Darstellung und Vergleich mehrerer Möglichkeiten zur Umsetzung eines sequentiellen HR-Prozesses im RESTful API-Umfeld

Projektarbeit 1

im Rahmen der Prüfung zum

Bachelor of Science (B.Sc.)

des Studienganges Wirtschaftsinformatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Tom Wolfrum

- Sperrvermerk -

Abgabedatum:	4. September 2023
Bearbeitungszeitraum:	05.06.2023 - 03.09.2023
Kurs:	WWI22B5
Ausbildungsfirma:	SAP SE Dietmar-Hopp-Allee 16 69190 Walldorf, Deutschland
Betreuer der Ausbildungsfirma:	Steven Rösinger
Gutachter der Dualen Hochschule:	Paul Peitz

Sperrvermerk

Die nachfolgende Arbeit enthält vertrauliche Daten der:

SAP SE
Dietmar-Hopp-Allee 16
69190 Walldorf, Deutschland

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen ausserhalb des Prüfungs- und Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung des Dualen Partners vorliegt.

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Projektarbeit 1 mit dem Thema:

> *Darstellung und Vergleich mehrerer Möglichkeiten zur Umsetzung eines sequentiellen HR-Prozesses im RESTful API-Umfeld*

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, den 18. Juli 2023

Wolfrum, Tom

Inhaltsverzeichnis

Abkürzungsverzeichnis

SaaS	Software-as-a-Service
AIS	Application Innovation Services
HCM	Human Capital Management
API	Application Programming Interface
REST	Representational State Transfer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
RAP	Restful Application Programming Model
ABAP	Advanced Business Application Programming
CDS	Core Data Service
BO	Business Object
UI	User Interface (Benutzeroberfläche)
TCO	Total Cost of Ownership (Gesamtkosten für Entwicklung und Betrieb)
BW	Business Workflow
BE	Business Events
bgPF	background Processing Framework
LUW	Logical unit of work
bgRFC	background Remote Function Call
BTP	Business Technology Platform

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Unternehmensprofil und Anwendungsbezug

SAP SE ist ein börsennotierter Softwarekonzern mit Sitz in Walldorf. Das Hauptgeschäft des 1972 gegründeten Unternehmens ist die Entwicklung von Unternehmenssoftware zur Abwicklung von Geschäftsprozessen. Heute erwirtschaften 105.000 Mitarbeiter in 157 Ländern einen Umsatz von ca. 30 Mrd. €. Erfolgreich wurde das Unternehmen mit dem Verkauf von ERP Standardsoftware. In den letzten Jahren stand die Transformation des gesamten Produkt-Portfolios in Richtung Cloud-Services als Abo-Modell im Fokus der Unternehmensstrategie.[1]

Die Abteilung AIS HCM ist Teil des Unternehmensbereichs Product Engineering und zuständig für 2nd-Level-Support und Eigenentwicklungen der SAP Personallösung HCM. Zudem stellt die Abteilung mehrere SAP Fiori Apps als Self-Service für Mitarbeiter bereit. Durch das hohe Nutzungsvolumen dieser Apps und der somit großen betriebswirtschaftlichen Relevanz, sind diese und auch der Untersuchungsgegenstand dieser Arbeit für das Produkt HCM von großer Bedeutung.

1.2 Motivation und Problemstellung

Im folgenden Kapitel soll dargestellt werden, welche Probleme sich durch gewisse technische Veränderungen der SAP Produkte ergeben und sich somit für eine wissenschaftliche Untersuchung im Rahmen dieser Arbeit anbieten.

Die von der Abteilung betriebenen Fiori Apps, die schon im Zusammenhang der Einleitung angesprochen wurden, sind auf Basis des Frameworks SAP UI5 Freestyle für ein älteres Produkt - SAP ERP - entwickelt worden. Durch die strategische Entscheidung HCM im neuen S/4 HANA System (”S/4” abgekürzt) durch die neue cloudbasierte Personallösung SuccessFactors abzulösen war dieser Umstand ursprünglich kein Problem.

[1] Vgl. SAP 2023a.

Diese Entscheidung wurde aufgrund fehlender Funktionalitäten in SuccessFactors und hoher verlässlicher Einnahmen durch Wartungsverträge für HCM revidiert und HCM ist Bestandteil von S/4. Somit finden die S/4-Design-Guidelines darauf Anwendung, die z. B. Oberflächen-Design oder zu verwendende Technologien, festlegen. Fiori Apps müssen dadurch die Technologie Fiori Elements verwenden. Aus Praktikabilitätsgründen dürfen existierende Apps auf Basis der älteren Technologie in S/4 weiterbetrieben werden und nur neu entwickelte Apps müssen Fiori Elements verwenden.

Diese Situation sorgt für ein Problem in Geschäftsprozessen, die über solche Apps abgebildet werden sollen. Das Framework Fiori Elements generiert das gesamte Front-End der Anwendung selbstständig. Das erleichtert auf der einen Seite die Entwicklung der Apps, auf der anderen Seite kann dadurch keine eigene Programmlogik mehr im Front-End eingebaut werden. Zudem wird die Kommunikation mit dem Back-End über eine RESTful API, die zustandslos angelegt ist, abgewickelt. Auch wenn eine RESTful-API viele Vorteile mit sich bringt, sind Anwendungen, deren Prozesse asynchrone Kommunikation benötigen nur noch schwer abbildbar.

In der vorliegenden Arbeit soll nun untersucht werden, wie sich solche asynchronen Prozesse, trotz den eben dargelegten Einschränkungen trotzdem im neuen S/4 HANA Umfeld mit den neueren Technologien umsetzen lassen.

1.3 Aufbau und Ziel der Arbeit

Im Folgenden wird der Aufbau und das Ziel der Arbeit thematisiert.

Als erstes wird im einleitenden Kapitel die SAP und die Abteilung AIS HCM, in der die Praxisphase absolviert wurde, kurz vorgestellt und somit der Anwendungsbezug der Arbeit hergestellt. Danach wird mit der Motivation und Problemstellung der Untersuchungsgegenstand und die Bedeutung der Arbeit für die Abteilung und die Kunden erläutert. Danach soll die Arbeit klar von verwandten Themen abgegrenzt werden, um einen klaren Rahmen für die Untersuchung zu schaffen. Im methodischen Vorgehen werden dann abschließend für die Einleitung noch auf die wissenschaftlichen Methoden, die verwendet wurden, um die Untersuchungsergebnisse zu erhalten, eingegangen. Der Hauptteil der Arbeit besteht aus zwei Teilen: Im ersten Teil werden die theoretischen Grundlagen der Arbeit gelegt. Hier werden die Designprinzipien einer RESTful API, wie

diese im RESTful Application Programming Model der SAP eingesetzt werden und die Technologie Fiori Elements näher beleuchtet. Der praktische Hauptteil stellt drei Ansätze vor, wie das in der Problemstellung thematisierte Problem gelöst werden kann. Hierfür werden die Technologien Business Workflows, Business Events und das Background Processing Framework vorgestellt und im Bezug auf Stärken und Schächen sowie Effizienz und Robustheit verglichen. Diese Ergebnisse werden dann in einer Entscheidungsmatrix dargestellt. Das Ziel soll es sein, dass diese Entscheidungsmatrix klare Tendenzen gibt, welche der untersuchten Technologien sich in einem konkreten Anwendungsfall für das Abbilden von asynchronen Prozessen im RESTful API Umfeld anbietet. Im Schlussteil werden die Ergebnisse der Arbeit nochmals zusammengefasst, eine konkrete Handlungsempfehlung für die Lösung dieses Problems gegeben und die Ergebnisse abschließend kritisch Reflektiert und ein Ausblick auf zukünftige Entwicklungen gegeben.

1.4 Abgrenzung

Der Zweck der vorliegenden Arbeit ist es, die drei vorgestellten Technologien vergleichend zu bewerten und je nach Anwendungsfall eine Handlungsempfehlung im Bezug auf eine sich anbietende Technologie zu geben. Über diese drei Technologien hinaus werden keine anderen Möglichkeiten asynchrone Prozesse abzubilden, wie z. B. im Cloud Application Programming Model (CAP) behandelt. Außerdem findet aufgrund des beschränkten Umfangs der Arbeit lediglich ein Vergleich der Technologien statt und keine direkte Implementierung dieser in einem konkreten Anwendungsfall. Hierfür sei auf die offizielle Dokumentation der SAP mit Showcases für die respektiven Technologien verwiesen.

1.5 Methodisches Vorgehen

Nachdem die drei Ansätze vorgestellt wurden, sollen diese im Bezug auf mehrere Kriterien betrachtet und anhand dieser miteinander verglichen werden. Diese Kriterien werden bei den einzelnen Ansätzen durch Experteninterviews bewertet und dann die betrachteten Ansätze anhand dieser Kriterien gegenübergestellt. So kommt dann die Entscheidungsmatrix zu Stande, welche Technologie sich bei welchen Anforderungen und Rahmenbedingungen anbietet.

