Abschlussprüfung Sommer 2022

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Verbesserung einer Anwendung zur Meldung von aktuellen Verkaufspreisen von Wettbewerbern

SAP-Anwendung „**Report Competitor Prices**“

Vorgelegt von:

Mohammed Alsaloum Altarkawi

Sönke-Nissen-Allee 2A

21509 Glinde

Prüfungsnummer:

131-54246

Ausbildungsbetrieb:

Implico GmbH

Weidestraße 120 B

22083 Hamburg

Ausbilder: Thorsten Klingspor (Tel.: +49 40 270936 263)

Betriebliche Betreuerin: Anne Luther (Tel.: +49 40 270936 391)

Inhaltsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

- ABAP - Advanced Business Application Programming
- CDS – Core Data Service
- CRUD – (Create, Read, Update, Delete) Methoden
- ERD – Entity Relationship Diagram
- GUID – Global Unique Identifier
- IMDB – In-Memory-Datenbank
- IS-OIL – Industry Solution Oil and Gas
- JS – Javascript
- RFNO – Retail Fuel Network Operations
- SDM – Secondary Distribution Management for Oil & Gas
- SQL – Structured Query Language
- UI – User Interface (Benutzeroberfläche)
- URI – Uniform Resource Identifier
- VDM – Virtual Data Model

1 Einleitung

Die folgende Dokumentation beschreibt das Projekt zur Verbesserung einer Anwendung zur Meldung von aktuellen Verkaufspreisen von Wettbewerbern, das im Rahmen der Ausbildung zum Fachinformatiker Fachrichtung Anwendungsentwicklung durchgeführt wurde.

Dieses Kapitel stellt das beteiligte Unternehmen vor und definiert den Projektantrag bzw. beschreibt den Projektziel.

1.1 Vorstellung des eigenen Unternehmens

Der Ausbildungsbetrieb ist Implico GmbH, ein Anbieter von Software, Beratung und Dienstleistungen für die Mineralölindustrie. Implico befindet sich in Downstream-Branche. Der Unternehmen entwickelt eine SAP-Anwendung für den Teil „Secondary Distribution“. Downstream Secondary Distribution ist der Prozess der Verteilung der Produkte von der Raffinerie/Tanklager bis zum Endverbraucher. Endverbraucher können private oder gewerbliche Haushalte oder auch Tankstellen sein. Die Prozesse, die unterstützt werden, sind unter anderem die Lieferungen und die Abrechnung gegenüber unterschiedlichen Geschäftspartnern.

Implico ist ein Entwicklungspartner von SAP SE, die Software-Industrielösungen in vielen Branchen entwickelt. Die Implico SAP-Produkte basieren auf der Industrielösung “Öl und Gas“. Die Anwendung im Projekt wurde auf einem und für ein SAP-System entwickelt.

1.2 Projektbeschreibung

Das Projekt umfasst das Reengineering der bestehenden Anwendung „Report Competitor Prices“. Die Anwendung gibt die Möglichkeit, Preise der Materialien an Wettbewerber-Tankstellen zu melden.

Wettbewerber-Tankstelle sind in der Regel Tankstellen, die aus einem anderen Tankstellennetz sind, sich aber in der Nähe eines eigenen Tankstellen-Standortes befinden.

Die Anwendung ist in dem größeren Produkt integriert. Implico erstellt ein SAP Add-on. Dieses ist in zwei Hauptkomponenten unterteilt. Eines davon ist Retail Fuel Network Operations (RFNO). Die RFNO-Komponente bietet Interfaces und Funktionen, um die Geschäftsvorfälle in Vertriebsnetzen, deren Hauptgeschäft der Verkauf von Kraftstoffen an Endverbraucher ist, zu automatisieren. Dazu gehört die Verwaltung der Vertriebsnetze und die Abstimmung und Abrechnung der Geschäftsvorfälle an einem Standort.

1.3 Projektziel

Das Ziel des Projekts ist Umstrukturierung der Benutzeroberfläche der Anwendung „Report Competitor Prices“. Wegen der Besonderheiten der Entwicklung im SAP-Umfeld, ist die Gestaltung eines gesamten Features innerhalb des Zeitrahmen von 70 Stunden nicht möglich.

