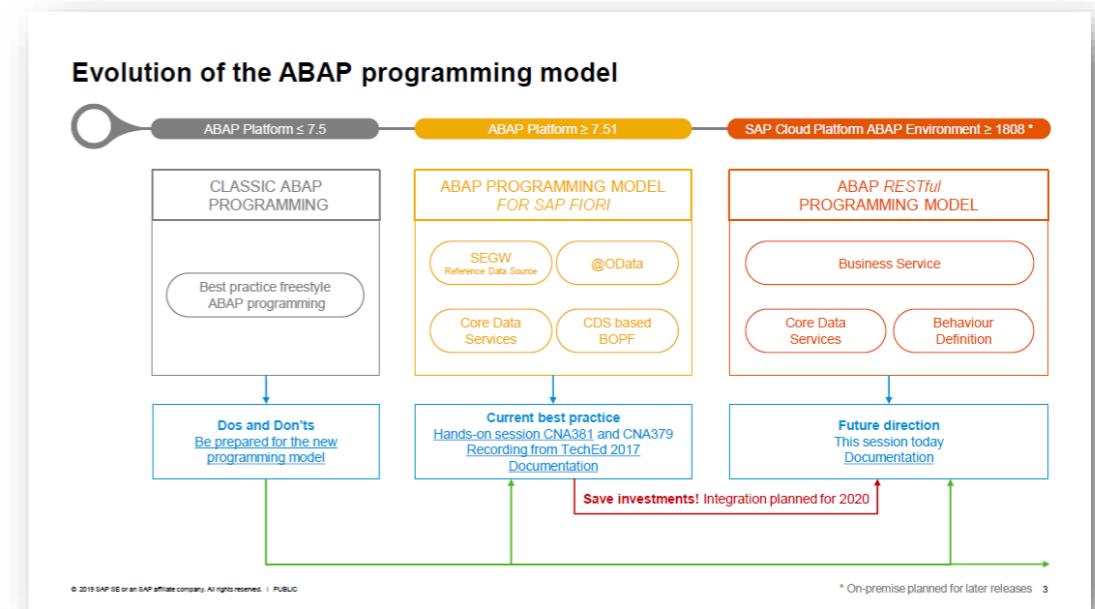
- Die virtuellen Datenmodelle sind die **Grundlage** für viele neue Funktionen im SAP S/4 HANA
- Die **HANA-Datenbank** ermöglicht einen schnellen Zugriff auf bisher eher unperformante Datenquellen  
→ **S/4 Embedded Analytics** für das operative Reporting (macht ein Data Warehouse Systeme deshalb nicht unbedingt obsolet)
- Viele der neuen webbasierenden FIORI Anwendungen bauen auf VDMs auf
- **Key User Tools**, Entwicklungsmethoden und weitere Werkzeuge im SAP Backend
- Kompatibilität zur alten SAP Welt...



# Virtuelle Datenmodelle (VDM) – SAP Entwickler



- Die VDM werden mit der **Core Data Services (CDS)** Technologie entwickelt
- SAP hat sehr früh darauf hingewiesen, dass die SAP Entwicklung sich in die Richtung der **CDS weiterentwickeln** wird
- Es ist **gut dokumentiert** und sogar in älteren Releases in Teilen bereits verfügbar
- In bestimmten Runtime Umgebungen (z.B. **ABAP in der Cloud**, Codename „Steampunk“) geht vieles gar nicht mehr ohne CDS...



## Availability of CDS in SAP Platforms

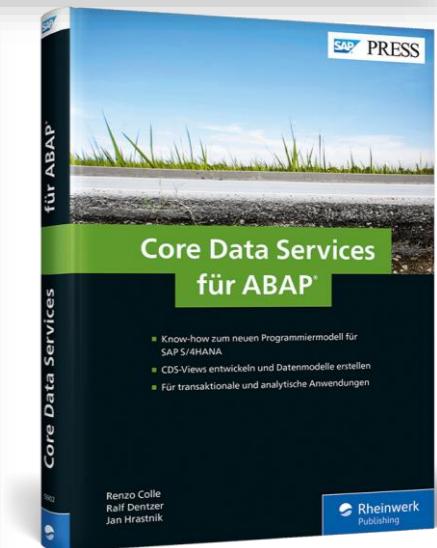
The Core Data services are available in below mentioned SAP Platforms:

1. SAP NetWeaver 7.50, SP01, or higher.
2. SAP NetWeaver 7.4 SP05
3. SAP HANA SPS6
4. SAP Business Suite EHP7 (Suite on HANA)
5. S/4HANA
6. SAP Business Warehouse 7.3

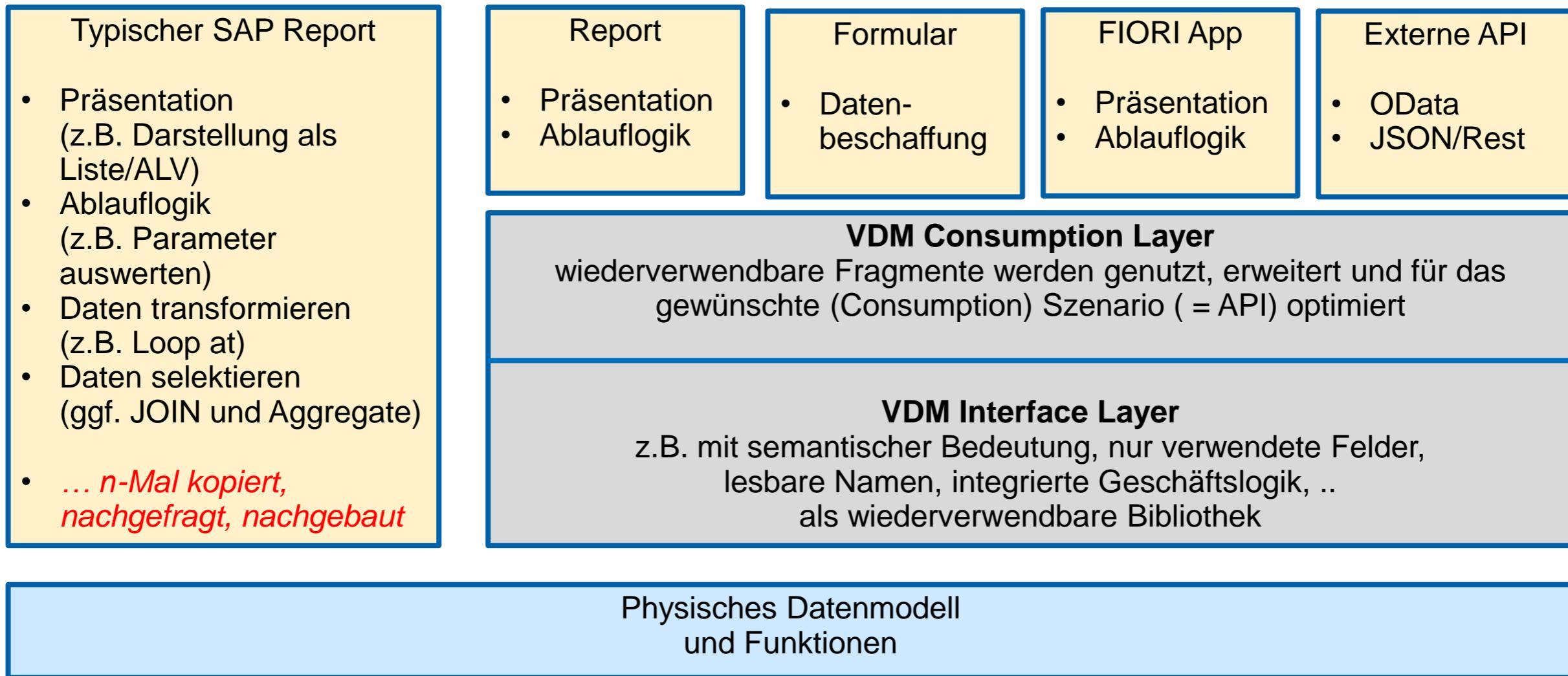
The screenshot shows a SAP blog post by Tushar Sharma. The post title is "ABAP Core Data Services – Introduction (ABAP CDS view)". It includes a profile picture of Tushar Sharma, the date (September 9, 2017), and a 6-minute read time. Below the title is a "Table of Content" table:

S.no	Topics
1.	Introduction
2.	Architecture Overview
3.	CDS Releases
4.	Availability of CDS in SAP Platforms
5.	Motivation behind Core Data Services
6.	Next Blog's (Follow)
7.	Credits

On the right side, there are sections for "Assigned tags" (SAP HANA, ABAP Development, SAP Access Control, ABAP CDS view, ABAP Core Data Service Views), "Related Blog Posts", and "Related Questions".



# Virtuelle Datenmodelle (VDM) – Motivation Kunde



# Was ist CDS?



## Konsumenten

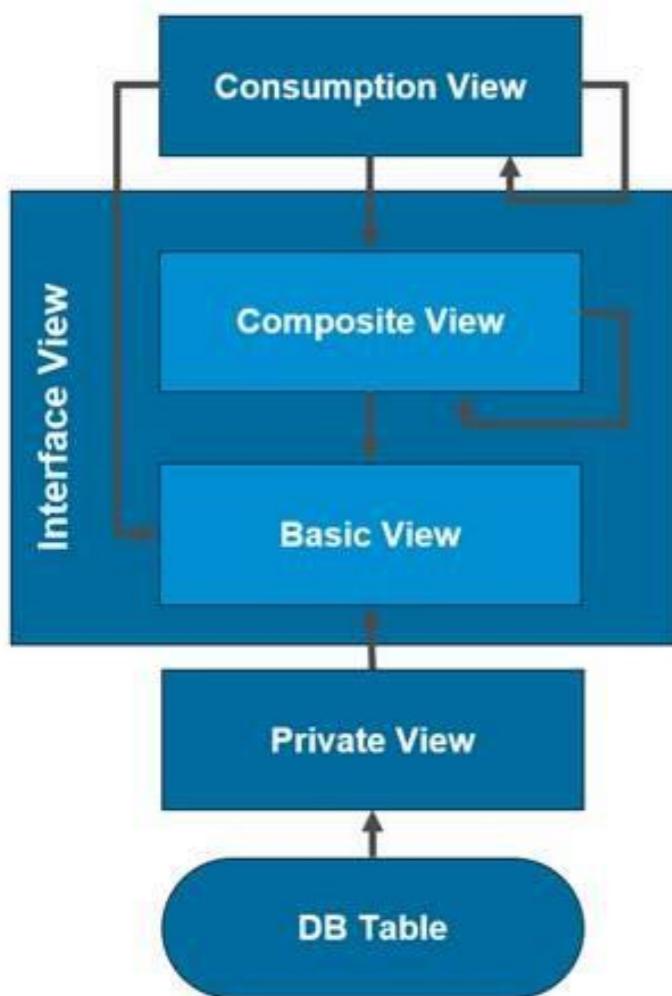
- ABAP Geschäftslogik
- Schnittstellen

## CDS Layer (virtuelle Datenmodelle)

## Physische Datenbanktabellen

- CDS = Core Data Services ist eine **neue Technologie** der SAP
- Mit der CDS Technologie entsteht eine **neue datenbanknahe Zwischenschicht** zwischen bisheriger ABAP Programmlogik und den physischen Datenbanktabellen
- Durch sogenannte CDS Views werden „gewünschte Datenmodelle“ erstellt
- CDS Views und andere Fragmente sind **wiederverwendbar** und sollten auch wie eine Bibliothek verwendet und gepflegt werden
- CDS Views können Teile der **Geschäftslogik** abbilden, was die ABAP Programmierung u.ä. vereinfacht, wenn dort das „Wissen des Unternehmens“ implementiert wird
- CDS Views bilden auch sehr **komplexe Datenmodelle** ab und werden von der Datenbank optimiert behandelt  
→ „**Views on Demand**“ (= JOINs u.ä. werden nur ausgeführt, wenn benötigt)
- CDS bietet spezielle Funktionen für die **Web-Entwicklung, Schnittstellen und Analytics** und wird mit jeder neuen Version mächtiger ....

# CDS – wichtige Begriffe



- „Consumption“ Schicht
  - Ist die sichtbare Schicht aus der Sicht des Anwenders bzw. der Anforderung
  - Jede konkrete Anforderung hat eine eigene Consumption Schicht und ist dadurch auch bei Änderungen stabil
  - Die Consumption Schicht bedient sich aus einem „Baukasten“ vordefinierter Datenmodelle → „Interface Schicht“
- „Interface“ Schicht
  - Die Interface Schicht bildet das „Bauskastensystem“ und damit die Wiederverwendbarkeit ab
  - Hier werden die grundlegenden Datenmodelle des Kunden abgebildet, z.B. ein typischer Kundenauftrag
  - Aus den physischen Tabellen werden hier „semantische Objekte“ mit unterschiedlichen Eigenschaften (z.B. DB Tabelle VBAK → Objekt „Kundenauftrag“ + Objekt „Kundenangebot“)
- „Composite Views“
  - Semantische Objekte mit den (maximal) verfügbaren Feldern und Beziehungen zu anderen Objekten
- „Basic Views“
  - Zwischenschicht, um die wesentlichen Felder, deren Namen und Eigenschaften festzulegen
  - Ggf. bereits komplexe JOIN-Verbindungen hinterlegt
- „Private Views“
  - Optionale Zwischenschicht, die nicht offiziell zur Verfügung steht

# Ein CDS View Beispiel

```
@AbapCatalog.sqlViewName: 'ZC_DMORD_SEL'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'Demo Order Select - consumption'  
  
@VDM.viewType: #CONSUMPTION  
@OData.publish: true  
  
define view ZC_DEMO_ORDER_SELECT  
  as select from I_SalesDocumentBasic  
{  
  //I_SalesDocumentBasic  
  key SalesDocument,  
  TotalNetAmount,  
  TransactionCurrency,  
  CreatedByUser,  
  CreationDate,  
  CreationTime,  
  SalesOrganization,  
  _SalesOrganization._Text.SalesOrganizationName,  
  SalesOffice,  
  _SalesOffice._Text.SalesOfficeName  
  
} where SDDocumentCategory = 'C' // only Orders
```



The diagram illustrates the annotations for a CDS View definition. Red lines connect specific code elements to their corresponding descriptions on the right side:

- Annotation: @AbapCatalog.sqlViewName: 'ZC\_DMORD\_SEL' → Description: View für den Zugriff von Standard ABAP (SE16N)
- Annotation: @AccessControl.authorizationCheck: #CHECK → Description: Rolle „Consumption“ im Virtuellen Datenmodell
- Annotation: @EndUserText.label: 'Demo Order Select - consumption' → Description: Annotation, um automatisch eine OData Freigabe zu generieren (Erst ab 7.5x)
- Annotation: define view ZC\_DEMO\_ORDER\_SELECT → Description: Name des CDS Views (für Eclipse)
- Annotation: as select from I\_SalesDocumentBasic → Description: Name des aufgerufenen CDS Views bzw. Tabelle
- Annotation: //I\_SalesDocumentBasic → Description: Verfügbare Feldliste und Herkunft
- Annotation: key SalesDocument, TotalNetAmount, TransactionCurrency, CreatedByUser, CreationDate, CreationTime, SalesOrganization, \_SalesOrganization.\_Text.SalesOrganizationName, SalesOffice, \_SalesOffice.\_Text.SalesOfficeName → Description: Felder aus dem Beziehungswissen (Assoziationen)
- Annotation: } where SDDocumentCategory = 'C' // only Orders → Description: Vordefinierte Filter

# Das OData Protokoll



OData - the best way to REST

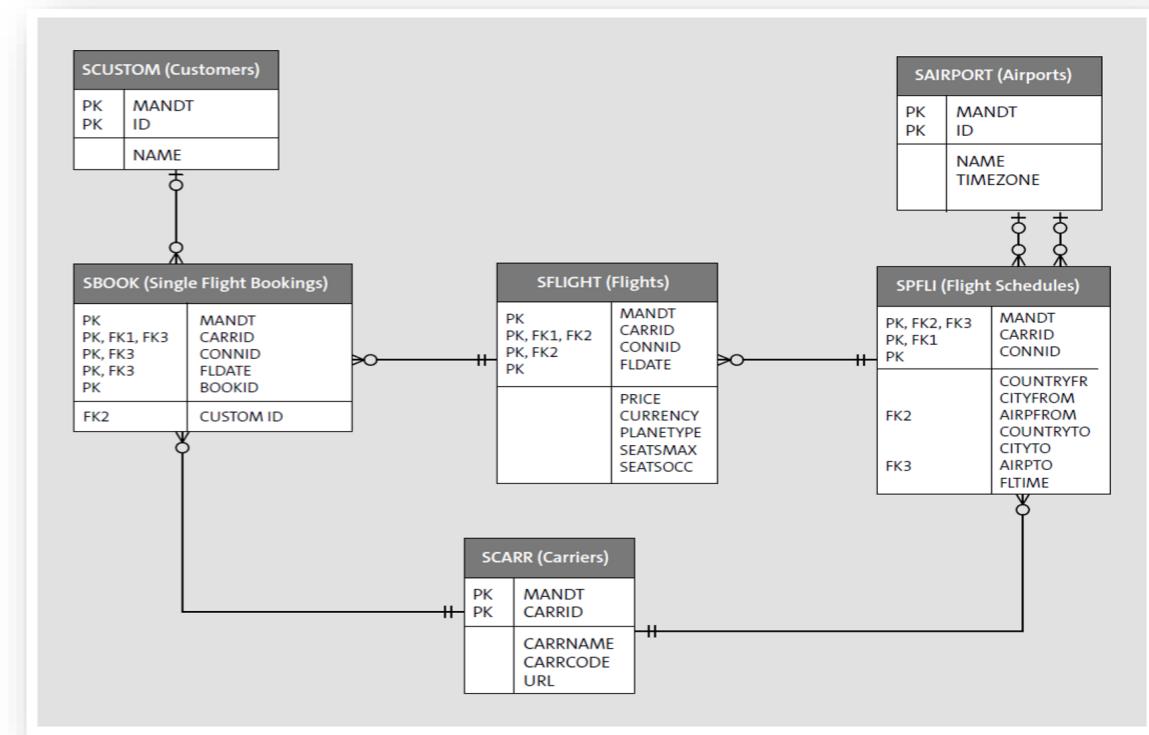
An **open protocol** to allow the creation and consumption of **queryable** and **interoperable RESTful APIs** in a **simple** and **standard** way.