2 Theoretische Grundlagen

Im Folgenden sollen die theoretischen Grundlagen für die nachfolgende vergleichende Darstellung der Umsetzung sequentieller Prozesse im REST-Umfeld gelegt werden. Zuerst wird grundsätzlich erklärt, was eine RESTful-API ist und danach die Umsetzung von REST in ABAP näher erleutert. Abgeschlossen wird der theoretische Teil der Arbeit mit einer Darstellung von Fiori Elements.

2.1 RESTful Application Programming Interface

Eine API ist eine Schnittstelle, über die verschiedene Softwareanwendungen miteinander kommunizieren können. Die API definiert die Methoden, Protokolle und Tools, die für den Zugriff auf die Funktionen und Daten einer Softwareanwendung verwendet werden können. Somit standardisiert eine API die Kommunikation verschiedener Anwendungen und ermöglicht den Zugriff auf bereitgestellte Daten, ohne dass die zugreifende Anwendung die interne Logik oder Implementierung der anderen Anwendung kennen muss.

Eine RESTful-API ist eine spezielle Schnittstelle, die den Designkonventionen nach REST folgt.

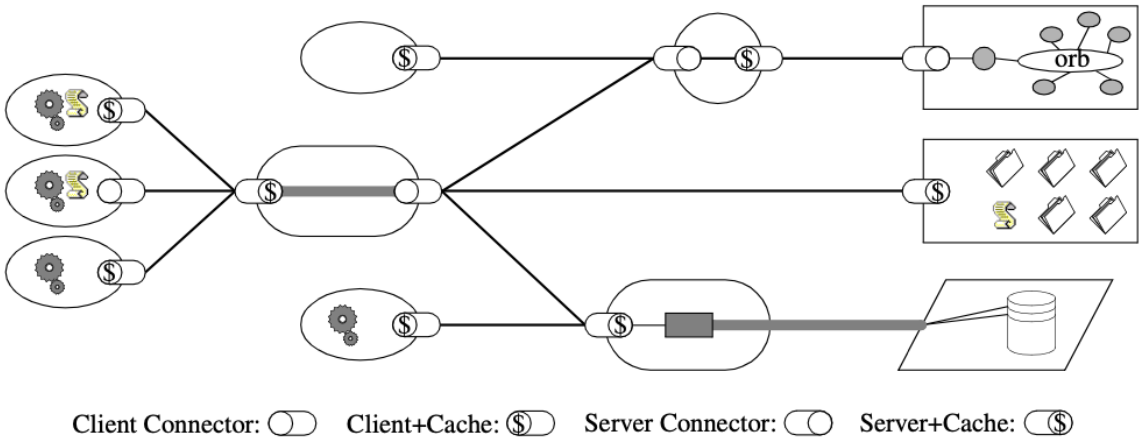

Abbildung 1: REST Designprinzipien, Abgerufen von Fielding 2000 am 05.07.2023.

Client-Server-Architektur

Das erste Prinzip ist die Client-Server-Architektur. Das bedeutet, dass die Benutzeroberfläche von den gespeicherten Daten getrennt wird. Die Benutzeroberfläche und Sitzung existiert nur auf dem Client und die gespeicherten Daten oder zur verfügung gestellten Funktionen existieren nur auf dem Server. Somit wird die Portierbarkeit und Skalierbarkeit des Gesamtsystems verbessert. Zudem wird die Möglichkeit einer unabhängigen Weiterentwicklung der verschiedenen Komponenten sichergestellt.[1]

Zustandslosigkeit

Zudem soll eine RESTful-API zustandslos angelegt sein. Das heißt im Genaueren, dass die Kommunikation der verschiedenen Parteien zustandslos sein muss. Es muss für den Server somit möglich sein, die Anfrage des Clients vollständig zu verstehen bzw. zu verarbeiten, ohne zusätzlich auf vergangene Anfragen zugreifen zu müssen. Auf der anderen Seite bedeutet das auch, dass der Client jede Antwort des Servers ohne zusätzliche Inforamtionen, die eventuell zu einem früheren Zeitpunkt angefordert wurden, verstehen können muss. Das heißt, dass in jeder Anfrage immer alle notwendigen Informationen mitgeschickt werden müssen und von keinem "Vorwissen" ausgegangen werden darf. Das hat wiederum zur Folge, dass Sitzungsinformationen ausschließlich auf dem Client gespeichert werden. Durch diese Bedingung verbessert sich die Skalierbarkeit weiter, da der Server Ressourcen, die ansonsten für die Speicherung der Stati der Requests benötigt würden, nicht freihalten muss. Zudem steigt die Zuverlässigkeit der Schnittstelle, da bei einem Fehler immer nur ein Request betrachtet werden muss. Somit ist ein Fehler einfacher behebbar und hat keine Auswirkungen auf andere Anfragen. Damit einher geht auch ein vereinfachtes Monitoring, da immer nur eine Request betrachtet werden muss und nicht erst eine Kette zusammenhängender Anfragen nachvollzogen werden muss.[2]

Caching

Die dritte Designkonvention besagt, dass auf der Client Seite ein Cache vorhanden sein muss. Durch das implizite oder explizite Markieren von Daten als cache-fähig dürfen

[1] Vgl. Fielding 2000.
[2] Vgl. Fielding 2000.

die Anfrage-Daten vom Client für spätere identische Requests wiederverwendet werden. Durch dieses Caching von Daten ist es möglich manche Client-Server Interaktionen teilweise oder ganz zu vermeiden, wodurch die Netzwerkauslastung und Skalierbarkeit verbessert wird. Jedoch birgt die Verwendung eines Caches das Risiko, dass die Daten im Cache im Vergleich zu den auf dem Server gespeicherten Daten schon veraltet sind, was gegebenenfalls zu Fehlern in der weiteren Verarbeitung führen könnte.[3]

Einheitliche Schnittstelle

Das vierte Prinzip und zentrales Unterscheidungsmerkmal von REST ist das einheitliche Interface zwischen den verschiedenen Komponenten. Hierdurch wird die Systemarchitektur durch das Prinzip der Generalität vereinfacht. Die Schnittstelle ist einfacher benutzbar. Zudem wird eine unabhängige Weiterentwicklung der verschiedenen kommunizierenden Kompoenenten gewährleistet, da die Implementierung der einzelnen Komponenten von den angebotenen Services getrennt wird. Jedoch entsteht durch die einheitliche Schnittstelle auch ein Effizienzverlust, da diese nicht an die Bedürfnisse einer speziellen Anwendung angepasst werden kann. Um ein einheitliches Interface zu erreichen, finden mehrere Beschränkungen auf die Schnittstelle Anwendung: Die Ressourcen der Schnittstelle sollen eindeutig identifizierbar sein. Eine Ressource ist eine vom Interface bereitgestellte Information, die eindeutig über einen URI identifizierbar ist. Die Informationen, die durch die Ressourcen repräsentiert werden können statisch festgelegt sein, oder sich auch im Zeitablauf verändern. Zudem kann eine Ressource auch existieren, ohne das die Information schon existiert. Das erleichtert die Verarbeitung verschiedener Informationsarten, da auf abstrakter Ressourcenebene nicht zwischen bestimmten Typen unterschieden wird. Außerdem kann so die benötigte Information auch noch zu einem späten Zeitpunkt, je nach Inhalt der Anfrage, festgelegt werden. Zudem hat jeder Service, der nach den REST Prinzipien entworfen ist eine URL, also eine eindeutige Adresse. Durch diese URL ist der Zugriffsweg zum Webservice standardisiert. Durch diese eindeutig identifizierbaren Ressourcen und Services wird zudem die Kombinierbarkeit verschiedener Ressourcen eines Services bzw. von verschiedenen Services in einem größerem System erleichtert. Eine weitere Beschränkung für die Schnittstelle ist die Verwendung von Repräsentationen zur Veränderung von Ressourcen. Eine Repräsentation ist eine Folge von Bytes, die eine Ressource in einer bestimmten Darstellung und zugehörige Metadaten abbildet. Somit

[3] Vgl. Fielding 2000.

kann eine Ressource vom Server in verschiedenen Repräsentationen, je nach Anfrage, zurückgegeben werden. Zudem werden mit der Repräsentation alle Informationen, wie die Ressource verändert werden kann, mitgeschickt. Veränderungen der Ressource finden nur über die Repräsentation statt. Des weiteren sollen Antworten des Servers auf Anfragen selbsterklärend sein. Dass heißt, das Standard-Methoden und -Datentypen verwendet werden, um die Ressource zu verändern oder Informationen auszutauschen. Diese Standard-Methoden sind zwar in REST selbst nicht festgelegt, werden aber normalerweise durch die Verwendung des Protokolls auf der Anwendungsschicht definiert. Für das meistens im Internet verwendete HTTP-Protokoll sind diese z. B. : GET (gewünschte Ressource vom Server anfordern), POST (neue Ressource unterhalb angegebener Ressource einfügen) oder PUT (angegebene Ressource anlegen bzw. ändern).[4] Die letzte Beschränkung wird als "Hypermedia as the Engine of Application State" bezeichnet. Hiermit ist gemeint, dass die Interaktion mit einer API dynamisch über Hypermedien abläuft. Somit ist auf der Client-Seite nur Basiswissen über Hypertext und fast kein Wissen über die Interaktion mit der spezifischen Schnittstelle nötig. Somit können Client und Server voneinander entkoppelt werden, da der Server dem Client neben den angeforderten Informationen dynamisch mögliche Interaktionen zurückgibt.[5]