1.4 Projektumfeld

Der Auftraggeber des Projekts ist die Beratungsabteilung von Implico GmbH. Die Berater stehen in Kontakt mit den Kunden und sind vor Ort, um ihre Arbeitsprozesse so gut wie möglich zu verstehen und zu optimieren. Aus einem Gespräch mit den Kunden, ist die Idee von der Umstrukturierung der bestehenden Anwendung entstanden.

1.5 Projektschnittstellen

In der Entwicklung besteht immer eine klare Schnittstelle mit der Beratungsabteilung. Sie kennen die Bedürfnisse der Kunden und kommunizieren sie weiter. Die Zusammenarbeit mit der UI-Expertin und Mentorin und mit erfahrenen Entwicklern ist sehr lehrreich und verbessert die Qualität des Projekts.

2 Projektplanung

Nachfolgend werden Informationen zur Projektplanung, Umgang mit den Ressourcen im Entwicklungsprozess, Projektabweichungen, gegeben.

2.1 Projektphasen

Für die Planung und Umsetzung des Projekts wurde einen Zeitplan erstellt. Dieses Projekt fand im Zeitraum vom 04. April 2022 bis zum 16. Mai 2022 statt. Es wurden 70 Stunden zur Verfügung gestellt. In Tabelle 1 Finden Sie die grobe Zeitplanung. (Für eine Bildschirmaufnahme von einer detaillierten Planung, siehe Soll-Ist-Vergleich).

Phase	Dauer in Stunden
Analyse	5
Mockup	4
Implementierung	43
Test	4
Abschlussphase	14
Summe	70

Tabelle 1: Zeitplan zur Realisierung des Projekts

2.2 Abweichung vom Projektantrag

Nach gründlicher Analyse und in Abstimmung mit dem Abteilungsleiter wurde sich für einen Backendumbau vom ABAP auf CDS Views entschieden. Dies führt zu einer sehr flexiblen Anwendung, welche durch die Annotation gesteuert werden kann. Allerdings kann damit aktuell keine Code-Logik entwickelt werden, um die Materialpreise aus den RFNO-Belegen zu holen. Die RFNO-Belege sammeln Information von allen Vorgängen einer Tankstelle. Diese könnten Daten über Lieferung. Verkauf. Zählerstsand, Materialpreise etc sein. Die RFNO-Belege sollen nicht von jedem geändert werden. Hier wurde eine zusätzliche Tabelle für die Preise, die die Materialien und die Wettbewerber als Fremdschlüsselfelder und Felder für die Preise und die letzte Änderung hat, erstellt. Diese Code-Logik kann mit einem neuen Auftrag nach Abschluss des aktuellen Projekts durch eine Erweiterung realisiert werden.

2.3 Ressourcenplanung

Hardware:

- Laptop
- Virtual Desktop Interface auf SAP internem Netzwerk
- Zugang zum Implico System auf SAP-Netzwerk

Software:

- SAP Web IDE
- ABAP Development Tool
- Citrix
- Figma
- Microsoft Visio

Personal:

- UI-Expertin
- Berater – Kommunikation der Anforderungen, Review der Prototypen
- Entwicklerin – Code Review
- Auszubildender – Ausführung des Projekts

2.4 Entwicklungsprozess

Vor dem Anfang der Entwicklung musste der Auszubildende sich für einen Entwicklungsprozess entscheiden. Die Anwendung wurde als Full-Stack entwickelt. In Full-Stack-Entwicklung ist die Entwicklung vom Backend und Frontend gekoppelt und wird durch den gleichen Entwickler durchgeführt. Wegen der kombinierten Form der Arbeit, kann es schwer sein, allen Prozessen gleichzeitig zu folgen, deswegen wurde die agile Form gewählt. Die Teilung der Aufgaben in atomische Aufträge hat es leichter gemacht, sicherzustellen, dass die Funktionalität richtig ist und die Kommunikation zwischen Backend und Frontend funktioniert wie erwartet und gewollt. Durch das

Testen jeder neuen Funktionalität während der Entwicklung wurden Bugs gefunden und sie waren leicht zu reparieren, weil die Änderungen zwischen den Tests relativ klein waren.

3 Planungs- und Analysephase

Nachfolgend werden Informationen zur Projektplanung nach dem Soll/Ist-Analyse, Projektkosten, gegeben.