<https://www.odata.org/>

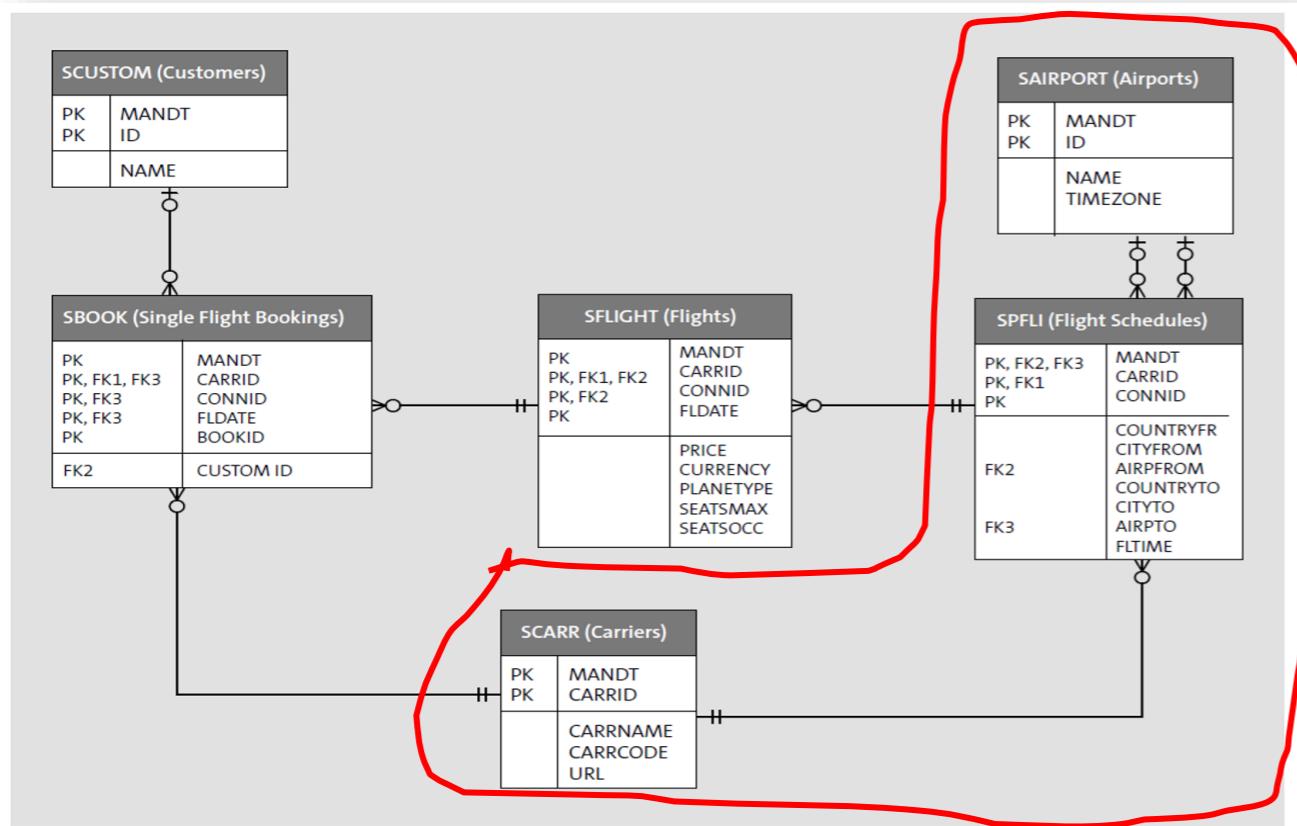
- OData („Open Data Protocol“) ist das **wichtigste SAP Protokoll** bei der Bereitstellung von Daten **für das Web** (z.B. FIORI Apps) und OASIS Standard
- OData ist eine Schnittstellentechnologie – man nennt es auch das „**JDBC für das Internet**“ -, mit der tabellenähnliche Daten im Internet abgefragt werden können
- Es ist auch **außerhalb des SAP Bereiches** bekannt, verfügbar und es gibt für viele Programmiersprachen entsprechende **Bibliotheken**
- Sollte man kennen und wird von der CDS Technologie unterstützt...

# Virtuelle Datenmodelle (VDM Praxis)

Ein kleines Beispiel, wie man aus Alt Neu machen kann...



# Ausgangssituation – Das „SAP Flight Model“



- Das **SAP Flugdatenmodell** ist seit Jahren die Grundlage für SAP Entwicklerschulungen und in allen SAP System vorhanden
- Für die Demo wurde der spezielle Ausschnitt des **Flugplans** mit Verweisen auf die Tabellen für die **Fluglinien** und **Flughäfen** verwendet
- Diese werden in den folgenden Schritten über die **CDS Technologie** „modernisiert“ und stehen danach als **Interface Layer** als Bibliothek zur Verfügung
- Auf Basis dieser Bibliothek entstehen zwei Konsumentenszenarien (**CDS Consumer Layer**), die danach in verschiedenen Schnittstellentechnologien verwendet werden

The image shows a flight information display board with two main sections: "Departures" and "Arrivals".

Terminal	Flight	Destination	Time	Gate	Remark	Terminal	Flight	Destination	Time	Gate	Remark
Terminal 0	BR117	Edinburgh	06:45	10		Terminal 0	RK1250	Berlin	08:40	125	125
Terminal 0	CX1347	Bucharest	06:50	10		Terminal 0	RC1550	Bordeaux	07:15	125	125
Terminal 0	BR165	Brussels	06:55	10		Terminal 0	KP1030	Brussels	07:25	125	125
Terminal 0	BR3925	Amsterdam	07:05	12	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal 0	BR872	Istanbul	07:10	12	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal 0	RR6706	Copenhagen	07:15	13	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal 0	RY6554	Glasgow	07:15	13	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal Z	RF7651AY	Zurich	07:25	14	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal 0	RF7651CD	Rome	07:35	15	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal 0	DLB611CD	Newcastle	07:45	16	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal 0	RZ300RF	Dusseldorf	07:50	16	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal 0	RZ100ZOL	Berlin	07:55	16	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125
Terminal 0	YK100ZOL	London	08:00	17	CHECK-IN	Terminal 0	YK1030	Paris	07:25	125	125

<https://blog.sap-press.com/what-is-sflight-and-the-flight-and-booking-data-model-for-abap>

[https://help.sap.com/SAPHelp\\_nw73/helpdata/en/cf/21f304446011d189700000e8322d00/frameset.htm](https://help.sap.com/SAPHelp_nw73/helpdata/en/cf/21f304446011d189700000e8322d00/frameset.htm)

# Vorher - Nachher

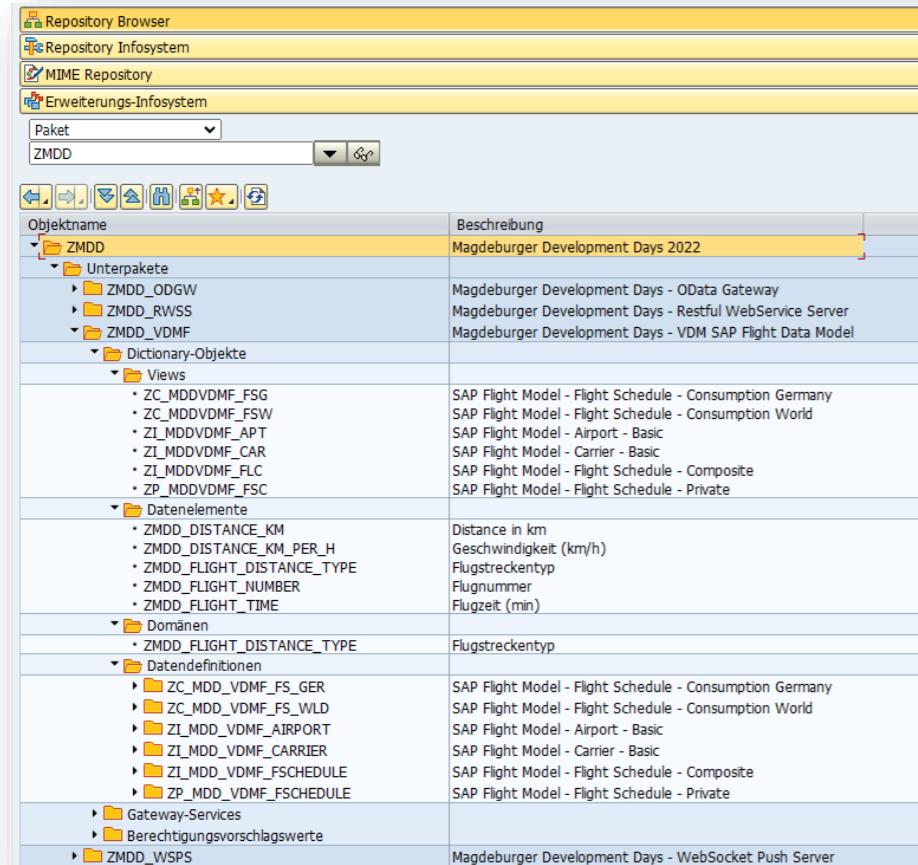


A screenshot of the SAP Data Browser (SE16n) interface. The title bar says "Dictionary: Tabelle anzeigen". The menu bar includes "Tabelle", "Bearbeiten", "Springen", "Hilfsmittel", "Zusätze", "Umfeld", "System", and "Hilfe". The toolbar has various icons for file operations. The main window shows a table named "SPFLI" with 16 rows. The columns are: Feld, Key, In.., Datenelement, Datentyp, Länge, DezZ..., KoordSyst..., Kurzbeschreibung. The table contains data such as MANDT, CARRID, CONNID, COUNTRYFR, CITYFROM, AIRPFROM, COUNTRYTO, CITYTO, AIRPTO, FLTTYPE, DEPTIME, ARRTIME, DISTANCE, DISTID, FLTYPE, and PERIOD. The data is presented in a grid format with some columns having dropdown menus.

- Vorher entspricht dem Data Browser Blick (SE16n) auf Tabelle SPFLI
- Das Ergebnis „Nachher“ ist ein JSON Objekt, z.B. für Schnittstellen
- Änderungen zum „SAP Original“: sprechende Namen in „CamelCase“, Typkonvertierung inkl. Bezeichnungswechsel, ein eindeutiger kombinierter Schlüssel, Zusatzinformationen zum Flughafen/ Fluggesellschaft, Entfernung umgerechnet in „km“, berechnete Durchschnittsgeschwindigkeit in km/h, Kategorie Streckentyp

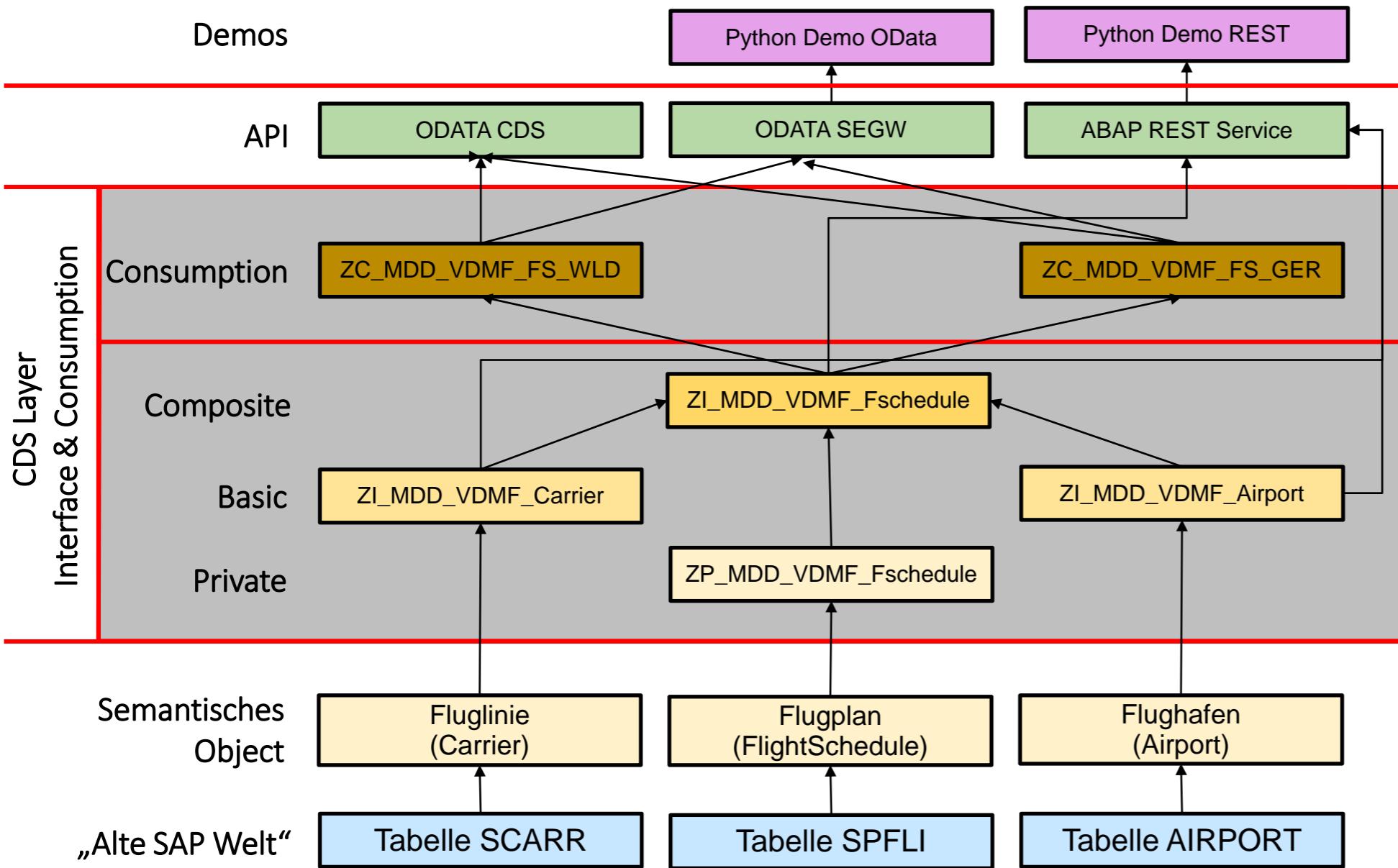
```
{
    "FlightNumber": "UA3517",
    "CarrierId": "UA",
    "CarrierName": "United Airlines",
    "CarrierUrl": "http://www.ual.com",
    "ConnectionId": "3517",
    "CountryFrom": "DE",
    "CityFrom": "FRANKFURT",
    "AirportFrom": "FRA",
    "AirportFromName": "Frankfurt/Main, FRG",
    "DepartureTime": "PT10H40M00S",
    "CountryTo": "US",
    "CityTo": "NEW YORK",
    "AirportTo": "JFK",
    "AirportToName": "New York JF Kennedy, USA",
    "ArrivalTime": "PT12H55M00S",
    "DaysLater": 0,
    "FlightTime": 495,
    "DistanceKM": "6162.0000",
    "DistanceKMH": "746.90",
    "FlightIsCharter": "",
    "FlightType": "L"
}
```

# ABAP Server Code (SAPGUI SE80, Eclipse)

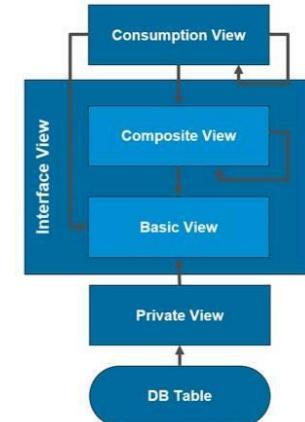


- Der **ABAP Code** für das VDM befindet sich im Unterpaket ZMDD\_VDMF des **ZMDD Paketes**
- Erinnerung: Das komplette Paket ZMDD ist als **Github Repository** verfügbar und kann per **abapGit** in eigene SAP Systeme importiert werden

# VDM „MDD Flugdaten“ – Big Picture



Zur Erinnerung:  
 Mit der CDS Technologie werden wieder verwendbare Datenbank-Fragmente geschaffen, die am besten das gewünschte Datenmodell abbilden und auch Geschäftslogik in den CDS Layer verlagern



# Basic Views „Carrier“ & „Airport“



```
@AbapCatalog.sqlViewName: 'ZI_MDDVDMF_CAR'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'SAP Flight Model - Carrier - Basic'  
  
@VDM.viewType: #BASIC  
  
define view ZI_MDD_VDMF_Carrier  
  as select from scarr  
{  
  key carrid as CarrierId,  
    carrname as CarrierName,  
    currcode as CarrierCurrencyCode,  
    url      as CarrierUrl  
}
```

```
@AbapCatalog.sqlViewName: 'ZI_MDDVDMF_APT'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'SAP Flight Model - Airport - Basic'  
  
@VDM.viewType: #BASIC  
  
define view ZI_MDD_VDMF_Airport  
  as select from sairport  
{  
  key id      as AirportId,  
    name     as AirportName,  
    time_zone as AirportTimeZone  
}
```

- In diesem Schritt werden hauptsächlich die originalen SAP **Feldnamen** durch sprechende **CamelCase** Namen ersetzt
- Der CDS View wird mit der Annoation **@VDM.viewType** als **BASIC** View klassifiziert
- Mit der Annoation **@AbapCatalog.sqlViewName** wird ein Name für einen generierten SQL View mitgegeben, der in älteren SAP Programmen oder als Typdefinition verwendet werden kann  
Achtung: max. 16 Stellen (vs. 30 Stellen beim CDS View)