Schichtenarchitektur

Die Systemarchitektur soll zudem in Schichten aufgebaut sein. Das heißt, dass eine Schicht jeweils nur die nächste darunter- und darüberliegende Schicht sehen und mit ihr interagieren kann. Durch diese Architektur wird die Komplexität des Gesamtsystems reduziert und die unabhängige Weiterentwicklung der einzelnen Schichten gefördert. Zudem können veraltete Dienste abgekapselt werden und und neue somit von diesen getrennt werden. Durch die Aufteilung der Architektur in Schichten kann zudem redundante oder selten benutzte Funktionalität in eine "Zwischenschicht" ausgelagert werden. Durch diese Aulagerung verbessert sich zudem die Skalierbarkeit des Systems, da load-balancing, also die Lastverteilung eines Services auf mehrere Netzwerke oder Prozessoren, ermöglicht wird. Dennoch bringt die eine Schichtenarchitektur auch Nachteile mit sich. Durch die Kapselung der Dienste und Funktionalitäten in Schichten steigt der Verwaltungs- und Wartungsaufwand des Gesamtsystems. Zudem sinkt auch die Geschwindigkeit, mit der

[4] Vgl. Fielding 2000.
[5] Vgl. Fielding 2008.

Daten verarbeitet werden, da die Anfrage im Verarbeitungsprozess wesentlich mehr Schnittstellen passieren muss. Dieser Nachteil kann jedoch durch die Verwendung von geteilten Caches in den Zwischenschichten kompensiert werden, da durch diese Caches die Anzahl der Schnittstellen, die die Anfrage passieren muss, reduziert werden kann. Ein weiterer Vorteil der Schichtenarchitektur ist, dass die Anfragen selektiv von den einzelnen Schichten verändert werden können, da der Inhalt dieser selbst-beschreibend und die Bedeutung der Nachricht für die Zwischenschichten sichtbar ist.[6]

Code on Demand

Die sechste (optionale) Designkonvention von REST besagt, dass wenn nötig Code in Form von Skripten oder Apps über die Schnittstelle vom Client heruntergeladen und ausgeführt werden kann. Dies vereinfacht die Programmlogik des Clients, da weniger Programme schon im Voraus vorhanden sein müssen. Zudem wird dadurch die Erweiterbarkeit eines Systems verbessert, da auch nach dem initalen Installieren eines Systems, dieses noch durch das Bereitstellen von Code über die Schnittstelle erweitert werden kann.[7]

2.2 ABAP Restful Application Programming Model

Im Folgenden wird das Restful Application Programming Model für ABAP vorgestellt.

ABAP RAP ist ein Programmiermodell der SAP auf Basis von ABAP, das die Architektur für die Entwicklung von OData-Services definiert. Es basiert auf den Designprinzipien nach REST. Mit diesem Modell können sowohl Web APIs veröffentlicht und Business Events erzeugt als auch Fiori Apps entwickelt werden. RAP ist nach dem "Classic ABAP Programming" und dem "ABAP Programming Model for SAP Fiori" die dritte Evolutionsstufe des ABAP Programming Model. RAP kann sowohl in Cloud-, als auch in on-premise Systemen eingesetzt werden.[8]

RAP baut im Allgemeinen auf drei Säulen auf: Anders als in den vorhergehenden Programmiermodellen, wo mehrere Tools zum entwickeln von z. B. einer Fiori App nötig

[6]Vgl. Fielding 2000.
[7]Vgl. Fielding 2000.
[8]Vgl. SAP 2023b.

waren, sind sollen in RAP Implementierungsaufgaben in einer Entwicklungsumgebung integriert werden um einen standardisierteren Entwicklungsprozess zu gewährleisten und diesen für den Entwickler zu vereinfachen. Die zweite Säule ist die Programmiersprache ABAP: Durch Erweiterungen und Anpassungen ist es möglich diese für die Entwicklung mit RAP zu verwenden. Hierbei kommen Technologien wie CDS-Views zum Einsatz um Daten-Modelle zu definieren. Zudem können vorgefertigte APIs für generische Entwicklungsaufgaben verwendet werden. Die dritte Säule sind umfassende Frameworks, die dem Entwickler helfen, effizient und in kurzer Zeit eine Anwendung zu entwickeln, da einzelne Bausteine automatisch generiert werden und an bestimmten Stellen noch anwendungsspezifische Logik eingefügt werden kann.[9]

Architektur eines Business Services in RAP

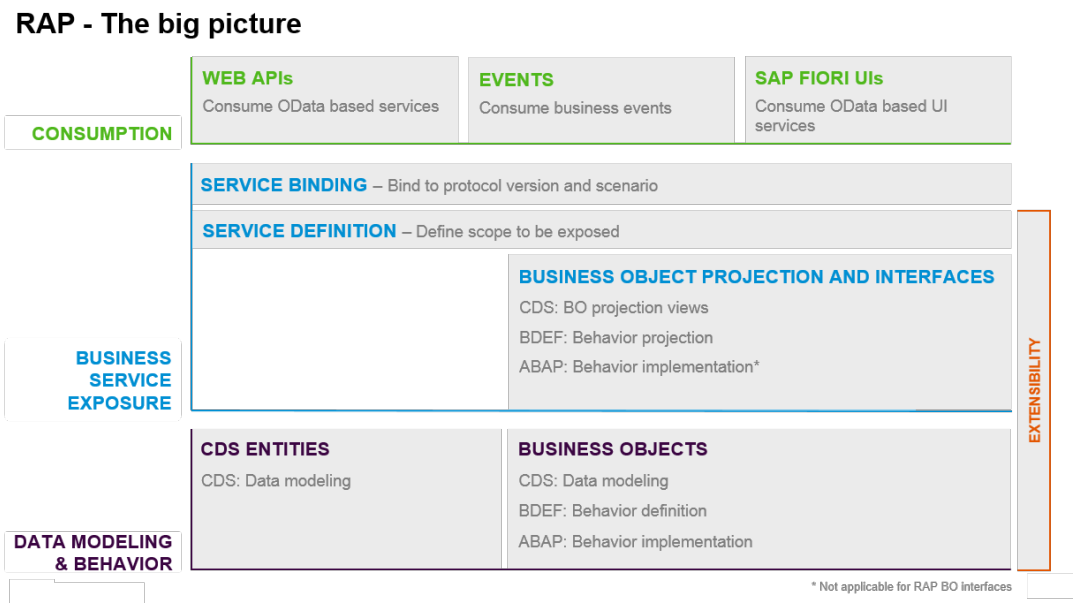

Abbildung 2: RESTful Application Programming Model Architektur, Abgerufen von SAP 2023b am 09.07.2023.

[9]Vgl. SAP 2023b.

Modellierung von Daten und Verhalten

Auf der untersten Ebene werden die benötigen Daten und das beabsichtigte Verhalten modelliert. Dies kann entweder durch CDS-Views oder durch Business Objects geschehen. Core Data Services sind ein Framework um ein Datenmodell, basierend auf der HANA-Datenbank, zu definieren und zu organisieren. Es können selektiv die benötigten Daten aus den Datenbanktabellen ausgewählt werden. Genauer werden alle benötigten Spalten aus einer oder mehreren Tabellen ausgewählt und diese bei Bedarf mit Annotationen versehen, um speziellere Anforderungen zu erfüllen. Es kann bei Bedarf auch noch nach Datensätzen, die gewisse Bedingungen erfüllen gefiltert werden. Somit strukturiert und gruppiert ein CDS-View die benötigten Daten. Die SQL-Abfrage, um diese Daten von der Datenbank abzurufen ist in dem CDS-View integriert. Der Zweck eines CDS-Views ist jedoch lediglich das Lesen und Strukturieren der Daten; es können hiermit keine Daten verändert werden.[10]

Eine andere Möglichkeit ein Datenmodell zu erzeugen sind Business Objects. Diese bieten zudem die Implementierung von Behaviors (Verhalten) und einer Laufzeit. Ein BO ist aus struktureller Sicht ein hierarchisch aufgebauter Baum aus mehreren Knoten, die Daten enthalten und durch Eltern-Kind-Beziehungen (sogenannte Kompositionen) miteinander verknüpft sind. Diese Knoten werden durch CDS Entitäten dargestellt. Der hierarchisch oberste Wurzelknoten stellt dabei die Repräsentation des BO an sich dar. Um das Verhalten eines BO zu spezifizieren, muss eine ”Business Object Behavior definition” (”Behavior definition” abgekürzt) angelegt werden. Dieses ABAP Objekt beschreibt das gewünschte Verhalten des BO in RAP. Die Behavior definition bezieht sich immer auf die Wurzel-CDS-Entität eines BO und wird als ABAP Klasse implementiert. Ein Verhalten beschreibt welche Operationen für ein BO verfügbar sein sollen. Es besteht außerdem noch aus der ”Behavior characteristic”, die zusätzliche Eigenschaften, wie z. B. Autorisierungen für Operationen festlegt. Operationen sind z. B. create() für das Erstellen, update() für das Aktualisieren und delete() für das Löschen eines Datensatzes. Diese modify-Operationen verändern im Gegensatz zu den read-Operationen auch die tatsächlichen Daten auf der Datenbank. Die Laufzeit eines BO besteht aus 2 Teilen: In der Interaktionsphase werden durch das Ausführen von Operationen Daten gelesen und/ oder verändert. Diese Veränderungen werden zunächst in einem ”transactional buffer”

[10] Vgl. SAP 2023b.

