



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 29. Juni 2023

Wolfrum, Tom

Inhaltsverzeichnis

Abkürzungsverzeichnis	IV
Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
1 Einleitung T-Shirt Size: L	1
1.1 Unternehmensprofil und Anwendungsbezug T-Shirt Size: S	1
1.2 Problemstellung T-Shirt Size: S	2
1.3 Motivation T-Shirt Size: S	2
1.4 Abgrenzung T-Shirt Size: S	2
1.5 Methodisches Vorgehen T-Shirt Size: S	2
2 Theoretische Grundlagen T-Shirt Size: XL	3
2.1 RESTful Application Programming Interface (API) T-Shirt Size: L	3
2.2 ABAP Restful Application Programming Model (RAP) T-Shirt Size: L	5
2.3 SAP Fiori Elements T-Shirt Size: L	5
3 Praktischer Teil T-Shirt Size: XL	6
3.1 Lösungsansätze	6
3.1.1 Workflows T-Shirt Size: L	6
3.1.2 Business Events T-Shirt Size: L	6
3.1.3 Background Processing Framework (bgpf) T-Shirt Size: L	6
3.2 Entscheidungsmatrix T-Shirt Size: M	7
4 Schlussbetrachtungen T-Shirt Size: L	8
4.1 Zusammenfassung T-Shirt Size: M	8
4.2 Handlungsempfehlung T-Shirt Size: S	8
4.3 Reflexion der Arbeit und Ausblick T-Shirt Size: S	8

Abkürzungsverzeichnis

SaaS	Software-as-a-Service
AIS	Application Innovation Services
HCM	Human Capital Management
API	Application Programming Interface
REST	Representational State Transfer

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung **T-Shirt Size: L**

1.1 Unternehmensprofil und Anwendungsbezug **T-Shirt Size: S**

SAP ist ein börsennotierter Softwarekonzern mit Sitz in Walldorf. Das Unternehmen wurde 1972 von 5 IBM-Mitarbeitern, darunter Hasso Plattner und Dietmar Hopp gegründet. Das Hauptgeschäft ist die Entwicklung von Unternehmenssoftware zur Abwicklung von Geschäftsprozessen, unter anderem in den Bereichen Buchführung, Controlling, Vertrieb, Einkauf, Produktion, Lagerhaltung, Transport und Personalwesen. Für das Unternehmen arbeiten heute 105.000 Mitarbeiter an Standorten in 157 Ländern und erwirtschaften einen Umsatz von ca. 29,5 Mrd. €. Erfolgreich wurde das Unternehmen mit seinem Standardsoftwarepaket SAP R/2 für Großrechnersysteme und später mit SAP R/3 für Client-Server-Systeme. Die Vorstellung der Mittelstandslösung SAP ByDesign im Jahr 2007 als Cloud-Produkt läutete die bis heute andauernde Transformation der gesamten Produktpalette in Richtung Cloud/ SaaS ein, die 2015 mit der Einführung von S/4 HANA als Hauptprodukt noch einmal verstärkt wurde.

Die Abteilung AIS HCM ist Teil des Product Engineering Unternehmensbereichs und zuständig für den 2nd-Level-Support und Eigenentwicklungen für die SAP on-Premise Personallösung HCM. Die Kunden der Abteilung sind Unternehmen die HCM verwenden und zusätzlich Wartungsverträge mit der SAP abgeschlossen haben. Zudem stellt die Abteilung mehrere SAP Fiori Apps als Self-Service für Mitarbeiter z.B. um Urlaub zu beantragen und Manager z.B. um Urlaubsanträge zu bearbeiten, bereit. Diese Apps sind für das Produkt HCM aufgrund des hohen Nutzungsvolumens von großer betriebswirtschaftlicher Bedeutung.