# Private View „FlightSchedule“



```
@AbapCatalog.sqlViewName: 'ZP_MDDVDMF_FSC'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'SAP Flight Model - Flight Schedule - Private'  
  
@VDM.viewType: #BASIC  
  
define view ZP_MDD_VDMF_FSchedule  
  as select from spfli  
{  
  key carrid  
  key connid  
  
  // departure from  
  countryfr  
  cityfrom  
  airpfrom  
  deptime  
  // arrival at  
  countryto  
  cityto  
  ...  
  // ---- conversions  
  // combined Flight Number  
  cast(concat(carrid, connid) as zmdd_flight_number) as FlightNumber,  
  // calculate distance in KM  
  cast(unit_conversion(  
    quantity => distance,  
    source_unit => distid,  
    target_unit => cast('KM' as abap.unit),  
    error_handling => 'SET_TO_NULL' ) as zmdd_distance_km) as DistanceKM,  
  // calcuate the speed as float  
  cast(distance * 60 as abap.fltp) / cast(flttime as abap.fltp) as DistancePerHour  
}
```

- Der Private View wurde zwischengeschaltet, um einige Berechnungen schon in einer Zwischenschicht durchzuführen und spätere Wiederverwendung zu unterstützen
- Zusätzliche Anpassungen:
  - Kombinierter Key als Feld „FlightNumber“ (Schlüssel der Tabelle sind 2 Felder)
  - Einheitenumrechnung mit CDS Funktion die Flugentfernung (Distance) kommt nicht nur in KM sondern auch in MI, daher hier Vereinheitlichung
  - Berechnung der Durchschnittsgeschwindigkeit als km/h

# Composite View „FlightSchedule“



```
@AbapCatalog.sqlViewName: 'ZI_MDDVDMF_FLC'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'SAP Flight Model - Flight Schedule - Composite'

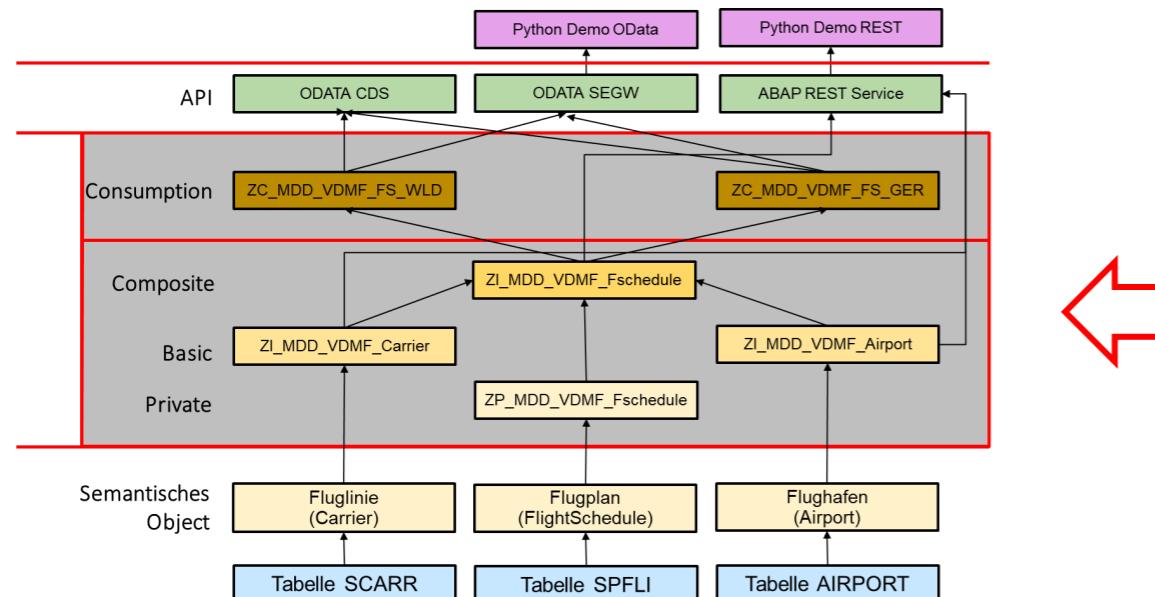
define view ZI_MDD_VDMF_FSchedule
  as select from ZP_MDD_VDMF_FSchedule

    association [1]      to ZI_MDD_VDMF_Carrier as _Carrier      on _Carrier.CarrierId = $projection.CarrierId
    association [0..1]   to ZI_MDD_VDMF_Airport as _AirportFrom on _AirportFrom.AirportId = $projection.AirportFrom
    association [0..1]   to ZI_MDD_VDMF_Airport as _AirportTo   on _AirportTo.AirportId = $projection.AirportTo

{
  // key info + combined info field
  key CarrierId,
  key ConnectionId,
  FlightNumber,
  // departure
  CountryFrom,
  ...
  // flight information
  cast(FlightTime as zmdd_flight_time) as FlightTime,
  DistanceKM,
  FLTP_TO_DEC(DistancePerHour as zmdd_distance_km_per_h) as DistanceKMH,
  FlightIsCharter,
  // additional Information
  cast(case
    when FlightTime <= 120 then 'S'
    when FlightTime > 360 then 'L'
    else 'M'
    end as zmdd_flight_distance_type)
  as FlightType,          // Special calculation for
filtering
  // Associations
  _Carrier,
  _AirportFrom,
  _AirportTo
}
```

- Eine wichtige Eigenschaft von **Composite Views** ist, dass sie die Navigation zu anderen Entitäten über Assoziationen modellieren und an Ihre Erben weitergeben
- Assoziationen werden dann als **Navigationsfunktionen** verwendet (z.B. Odata)
- „Views on Demand“ – JOIN  
Assoziationen werden nur auf der Datenbank ausgeführt, wenn vom Aufrufer verwendet
- Zusätzliche Anpassungen:
  - **Feldreihenfolge** (z.B FlightNumber)
  - **Semantic Type Casting** (Grund OData Fehler, Bezeichnung)
  - **Technical Type Casting** (Float -> Dec für bessere Lesbarkeit)
  - **Klassifikation** der Flugzeit in S,M,L
  - **Assoziationen freigeben** für Erben (explizit in Feldliste aufgenommen)

# Zwischenstand – der CDS Interface Layer ist fertig



- Über die Eclipse Tools „Open with“ → „Data Preview“ können die Daten bereits im Eclipse schnell angezeigt werden
- Eclipse unterstützt auch eine Navigation auf den Assoziationen

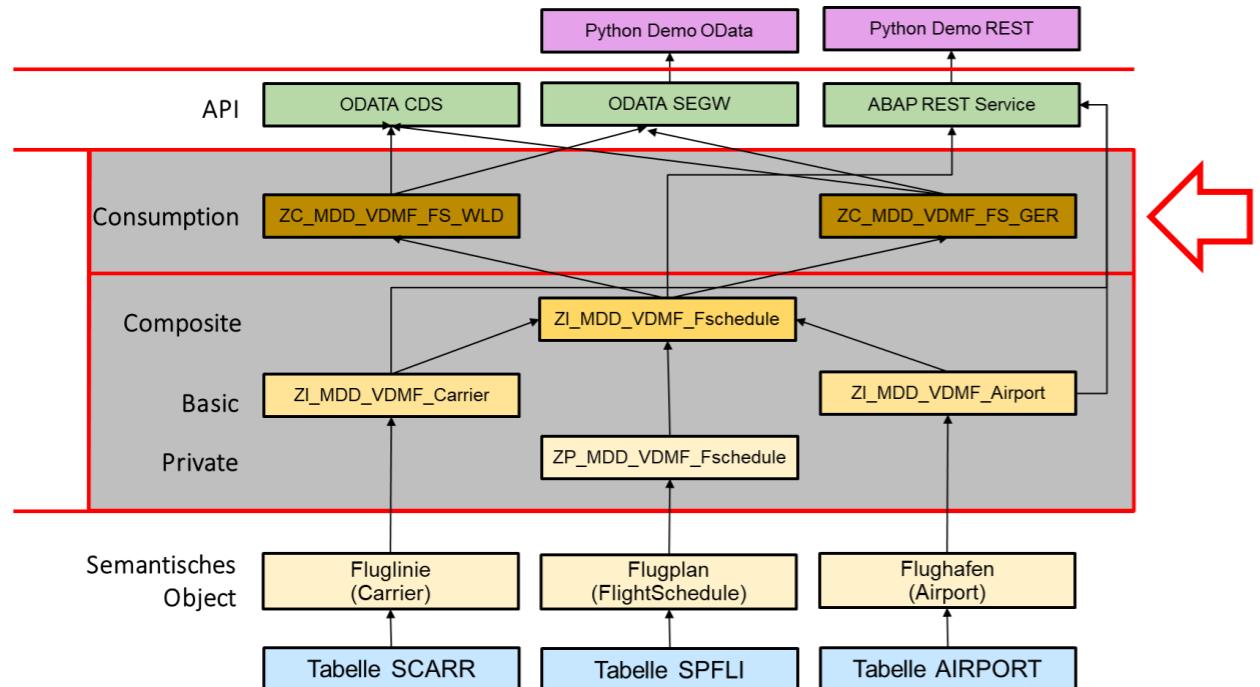
The screenshot shows the Eclipse Data Preview interface with the following details:

- Raw Data View:** Displays flight schedule data with columns: Carrie, Connector, FlightNumt, CountryFrom, CityFrom, AirportFrom, DepartureTir, Country, City, and Airport. A red arrow points from the diagram to this view.
- Contextual Menu:** Shows options like "Quick filter on [IT]", "Distinct values for [CountryFrom]", "Follow Association", "Copy row", and "Copy all rows as ABAP value statement". A red arrow points from the menu to the "Follow Association" option.
- List of Associations:** Shows associations: \_AirportFrom → ZI\_MDD\_VDMF\_Airport [0..1], \_AirportTo → ZI\_MDD\_VDMF\_Airport [0..1], and \_Carrier → ZI\_MDD\_VDMF\_Carrier [0..1]. A red arrow points from the menu to this list.

The screenshot shows the Eclipse Data Preview interface with the following details:

- Raw Data View:** Displays carrier data with columns: Carrie, CarrierName, CarrierCurrencyCo, and CarrierId. A single row for Alitalia is selected.
- Status Bar:** Shows "1 rows retrieved - ms".

# Consumption Layer - Überblick



- CDS Consumption Views stellen eine Art „API“ dar  
→ **eine Anforderung = 1 Consumption View**
- Auf Basis der soeben geschaffenen „Bibliothek“ entstehen in diesem Schritt zwei unterschiedliche CDS Consumption Views, die beide den Composite View „ZI\_MDD\_VDMF\_Fschedule“ aus dem **CDS Interface Layer** verwenden, aber unterschiedlich nutzen
- **Flugplan Deutschland** → ZC\_MDD\_VDMF\_FS\_GER
  - Zugriff über eindeutigen kombinierten Schlüssel FlightNumber
  - Einschränkung auf deutsche Flughäfen und ohne Charter
- **Flugplan Welt** → ZC\_MDD\_VDMF\_FS\_WLD
  - Zugriff über eindeutigen kombinierten Schlüssel FlightNumber
  - Zusatzinformationen für Fluggesellschaft und Flughafen
  - Keine Einschränkungen (inkl. Charter-Flüge)
- Diese Consumption Layer können danach für verschiedene Anwendungen (Oberflächen, Schnittstellen, Reports) mit gleichen Anforderungen dienen → "Single Source of Data"

Departures		Arrivals	
Terminal	Flight	Destination	Time
terminal A	BR117	Edinburgh	06:45
terminal B	CX197	Bucharest	06:50
terminal R	BR165	Brussels	06:55
terminal R	BR325	Asterdam	07:05
terminal D	BRB72	Istanbul	07:10
terminal D	BRB76	Copenhagen	07:15
terminal Z	AR6706	Glasgow	07:20
terminal B	AY6554	Zurich	07:25
terminal D	RF7651RY	Rome	07:30
terminal D	DLB811CD	Newcastle	07:35
terminal R	BZ 300RF	Dusseldorf	07:40
terminal R	BZ 3007DL	Berlin	07:45
terminal R	BZ 3007DL	Hamburg	07:50
terminal R	BZ 3007DL	Dortmund	07:55
terminal R	BZ 3007DL	Köln	08:00

# Consumption View „FlightScheduleGermany“



```
@AbapCatalog.sqlViewName: 'ZC_MDDVDMF_FSG'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'SAP Flight Model - Flight Schedule - Consumption Germany'  
  
@VDM.viewType: #CONSUMPTION  
  
define view ZC_MDD_VDMF_FS_GER  
as select from ZI_MDD_VDMF_FSchedule {  
    key FlightNumber,  
        CarrierId,  
        ConnectionId,  
        CountryFrom,  
        CityFrom,  
        AirportFrom,  
        DepartureTime,  
        CountryTo,  
        CityTo,  
        AirportTo,  
        ArrivalTime,  
        DaysLater,  
        FlightTime,  
        DistanceKM,  
        DistanceKMH,  
        FlightIsCharter,  
        FlightType,  
        /* Associations */  
        _AirportFrom,  
        _AirportTo,  
        _Carrier  
}  
  
where FlightIsCharter <> 'X'  
and ( CountryFrom = 'DE' or CountryTo = 'DE' )
```

- Über die @VDM-Annotation wird der View als „konsumierbar“ gekennzeichnet
- Diese Annotation wird in verschiedenen Tools geprüft
- Die bisherigen Tabellenschlüssel wurden hier ersetzt durch einen kombinierten Key „FlightNumber“
- Die geerbten Assoziationen werden alle freigegeben, damit sie auch für OData u.ä. verwendbar sind  
(Tipp: man kann also auch bewusst ausblenden und sogar umbenennen)
- Über die WHERE Klausel werden nur Deutsche Flughäfen sowie Charter-Flüge gefiltert
  
- Hinweis:  
Wenn der CDS View später OData-enabled werden soll, gibt es eine Längenschränkung: an den späteren Service wird „\_CDS“ angehängt und er darf nur insgesamt 30 Stellen lang werden

# Consumption View „FlightScheduleWorld“



```
@AbapCatalog.sqlViewName: 'ZC_MDDVDMF_FSW'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'SAP Flight Model - Flight Schedule - Consumption World'  
  
@VDM.viewType: #CONSUMPTION  
  
define view ZC_MDD_VDMF_FS_WLD  
as select from ZI_MDD_VDMF_FSchedule  
{  
    key FlightNumber,  
    CarrierId,  
    _Carrier.CarrierName,  
    _Carrier.CarrierUrl,  
    ConnectionId,  
    CountryFrom,  
    CityFrom,  
    AirportFrom,  
    _AirportFrom.AirportName as AirportFromName,  
    DepartureTime,  
    CountryTo,  
    CityTo,  
    AirportTo,  
    _AirportTo.AirportName as AirportToName,  
    ArrivalTime,  
    DaysLater,  
    FlightTime,  
    DistanceKM,  
    DistanceKMH,  
    FlightIsCharter,  
    FlightType,  
    /* Associations */  
    _AirportFrom,  
    _AirportTo,  
    _Carrier  
}
```

- Bis auf die WHERE-Filter-Klausel alles wie beim Germany-Beispiel:
  - Freigabe der Assoziationen
  - VDM-Annotation
- Anpassungen:
  - Der kombinierte Schlüssel „FlightNumber“ ist der eindeutige Schlüssel für diese Entität (Ja, das geht)
  - Zusatzinfos für Flughafen (Airport) und Fluglinie (Carrier) über die Assoziationen
    - Assoziationen werden jetzt tatsächlich von der Datenbank bearbeitet („Views on Demand“)

# Unser virtuelles Datenmodell ist fertig!



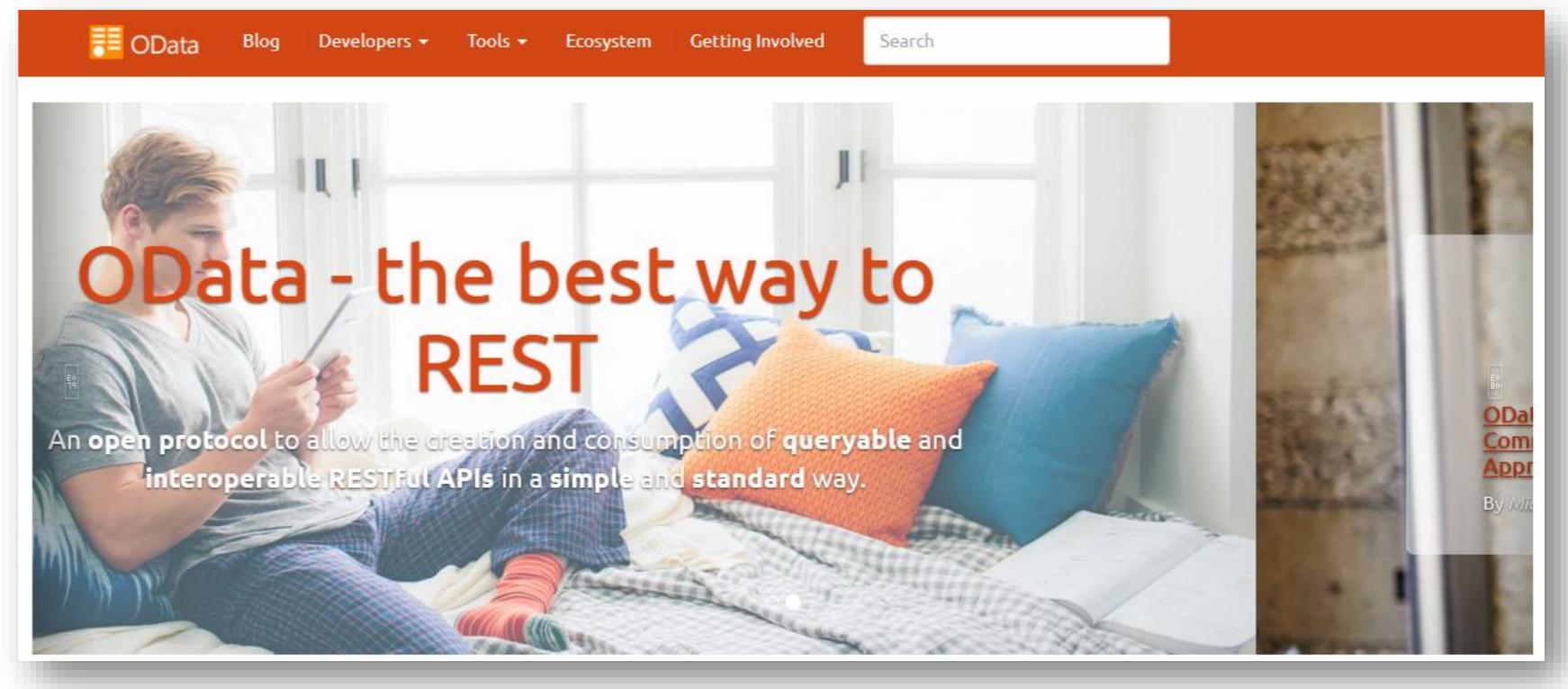
*Grundsätzlich könnte man jetzt im SAP ABAP auf dem Server auf Basis der geschaffenen CDS Fragmente die CDS Bibliothek weiter ausprägen, neue Consumption Views aufgrund neuer Anforderungen anlegen ...*

*... oder sich ab jetzt um Schnittstellen kümmern.*

# OData API

Der SAP Standard spricht  
am liebsten OData...