(Pufferspeicher) gespeichert und nachdem alle Änderungen durchgeführt wurden in der sog. ”save sequence” auf der Datenbank persistiert.[11]

Veröffentlichung des Business Service

Die zweite Ebene sorgt für das Projizieren von BOs und die Erstellung sowie Veröffentlichung von Business Services. Ein Business Service ist in RAP ein RESTful Service, der Repräsentationen von Ressourcen veröffentlicht, die dann von Konsumenten abgerufen werden können. Die Bestandteile eines Business Services werden später noch genauer erläutert. Die Projektion eines BO ist notwendig, um es flexibel konsumieren zu können, da dieses an sich komplett unabhängig vom OData-Service ist. Das BO an sich stellt die maximal möglichen Funktionen und Daten bereit, die service-unabhängig implementiert und ggf. durch die Projektion auf die für den Service relevanten Aktionen und Daten eingeschränkt werden. Zudem können genauere Anpassungen z. B. im Bezug auf die Darstellung auf einer Benutzeroberfläche über UI-Annotationen erfolgen, die aber nicht Teil des Datenmodells sein sollen. Eine zusätzliche Projektions-Schicht hat mehrere Vorteile: Zum einen kann das zugrundeliegende BO angepasst und erweitert werden, ohne dass der darauf aufbauende Service davon betroffen ist. Zum anderen können verschiedene Projektions-Views für verschiedene Anforderungen erstellt werden, die alle dasselbe BO wiederverwenden. Zudem können die Daten und Funktionen eines Services für eine Fiori App oder Web API veröffentlicht werden. Des Weiteren können Services somit auch rollenbaisert veröffentlicht werden, sodass unterschiedliche Daten und Funktionen für unterschiedliche Anwender bereitgestellt werden können. Um eine solche Projektionsschicht zu erstellen, muss zusätzlich eine CDS Projection View erstellt werden, um die speziellen Daten einer Projektion darzustellen. Dieser basiert auf der CDS View des BO und erzeugt selbst keine neue SQL-View, sondern nur eine Repräsentation der dargestellten Entitäten. Um eine CDS-Entität eines BO zu projizieren müssen die Wurzel-Entität sowie alle Eltern-Entitäten ebenfalls projiziert sein. Zudem wird auch eine Projection Behavior Definition benötigt, die alle Verhalten, die für einen speziellen Service veröffentlicht werden sollen projiziert.[12]

Nachdem Teile des BO projiziert wurden und die zugehörigen Artefakte erstellt wurden, muss in der zweiten Ebene ein Service definiert werden. In einer ”business service

[11] Vgl. SAP 2023b.
[12] Vgl. SAP 2023b.

definition" (abgekürzt "Service Definition") wird festgelegt, welche CDS Entitäten eines Datenmodells, also welche Daten, in einen bestimmten Service veröffentlicht werden sollen. Die Service Definition stellt eine protokoll-unabhängige und Konsumenten-spezifische Sichtweise auf das Datenmodell dar. Es können auch mehrere CDS-Entitäten oder eine komplette BO Struktur in einem Service veröffentlicht werden. Dafür muss in der Service Definition die hierarchisch höchste Entität des BO, die veröffentlicht werden soll, markiert werden. Diese dient dann als Einstiegspunkt für den Service.[13]

Als letzter Schritt in der zweiten Ebene muss noch das Kommunikationsprotokoll des Service im Service Binding definiert werden. Ein häufiges Beispiel wäre hierbei OData, für das Bereitstellen von Daten in einer Fiori App. Ein Service Binding bezieht sich immer direkt auf eine oder mehrere Service Definitions. Es können auch mehrere Service Bindings basierend auf einer Service Definition erstellt werden. Das ist z. B. hilfreich, wenn derselbe Service mit unterschiedlichen Kommunikationsprotokollen veröffentlicht werden soll, da durch die Trennung von Service Definition und Binding das Protokoll von der Geschäftslogik getrennt wird. Somit kann der Entwicklungsaufwand für einen Service erheblich reduziert werden. Ein Service kann für die Protokolle OData Version 2 und 4 veröffentlicht werden, wenn das Ziel ist, die Daten in einer Fiori App darzustellen. OData ermöglicht außerdem das Erstellen von HTTP-basierten Services, deren Ressourcen über URIs identifizierbar sind und über HTTP-Nachrichten abgerufen und modifiziert werden können. Dies korrespondiert wiederum mit den Designkonventionen nach REST aus dem vorhergehenden Kapitel. Zudem stehen noch die Protokolle Information Access für Analysezwecke und SQL zur Verfügung. Ein Service kann grundlegend auf zwei Arten veröffentlicht werden: Entweder als UI Service mit den Protokollen OData oder Information Access, indem man dem durch UI-Annotationen eine Fiori Elements oder andere Benutzeroberfläche hinzufügt. Für alle anderen Anwendungsfälle wird der Service als Web API veröffentlicht, die von Clients über das Web mit OData konsumiert werden kann. Wenn ein Service veröffentlicht ist, kann er jedoch im Standard nur innerhalb der Systemlandschaft abgerufen werden. Ein Service kann zudem in mehreren Versionen existieren. Dies geschieht durch das Hinzufügen oder Entfernen von zusätzlichen Service Definitions zu einem Service Binding. Damit kann ein Service geändert oder erweitert werden.[14]

[13] Vgl. SAP 2023b.
[14] Vgl. SAP 2023b.

Konsumieren des Services

Die dritte und oberste Ebene der RAP Architektur ist für das Konsumieren der Daten eines Business Services verantwortlich. Es gibt zwei Möglichkeiten, diese Daten durch einen Service zu konsumieren. Die erste Möglichkeit ist durch eine Web API. Die Metadaten eines so veröffentlichten Services enthalten keine Informationen über eine Benutzeroberfläche für die Darstellung der Informationen. Der Zugriff auf die bereitgestellten Daten erfolgt über eine öffentliche Schnittstelle des OData Services. Eine weitere Möglichkeit einen Service zu konsumieren ist innerhalb einer Fiori Elements Anwendung als UI Service. Hier werden die Konfigurationen für die Benutzeroberfläche und das Front-End der Anwendung, die im Back-End als Annotationen in den CDS Entitäten festgelegt wurden, über die Metadaten mitgegeben. Somit kann das Fiori Elements Framework aus diesen Metadaten direkt eine fertige UI generieren. Als letzte Möglichkeit ein BO zu konsumieren sind noch Business Events zu nennen. Diese werden hier nur kurz genannt und in einem späteren Kapitel detaillierter beschreiben.[15]

2.3 SAP Fiori Elements

Fiori Elements ist ein Framework zum Entwickeln benutzerfreundlicher und ansprechender Anwendungen. Apps werden auf Basis von OData-Services und UI-Annotationen in den CDS Entitäten durch ein umfassendes Framework fast automatisch generiert. Somit ist im Gegensatz zur älteren SAP UI5 Freestyle Technologie kein JavaScript Coding mehr nötig um das Front-End zu programmieren. Elements benutzt vordefinierte Layouts und Controller für Aktionen der App.[16]

Fiori Elements bietet drei zentrale Vorteile: Es soll dabei helfen, dass sich Entwickler auf die spezifische Geschäftsprozess-Logik und die Back-End Entwicklung fokussieren und somit weniger Zeit für die Programmierung der Benutzeroberfläche benötigen, was insgesamt zu einer verkürzten Entwicklungszeit von Apps und somit auch für niedrigere Entwicklungskosten sorgt. Zudem wird die Kontinuität des UI über alle Fiori Apps hinweg und die Übereinstimmung der Apps mit den SAP Designkonventionen sichergestellt. Die Benutzer der Apps haben so eine einheitliche Benutzungserfahrung (Layout, Navigation,

[15] Vgl. SAP 2023b.
[16] Vgl. SAP 2022a.