3.1 Ist-Analyse

Die TankstellenzentralmitarbeiterInnen geben die Preise von den jeweiligen Materialien eines Wettbewerbers in die Anwendung „Report Competitor Prices“ bei Änderungen ein. Früher haben die MitarbeiterInnen die Preise einmal täglich angepasst. Denn es ändert sich nur einmal am Tag. Ein zusätzliches Feature an den Benutzeroberflächen in veralteten Technologien einzufügen, erfordert zusätzlichen Wartungsaufwand und ebenso fragen Kunden nach Anwendungen mit neuesten Technologien. Diese Anforderungen von Seite der Kunden nehmen mehr zu.

3.2 Soll-Analyse

Kunden wünschen sich, dass die TankstellenzentralmitarbeiterInnen einen besseren Blick auf die Materialpreise der Wettbewerber haben, denn die Preiseändern sich mehrmals an einem Tag. Zusätzlich wollen die TankstellenmitarbeiterInnen einen Verlauf für die vorherige Materialpreise eines Wettbewerbers analysieren. Um die Produktivität der Benutzer ebenfalls zu erhöhen und die entsprechende Kundenanfragen zu erledigen, wird eine neue Benutzeroberfläche benötigt. Mit den neuen SAPUI5-Elementen z.B SmartTable, ist die Bedienung einfacher für Mitarbeiter ohne Vorkenntnisse in der Nutzung von SAP-Oberflächen und die Kundenanforderungen umsetzbar.

3.3 Wirtschaftlichkeitsanalyse und Nutzwertanalyse

Die Wirtschaftlichkeit läßt sich auf dem Grund der Integration in einem SAP Add-on nicht exakt bestimmten. Durch die Umsetzung der SAP-Vorgaben und Kundenwünsche zur Verwendung der neuesten Technologien erwarten wir eine Steigerung der Attraktivität des gesamten Moduls. Die Wartungsarbeit von den Entwicklern auf diese bestehende Anwendung beträgt im Durchschnitt vier Stunden Arbeit monatlich. Das ist eine grobe Schätzung. Die Verbesserung der bestehenden Anwendung auf eine neue Technologie erspart jeden Monat die zusätzliche Wartungssaufwand der Entwickler. Durch die neue Änderung und die Nutzung der neuen Technologien wie SmartTable und CDS-Views wird die Zeit ebenfalls für Analyse erspart. Da die CDS-Views einen großen Teil vom ABAP-Code, sowie Smartelemente vom Frontend-Code übernehmen. Dies führt auch zur Verringerung des Quellcodes. Ein kurzer Quellcode ist weniger fehleranfällig und erleichtert die Korrekturen.

3.4 Projektkosten

Für die Entwicklung des Projekts war die Beschaffung neuen Materials nicht notwendig. Die Entwicklung wurde auf einem Firmenrechner und VDI durchgeführt, die für die Entwicklung des Produkts verwendet werden, sie sind Teil der Fixkosten.

Die Personalkosten sind eine Schätzung, da wegen Datenschutz das genaue Gehalt nicht mitgeteilt wird. Sie beinhalten auch die Sozialkosten. Es wird für ein Auszubildene 15 € pro Stunde und einen Entwickler oder Berater 30 € pro Stunde gerechnet. Die Nutzung des SAP-Systems ist zum einzelnen Projekt nicht zu verfolgen. Es wird für das gesamte Add-On verwendet.

Eine Kosten-Nutzen-Analyse ist nicht möglich, weil die App ein Teil eines großen Produkts ist und die Entwicklung der einzelnen Features und Apps basieren auf die Wünsche von Kunden, die durch die Berater kommuniziert werden. Das Einkommen von einer Teilanwendung ist unmöglich zu berechnen.

Hardware- und Softwarekosten:

Beschreibung	Zeit in Stunden	Kosten in Euro
Computer	70	67
VDI Miete	70	3

Tabelle 2: Hardware- und Softwarekostenrechnung

Personalkosten:

Beschreibung	Zeit in Stunden	Kosten in Euro
Auszubildender	70	1050
Begleitung durch erfahrene Entwickler	1	30
Berater	1	30
UI-Expertin	1	30

Tabelle 3: Personalkostenrechnung

Die ungefähre Kostenschätzung liegt bei 1.210€ für das gesamte Projekt.

4 Entwurfsphase

Nachfolgend werden Informationen zur Zielplattform, zu den Technologien, die in der Umsetzung des Projekts, benutzt wurden, gegeben.