<https://www.odata.org>



# Der Glücksfall: OData per CDS Annotation



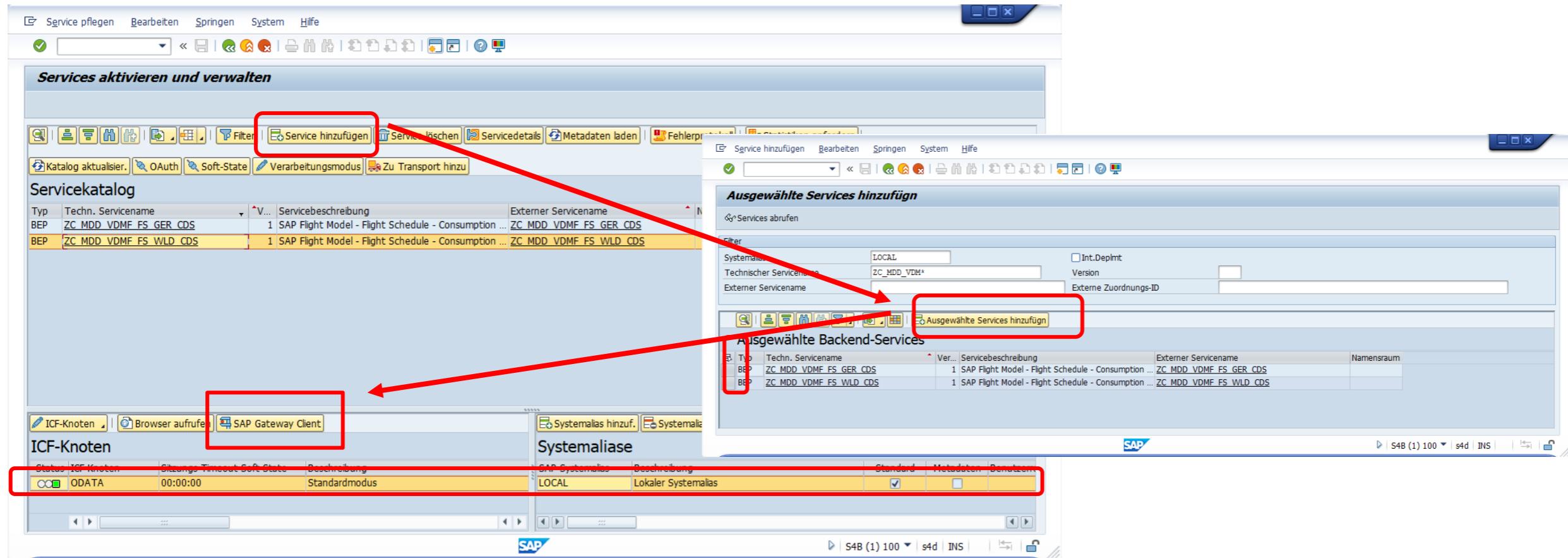
```
D [S4B] ZC_MDD_VDMF_FS_WLD ×

1 @AbapCatalog.sqlViewName: 'ZC_MDDVDMF_FSW'
2 @AbapCatalog.compiler.compareFilter: true
3 @AbapCatalog.preserveKey: true
4 @AccessControl.authorizationCheck: #CHECK
5 @EndUserText.label: 'SAP Flight Model - Flight Schedule - Consumption World'
6
7 @VDM.viewType: #CONSUMPTION
8 @OData.publish: true
9
10 define view ZC_MDD_VDMF_FS_WLD
11   as select from ZT_MDD_VDMF_FSchedule
```

- In neueren SAP Systemen wird die notwendige OData API per Annotation „@OData.publish: true“ generiert
- **Achtung:**  
Der OData-Service muss zusätzlich noch in der Transaktion „/WFND/MAINT\_SERVICE“ aktiviert werden (Sicherheitsaspekte)  
Die Warnung (links) weist darauf hin
- Nach der Aktivierung steht die neue OData API als Service auf dem SAP Server im Pfad „/sap/opu/odata/sap/\*“ zur Verfügung
- Für ältere Systeme gibt es Workarounds...

# OData Services aktivieren – /IWFND/MAINT\_SERVICE

BA



- OData Services werden danach – ggf. von der SAP Basis – in Transaktion /IWFND/MAINT\_SERVICE aktiviert
- Diese Transaktion stellt auch Testmöglichkeiten zur Verfügung (SAPGUI oder Browser)
- Voraussetzung: die Konfiguration (unterer Bereich) ist vollständig gepflegt

# OData Testtool im SAP



The image displays two side-by-side screenshots of the SAP Gateway Client interface, version S4D (4) 100.

**Left Screenshot (Request):**

- Toolbar:** SAP Gateway Client, Bearbeiten, Springen, Metadaten, System, Hilfe.
- Request-URI:** /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/?\$format=json
- HTTP-Request (Content):**

```

1  {
2    "d" : {
3      "EntitySets" : [
4        "ZC_MDD_VDMF_FS_GER",
5        "ZI_MDD_VDMF_Airport",
6        "ZI_MDD_VDMF_Carrier"
7      ]
8    }
9  }

```

**Right Screenshot (Response):**

- Request-URI:** /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/?\$format=xml
- HTTP-Response (Header):**

Header-Name	Wert
~status_code	200
~status_reason	OK
- HTTP-Response (Content):**

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<app:service xmlns:app="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:sap="http://www.sap.com/Protocols/SAPData"
  xml:lang="de" xml:base="http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/">
  <app:workspace>
    <atom:title type="text">Data</atom:title>
    <app:collection sap:createable="false" sap:updatable="false" sap:deletable="false" sap:content-version="1">
      <atom:title type="text">ZC_MDD_VDMF_FS_GER</atom:title>
      <atom:link href="ZC_MDD_VDMF_FS_GER" rel="self" type="application/atom+xml"></atom:link>
      <atom:member title="SAP Flight Model - Flight Schedule - Consumption Germany"></atom:member>
    </app:collection>
    <app:collection sap:createable="false" sap:updatable="false" sap:deletable="false" sap:content-version="1">
      <atom:title type="text">ZI_MDD_VDMF_Airport</atom:title>
      <atom:link href="ZI_MDD_VDMF_Airport" rel="self" type="application/atom+xml"></atom:link>
      <atom:member title="SAP Flight Model - Airport"></atom:member>
    </app:collection>
    <app:collection sap:createable="false" sap:updatable="false" sap:deletable="false" sap:content-version="1">
      <atom:title type="text">ZI_MDD_VDMF_Carrier</atom:title>
      <atom:link href="ZI_MDD_VDMF_Carrier" rel="self" type="application/atom+xml"></atom:link>
      <atom:member title="SAP Flight Model - Carrier"></atom:member>
    </app:collection>
  </app:workspace>
  <atom:link href="http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/" rel="latest-version"></atom:link>
  <atom:link href="http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/" rel="self" type="application/atom+xml"></atom:link>

```

- [http://10.17.1.12/sap/opu/odata/sap/ZC\\_MDD\\_VDMF\\_FS\\_WLD\\_CDS/?\\$format=xml](http://10.17.1.12/sap/opu/odata/sap/ZC_MDD_VDMF_FS_WLD_CDS/?$format=xml)
- /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZI\_MDD\_VDMF\_Airport('FRA')
- /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/?\$format=xml
- /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/?\$format=json
- /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZC\_MDD\_VDMF\_FS\_GER?\$format=json
- /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZC\_MDD\_VDMF\_FS\_GER('LH0400')?\$format=json

# OData Testtool im SAP (2) – Entities und Entity



The screenshot displays the SAP Gateway Client interface with two separate requests:

**Request 1 (Left):** GET /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZC\_MDD\_VDMF\_FS\_GER?\$format=json

**Request 2 (Right):** GET /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZC\_MDD\_VDMF\_FS\_GER('LH0400')?\$format=json

**HTTP-Response (Left):**

```
Header-Name Wert
~status_code 200
~status_reason OK
sap-processing-info ODataBEP+,crp+,RAL=,st=X,MedCacheHub=SHM,MedCacheBEP=SHM,codeployed=X,softstate=0

55 "DepartureTime" : "PT19H00M00S",
56 "CountryTo" : "DE",
57 "CityTo" : "FRANKFURT",
58 "AirportTo" : "FRA",
59 "ArrivalTime" : "PT21H05M00S",
60 "DaysLater" : 0,
61 "FlightTime" : 125,
62 "DistanceKM" : "1359.8956",
63 "DistanceKMH" : "405.60",
64 "FlightIsCharter" : "",
65 "FlightType" : "M",
66 "to_AirportFrom" : {
67     "_deferred" : {
68         "uri" : "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZC_MDD_VDMF_FS_GER('AZ0555')/to_"
69     }
70 },
71 "to_AirportTo" : {
72     "_deferred" : {
73         "uri" : "http://s4d.17.ucc.md/sap/cpu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZC_MDD_VDMF_FS_GER('AZ0555')/to_"
74     }
75 },
```

**HTTP-Response (Right):**

```
Header-Name Wert
~status_code 200
~status_reason OK
sap-processing-info ODataBEP+,crp+,RAL=,st=X,MedCacheHub=SHM,codeployed=X,softstate=0

1 {
2     "d" : {
3         "__metadata" : {
4             "id" : "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZC_MDD_VDMF_FS_GER('LH0400')",
5             "uri" : "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZC_MDD_VDMF_FS_GER('LH0400')",
6             "type" : "ZC_MDD_VDMF_FS_GER_CDS.ZC_MDD_VDMF_FS_GERTYPE"
7         },
8         "FlightNumber" : "LH0400",
9         "CarrierId" : "LH",
10        "ConnectionId" : "0400",
11        "CountryFrom" : "DE",
12        "CityFrom" : "FRANKFURT",
13        "AirportFrom" : "FRA",
14        "DepartureTime" : "PT10H10M00S",
15        "CountryTo" : "US",
16        "CityTo" : "NEW YORK",
17        "AirportTo" : "JFK",
18        "ArrivalTime" : "PT11H34M00S",
19        "DaysLater" : 0,
20        "FlightTime" : 444,
21        "DistanceKM" : "6162.0000",
```

# OData Testtool im SAP (3) – Assoziationen



The screenshot displays the SAP Gateway Client interface with two separate requests:

**Request 1 (Left):** /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZI\_MDD\_VDMF\_Carrier('LH')?format=json

**Request 2 (Right):** /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZI\_MDD\_VDMF\_Carrier?\$format=json

**HTTP-Response (Left):** Verarbeitungszeit = 54 ms

**HTTP-Request Headers:**

Header-Name	Wert
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,codeployed=X,softstate=
sap-metadata-last-updated	Wed, 11 May 2022 08:54:36 GMT

**HTTP-Response Body (JSON):**

```
{  
  "d": {  
    "_metadata": {  
      "id": "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZI_MDD_VDMF_Carrier('LH')",  
      "uri": "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZI_MDD_VDMF_Carrier('LH')",  
      "type": "ZC_MDD_VDMF_FS_GER_CDS.ZI_MDD_VDMF_CarrierType"  
    },  
    "CarrierId": "LH",  
    "CarrierName": "Lufthansa",  
    "CarrierCurrencyCode": "EUR",  
    "CarrierUrl": "http://www.lufthansa.com"  
  }  
}
```

**HTTP-Response (Right):** Verarbeitungszeit = 68 ms

**HTTP-Request Headers:**

Header-Name	Wert
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,MedCacheBEP=SHM,codeployed=,softstate=
sap-metadata-last-updated	Wed, 11 May 2022 08:54:36 GMT

**HTTP-Response Body (JSON):**