Suche, ...) über alle Apps hinweg. Zudem stellt das zentrale Framework auch sicher, dass der Programmcode für die Benutzeroberflächen der Apps immer sofort funktioniert und bietet außerdem weitere Funktionen wie Übersetzungen und Unterstützung für mobile Endgeräte. Diese Vorteile tragen wiederum zu einer reduzierten Entwicklungszeit und somit einem Kostenersparnis bei.[17]

Vergleich Fiori Elements - SAP UI5 Freestyle

Verglichen mit SAP UI5 Freestyle (Freestyle abgekürzt), der anderen Technologie, um Fiori Apps zu entwicklen, lassen sich folgende Unterschiede feststellen: Die generelle Herangehensweise bei Fiori Elements zielt eher auf das effiziente und schnelle Entwickeln und Veröffentlichen einer App ab. Im Gegensatz dazu, liegt der Fokus bei Freestyle eher auf der Flexibilität, spezielle Anforderungen, die ggf. auch nicht mit dem Elements Standard übereinstimmen abbilden zu können. Das wirkt sich auch auf die Möglichkeiten im UI-Design aus: In Fiori Elements ist der Entwickler an die vordefinierten SAP Vorlagen gebunden, während es bei den Freestyle Designs auch möglich ist, mit einer leeren Vorlage zu beginnen. Das hat aber auch zur Folge, dass bei Freestyle wesentlich mehr Webentwicklungs Kenntnisse nötig sind, da die Oberfläche mithilfe von JavaScript Coding selbst programmiert werden muss. In Fiori Elements hingegen wird die gesamte Oberfläche vom Elements Framework generiert und muss nur durch UI-Annotationen in den CDS-Entitäten an die Wünsche des Kunden angepasst werden. Was die Wartbarkeit und entwicklungstechnischen Freiheiten der beiden Technologien angeht, sind auch bei Freestyle größere Spielräume vorhanden: Dadurch, dass die Oberfläche selbst programmiert wird, ist es möglich eine Logik in das Front-End einzubauen. Diese Tatsache ist für die Problemstellung der Arbeit sehr relevant, da sequentielle Prozesse, die asynchrone Kommunikation benötigen, sich durch eben diese Logik leicht abbilden lassen. Da in Fiori Elements die das UI generiert wird und die Logik von SAP kommt, fehlt diese Gestaltungsmöglichkeit, was die Umsetzung der angesprochenen Geschäftsprozesse erschwert. Dennoch muss man sagen, dass in allen anderen Fällen Fiori Elements erhebliche Vorteile bietet, da bei der Entwicklung einer App sehr viel Zeit und somit auch Geld gespart werden kann, was somit auch die TCO reduziert.[18]

[17] Vgl. SAP 2022a.
[18] Vgl. SAP 2023c.

Model-View-Controller Konzept

Beide Fiori Technologien, Elements und Freestyle verwenden das Model-View-Controller Konzept, auf welches im Folgenden genauer eingegangen wird.

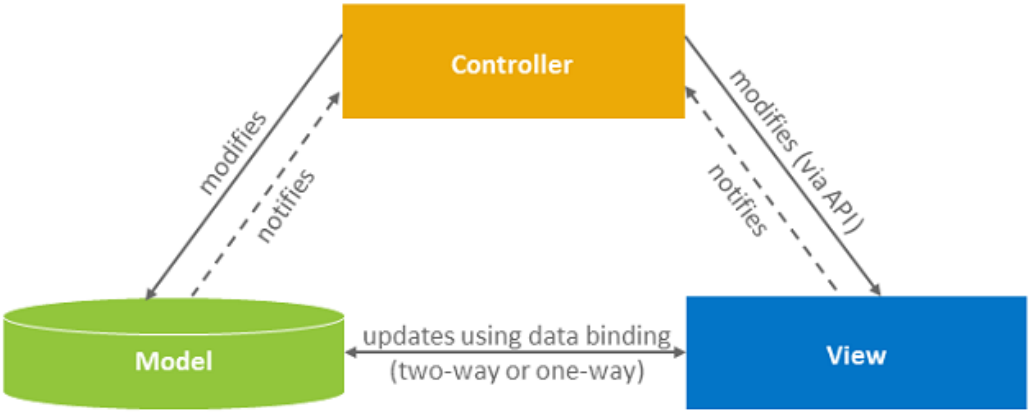

Abbildung 3: Model-View-Controller Konzept, Abgerufen von SAP 2023d am 13.07.2023.

Das Konzept unterteilt eine Anwendung in ein Modell, einen View und einen Controller. Das Modell verwaltet die Daten der Anwendung und stellt Methoden bereit, um die Daten aus der Datenbank abzurufen und bei Bedarf zu bearbeiten. Fiori unterstützt Client- und serverseitige Modelle. Das heißt, dass das Modell entweder komplett vom Client heruntergeladen wird und dort dann lokal existiert, oder bei serverseitigen Modellen nur die jeweils angefragten Daten vom Client beim Server angefordert werden. Daraus resultiert, dass clientseitige Modelle sich aus Performancegründen nur bei kleinen Datenmodellen empfehlen lassen und sich für groSSe Modelle eher serverseitige Datenmodelle anbieten. Wenn das Modell auf der Client Seite existieren soll, kann entweder ein JSON-, XML- oder Ressourcen-Modell verwendet werden. Serverseitig kann ein OData-Service in den Versionen zwei und vier verwendet werden. Es ist auch möglich für verschiedene Bereiche einer Anwendung jeweils verschiedene Modelle zu definieren und diesen auch modellspezifisch über den Controller verschiedene Interaktionsmöglichkeiten mit dem View zu geben. Der View definiert und rendert die Benutzeroberfläche und legt somit das Aussehen der App für den Anwender fest und visualisiert die Daten des Models. Standardmäßig werden XML- und JSON-Views unterstützt. Zudem kann man einen View als eigene Klasse selbst programmieren. Der Controller ist für die Interaktionen des

Benutzers mit der App zuständig und enthält Methoden, die die Interaktion zwischen Modell und View regeln. Er enthält den Programmcode für die Geschäftsprozesslogik hinter der Anwendung. Zudem kann der Controller in Methoden oder zu bestimmten Zeitpunkten, wie z. B. bei Start oder Beenden der Anwendung Events auslösen. Diese Events können dann von anderen Controllern empfangen und verarbeitet werden. Somit können eventgesteuert bestimmte Aktionen in der Anwendung ausgelöst werden. "Data-binding" wird verwendet, um die Daten des Modells und den View zu synchronisieren und das Bearbeiten dieser Daten direkt auf der Benutzeroberfläche zu ermöglichen. Das Data binding definiert, wie das Model und der View miteinander kommunizieren. Eine Art, die Daten des Models an die UI zu binden, ist das One-way Binding, welches nur in eine Richtung Daten bindet. Beispielsweise werden hier nur werden Veränderungen der Daten im Modell in die UI synchronisiert und nicht in die Gegenrichtung. Beim Two-way Binding ist diese Bindung in beide Richtungen vorhanden. Wenn sich also Daten entweder im View oder im Model ändern werden diese Änderungen dirket mit der jeweils anderen Komponente synchronisiert. Das Ziel des Konzepts ist es, die Daten der App logisch von den Interaktionen des Benutzers zu trennen. Eine Anwendung in die genannten drei Teile aufzuteilen hat folgende Vorteile: Der Programmcode der Anwendung ist leichter lesbar, wartbar und erweiterbar, da er in logisch zusammenhängende Teile aufgeteilt ist. Somit ist es möglich die Darstellung der Daten in der View zu verändern, ohne die darunterliegende Logik oder das Datenmodell zu ändern und mit mehreren Views mehrere Darstellungsweisen, je nach Anwendungsfall zu erstellen.[19]

Floorplans in Fiori Elements

Apps sind in Fiori Elements immer auf vordefinierten Layouts aufgebaut. Diese können am Anfang der Entwicklung einer App ausgewählt werden und geben das generelle Aussehen und Funktionen vor.

[19] Vgl. SAP 2023d.

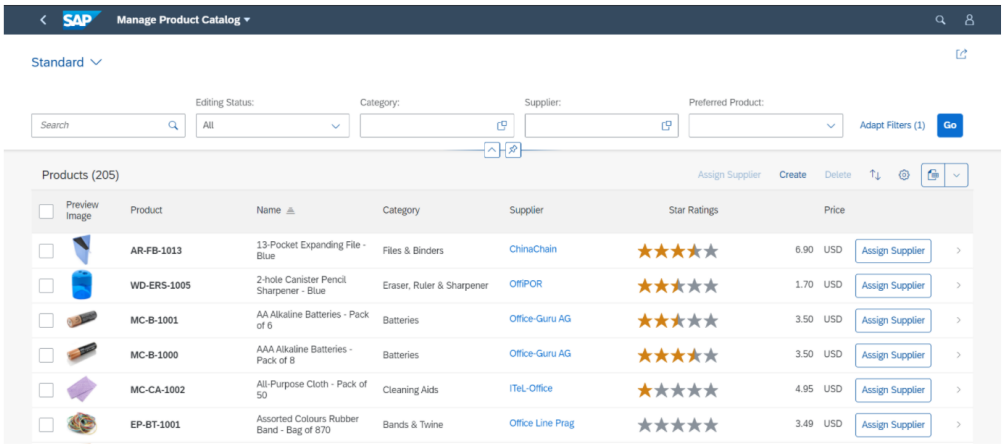

Abbildung 4: Fiori Elements List Report Floorplan. Abgerufen von SAP 2022a am 13.07.2023.