4.1 Zielplattform

Die "Report Competitor Price“- Anwendung ist ein Teil von dem RFNO-Modul. Wie oben beschrieben, ist das Add-On für das SAP-System konzipiert. Die Teile der Benutzeroberfläche des Add-Ons sind in unterschiedlichen SAP-Oberflächen Technologien geschrieben. Die neueste Oberfläche basiert auf der SAPUI5-Javascript-Bibliothek mit dem Konzept von Fiori-Apps. Fiori-Apps sind modulare Anwendungen mit spezialisierten Funktionen, die gemeinsam eine komplexe Aufgabe erfüllen.

Die Backend-Funktionalität ist in der Programmiersprache ABAP und CDS-Views geschrieben und ihre Schnittstelle zur Benutzeroberfläche ist ein OData-Service.

4.2 Architekturdesign

Die Anwendung ist eine Fiori-Applikation. Fiori ist ein Benutzeroberflächen-Konzept von SAP SE. Dieses Konzept standardisiert die Struktur der modularen Applikationen, die die SAPUI5-Bibliothek verwenden. Die Applikation kommuniziert mit der SAP-Datenbank durch einen OData-Service. Im Rahmen des Projekts wurden das Backend und Fiori-Frontend entwickelt.

Nachfolgend werden zunächst die verschiedenen Technologien beschrieben, die für das Backend verwendet wurden. Danach folgen die drei verschiedenen für das Frontend.

4.2.1 Backend

Das Backend wird in sowohl CDS-Views als auch ABAP implementiert. Das Backend wird auf einer Datenbank mit der Datenbanktechnologie SAP HANA entwickelt. Es handelt sich um eine In Memory Database (IMDB). IMDB sind Datenbanken, die Daten im sehr schnellen RAM bereithalten. In klassischen Datenbanken werden die Daten auf langsameren Festplatten gespeichert. Die Bereitstellung auf einem schnelleren Medium führt zu schnelleren Geschwindigkeiten für die Lektüre der Daten und damit auch die Möglichkeit mehr Prozessoren zu verbinden, um die Analyse zu beschleunigen.

CDS-Views

Die Datenbeschaffung findet nicht direkt von den Datenbanktabellen statt. CDS-Views werden als Zwischenschicht verwendet. CDS-Views sind ein SAP-Werkzeug, mit dem man schnelle Anfragen an die Datenbank machen kann. Die existierenden CDS-Views liefern Details über die Wettbewerber, für die der Benutzer zuständig ist. Während dieses Projekts wurden diese CDS-Views erweitert und ein OData-Service erstellt, um Details nicht nur über die Wettbewerber zu liefern, sondern auch über die Materialien, die jedem Wettbewerber zugordnet. Dieses Beispiel zeigt, dass die CDS-Views modular sind, und die Arbeit in diesem Projekt könnte auch für weitere Anwendungen wertvoll sein. Weiterhin kann man im OData-Service durch die neueingeführten SAP-Annotationen die Darstellung von den

Benutzeroberflächen festlegen. Filtereigenschaften von den Tabellen, Draft-Handling, Eingabeprüfungen, semantische Zusammenhänge zwischen Werte und ihre Einheiten uvm. können hier definiert werden. Das heißt, dass die CDS-Annotationen genug Daten liefern, die in der SAP Web IDE die Nutzung der App-Templates ermöglichen. Dadurch kann man mit wenig Programmieraufwand eine Fiori-Anwendung auf die Beine bringen. Die CDS-Views ersetzen/verschalen kurz gesagt komplexe ABAP SQL-Anweisungen und bieten darüber wesentliche Vorteile wie: Generierung von Fiori-Oberflächen, Nutzung der analytischen S/4HANA Funktionen, dokumentiertes und verständlicheres Datenmodell. Dies gliedert die Datenbanktabellen als VDM (Virtual Data Model) ein. Die VDMs basieren auf drei Schichten:

- Datenbanktabellen
- Basic Interface View
- Consumption View

OData-Service

Entitäten, Funktionen, Funktionsimporte und Navigation werden in diesem Tool beschrieben. Die Klassen werden auf dieser Basis generiert. In den generierten Klassen sind die CRUD Funktionen definiert und werden in der Erweiterungsklasse neudefiniert.