```
{  
  "d": {  
    "results": [  
      {  
        "_metadata": {  
          "id": "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZI_MDD_VDMF_Carrier('AA')",  
          "uri": "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZI_MDD_VDMF_Carrier('AA')",  
          "type": "ZC_MDD_VDMF_FS_GER_CDS.ZI_MDD_VDMF_CarrierType"  
        },  
        "CarrierId": "AA",  
        "CarrierName": "American Airlines",  
        "CarrierCurrencyCode": "USD",  
        "CarrierUrl": "http://www.aa.com"  
      },  
      {  
        "_metadata": {  
          "id": "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZI_MDD_VDMF_Carrier('AC')",  
          "uri": "http://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZI_MDD_VDMF_Carrier('AC')",  
          "type": "ZC_MDD_VDMF_FS_GER_CDS.ZI_MDD_VDMF_CarrierType"  
        },  
        "CarrierId": "AC",  
        "CarrierName": "Air Canada",  
        "CarrierCurrencyCode": "CAD",  
        "CarrierUrl": "http://www.aircanada.ca"  
      }  
    ]  
  }  
}
```

- /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZI\_MDD\_VDMF\_Carrier?\$format=json
- /sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZI\_MDD\_VDMF\_Carrier('LH')?format=json

# OData - Im Browser testen



The screenshot displays three browser tabs, each showing a different view of SAP OData results:

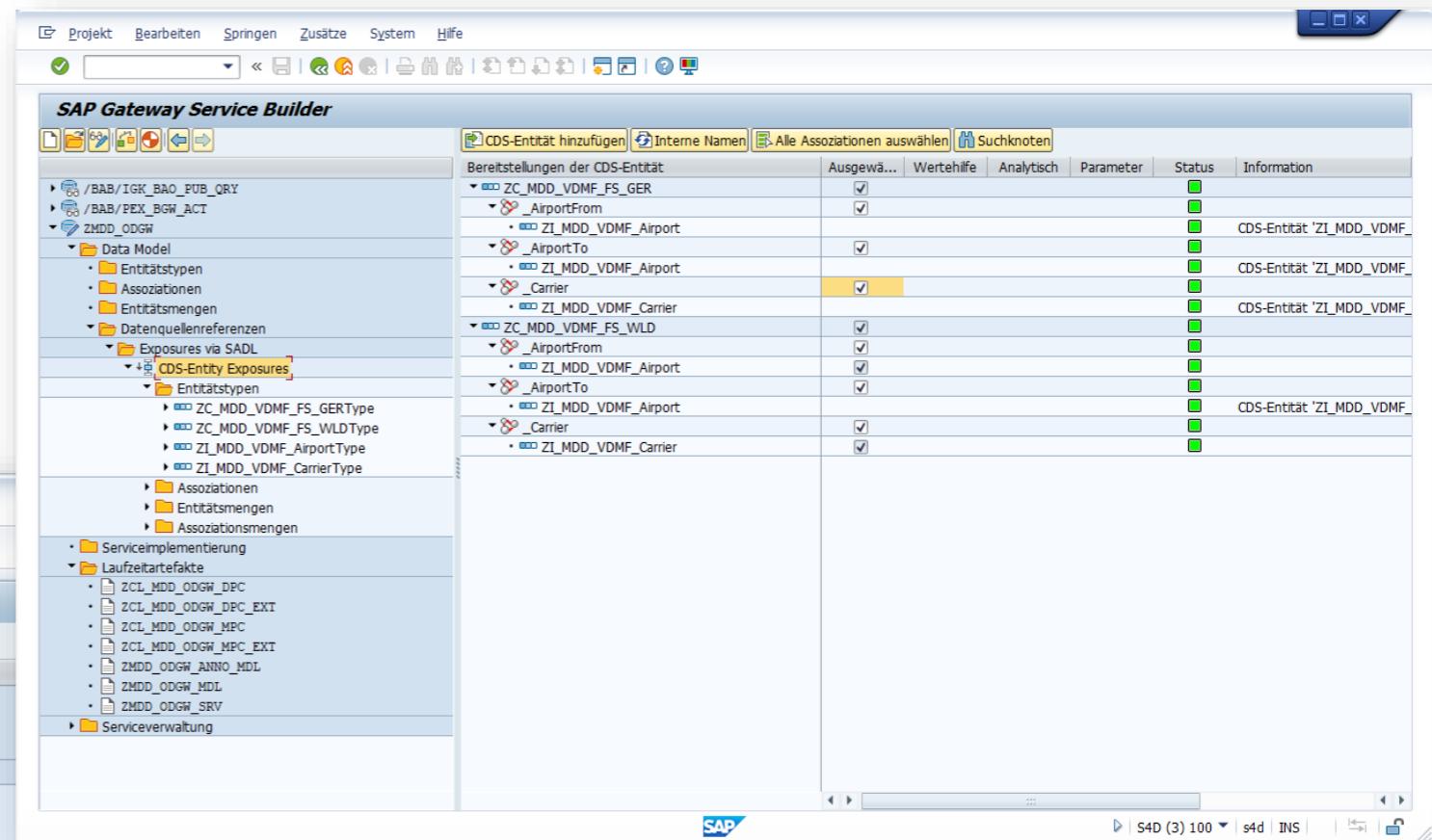
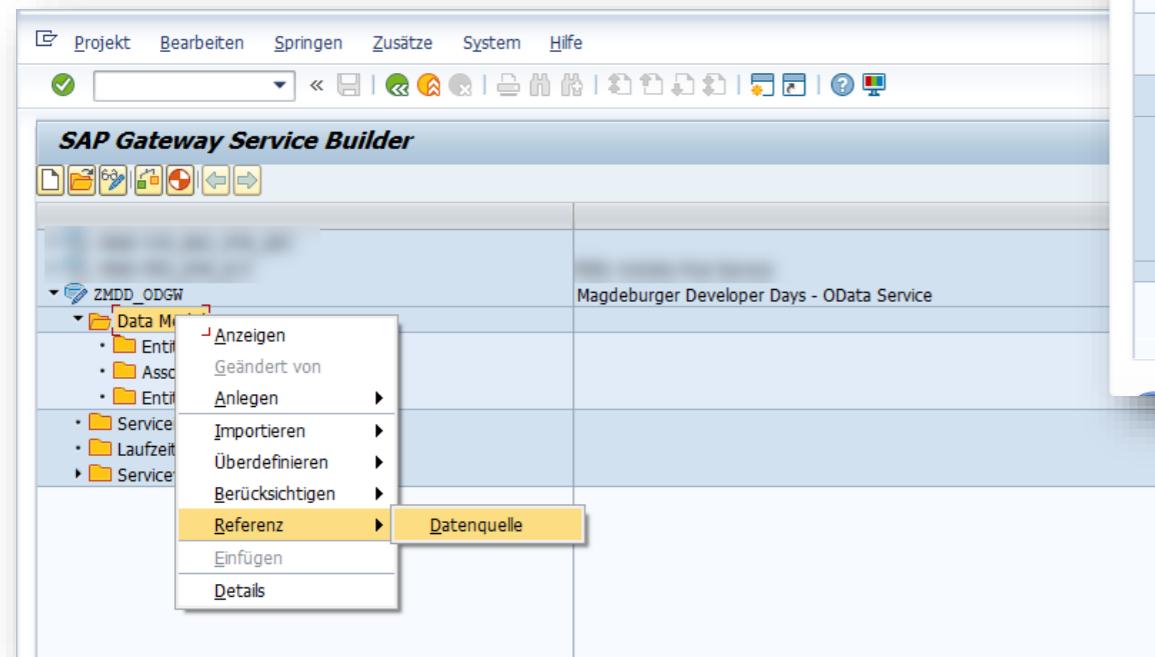
- Top Left Tab:** Shows the XML representation of the data. The page title is "Nicht sicher | https://s4d.17.ucc.md/sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/?sap-client=100". The content is a large XML document representing the entity sets and their metadata.
- Top Right Tab:** Shows the JSON representation of the data. The page title is "Nicht sicher | https://s4d.17.ucc.md/sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/ZC\_MDD\_VDMF\_FS\_GER?sap-client=100...". The content is a JSON object with the key "d" pointing to an array of entity set names: ["ZC\_MDD\_VDMF\_FS\_GER", "ZI\_MDD\_VDMF\_Airport", "ZI\_MDD\_VDMF\_Carrier"].
- Bottom Tab:** Shows the XML representation of the data again, similar to the top-left tab but with a different URL. The page title is "Nicht sicher | https://s4d.17.ucc.md/sap/opu/odata/sap/ZC\_MDD\_VDMF\_FS\_GER\_CDS/?sap-client=100&\$format=json". The content is a JSON object with the key "d" pointing to the same array of entity sets: {"d": {"EntitySets": ["ZC\_MDD\_VDMF\_FS\_GER", "ZI\_MDD\_VDMF\_Airport", "ZI\_MDD\_VDMF\_Carrier"]}}.

- [https://s4d.17.ucc.md/sap/opu/odata/sap/ZC\\_MDD\\_VDMF\\_FS\\_GER\\_CDS/?sap-client=100](https://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/?sap-client=100)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZC\\_MDD\\_VDMF\\_FS\\_GER\\_CDS/?sap-client=100&\\$format=json](https://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/?sap-client=100&$format=json)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZC\\_MDD\\_VDMF\\_FS\\_GER\\_CDS/ZC\\_MDD\\_VDMF\\_FS\\_GER?sap-client=100&\\$format=json](https://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZC_MDD_VDMF_FS_GER?sap-client=100&$format=json)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZC\\_MDD\\_VDMF\\_FS\\_GER\\_CDS/ZI\\_MDD\\_VDMF\\_Airport?sap-client=100&\\$format=json](https://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZI_MDD_VDMF_Airport?sap-client=100&$format=json)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZC\\_MDD\\_VDMF\\_FS\\_GER\\_CDS/ZI\\_MDD\\_VDMF\\_Airport\('FRA'\)?sap-client=100&\\$format=json](https://s4d.17.ucc.md/sap/opu/odata/sap/ZC_MDD_VDMF_FS_GER_CDS/ZI_MDD_VDMF_Airport('FRA')?sap-client=100&$format=json)

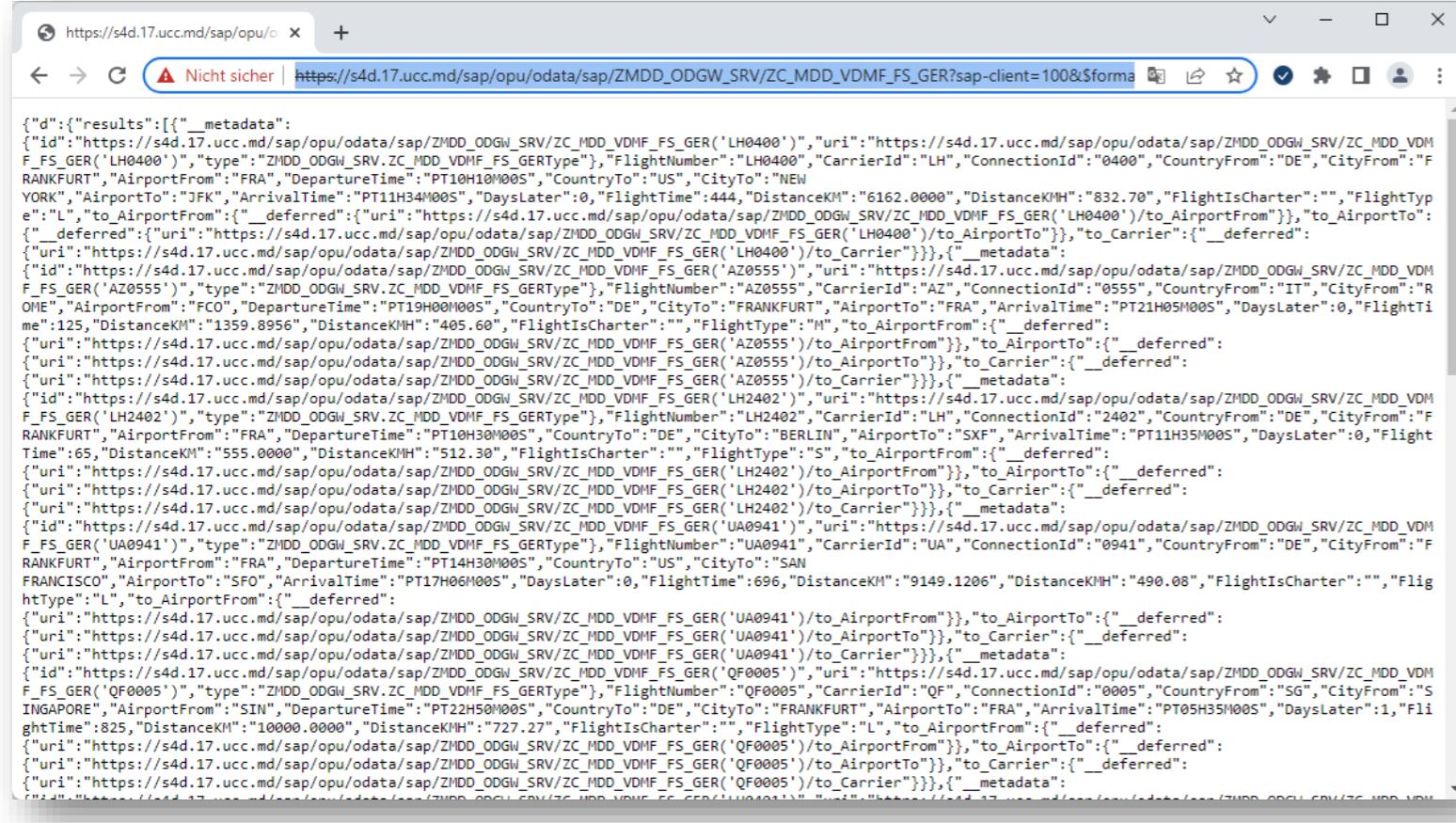
# Workaround - OData via SEGW



- In älteren Releases wurde für die Erstellung von OData-Services die Transaktion SEGW verwendet
- Diese Option können ältere Systeme in drei Varianten verwenden:
  - 1. „CDS Entity Exposure“ → neuere SAP Systeme
  - 2. Referenz auf CDS View → etwas ältere Systeme
  - 3. Eigene Implementierung → sehr alte Systeme
- Option 1 und 3 wurden für diese Demo implementiert



# Workaround: OData via SEGW – Test im Browser



- [https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD\\_ODGW\\_SRV/?sap-client=100](https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD_ODGW_SRV/?sap-client=100)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD\\_ODGW\\_SRV/ZC\\_MDD\\_VDMF\\_FS\\_GER?sap-client=100](https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD_ODGW_SRV/ZC_MDD_VDMF_FS_GER?sap-client=100)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD\\_ODGW\\_SRV/ZC\\_MDD\\_VDMF\\_FS\\_GER?sap-client=100](https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD_ODGW_SRV/ZC_MDD_VDMF_FS_GER?sap-client=100)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD\\_ODGW\\_SRV/ZC\\_MDD\\_VDMF\\_FS\\_GER?sap-client=100&\\$format=json](https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD_ODGW_SRV/ZC_MDD_VDMF_FS_GER?sap-client=100&$format=json)

# Workaround: OData via SEGW - für ältere Systeme



The screenshot illustrates the SAP Gateway Service Builder interface, specifically the process of creating a CDS Entity and associating it with a service implementation.

**SAP Gateway Service Builder (Main Window):**

- The left sidebar shows a project structure with nodes like /BAB/IGK\_BAO\_PUB\_QRY, /BAB/PEX\_BGW\_ACT, and ZMDD\_ODGW.
- The main area displays the "Bereitstellungen der CDS-Entität" (Deployment of CDS Entity) table. A row for "ZC\_MDD\_VDMF\_FSW" is selected, showing associations to "AirportFrom" and "Carrier".
- A context menu is open over the "FlightScheduleWorldSet" entry in the "Serviceimplementierung" folder, with options like "Anzeigen", "Geändert von", "Zu Datenquelle zuordnen", and "Details".

**Assistant - Schritt 1 von 3: Aus ABAP-Dictionary-Struktur importieren (Import Wizard Step 1 of 3):**

- The "Name" field contains "FlightScheduleWorld".
- The "Entitätstyp" radio button is selected.
- The "ABAP-Struktur" dropdown is set to "ZC\_MDD\_VDMF\_FSW".
- The "Standardentitätsmengen anlegen" checkbox is checked.
- Buttons at the bottom include "Weiter" (Next) and "Abbrechen" (Cancel).

# Workaround: OData via SEGW - für sehr alte Systeme



- Über das SEGW Projekt werden im Hintergrund ABAP Klassen generiert
- Die entsprechende \*\_DPC\_EXT Klasse enthält entsprechende Methoden für die modellierten Entitäten
- Diese sind bei Bedarf zu implementieren
- In den meisten Fällen reicht ein einfaches Coding (siehe rechts), mit denen man die CDS Views nutzen kann
- Sollten Filter, Sortierung oder Offset-Zugriff gewünscht sein, muss dies in dieser Variante selbst implementiert werden
- Tipps:
  - OData per SEGW bietet noch funktionale Aufrufe ohne Entität – genannt Function Import“, die wieder über Vererbung dieser Klasse implementiert werden
  - Selbst wenn man in neueren Systemen zuhause ist, kann sich eine SEGW Implementierung lohnen, da man hier die Services unter einem Endpunkt zusammen führen kann und auch den „Function Import“ für schreibende Zugriffe nutzen kann

The screenshot shows the SAP Class Builder interface for a class named ZCL\_MDD\_ODGW\_DPC\_EXT. The left pane displays the Repository Browser with the package ZMDD\_ODGW selected. The right pane shows the class structure and a method implementation.

**Class Structure:**

- Paket: ZMDD\_ODGW
- Objektname: ZCL\_MDD\_ODGW\_DPC\_EXT
- Beschreibung: Magdeburger Development Days
- Klassenbibliothek:
  - ZCL\_MDD\_ODGW\_DPC
  - ZCL\_MDD\_ODGW\_DPC\_EXT
    - Methoden:
      - FLIGHTSCHEDULEWO\_GET\_ENTITYSET
- Methoden:
  - FLIGHTSCHEDULEWO\_GET\_ENTITYSET
    - METHOD flightschedulewo\_get\_entityset.
      - \* WORKAROUND: NO FILTERS, SORTING, ... Available
      - SELECT \*  
FROM zc\_mdd\_vdmf\_fs\_wld  
INTO CORRESPONDING FIELDS OF TABLE @et\_entityset.

# Workaround: Odata via SEGW – Test für ältere Systeme



The screenshot shows a browser window with two tabs. The left tab displays an error message in JSON format:

```
{"error": {"code": "/IWBEP/CM_MGW_RT/021", "message": {"lang": "de", "value": "Methode 'FLIGHTSCHEDULEWO_GET_ENTITY' in Datenanbieterklasse nicht implementiert"}, "innererror": {"application": {"component_id": "", "service_namespace": "/SAP/", "service_id": "ZMDD_ODGW_SRV", "service_version": "0001"}, "transactionid": "D262607D47D40080E000625EAFF51FF6", "timestamp": "20220511094911", "Error_Resolution": {"SAP_Transaction": "For backend administrators: use ADT feed reader or Gateway Error Log\\ or run transaction /IWFND/ERROR_LOG on SAP Gateway hub system and see entries with the timestamp above for more details", "SAP_Note": "See SAP Note 1797736 error analysis (https://service.sap.com/sap/support/notes/1797736)"}, "errordetails": [{"code": "/IWBEP/CX_MGW_NOT_IMPL_EXC", "message": "Methode 'FLIGHTSCHEDULEWO_GET_ENTITY' in Datenanbieterklasse nicht implementiert", "propertyref": "", "severity": "error", "transition": false, "target": ""}]}}}
```

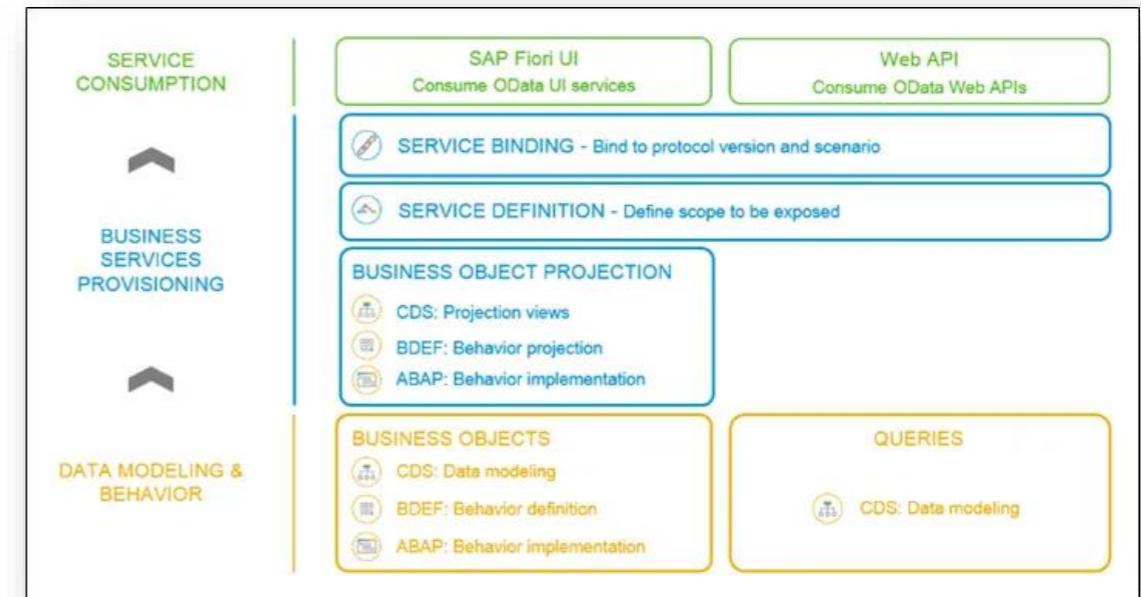
The right tab displays a large list of flight schedule entries in JSON format, showing various flight details like departure and arrival times, airports, and flight numbers.

- Methoden, die nicht implementiert wurden, führen zu solchen Fehlermeldungen
- Zugriff auf einzelnen Datensatz ist nicht implementiert, der Zugriff auf die Liste klappt
- Test-URLs:
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD\\_ODGW\\_SRV/?sap-client=100&\\$format=json](https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD_ODGW_SRV/?sap-client=100&$format=json)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD\\_ODGW\\_SRV/FlightScheduleWorldSet?sap-client=100&\\$format=json](https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD_ODGW_SRV/FlightScheduleWorldSet?sap-client=100&$format=json)
  - [https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD\\_ODGW\\_SRV/FlightScheduleWorldSet\('LH0400'\)?sap-client=100&\\$format=json](https://s4d.17.ucc.md/sap/opu/odata/sap/ZMDD_ODGW_SRV/FlightScheduleWorldSet('LH0400')?sap-client=100&$format=json)

# Das war es mit CDS und OData! - Zusatzinformationen



- OData über CDS Views deckt vor allem **lesende Zugriffe** ab
- **Schreibende Zugriffe** werden am besten über SEGW Projekte gelöst:
  - ODATA Methoden POST, PUT, DELETE implementieren
  - FUNCTION IMPORT verwenden
- Ab S/4 HANA 1909 (besser 2020) steht mit dem **RESTful ABAP Programming Model (RAP)** eine neue Technologie zur Verfügung mit der man auf Basis CDS CRUD Konzepte umsetzen kann
- Es gibt viele weitere **UI Annotationen**, die für die Gestaltung von generierten Oberflächen verwendet werden oder via OData auch für Schnittstellen verwendet werden können: z.B. Beschreibungstexte, Position in Liste usw.
- Über sogenannte **CDS Metadata Extensions** können UI Annotationen separat vom CDS View gepflegt werden
- Mit **FIORI Elements** werden auf Basis der CDS Annotationen FIORI Anwendungen generiert (z.B. auf Basis Visual Studio Code)
- Es gibt spezielle CDS Syntax und Erweiterungen für **Analytics...**

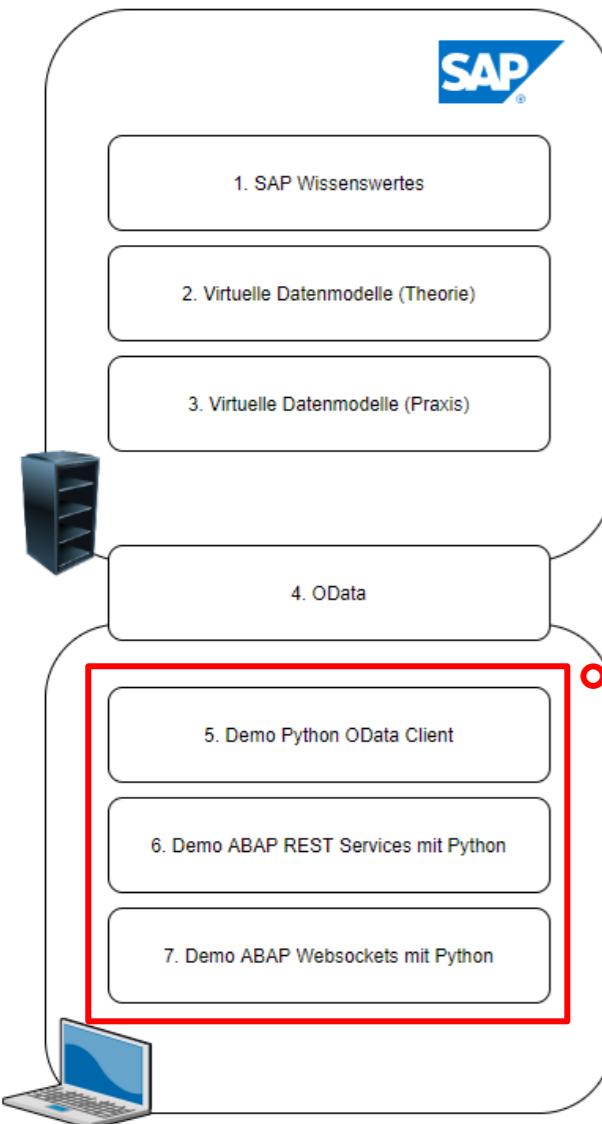


- Weitere Informationen
  - <https://github.com/MDJoerg/SapCdsWorkshopBasic.git>
  - <https://blogs.sap.com/2019/10/25/getting-started-with-the-abap-restful-programming-model/>
  - <https://blogs.sap.com/2021/10/18/modernization-with-rap/>



SAP Stammtisch CDS Workshop

# An SAP andocken ... mit Python – Demo Time!



## ▪ Warum Python?

- Java war mir zu aufwendig, Microsoft kann ich nicht...
- Python kommt im SAP Umfeld gerade in Mode
- Ich wollte noch mal eine neue Sprache lernen und Python mal selbst mit SAP Szenarien testen
- Tipp: openSAP Schulung → <https://open.sap.com/courses/python1/>
- Für Python gibt es auch Bibliotheken für OData:  
<https://www.odata.org/libraries/>

openSAP Über openSAP Kursbereiche Kurse Microlearning Podcasts Neuigkeiten Deutsch Anmelden Registrieren

Python for Beginners Christian Drumm, Stephan Jacobs

Kurs ist verfügbar

Lernmaterial Diskussionen Fortschritt Lernräume Kursdetails Ankündigungen

python™

Join this free online course to learn how to program with Python. You'll be introduced to the fundamentals of the programming language like variables, data types, and loops. More complex topics like functions, libraries, and file input and output will also be covered. At the end of the course, you'll be able to write simple Python programs to be prepared for your next programming challenges.

5 April 2022 - 1 June 2022 Kurssprache: English Für den Kurs einschreiben

Teilen Tweet Mitteilen Mail

Python for Beginners 03:54

# Python – Einfacher OData Call