1.2 Problemstellung **T-Shirt Size: S**

Bestehendes Problem, Auslöser (Umstieg auf Business Objects? -> Was genau heiSSt das? Von was aus wird umgestiegen? Wieso wird umgestiegen?) erklären (Umstieg zu Fiori Elements, dadurch keine Logik mehr im Frontend, dadurch ist stateless angelegte REST-API ein problem für sequentielle Prozesse, die asynchrone Kommunikation verwenden) -> Somit müssen Möglichkeiten, wie man diese Prozesse trotzdem umsetzen kann, ausloten

Durch diesen Umstieg der xy auf BusinessObjects wird es notwendig, die von der Abteilung betriebenen Apps von UI5 Freestyle auf das neuere Framwork Fiori Elements umzustellen.

1.3 Motivation **T-Shirt Size: S**

Wieso ist das Thema bzw. die Arbeit wichtig?

1.4 Abgrenzung **T-Shirt Size: S**

Hier die Arbeit klar abgrenzen, also was genau behandelt wird und eben was auch nicht betrachtet wird (z.b.) das keine Implementierung der vorgestellten Ansätze erfolgen soll, (wenn dann nur kleiner Prototyp) Ggf. auch kurz darauf eingehen welche Ansätze auch nicht beleutet werden

1.5 Methodisches Vorgehen **T-Shirt Size: S**

Wie werden die Ansätze betrachtet? Wie kommt die Handlungsempfehlung zustande?

Nachdem zunächst im theoretischen Teil der Arbeit die Grundlagen für die verschiedenen Ansätze sequentielle Prozesse im RESTful-API Umfeld abzubilden gelegt werden,

2 Theoretische Grundlagen **T-Shirt** **Size: XL**

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 wird auf die Umsetzung von REST in ABAP näher erleutert. Abgeschlossen wird der theoretische Teil der Arbeit mit einer Darstellung von Fiori Elements, dem Framework zur Entwicklung von Fiori Apps.

2.1 RESTful Application Programming Interface (API) **T-Shirt Size: L**

Erstmal die theoretischen Grundlagen einer RESTful-API zu legen.

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.

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.

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 und 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 Informationen, 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 eine 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.

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 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.

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. Zudem wird eine unabhängige Weiterentwicklung der verschiedenen Komponenten 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 Identifika-

tion von Ressourcen, Ressourcenmanipulation durch Repräsentationen, selbst-erklärende Nachrichten und die Definition des Zustands der API durch die zugrundeliegenden Daten.

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 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 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.

2.2 ABAP Restful Application Programming Model (RAP) T-Shirt Size: L

Hier dann nochmal auf ABAP RAP eingehen, wie REST hier umgesetzt wird

2.3 SAP Fiori Elements T-Shirt Size: L

Erklären was Fiori Elements ist, auch auf technische Details eingehen (keine Logik im Frontend, Framework)

3 Praktischer Teil **T-Shirt Size: XL**

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

3.1 Lösungsansätze

Jetzt werden 3 verschiedene Lösungsansätze vorgestellt, wie man trotzdem sequentielle Prozesse/ asynchrone Kommunikation umsetzen kann

3.1.1 Workflows **T-Shirt Size: L**

Das ist die bisherige alte/ ineffiziente Lösung. Diese wird zwar noch vorgestellt, soll aber überdacht werden.

3.1.2 Business Events **T-Shirt Size: L**

Eine Option wären die Verwendung von Business Events. Hier auch ggf. auf Probleme mit Event-Mesh (Cloud- bzw. BTP-Komponente) für onPremise-Systeme eingehen -> lokale Verarbeitung der Business Events?

3.1.3 Background Processing Framework (bgpf) **T-Shirt Size: L**

Andere Option wäre das Background Processing Framework über Background remote function calls.

3.2 Entscheidungsmatrix T-Shirt Size: M

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

4 Schlussbetrachtungen **T-Shirt Size: L**

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 **T-Shirt Size: M**

Hier nochmal die ganze Arbeit zusammenfassen

4.2 Handlungsempfehlung **T-Shirt Size: S**

Konkrete Handlungsempfehlung für die Praxis abgeben

4.3 Reflexion der Arbeit und Ausblick **T-Shirt Size: S**

Ergebnisse reflektieren und einen Ausblick auf zukünftige Entwicklungen geben