Auf dem Bild ist beispielhaft das Layout List Report zu sehen. Dieser kann verwendet werden, wenn der Zweck der Anwendung das Arbeiten mit einer Liste von Dingen z. B. Produkten ist. Diese werden dann in einer sortierten Tabelle dargestellt, in der der Benutzer nach Einträgen sortieren, filtern und suchen kann. Hier müssen nur die Produktdaten über einen OData Service bereitgestellt werden und die Darstellung der Daten über UI-Annotationen nach Bedarf angepasst werden. Die restliche Benutzeroberfläche mit Navigation, Suchleisten und Aktionen wird vollständig durch das Framework generiert. Der List Report wird häufig als Einstiegspunkt in einer App verwendet um nach bestimmten Objekten zu suchen und mit diesen dann in einer Object Page näher zu interagieren.

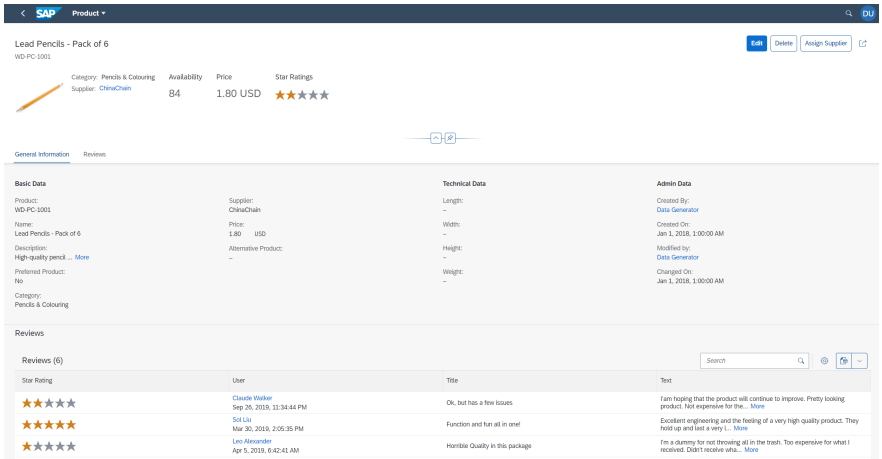

Abbildung 5: Fiori Elements Object Page Floorplan. Abgerufen von SAP 2022a am 13.07.2023.

Das List Report Layout wird häufig mit dem Object Page Floorplan in einer App kombiniert, wenn es darum geht, mit den einzelnen Objekten in der Liste zu arbeiten. Die Objekt-Ansicht wird durch einen Klick auf das gewünschte Objekt in der Listen-Ansicht aufgerufen. Hier können dann weitere Informationen zu z. B. einem Produkt angezeigt und bei Bedarf auch bearbeitet werden. Des Weiteren gibt es noch Floorplans für eine Worklist, eine Liste mit Unterstützung für diverse Analyseverfahren, wie z. B. Drill-Down und ein Dashboard.

3 Praktischer Teil

Im Folgenden werden die verschiedene praktische Lösungsansätze vorgestellt und anhand verschiedener Kriterien gegeneinander abgewogen, sodass am Ende eine Handlungsmatrix erstellt werden kann.

3.1 Lösungsansätze

Zunächst sollen drei verschiedene Technologien vorgestellt werden, mit denen sich asynchrone Prozesse mit sequentieller Kommunikation, trotz den Einschränkungen durch RAP und Fiori Elements, umsetzen lassen.

3.1.1 Business Workflows

Der erste mögliche Ansatz sind Business Workflows (BW abgekürzt). BWs können benutzt werden um jeglichen Geschäftsprozess im SAP-System abzubilden. Sie decken das Spektrum von einfachen Genehmigungsprozessen bis hin zu komplexen Abläufen ab. Sie eignen sich vor allem für repetitive Prozesse mit mehreren Bearbeitern. BWs können zudem zur Fehelerbehandlung in anderen Prozessen oder eventgesteuert eingesetzt werden. Mit Workflows können durch die Benutzung der bereits bestehenden Funktionen und Transaktionen des SAP-Systems neue Geschäftsprozesse abgebildet werden. Die Funktionen und Transaktionen an sich werden dabei durch den Workflows nicht verändert. In Kombination mit Organisationsmanagement können die einzelnen Schritte des BW durch bestimmte Akteure ausgeführt werden. Das kann auch auf bestimmte Stellen abstrahiert werden, um von personellen Veränderungen innerhalb des Unternehmens unabhängig zu sein. Workflows können auch untereinander durch das Versenden und Konsumieren von Nachrichten kommunizieren. Diese Kommunikation ist auch zwischen verschiedenen SAP-Systemen über das Internet mit XML-Dokumenten möglich.[1]

[1] Vgl. SAP 2022b.

Aufbau eines Business Workflows

Zunächst wird die Definition der Aufbau eines Business Workflows beschrieben. Diese lässt sich in vier Bereiche unterteilen.

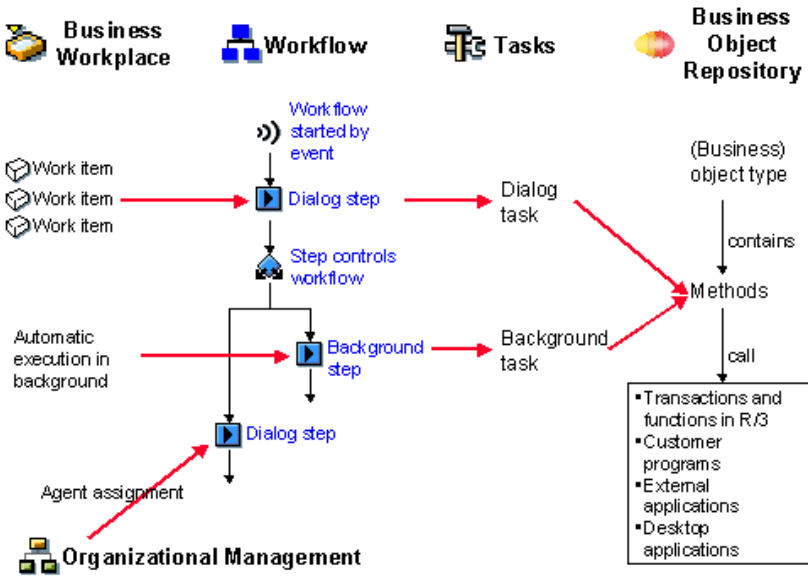

Abbildung 1: Aufbau eines Business Workflows, Abgerufen von SAP 2022b am 17.07.2023.

Der Business Workplace ist der Ort in einem SAP-System, in dem der Endanwender ”work items” (übersetzt aus dem Englischen: ”Arbeitspakete” oder ”Aufgaben”) abhängig vom Zeitpunkt im Geschäftsprozess und den Berechtigungen des Users ausführen kann. Ein work item stellt zur Laufzeit des BW einen Schritt des Prozesses dar, der ausgeführt wird. Es werden hier jedoch nicht alle work items angezeigt. So werden z. B. solche, die einem Prozess-Schritt, der im Hintergrund ausgeführt werden soll, zugeordnet sind, hier nicht angezeigt.[2]

Ein Workflow muss vor Ausführung in der ”workflow definition” angelegt werden. Diese Definition legt die Reihenfolge der auszuführenden Schritte des Prozesses fest und enthält zudem Kontrollschritte. Zusätzlich können noch Bearbeiter und Fristen für bestimmte Schritte festgelegt werden, die dann zur Laufzeit des Workflows vom ”Work item manager” verwaltet werden. Es gibt viele Arten von Schritten, die gängigen Konzepten in der Programmierung ähneln, wie z. B. normale Aktivitäten, Fallunterscheidungen,

[2]Vgl. SAP 2022b.