```
import sap_context
import requests
import json

# build service url
odata_url = "http://" + sap_context.sap_host + ":" +
sap_context.sap_http_port
odata_url += sap_context.sap_odata_endpoint + "/" +
sap_context.sap_odata_flight_schedule_world +
sap_context.sap_odata_params
print("OData URL: ", odata_url)

# call sap odata service api
response = requests.get(odata_url)

if response.status_code != 200:
    print(f"Error {response.status_code}: {response.text}")
else:
    if response.text.find("{}") != 0:
        print("Response:", response.text)
        print("Response is not a JSON answer?!")
    else:
        json_response = json.loads(response.text)
        #print(json_response)
        print("JSON output detected")
        json_results = json_response["d"]["results"]
        print("====")
        index = 0
        for json_result in json_results:
            index += 1
            print("\nResult Index: ", index)
            for json_attribute in json_result:
                print(json_attribute, " = ", json_result[json_attribute])
```

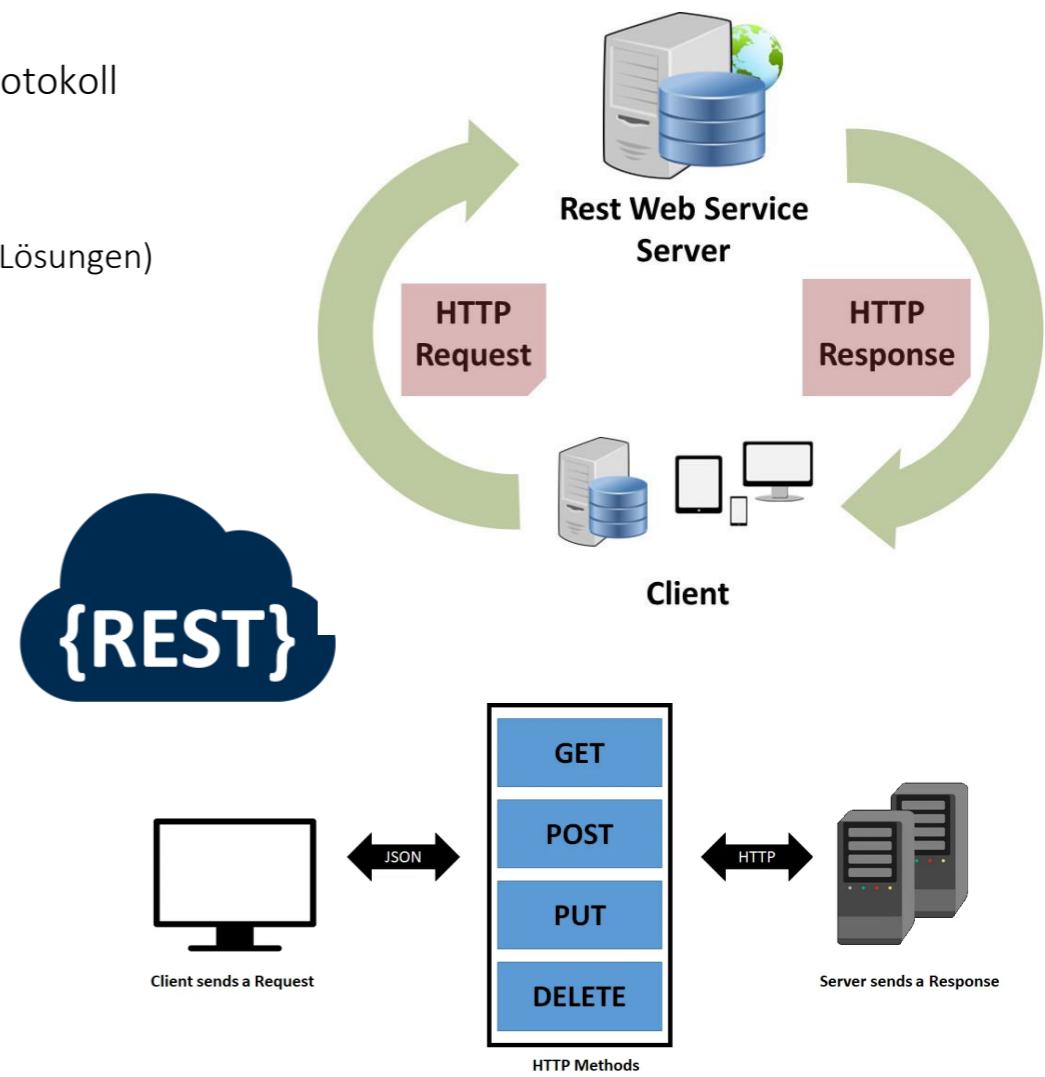
demo\_sap\_odata\_get\_simple.py

Result Index: 24  
...  
FlightNumber = SQ0988  
CarrierId = SQ  
ConnectionId = 0988  
CountryFrom = SG  
CityFrom = SINGAPORE  
AirportFrom = SIN  
DepartureTime = PT16H35M00S  
CountryTo = JP  
CityTo = TOKYO  
AirportTo = TYO  
ArrivalTime = PT00H15M00S  
DaysLater = 1  
FlightTime = 400  
DistanceKM = 5029.2000  
DistanceKMH = 468.75  
FlightIsCharter =  
FlightType = L  
...

# REST API – Let's talk about REST



- OData ist bereits ein **RESTful Webservice** – allerdings mit Vorgaben durch das OData Protokoll
- Neben OData, sind im SAP Umfeld auch **SOAP und andere Protokolle** üblich
  - Einen Überblick über mögliche APIs der SAP Lösungen findet man auf <https://api.sap.com>
  - Von den hier publizierten APIs profitieren vor allem neuere Lösungen (z.B. S/4 HANA, SAP Cloud Lösungen)
  - Für wichtige SAP Objekte im ERP Backend fehlen noch einige APIs
- SAP bietet auch in älteren Systemen die **ABAP REST Library (ARL)**
  - SAP Hilfe ABAP Rest Library (ARL)  
[https://help.sap.com/doc/saphelp\\_nw74/7.4.16/en-us/28/50217946b54e718e1f4afb35c4c283/content.htm?no\\_cache=true](https://help.sap.com/doc/saphelp_nw74/7.4.16/en-us/28/50217946b54e718e1f4afb35c4c283/content.htm?no_cache=true)
  - ARL SAP Tutorial  
[https://help.sap.com/doc/saphelp\\_nw74/7.4.16/en-us/0f/5fb77942744afe94afafa78df57b70/content.htm?no\\_cache=true](https://help.sap.com/doc/saphelp_nw74/7.4.16/en-us/0f/5fb77942744afe94afafa78df57b70/content.htm?no_cache=true)
  - Step By Step Tutorial für einfaches GET  
<https://de-veloped.de/?p=314>
  - Umfangreiches Tutorial für EPM Datenmodel (nicht nur GET)  
<https://developers4sap.blog/rest-services-in-abap/>



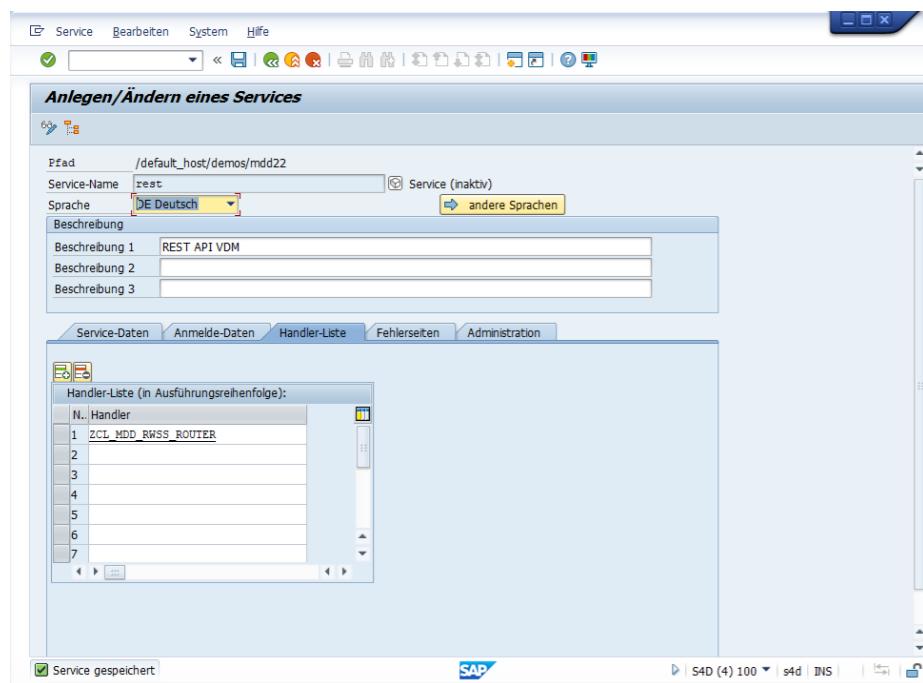
Bildquellen:

- <https://www.opc-router.de/was-ist-rest>
- <https://gocoding.org/de/Was-sind-ruhige-Webdienste%3F/>
- <https://restfulapi.net/>

# ABAP REST – Die „Magie“ - Router



1. Router Klasse anlegen
2. Router Klasse als Knoten in Transaktion SICF registrieren
3. Ressourcenklasse(n) anlegen
4. Im Router Zugriffspfade definieren und Ressourcenklasse(n) binden
5. Testen



```
METHOD if_rest_application~get_root_handler.  
* ----- constants  
DATA lc_carrier TYPE seoclsname VALUE 'ZCL_MDD_RWSS_RES_CARRIER'.  
DATA lc_airport TYPE seoclsname VALUE 'ZCL_MDD_RWSS_RES_AIRPORT'.  
DATA lc_fls_wld TYPE seoclsname VALUE 'ZCL_MDD_RWSS_RES_FLS_WORLD'.  
DATA lc_fls_ger TYPE seoclsname VALUE 'ZCL_MDD_RWSS_RES_FLS_GER'.  
* ----- init  
DATA(lr_router) = NEW cl_rest_router( ).  
ro_root_handler = lr_router.  
* ----- add path  
* carrier  
lr_router->attach( iv_template = '/Carrier' iv_handler_class = lc_carrier ).  
lr_router->attach( iv_template = '/Carrier/{id}' iv_handler_class = lc_carrier ).  
* airport  
lr_router->attach( iv_template = '/Airport' iv_handler_class = lc_airport ).  
lr_router->attach( iv_template = '/Airport/{id}' iv_handler_class = lc_airport ).  
* Flight Schedule  
* World  
lr_router->attach( iv_template = '/FlightSchedule' iv_handler_class = lc_fls_wld ).  
lr_router->attach( iv_template = '/FlightSchedule/{id}' iv_handler_class = lc_fls_wld ).  
* Germany  
lr_router->attach( iv_template = '/FlightSchedule/Germany' iv_handler_class = lc_fls_ger ).  
lr_router->attach( iv_template = '/FlightSchedule/Germany/{id}' iv_handler_class = lc_fls_ger ).  
ENDMETHOD.
```

# ABAP REST – Die „Magie“ (2) – READ



- Die Ressourcenklassen redefinieren die entsprechenden HTTP-Methoden (z.B. GET, POST, DELETE)
- Der Zugriff auf die CDS Views der Interface Layer Schicht ist möglich
- Die get\_param und send\_\*-Methoden werden von einer eigenen (Zwischen-)Klasse über Vererbung bereitgestellt

Paket  ZMDD

Objektname	Beschreibung
ZMDD	Magdeburger Development Days 2022
Unterpakete	
ZMDD_ODGW	Magdeburger Development Days - OData Gateway
ZMDD_RWSS	Magdeburger Development Days - Restful WebService Server
Klassenbibliothek	
Klassen	
ZCL_MDD_RWSS_RES	MDD22 - REST Ressource Base
ZCL_MDD_RWSS_RES_AIRPORT	MDD22 - REST Ressource Airport
Superklassen	
Attribute	
Methoden	
Geerbte Methoden	
Redefinitionen	
IF_REST_RESOURCE~DELETE	DELETE Method
IF_REST_RESOURCE~GET	GET Method
IF_REST_RESOURCE~POST	POST Method
Typen	
ZCL_MDD_RWSS_RES_CARRIER	MDD22 - REST Resource Carrier
ZCL_MDD_RWSS_RES_FLS_GER	MDD22 - REST Ressource Flight Schedule - Germany
ZCL_MDD_RWSS_RES_FLS_WORLD	MDD22 - REST Ressource Flight Schedule - World
ZCL_MDD_RWSS_ROUTER	MDD22 - Example REST Router
Superklassen	
Attribute	
Methoden	
Geerbte Methoden	
Redefinitionen	
IF_REST_APPLICATION~GET_ROOT_HANDLER	Creates the REST Resource Tree
HANDLE_CSRF_TOKEN	Handle CSRF token
Typen	
ZMDD_VDMF	Magdeburger Development Days - VDM SAP Flight Data Model
ZMDD_WSPS	Magdeburger Development Days - WebSocket Push Server

ABAP Code in Paket ZMDD\_RWSS - Magdeburger Development Days - Restful WebService Server

```

METHOD if_rest_resource~get.

* ----- check params
DATA(lv_id) = get_param( 'id' ).

* ----- execute request
IF lv_id IS INITIAL.
* -- set request
SELECT *
FROM zi_mdd_vdmf_airport
INTO TABLE @DATA(lt_data).

IF lt_data IS NOT INITIAL.
  send_as_json( lt_data ).
ELSE.
  send_error( ).
ENDIF.

* -- single request
SELECT SINGLE *
FROM zi_mdd_vdmf_airport
INTO @DATA(ls_data)
WHERE airportid = @lv_id.