Funktionsimporte sind Teil der OData-Services. Man kann eine Anfrage mit dem URI für die Funktion vom Frontend stellen und die Rückgabe der Funktion, die auf dem Backend gelaufen ist, als Antwort bekommen. Eine Entität für die Rückgabe ist notwendig.

ABAP

ABAP ist eine proprietäre Programmiersprache von SAP SE. Das ist eine klassenorientierte Sprache. Für die Implementierung des OData-Service sind verschiedene Klassen nötig. Muster für diese Klassen werden durch den SAP-Gateway-Builder (SEGW) generiert.

4.2.2 Frontend

Das Frontend wird als Fiori-Applikation mit der SAPUI5-Bibliothek gebaut. Das gewählte Entwurfsmuster ist MVC. MVC ermöglicht eine klare Trennung zwischen der Oberfläche, Logik und der Datenbeschaffung innerhalb der Applikation. Diese Form ist sehr flexibel und ermöglicht die Wiederverwendbarkeit von vielen Komponenten.

Models

Ein wichtiges Modell ist i18n. In diesem Model werden alle Texte, die in der Anwendung verwendet werden, gespeichert. Diese Datei und damit die ganze Applikation kann übersetzt werden. Die Produkte von Implico GmbH sind in mehreren Sprachen verfügbar.

Views

Die Views sind in XML geschrieben. Durch SAPUI5 ist die Datenbindung in vielen Fällen direkt in der View möglich. Das hilft einen lesbareren Code zu schreiben und reduziert die Anzahl an Javascript-Funktionen im Controller mit wenig interner Logik zu reduzieren. Die Datenbindung kann dann durch

„Formatters“ erweitert werden. Das sind Javascript-Funktionen, die nur für die Bearbeitung der Daten für die Präsentation zuständig sind. Sie werden oft in einem getrennten Modul definiert. Die Wiederverwendbarkeit der Formatters ist sehr stark. Die Entwicklung folgt die SAP-Entwicklungsrechtlinie für Umsetzung der UI-Elemente.

Controller

Controller werden in Javascript geschrieben und beinhalten die Logik der Anwendung. Viele Funktionen und Objekte basieren sich auf der API von SAPUI5. In SAPUI5 ist jeder Controller ein Modul, das durch die „define“-Methode vom Namespace sap.ui definiert wird.

4.3 Entwurf der Benutzeroberfläche

Das Design der Oberfläche wurde gemeinsam mit den Beratern geplant, die in Kontakt mit den Kunden stehen. Nach einem ersten Gespräch wurde ein interaktiver Design-Prototyp in Figma entworfen. In dem anschließenden Review-Termin konnte der Berater das Design testen. Im Gespräch wurden Anmerkungen angenommen, die dann in den funktionsfähigen Prototyp eingeflossen sind. (Für eine Bildschirmaufnahme, siehe …).

4.4 Datenmodell

Die benötigten Entitäten für die Umsetzung sind Standard-SAP-Tabellen, die mit ein anderen verbunden sind. Die Standard-SAP-Tabellen sind Datenbanktabellen, die aus anderen Modulen erstellt worden und von überall zugreifbar sind. Diese Datenbanktabellen können im SAP-System wiederverwendet werden. Die Verbindung zur Tabelle OIFSPBL auf dem Schlüsselfeld, der den Standort (LocationID) beinhaltet war unter anderem sehr wichtig, der ebenfalls auch als Fremdschlüssel in der Tabelle OIRBPBLPBL, die die Wettbewerber (Comp) beinhaltet. Durch die beiden Schlüsselfelder werden in der CDS-View Comp (Für eine Bildschirmaufnahme vom Aufbau einer CDS-View, siehe Basic CDS-View). weitere Daten aus anderen Tabellen wie Adresse aus der Standard-SAP-Tabelle ADRC geholt. In die Preis-Tabelle werden Einträge von den Materialpreise, Währung und das Datum als Felder sowie die Location, Wettbewerber und Material als Schlüsselfelder eingetragen. Die History-Tabelle hat das Datum zusätzlich als Schlüsselfeld.

5 Implementierungsphase

Nachfolgend werden Information zur Implementierung des Front-/Backends, Erklärung zum Umgang mit den SAP-Entwicklungsrichtlinien, gegeben.