Schleifen. Zudem gibt es Schritte zum Versenden von Nachrichten, Auslösen von Events, Benutzerentscheidungen, usw. Diese Schritte können entweder im Dialog mit einem Benutzer ausgeführt werden, wenn z. B. die Eingabe bestimmter Werte erforderlich ist, oder automatisch vom System im Hintergrund ausgeführt werden. Ein Workflow kann nicht nur manuell von einem Benutzer gestartet werden, sondern auch systemseitig von einem bestimmten Event ausgelöst werden. Hierfür muss in der Definition des Workflows das gewünschte Event als Auslöser angegeben werden. Wenn dann das Event auftritt, wird der Workflow automatisch gestartet. Im betrieblichen Kontext könnte hier z. B. ein Mitarbeiter einen Urlaubsantrag stellen, der dann den als Workflow abgebildeten Genehmigungsprozess auslöst. Das wäre ein Beispiel für einen asynchronen Prozess mit sequentieller Kommunikation.[3]

Die einzelnen Schritte, die innerhalb des Workflows ausgeführt werden, heißen Tasks und stellen grundlegende betriebliche Tätigkeiten dar. Die Dialog- und Hintergrund-Schritte in der Workflow Definition korrespondieren hier mit Dialog- oder Hintergrund-Tasks. Im Workflow bezieht sich ein Task immer auf eine Methode eines Objekttyps. Diese Methoden können automatisch ausführbar sein oder müssen aktiv von einem Benutzer gestartet werden. Eine Methode kann einerseits Transaktionen oder Funktionen innerhalb des ERP-Systems aufrufen. Spezielle Anforderungen können durch kundeneigene Logik, oder Schnittstellen zu anderen Systemen umgesetzt werden.[4]

Methoden, die innerhalb eines Workflows aufgerufen werden, sind immer Teil von Objekten. Diese Objekte können auch BOs sein. Im Allgemeinen ist ein Objekt ein konkreter Datensatz eines Objekttyps. Die Daten des Objekts werden durch seine Attribute definiert und die Aktionen, wie das Erstellen, Aktualisieren oder Löschen von Daten wird durch die Methoden des Objekts beschrieben. Einen weiteren wichtigen Teil von Objekten stellen Events dar. Diese werden ausgelöst, wenn bei einem Objekt seinen Status verändert. Das kann z. B. durch das Erstellen, Verändern oder Löschen von Daten passieren. Diese Events können dann unter anderem Workflows starten. Das "Business Object Repository" bietet eine Übersicht über alle in einem SAP-System verfügbaren Objekttypen. Man kann die bereits vorhandenen Objekttypen bei Bedarf anpassen oder neue erstellen.[5]

[3] Vgl. SAP 2022b.
[4] Vgl. SAP 2022b.
[5] Vgl. SAP 2022b.

3.1.2 Business Events

Der zweite Ansatz ist die Umsetzung mit Business Events (BE abgekürzt). Business Events sind Events die von BOs erzeugt und konsumiert werden können. Dieser event-gesteuerte Kommunikationsansatz, der die asynchrone Kommunikation zwischen dem Event-erzeugenden und -konsumierenden BO ermöglicht, wird von RAP im Standard unterstützt. Hier ist keine direkte Antwort des Empfängers nötig, ein BO erzeugt ein Event und dieses wird von anderen BOs dann weiterverarbeitet, ohne dass der Erzeuger weiterhin in diesen Prozess involviert ist Somit spricht man hier von einer einseitigen, also asynchronen Kommunikation. Ein BE stellt eine signifikante Veränderung eines BO dar, die im Zuge eines Behaviours erzeugt wird. Dem Event werden dann über dessen Metadaten alle nötigen Informationen, anhand derer die Weiterverarbeitung, je nach speziellem Anwendungsfall, stattfindet, mitgegeben.[6]

Ein BO kann als Event-Erzeuger oder Event-Konsument auftreten. Zuerst wird die Erzeuger-Seite betrachtet.

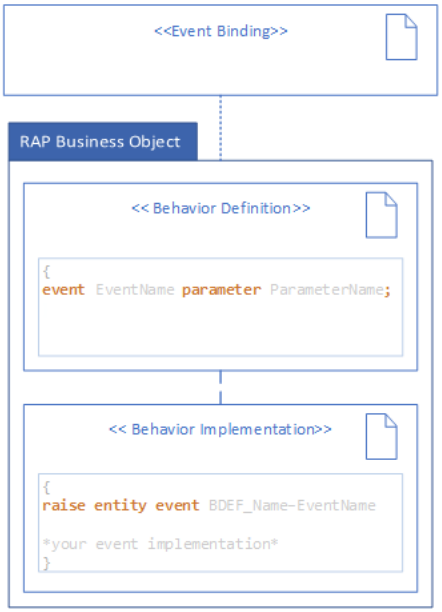

Abbildung 2: Business Object als Event-Erzeuger, Abgerufen von SAP 2023e am 17.07.2023.

[6]Vgl. SAP 2023e.

Ein Event wird in der Behaviour Definition eines BO erstmals definiert. Hier können dann Parameter für die weitere Verarbeitung mitgegeben werden, die dann in den Metadaten des Events mitgeschickt werden. Danach kann es in der dazugehörigen Behaviour Implementation innerhalb einer spziellen Operation des BO erzeugt werden. Ein Event wird zu dem Zeitpunkt in RAP Laufzeit erzeugt, nachdem die Veränderungen des BO, die das Event auslösen, auf der Datenbank persistiert wurden. Durch das Event Binding wird das Event noch einem speziellen Namensraum, BO und zugehöriger Operation zugeordnet.[7]

BOs können BEs nicht nur erzeugen, sonder nauch verarbeiten. Dies geschieht über das Event Consumption Model. Ein Event Consumption Model besteht aus einer Reihe von ABAP-Artefakten, mit denen man Events im SAP-System konsumieren kann. Diese Events müssen jedoch dem Standard eines Cloud Events entsprechen, da der Austausch über das SAP Event Mesh abgewickelt wird, was ein Dienst der SAP Business Technology Platform und somit komplett cloud-basiert ist. Hierfür muss ein inbound bzw. outbound Event-Binding für ein- bzw. ausgehende Events erstellt werden. Diese Bindings sind Teil des SAP Enterprise Event Enablement Frameworks, das den Austausch von Events zwischen dem S/4 System und der BTP regelt.[8]

3.1.3 Background Processing Framework

Die letzte Möglichkeit, die betrachtet wird, um asynchrone Prozesse abzubilden, stellt das ”background Processing Framework” (bgPF abgekürzt) dar.

Das bgPF ist ein Framework, das die Möglichkeit schafft asynchron zum Hauptprozess Logik auszulagern und auszuführen. Zuerst soll auf die grundlegende Funktionsweise des bgPF eingegangen werden.

[7] Vgl. SAP 2023e.
[8] Vgl. SAP 2022c.

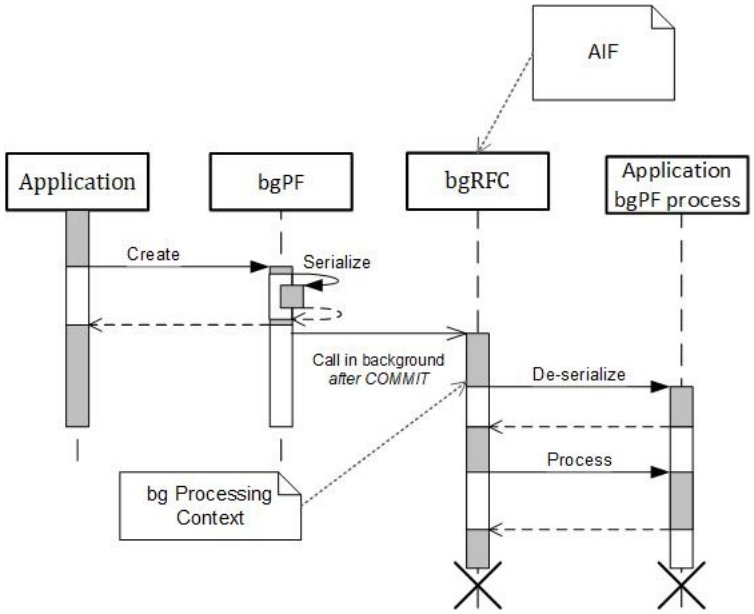

Abbildung 3: Funktionsweise des bgPF, Abgerufen von SAP 2023f am 17.07.2023.