IF ls_data IS NOT INITIAL.
  send_as_json( ls_data ).
ELSE.
  send_error( ).
ENDIF.

ENDIF.

ENDMETHOD.
```

# ABAP REST – Die „Magie“ (3) - CREATE



```
METHOD if_rest_resource~post.  
* ----- local data  
  DATA: ls_data TYPE sairport.  
  
* ----- check id  
  DATA(lv_id) = get_param( 'id' ).  
  IF lv_id IS INITIAL.  
    send_error( ).  
  ELSE.  
* ----- check json content  
  DATA(lv_json) = io_entity->get_string_data( ).  
  IF lv_json IS INITIAL OR lv_json(1) <> '{'.  
    send_error( ).  
* ----- check json format  
  ELSE.  
    cl_fdt_json=>json_to_data(  
      EXPORTING  
        iv_json = lv_json  
      IMPORTING  
        ev_meta =  
      CHANGING  
        ca_data = ls_data  
    ).  
    IF ls_data IS INITIAL  
      OR ls_data-name IS INITIAL.  
      send_error( ).  
    ELSE.  
      * ----- check existing  
      SELECT SINGLE id  
        FROM sairport  
        INTO ls_data-id  
        WHERE id = lv_id.  
      IF sy-subrc EQ 0.  
        send_error( ). " exists  
      ELSE.  
        * ----- create and fill output  
        * prepare  
        ls_data-mandt = sy-mandt.  
        ls_data-id = lv_id.  
        IF ls_data-time_zone IS INITIAL.  
          ls_data-time_zone = 'UTC'.  
        ENDIF.  
        * update database  
        MODIFY sairport FROM ls_data.  
        COMMIT WORK.  
        * send json output  
        send_as_json( ls_data ).  
      ENDIF.  
    ENDIF.  
  ENDIF.  
ENDMETHOD.
```

- CREATE Beispiel als ABAP Code (Paket ZMDD\_RWSS)
- Code aus ZCL\_MDD\_RWSS\_RES\_AIRPORT Methode IF\_REST\_RESOURCE~POST

# ABAP REST – Testen im Browser



"FLIGHTNUMBER": "LH0400", "CARRIERID": "LH", "CONNECTIONID": "0400", "COUNTRYFROM": "DE", "CITYFROM": "FRANKFURT", "AIRPORTFROM": "FRA", "ARRIVALTIME": "101000", "COUNTRYTO": "US", "CITYTO": "NEW YORK", "AIRPORTTO": "JFK", "ARRIVALTIME": "113400", "DAYSLATER": 0, "FLIGHTTIME": 444, "DISTANCEKM": 6162.0000, "DISTANCEKH": 832.70, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "AUA0017", "CARRIERID": "AA", "CONNECTIONID": "0017", "COUNTRYFROM": "US", "CITYFROM": "NEW YORK", "AIRPORTFROM": "JFK", "ARRIVALTIME": "110000", "COUNTRYTO": "US", "CITYTO": "SAN FRANCISCO", "AIRPORTTO": "SFO", "ARRIVALTIME": "141000", "DAYSLATER": 0, "FLIGHTTIME": 361, "DISTANCEKM": 4139.2327, "DISTANCEKH": 427.47, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "AZ0055", "CARRIERID": "AZ", "CONNECTIONID": "0555", "COUNTRYFROM": "US", "CITYFROM": "ROSE", "AIRPORTFROM": "FCO", "ARRIVALTIME": "190000", "COUNTRYTO": "DE", "CITYTO": "FRANKFURT", "AIRPORTTO": "FRA", "ARRIVALTIME": "210500", "DAYSLATER": 0, "FLIGHTTIME": 125, "DISTANCEKM": 1359.8596, "DISTANCEKH": 405.60, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "M"}, {"FLIGHTNUMBER": "LH2402", "CARRIERID": "LH", "CONNECTIONID": "2402", "COUNTRYFROM": "DE", "CITYFROM": "FRANKFURT", "AIRPORTFROM": "FRA", "ARRIVALTIME": "103000", "COUNTRYTO": "DE", "CITYTO": "BERLIN", "AIRPORTTO": "SXF", "ARRIVALTIME": "113500", "DAYSLATER": 0, "FLIGHTTIME": 65, "DISTANCEKM": 555.0000, "DISTANCEKH": 512.30, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "S"}, {"FLIGHTNUMBER": "UA0941", "CARRIERID": "UA", "CONNECTIONID": "0941", "COUNTRYFROM": "DE", "CITYFROM": "FRANKFURT", "AIRPORTFROM": "FRA", "ARRIVALTIME": "143000", "COUNTRYTO": "US", "CITYTO": "SAN FRANCISCO", "AIRPORTTO": "SFO", "ARRIVALTIME": "170600", "DAYSLATER": 0, "FLIGHTTIME": 696, "DISTANCEKM": 9149.1206, "DISTANCEKH": 1498.08, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "AUA0789", "CARRIERID": "AZ", "CONNECTIONID": "0789", "COUNTRYFROM": "DE", "CITYFROM": "TOKYO", "AIRPORTFROM": "TYO", "ARRIVALTIME": "114500", "COUNTRYTO": "IT", "CITYTO": "ROMA", "AIRPORTTO": "FCO", "ARRIVALTIME": "190000", "DAYSLATER": 0, "FLIGHTTIME": 940, "DISTANCEKM": 985.2787, "DISTANCEKH": 99.27, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "AUA0012", "CARRIERID": "AA", "CONNECTIONID": "0012", "COUNTRYFROM": "DE", "CITYFROM": "FRANKFURT", "AIRPORTFROM": "FRA", "ARRIVALTIME": "133000", "COUNTRYTO": "US", "CITYTO": "NEW YORK", "AIRPORTTO": "JFK", "ARRIVALTIME": "180000", "DAYSLATER": 0, "FLIGHTTIME": 455, "DISTANCEKM": 6162.0000, "DISTANCEKH": 812.57, "FLIGHTISCHARTER": "X", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "OP0007", "CARRIERID": "OP", "CONNECTIONID": "0005", "COUNTRYFROM": "SG", "CITYFROM": "SINGAPORE", "AIRPORTFROM": "SIN", "ARRIVALTIME": "225000", "COUNTRYTO": "DE", "CITYTO": "FRANKFURT", "AIRPORTTO": "FRA", "ARRIVALTIME": "053500", "DAYSLATER": 1, "FLIGHTTIME": 825, "DISTANCEKM": 10000.0000, "DISTANCEKH": 727.27, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "UQ0005", "CARRIERID": "SQ", "CONNECTIONID": "0015", "COUNTRYFROM": "US", "CITYFROM": "CITY", "ARRIVALTIME": "160000", "COUNTRYTO": "SG", "CITYTO": "SINGAPORE", "AIRPORTTO": "SIN", "ARRIVALTIME": "024500", "DAYSLATER": 2, "FLIGHTTIME": 1125, "DISTANCEKM": 13602.1754, "DISTANCEKH": 450.77, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "SQ0002", "CARRIERID": "SQ", "CONNECTIONID": "0002", "COUNTRYFROM": "SG", "CITYFROM": "SINGAPORE", "AIRPORTFROM": "SIN", "ARRIVALTIME": "170000", "COUNTRYTO": "US", "CITYTO": "SAN FRANCISCO", "AIRPORTTO": "SFO", "ARRIVALTIME": "192500", "DAYSLATER": 0, "FLIGHTTIME": 1105, "DISTANCEKM": 13602.1754, "DISTANCEKH": 458.93, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "LH0401", "CARRIERID": "LH", "CONNECTIONID": "0401", "COUNTRYFROM": "US", "CITYFROM": "NEW YORK", "AIRPORTFROM": "JFK", "ARRIVALTIME": "183000", "COUNTRYTO": "DE", "CITYTO": "FRANKFURT", "AIRPORTTO": "FRA", "ARRIVALTIME": "074500", "DAYSLATER": 1, "FLIGHTTIME": 435, "DISTANCEKM": 6162.0000, "DISTANCEKH": 849.93, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "DOL0106", "CARRIERID": "DL", "CONNECTIONID": "0106", "COUNTRYFROM": "US", "CITYFROM": "NEW YORK", "AIRPORTFROM": "JFK", "ARRIVALTIME": "193500", "COUNTRYTO": "DE", "CITYTO": "FRANKFURT", "AIRPORTTO": "FRA", "ARRIVALTIME": "093000", "DAYSLATER": 1, "FLIGHTTIME": 475, "DISTANCEKM": 6197.5837, "DISTANCEKH": 486.44, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "JL0407", "CARRIERID": "JL", "CONNECTIONID": "0407", "COUNTRYFROM": "JP", "CITYFROM": "TOKYO", "AIRPORTFROM": "NRT", "ARRIVALTIME": "133000", "COUNTRYTO": "DE", "CITYTO": "FRANKFURT", "AIRPORTTO": "FRA", "ARRIVALTIME": "173500", "DAYSLATER": 0, "FLIGHTTIME": 725, "DISTANCEKM": 9100.0000, "DISTANCEKH": 753.10, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "JL0408", "CARRIERID": "JL", "CONNECTIONID": "0408", "COUNTRYFROM": "DE", "CITYFROM": "FRANKFURT", "AIRPORTFROM": "FRA", "ARRIVALTIME": "202500", "COUNTRYTO": "JP", "CITYTO": "TOKYO", "AIRPORTTO": "NRT", "ARRIVALTIME": "154000", "DAYSLATER": 1, "FLIGHTTIME": 675, "DISTANCEKM": 9100.0000, "DISTANCEKH": 808.80, "FLIGHTISCHARTER": "X", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "AA0046", "CARRIERID": "AA", "CONNECTIONID": "0064", "COUNTRYFROM": "US", "CITYFROM": "SAN FRANCISCO", "AIRPORTFROM": "SFO", "ARRIVALTIME": "090000", "COUNTRYTO": "US", "CITYTO": "NEW YORK", "AIRPORTTO": "JFK", "ARRIVALTIME": "182500", "DAYSLATER": 0, "FLIGHTTIME": 321, "DISTANCEKM": 4139.2327, "DISTANCEKH": 480.74, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "M"}, {"FLIGHTNUMBER": "DL1699", "CARRIERID": "DL", "CONNECTIONID": "1699", "COUNTRYFROM": "US", "CITYFROM": "NEW YORK", "AIRPORTFROM": "JFK", "ARRIVALTIME": "171500", "COUNTRYTO": "US", "CITYTO": "SAN FRANCISCO", "AIRPORTTO": "SFO", "ARRIVALTIME": "093000", "DAYSLATER": 1, "FLIGHTTIME": 321, "DISTANCEKM": 4139.2327, "DISTANCEKH": 483.97, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "DL1984", "CARRIERID": "DL", "CONNECTIONID": "1984", "COUNTRYFROM": "US", "CITYFROM": "SAN FRANCISCO", "AIRPORTFROM": "SFO", "ARRIVALTIME": "100000", "COUNTRYTO": "US", "CITYTO": "NEW YORK", "AIRPORTTO": "JFK", "ARRIVALTIME": "182500", "DAYSLATER": 0, "FLIGHTTIME": 325, "DISTANCEKM": 4139.2327, "DISTANCEKH": 474.83, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "M"}, {"FLIGHTNUMBER": "LH2407", "CARRIERID": "LH", "CONNECTIONID": "2407", "COUNTRYFROM": "DE", "CITYFROM": "BERLIN", "AIRPORTFROM": "TXL", "ARRIVALTIME": "071000", "COUNTRYTO": "DE", "CITYTO": "FRANKFURT", "AIRPORTTO": "FRA", "ARRIVALTIME": "150000", "DAYSLATER": 0, "FLIGHTTIME": 165, "DISTANCEKM": 355.0000, "DISTANCEKH": 512.30, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "S"}, {"FLIGHTNUMBER": "UA0564", "CARRIERID": "UA", "CONNECTIONID": "5504", "COUNTRYFROM": "US", "CITYFROM": "CITY", "ARRIVALTIME": "150000", "COUNTRYTO": "DE", "CITYTO": "FRANKFURT", "AIRPORTTO": "FRA", "ARRIVALTIME": "071000", "DAYSLATER": 1, "FLIGHTTIME": 775, "DISTANCEKM": 9074.3443, "DISTANCEKH": 443.92, "FLIGHTISCHARTER": "X", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "AUA0786", "CARRIERID": "AA", "CONNECTIONID": "0786", "COUNTRYFROM": "IT", "CITYFROM": "ROME", "AIRPORTFROM": "FCO", "ARRIVALTIME": "120000", "COUNTRYTO": "JP", "CITYTO": "TOKYO", "AIRPORTTO": "TYO", "ARRIVALTIME": "0800", "DAYSLATER": 1, "FLIGHTTIME": 445, "DISTANCEKM": 985.2787, "DISTANCEKH": 474.58, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "AUA0798", "CARRIERID": "AA", "CONNECTIONID": "0798", "COUNTRYFROM": "IT", "CITYFROM": "ROME", "AIRPORTFROM": "FCO", "ARRIVALTIME": "103500", "COUNTRYTO": "JP", "CITYTO": "OSAKA", "AIRPORTTO": "KIX", "ARRIVALTIME": "0800", "DAYSLATER": 1, "FLIGHTTIME": 815, "DISTANCEKM": 9704.3443, "DISTANCEKH": 443.92, "FLIGHTISCHARTER": "X", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "OP0005", "CARRIERID": "OP", "CONNECTIONID": "0006", "COUNTRYFROM": "SG", "CITYFROM": "CITY", "ARRIVALTIME": "205500", "COUNTRYTO": "SG", "CITYTO": "SINGAPORE", "AIRPORTTO": "SIN", "ARRIVALTIME": "150500", "DAYSLATER": 1, "FLIGHTTIME": 670, "DISTANCEKM": 10000.0000, "DISTANCEKH": 895.52, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "SQ0158", "CARRIERID": "SQ", "CONNECTIONID": "0158", "COUNTRYFROM": "SG", "CITYFROM": "SINGAPORE", "AIRPORTFROM": "SIN", "ARRIVALTIME": "152500", "COUNTRYTO": "ID", "CITYTO": "JAKARTA", "AIRPORTTO": "JKT", "ARRIVALTIME": "160000", "DAYSLATER": 0, "FLIGHTTIME": 195, "DISTANCEKM": 9081.2326, "DISTANCEKH": 353.6, "FLIGHTISCHARTER": "X", "FLIGHTTYPE": "S"}, {"FLIGHTNUMBER": "UA0569", "CARRIERID": "UA", "CONNECTIONID": "0988", "COUNTRYFROM": "SG", "CITYFROM": "SINGAPORE", "AIRPORTFROM": "SIN", "ARRIVALTIME": "163500", "COUNTRYTO": "JP", "CITYTO": "TOKYO", "AIRPORTTO": "TYO", "ARRIVALTIME": "001500", "DAYSLATER": 1, "FLIGHTTIME": 400, "DISTANCEKM": 5029.2000, "DISTANCEKH": 468.75, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"}, {"FLIGHTNUMBER": "UA0516", "CARRIERID": "UA", "CONNECTIONID": "5116", "COUNTRYFROM": "US", "CITYFROM": "NEW YORK", "AIRPORTFROM": "JFK", "ARRIVALTIME": "162000", "COUNTRYTO": "DE", "CITYTO": "FRANKFURT", "AIRPORTTO": "FRA", "ARRIVALTIME": "054500", "DAYSLATER": 1, "FLIGHTTIME": 445, "DISTANCEKM": 6162.0000, "DISTANCEKH": 830.63, "FLIGHTISCHARTER": "", "FLIGHTTYPE": "L"},

## ■ Test-URLs

- <https://s4d.17.ucc.md/demos/mdd22/rest/Carrier/SR?sap-client=100>
  - <https://s4d.17.ucc.md/demos/mdd22/rest/Carrier?sap-client=100>
  - <https://s4d.17.ucc.md/demos/mdd22/rest/Airport?sap-client=100>
  - <https://s4d.17.ucc.md/demos/mdd22/rest/Airport/FRA?sap-client=100>
  - <https://s4d.17.ucc.md/demos/mdd22/rest/FlightSchedule?sap-client=100>
  - <https://s4d.17.ucc.md/demos/mdd22/rest/FlightSchedule/DL1699?sap-client=100>
  - <https://s4d.17.ucc.md/demos/mdd22/rest/FlightSchedule/Germany?sap-client=100>
  - <https://s4d.17.ucc.md/demos/mdd22/rest/FlightSchedule/Germany/LH0400?sap-client=100>
  - <https://s4d.17.ucc.md/demos/mdd22/rest/FlightSchedule/Germany/DL1699?sap-client=100>

# ABAP REST – Python Demo „Flugplan“



```
import sap_context
import requests
import json

# optional user input
rest_id = input("Optional: Enter a ID: ")

# build service url
rest_url = "http://" + sap_context.sap_host + ":" +
sap_context.sap_http_port
rest_url += sap_context.sap_rest_endpoint + "/" +
sap_context.sap_rest_flight_schedule_world
if rest_id != "":
    rest_url += "/" + rest_id
rest_url += sap_context.sap_rest_params
print("REST URL: ", rest_url)

# call sap odata service api
response = requests.get(rest_url)

if response.status_code != 200:
    print(f"Error {response.status_code}: {response.text}")
else:
    if response.text.find("{") != 0 and response.text.find("[") != 0:
        print("Response:", response.text)
        print("Response is not a JSON answer?!")
    else:
        json_response = json.loads(response.text)
        print(json_response)
```

> python demo\_sap\_rest\_get\_simple.py

Optional: Enter a ID: LH0400

REST URL:

http://10.17.1.12:80/demos/mdd22/rest/FlightSchedule/LH0400?sap-client=100

{'FLIGHTNUMBER': 'LH0400', 'CARRIERID': 'LH', 'CARRIERNAME': 'Lufthansa', 'CARRIERURL': 'http://www.lufthansa.com', 'CONNECTIONID': '0400', 'COUNTRYFROM': 'DE', 'CITYFROM': 'FRANKFURT', 'AIRPORTFROM': 'FRA', 'AIRPORTFROMNAME': 'Frankfurt/Main, FRG', 'DEPARTURETIME': '101000', 'COUNTRYTO': 'US', 'CITYTO': 'NEW YORK', 'AIRPORTTO': 'JFK', 'AIRPORTTONAME': 'New York JF Kennedy, USA', 'ARRIVALTIME': '113400', 'DAYSLATER': 0, 'FLIGHETIME': 444, 'DISTANCEKM': 6162.0, 'DISTANCEKMH': 832.7, 'FLIGHTISCHARTER': "", 'FLIGHTTYPE': 'L'}

> python demo\_sap\_rest\_get\_simple.py

Optional: Enter a ID:

REST URL:

http://10.17.1.12:80/demos/mdd22/rest/FlightSchedule?sap-client=100

[{'FLIGHTNUMBER': 'LH0400', 'CARRIERID': 'LH', 'CARRIERNAME': 'Lufthansa', 'CARRIERURL': 'http://www.lufthansa.com' ...]

# ABAP REST – Python Demo CR(U)D - Read