Nachdem das Interface für das bgPF in der Anwendung als eigene Klasse mit der jeweiligen Logik, die ausgelagert werden soll, implementiert wurde, kann es verwendet werden. Die Anwendung ruft dann zur Laufzeit diese Klasse auf und erstellt synchron ein bgPF-Objekt. Innerhalb dieses Objekts wird die Klasse dann serialisiert und mit einem Methodenaufruf für die spätere Ausführung gespeichert. In der Endphase des Hauptprozesses wird dann mit einem bgRFC, also einem Funktionsaufruf, der im Hintergrund ausgeführt wird, die Ausführung ausgelöst. Der bgRFC wird dann in einer neuen ABAP Laufzeit als neuer Prozess gestartet. Hier wird die serialisierte Klasse wieder deserialisiert und die implementierte Logik ausgeführt. Nach dem Ausführen wird der Prozess wieder beendet. Falls nötig, können durch das Bereitstellen eines eigenen Verarbeitungskontexts von der aufrufenden Anwendung spezifische Verarbeitungsprioritäten festgelegt werden. Dies ist zwingend notwendig, wenn die Verarbeitungsreihenfolge genau eingehalten werden muss. Der voreingestellte Kontext wird standardäSSig von anderen Anwendungen gemeinsam verwendet.[9]

Das Entkoppeln und asynchrone Ausführen gewisser Logik vom Hauptprozess bringt einige Vorteile mit sich: Zuerst wird die Robustheit und transaktionale Konsistenz des Hauptprozesses und somit auch die Konsistenz der Daten sichergestellt. Das bgPF bietet

[9]Vgl. SAP 2023f.

die Möglichkeit, diese Konsistenz zu gewährleisten, aber auch gleichzeitig relevantes Coding aus dem Hauptprozess auszulagern, was dessen Robustheit verbessert. Des Weiteren erscheint die Hauptanwendung für den Benutzer performanter, da gewisse Verarbeitungslogik in einer anderen Sitzung asynchron ausgeführt werden kann. Ein weiterer Vorteil ist die bessere Skalierbarkeit, wenn größere Prozesse in kleinere Teile unterteilt werden, die verteilt über mehrere Laufzeitumgebungen asynchron ausgeführt werden können. Die Skalierung einer Anwendung kann mit dieser Technologie dann komplett vom SAP-System automatisiert werden.[10]

bgPF innerhalb von RAP

Das bgPF kann auch innerhalb von RAP verwendet werden. Grundlegend besteht die Laufzeit eines RAP BOs aus einer Interaktions- und einer Speicherungsphase. In der Interaktionsphase werden über die Behaviours des BO dessen Daten ausgelesen oder verändert. Falls Änderungen an den Daten stattgefunden haben, werden diese jedoch nicht direkt auf der Datenbank persistiert, sondern zuerst in einem Puffer gespeichert. Nachdem alle Änderungen erfolgreich durchgeführt wurden, werden diese auf einmal in der Speicherungsphase auf die Datenbank geschrieben. Konkret wird die Methode des bgPF-Objekts, die das Speichern für die spätere asynchrone Ausführung übernimmt, in der zweiten Phase der Laufzeit des BOs aufgerufen und das Objekt somit auf der Datenbank gespeichert.[11]

3.1.4 Vergleich der Ansätze

Im Folgenden sollen die in den vorherigen Kapiteln vorgestellten Ansätze miteinander verglichen werden. Hierfür werden die Kriterien Komplexität der Implementierung, also welchen Aufwand das Umsetzen der Technologie im Anwendungsfall verursacht, die Auswirkungen auf die Systemlandschaft, also ob zusätzliche Systemkomponenten benötigt werden und die Performance der einzelnen Ansätze im Bezug auf Ausführungsdauer und Rechenlast verglichen werden. Weitere Kriterien sind zusätzliche Kosten die eine Lösung eventuell verursacht, die Abwärtskompatibilität zu älteren Systemen und die Flexibilität der Technologien im Bezug auf Gestaltungsmöglichkeiten von Anwendungsszenarien und

[10] Vgl. SAP 2023f.
[11] Vgl. SAP 2023f.

Integration mit anderen Technologien/ Frameworks. Abgeschlossen wird der Vergleich mit einer Betrachtung der Skalierbarkeit und Wartbarkeit der drei Umsetzungsmöglichkeiten.

3.2 Entscheidungsmatrix

Hier soll eine Entscheidungsmatrix entwickelt werden, welchen Lösungsansatz man in Abhängigkeit von mehreren Faktoren am besten verwenden soll (ersetzt auch weng mit die Zusammenfassung)

Vergleichskriterien:

- BTP Event Mesh Cloud nötig in Systemlandschaft, andere Lösungen laufen nur lokal -> mehr Kosten, Komplexität in Systemlandschaft, Datenschutz
- Kosten
- Performance (wshl verlieren BusinessEvents, Kommunikation über Systemgrenzen hinaus)
- Experteninterview: BusinessEvents (Martin Müller (+bgpf), Marcel Herrmanns, Daniel Wachs)
- Systemanalyse Vorlesung Kriterien für Software Folie (Leistung/ Effizienz, Flexibilität -> Betrieb/ Umgebungsbedingungen, Wartbarkeit, Portierbarkeit, Konformität, Skalierbarkeit)

mögliche Vergleichskriterien:
- Komplexität der Implementierung (Ranking: Workflow > Business Events > bgPF) -> Lines of Code, Artefakte
- Systemlandschaft (Cloudkomponente BTP bei Business Events nötig, bgPF/ Workflows brauchen das nicht)
- Performance (Rechenlast, Zeit) (BE nicht so gut, weil Kommunikation über Web zu Cloud über HTTP, bgPF <-> Workflows nehmen sich nicht viel)
- Kosten (Vorteil Workflows: Release (Abwärts-) Kompatibilität zu ECC (gibts schon immer), Drängen von Kunden auf Funktionsverfügbarkeit in alten Systemen) (BE: Lizenzkosten für BTP Komponente, Netzwerklast durch HTTP Kommunikation bei Kunde)
- Flexibilität (Workflows sehr flexibel, viele Anbindungen, sehr gute Integration zu zB. Emails) (Beiden anderen Techniken eher low level Code, wenig Anbindungen an andere Frameworks)

- Skalierbarkeit (Workflows (nicht aufteilbar), BE überhaupt nicht) (bgPF Autoskalierbar, -> siehe Praxiskapitel)
- Wartbarkeit (Workflows extrem gut wartbar -> komplett auf ABAP-Instanz debuggbar/ nachvollziehbar) (bgPF ähnlich) (BE schlechte Wartbarkeit, da über viele Systeme, Frameworks, intransparente Analysemöglichkeiten)

4 Schlussbetrachtungen

Im folgenden sollen die wichtigsten Ergebnisse noch einmal zusammengefasst, bewertet und eingeordnet werden. Zudem soll eine Handlungsempfehlung verfasst und die Arbeit einmal kritisch reflektiert werden. Abgeschlossen wird mit einem Ausblick auf weitere Entwicklungen.

4.1 Zusammenfassung

Hier nochmal die ganze Arbeit zusammenfassen

4.2 Handlungsempfehlung

Konkrete Handlungsempfehlung für die Praxis abgeben

4.3 Reflexion der Arbeit und Ausblick

Ergebnisse reflektieren und einen Ausblick auf zukünftige Entwicklungen geben

Literaturverzeichnis

[1] SAP. *Geschichte der SAP | Über SAP SE*. German. 2023. URL: https://www.sap.com/germany/about/company/history.html (Einsichtnahme: 13. 07. 2023).

[2] Fielding, R. „Architectural Styles and the Design of Network-based Software Architectures“. Diss. Irvine: University of California, 2000.

[3] Fielding, R. *REST APIs must be hypertext-driven ż Untangled*. 2008. URL: https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven (Einsichtnahme: 13. 07. 2023).

[4] SAP. *ABAP RESTful Application Programming Model | SAP Help Portal*. English. 2023. URL: https://help.sap.com/docs/btp/sap-abap-restful-application-programming-model/abap-restful-application-programming-model?locale=en-US (Einsichtnahme: 13. 07. 2023).

[5] SAP. *SAP UI5 Documentation, Developing Apps with Fiori Elements*. English. 2022. URL: https://ui5.sap.com//#/topic/03265b0408e2432c9571d6b3feb6b1fd (Einsichtnahme: 13. 07. 2023).

[6] SAP. *Developing SAP Fiori Applications with SAP Fiori Tools | SAP Help Portal*. English. 2023. URL: https://help.sap.com/docs/SAP_FIORI_tools/17d50220bcd848aa854c9c182d65b699/f09752ebcf63473e9194ea29ca232e56.html (Einsichtnahme: 13. 07. 2023).

[7] SAP. *SAPUI5: UI Development Toolkit for HTML5 | SAP Help Portal*. English. 2023. URL: https://help.sap.com/docs/SAPUI5/9c39e0e47bde4ed993d07d56b3ce3e08/95d113be50ae40d5b0b562b84d715227.html?locale=en-US&state=DRAFT&version=1.98_SAPUI5 (Einsichtnahme: 13. 07. 2023).

[8] SAP. *SAP Business Workflow | SAP Help Portal*. English. 2022. URL: https://help.sap.com/docs/SAP_NETWEAVER_700/109b51f56c531014b4e7c143c4b731a9/4f41e8a0dd89535fe10000000a421937.html?locale=en-US (Einsichtnahme: 17. 07. 2023).

[9] SAP. *Business Events | SAP Help Portal*. English. 2023. URL: https://help.sap.com/docs/btp/sap-abap-restful-application-programming-model/business-events (Einsichtnahme: 17. 07. 2023).

[10] SAP. *Creating an Event Consumption Model | SAP Help Portal*. English. 2022. URL: https://help.sap.com/docs/btp/sap-abap-development-user-guide/creating-event-consumption-model?locale=en-US (Einsichtnahme: 17. 07. 2023).

[11] SAP. *bgPF | SAP Wiki*. English. 2023. URL: https://wiki.one.int.sap/wiki/x/FhNsxg (Einsichtnahme: 17. 07. 2023).