```
import sap_context
import requests
import json

# optional user input
rest_id = input("Enter a Airport ID: ")
if rest_id == "":
    quit()

# build service url
rest_url = "http://" + sap_context.sap_host + ":" + sap_context.sap_http_port
rest_url += sap_context.sap_rest_endpoint + "/" +
sap_context.sap_rest_airport
rest_url += "/" + rest_id + sap_context.sap_rest_params
print("REST URL: ", rest_url)

# call sap odata service api
response = requests.get(rest_url)

if response.status_code != 200:
    print(f"Error {response.status_code}: {response.text}")
else:
    if response.text.find("{") != 0 and response.text.find("[") != 0:
        print("Response:", response.text)
        print("Response is not a JSON answer?!")
    else:
        json_response = json.loads(response.text)
        print(json_response)
```

> python demo\_sap\_rest\_airport\_read.py

Enter a Airport ID: FRA

REST URL:

http://10.17.1.12:80/demos/mdd22/rest/Airport/FRA?sap-client=100

{'AIRPORTID': 'FRA', 'AIRPORTNAME': 'Frankfurt/Main, FRG',  
'AIRPORTTIMEZONE': 'UTC+1'}

# ABAP REST – Python Demo CR(U)D - Create



```
import sap_context
import requests
import json

# ask user for new airport
airport_id = input("Enter the airport ID: ")
if airport_id == "":
    quit()

airport_name = input("Enter the airport name: ")
if airport_name == "":
    quit()

airport_timezone = input("Enter the airport timezone: ")

# build service url
rest_url = "http://" + sap_context.sap_host + ":" + sap_context.sap_http_port
rest_url += sap_context.sap_rest_endpoint + "/" + sap_context.sap_rest_airport
rest_url += "/" + airport_id + sap_context.sap_rest_params
print("REST URL: ", rest_url)
```

```
# build json
rest_data = {
    "NAME" : airport_name,
    "TIME_ZONE" : airport_timezone
}
rest_payload = json.dumps(rest_data)
print("REST Payload:", rest_payload)

# call sap odata service api
response = requests.post(rest_url, rest_payload)

if response.status_code != 200:
    print(f"Error {response.status_code}: {response.text}")
else:
    if response.text.find("{") != 0 and response.text.find("[") != 0:
        print("Response:", response.text)
        print("Response is not a JSON answer?!")
    else:
        json_response = json.loads(response.text)
        print("REST Response:", json_response)
```

> python demo\_sap\_rest\_airport\_create.py

Enter the airport ID: ZZZ

Enter the airport name: ZZZ Airport

Enter the airport timezone:

REST URL: http://10.17.1.12:80/demos/mdd22/rest/Airport/ZZZ?sap-client=100

REST Payload: {"NAME": "ZZZ Airport", "TIME\_ZONE": ""}

REST Response: {'MANDT': '100', 'ID': 'ZZZ', 'NAME': 'ZZZ Airport', 'TIME\_ZONE': 'UTC'}

# ABAP REST – Python Demo CR(U)D - Delete



```
import sap_context
import requests
import json

# ask user for new airport
airport_id = input("Enter the airport ID to delete: ")
if airport_id == "":
    quit()

# build service url
rest_url = "http://" + sap_context.sap_host + ":" + sap_context.sap_http_port
rest_url += sap_context.sap_rest_endpoint + "/" +
sap_context.sap_rest_airport
rest_url += "/" + airport_id + sap_context.sap_rest_params
print("REST URL: ", rest_url)

# call sap odata service api
response = requests.delete(rest_url)

if response.status_code != 200:
    print(f"Error {response.status_code}: {response.text}")
else:
    if response.text.find("{") != 0 and response.text.find("[") != 0:
        print("Response:", response.text)
        print("Response is not a JSON answer?!")
    else:
        json_response = json.loads(response.text)
        print("REST Response:", json_response)
```

> python demo\_sap\_rest\_airport\_delete.py

Enter the airport ID to delete: YYY

REST URL:

http://10.17.1.12:80/demos/mdd22/rest/Airport/YYY?sap-client=100

Error 400:

> python demo\_sap\_rest\_airport\_delete.py

Enter the airport ID to delete: ZZZ

REST URL:

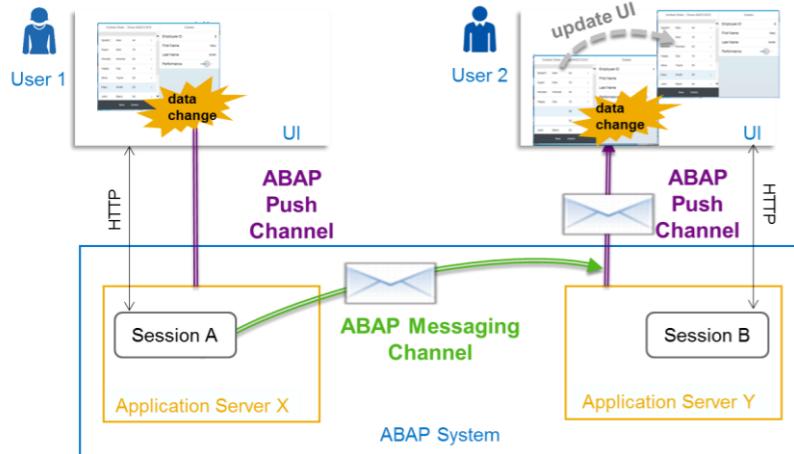
http://10.17.1.12:80/demos/mdd22/rest/Airport/ZZZ?sap-client=100

REST Response: {'MANDT': '100', 'ID': 'ZZZ', 'NAME': '', 'TIME\_ZONE': 'UTC'}

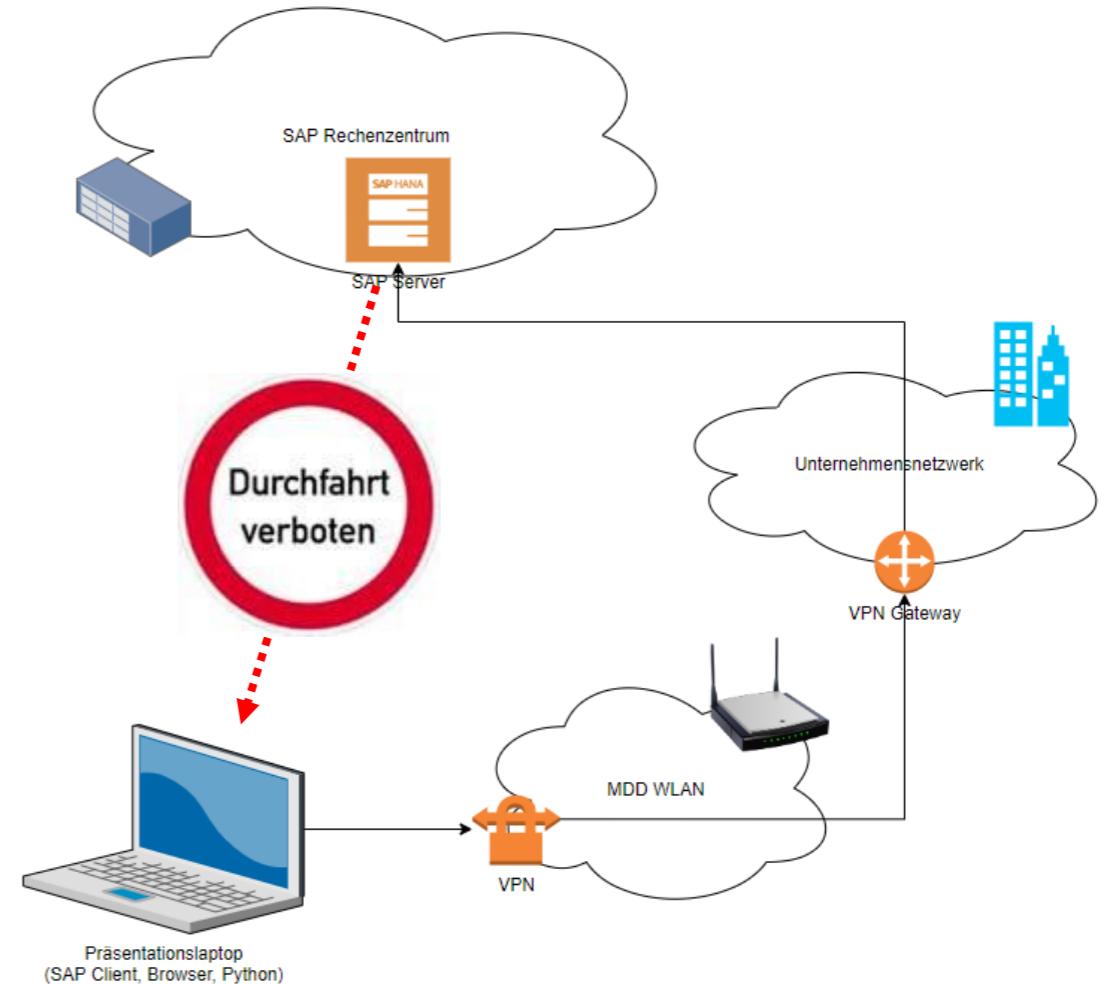
# MDD Bonus – Echtzeit vom SAP Server zum Client



Human Resources Example: User 1 and User 2 work on the same employees data



- Der Zugriff vom SAP Server auf den Client-Laptop (analog Server im Kundennetz) geht hier nicht!
- Lösung: Der Client baut eine Verbindung zum SAP Server auf und wartet  
→ Always Online mit WebSockets – Sogar in Echtzeit!
- Python Demo
  - Programm: demo\_sap\_websocket\_listen.py
- SAP ABAP Paket ZMDD\_WSPS
  - Programm ZMDD\_WSPS\_AMC\_SEND (Start mit SA38) zum Senden von Nachrichten
  - Programm ZMDD\_WSPS\_AMC\_RECEIVE zum Empfangen im SAPGUI
- Testanwendung mit Web-Oberfläche vom SAP Server:
  - [https://s4d.17.ucc.md/sap/bc/webdynpro/sap/wdr\\_test\\_apc\\_wsp?apc\\_application=zmd...&sap-client=100&sap-language=DE#](https://s4d.17.ucc.md/sap/bc/webdynpro/sap/wdr_test_apc_wsp?apc_application=zmd...)



# Bonus – Broadcast in Echtzeit vom SAP Server zum Client



The screenshot illustrates a real-time broadcast setup involving an SAP server and two Python clients.

**Listener SAP**: An SAP ABAP Push Channel (APC) interface window titled "Programm ZMDD\_WSPS\_AMC\_SEND". It shows a message box containing "broadcast:Das ist eine Nachricht an alle!" and a "CONSID" field. A red box highlights the message box. The SAP logo is visible at the bottom right of the window.

**Sender SAP**: A SAP ABAP Push Channel (APC) interface window titled "ABAP Push Channel (APC) für WebSocket Protocol". It shows an "APC Application" field set to "zmdd\_wspc\_apc" and a "Message" input field. Below the message area, it displays log output: "Connecting to wss://s4d.17.ucc.md/sap/bc/apc/sap/zmdd\_wspc\_apc", "Connected", "Receiving message", and "connected:KHM0ZF9TNERfMDAsVDExN19VMjY5MTZfTTApLDpDPTEwMCw6VT1KTVI=:SAP Backend S4D/100 - AMC enabled". A red box highlights the "Message" input field.

**Listener Python 1**: A command-line window showing the output of running "mdd22\_python>C:/ProgramData/Anaconda3/python.exe c:/Daten/Projects/github/MDJoerg/mdd22\_pyt...". It lists multiple client connections and their payloads, including "test-client1" and "test-client2". A red box highlights the text "Connected" in the log.

**Listener Python 2**: A second command-line window showing the same log output as Listener Python 1, indicating successful connections from both clients.

# Bonus – Broadcast in Echtzeit vom SAP Server zum Client (2)



The screenshot illustrates the real-time broadcast mechanism between an SAP server and clients. It consists of two windows:

- Top Window (SAP Application):** Shows the SAP interface for sending messages. A message box contains "broadcast:Das ist eine Nachricht an alle!". A red arrow points from this message box to the corresponding output in the terminal window.
- Bottom Window (Terminal):** Shows the command-line interface where a broadcast message is sent. The message "broadcast:Das ist eine Nachricht an alle!" is highlighted with a red box. A red arrow points from this box to the SAP application window, indicating the flow of the broadcast message.

**SAP Application Output (Top Window):**

```
C:\Daten\Projects\github\MDJoerg\mdd22_python>C:/ProgramData/Anaconda3/python.exe c:/Daten/Projects/github/MDJoerg/mdd22_pyth...  
Enter a client ID: test-client1  
Connect to SAP APC Websocket: ws://10.17.1.12:80/demos/mdd22/apc?sap-client=100  
> connect:test-client1:from python  
< connected:KHM0ZF9TNERfMDAsVDEzN19VMjY4NjlTTApLDpDPTEwMCw6VT1KTVI=:SAP Backend S4D/100 - AMC enabled  
< COMMAND detected: connected payload = 'KHM0ZF9TNERfMDAsVDEzN19VMjY4NjlTTApLDpDPTEwMCw6VT1KTVI=:SAP Backend S4D/100 - AMC enabled'  
< Echo: connect:test-client1:from python  
< COMMAND detected: Echo payload = 'connect:test-client1:from python'  
< connect:test-client1:from python  
< COMMAND detected: connect payload = 'test-client1:from python'  
< connect:test-client2:from python  
< COMMAND detected: connect payload = 'test-client2:from python'  
< broadcast:Das ist eine Nachricht an alle!  
< COMMAND detected: broadcast payload = 'Das ist eine Nachricht an alle!'
```

**Terminal Output (Bottom Window):**

```
C:\Daten\Projects\github\MDJoerg\mdd22_python>C:/ProgramData/Anaconda3/python.exe c:/Daten/Projects/github/MDJoerg/mdd22_pyth...  
Enter a client ID: test-client2  
Connect to SAP APC Websocket: ws://10.17.1.12:80/demos/mdd22/apc?sap-client=100  
> connect:test-client2:from python  
< Echo: connect:test-client2:from python  
< COMMAND detected: Echo payload = 'connect:test-client2:from python'  
< connect:test-client2:from python  
< COMMAND detected: connect payload = 'test-client2:from python'  
< connected:KHM0ZF9TNERfMDAsVDExN19VMjY5MTZfTTApLDpDPTEwMCw6VT1KTVI=:SAP Backend S4D/100 - AMC enabled  
< COMMAND detected: connected payload = 'KHM0ZF9TNERfMDAsVDExN19VMjY5MTZfTTApLDpDPTEwMCw6VT1KTVI=:SAP Backend S4D/100 - AMC enabled'  
< broadcast:Das ist eine Nachricht an alle!  
< COMMAND detected: broadcast payload = 'Das ist eine Nachricht an alle!'
```

# Bonus: Echtzeit im SAP – APC und AMC Infos



Weitere Informationsquellen:

## ■ Blogs

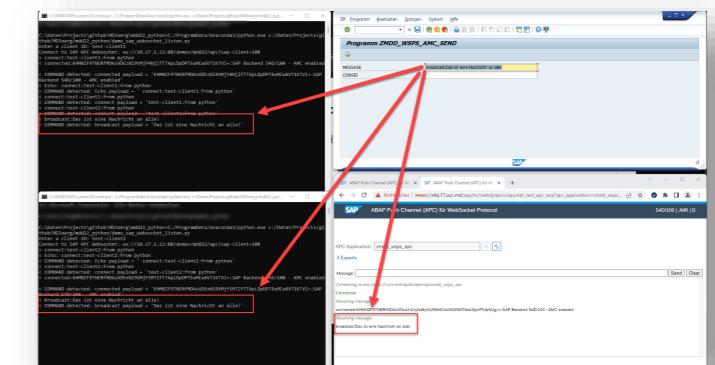
- Einführung ABAP Channels und Beispiele
- APC Beispiele im SAP Standard
- Step by Step Tutorial
- PCP Protocol - Javascript API
- Say Goodbye to Polling

- <https://blogs.sap.com/2014/11/27/introduction-to-abap-channels/>
- <https://blogs.sap.com/2016/06/09/abap-channels-examples/>
- <https://www.saptechnicalguru.com/apc/>
- <https://blogs.sap.com/2015/07/27/specification-of-the-push-channel-protocol-pcp/>
- <http://sapabapcentral.blogspot.com/2016/12/say-goodbye-to-polling-real-time-events.html>

## ■ Videos

- <https://youtu.be/Hr4ISLylusc>
- <https://youtu.be/kJiMAS9qPEQ>
- [https://youtu.be/l\\_J7Abb2KTQ](https://youtu.be/l_J7Abb2KTQ)
- <https://youtu.be/qwLBM6YLw9U>

- ABAP Channels - Real time eventing in ABAP
- How to implement collaboration using ABAP Push Channel (APC) and ABAP Messaging Channel (AMC)
- How to implement ABAP Push Channel (APC)
- How to implement ABAP Messaging Channel (AMC)



**Vielen DANK!**

**BA**

**VIELEN DANK FÜR EURE  
AUFMERKSAMKEIT**



**NOCH FRAGEN ?**

[makeameme.org](http://makeameme.org)