5.1 Backend Implementierung

Die Backend-Implementierung hatte während der Entwicklung die Priorität vor dem Frontend. In dem Backend wird das Datenmodell erschaffen, denn das Datenmodell die Grundlage für die Anwendung. Das Frontend kann anhand dessen orientieret werden. SAP empfiehlt, die SAP-Entwicklungsrichtlinien zu beachten. Dabei sind zum Beispiel die Standarddatenelemente für Preis-

und Währungsfelder zu benutzen. Dies führt zur Referenz mit den beiden Felder innerhalb der Datenbanktabelle.

5.1.1 Implementierung der Datenbanktabelle

Wie oben in der Projektabweichung genannt wurde, wurde zwei Tabellen (Preise, History) erstellt. Die Erstellung der Tabellen in SAP erfolgt über eine SAP-Anwendung "ABAP Dictionary“ mit der Transaction (SE11), in der nicht nur Tabellen, sondern andere ABAP-Objekte wie Datentypen, Data-Elemente. Die Anwendung bietet eine graphische Oberfläche zur Definition einer Datenbanktabelle. Jede Spalte wird durch ein Datenelement ein Datentyp zugeordnet. Wenn ein Datenelement verwendet wird, wird automatisch der Typ des Datenelements ergänzt. Ebenfalls muss die Datenbanktabelle einen Tabellentypen, Größenkategorie, Datenart, und andere technische Einstellungen haben. Nach der Aktivierung einer SAP-Datenbanktabelle ist sie in dem System erreichbar. Als Beispiel dient die Preis-Tabelle. (Für eine Bildschirmaufnahme von ABAP Dictionary, siehe Oberfläche der Anwendung ABAP Dictionary).

5.1.2 Implementierung der CDS Views

Für die Daten über Wettbewerber, Materialien, Preis-Verlauf wurden durch CDS-Views erstellt. Durch die drei Schichten der VDMs eine Basic View, um Logik und die Verbindung von verschiedenen Tabellen durch die Schlüsselfelder zu erschaffen, eine Consumption View jeweils, um die jeweilige Basic-View aufzurufen und die Annotationen zu beinhalten. Die Implementierung wurde mit ABAP Development Tool mit einem Z-Namensraum ausgeführt. Jeweilige CDS-Views greifen auf eine oder mehrere SAP-Datenbanktabellen über SELECT bzw. SQL-Statements zu. Ebenfalls hat eine CDS-View technische Befehle zur Einstellung der EntityName, SQLViewName etc. (Für eine Bildschirmaufnahme vom Aufbau einer Consumption CDS-View, siehe Consumption CDS-View).

5.1.3 Implementierung der Assoziation und Navigation

Eine Assoziation und Navigation wurden zwischen den Wettbewerbern- und Materialien-Entitäten in den CDS-Views durch die Schlüsselfelder hergestellt. Die Assoziation in den CDS-Views der Tabellen Materialien zu Wettbewerber ist definiert als many-to-many. Das ermöglicht die Nutzung des Parameters in einer HTTPS-Abfrage, und einfacher Data-Binding im Frontend, um alle Materialien eines Wettbewerbers zu beschaffen.

5.1.4 Implementierung der OData Entitäten

Ein OData-Service braucht CDS-Views, um die Entität und die Entitätstypen zu definieren. Jede View ist gleich eine Entität, Die Entitätsfelder werden ebenfalls automatisch durch OData-Service, außerdem erstellt die OData-Service die Navigation durch die in den CDS-Views gegebene Assoziation. Die Implementierung erfolgt über die Anwendung „SAP Gateway Service Builder“ mit der Transaction SEGW. Diese Anwendung ist für Erstellung bzw. Änderung der OData-Services. Bei Einfügen einer CDS-View zu einem OData-Service, wird aus der CDS-View eine Entität mit den dazugehörigen Feldern geschaffen. Die OData-Service muss über die Anwendung „Activate and

Maintain Services“ mit der Transaction /n/iwfnd/maint_service im System registriert werden. (Für eine Bildschirmaufnahme vom Aufbau des OData-Service, siehe OData-Service).

5.1.5 Generierung der Klassen

Nach der Erstellung der Entitäten durch CDS-Views werden die Klassen, die die CRUD Methoden beinhaltet, generiert. Für jede Entität kann bis zu fünf generierbaren Methoden sein (Create, Get_Entity, Get_EntitySet, Update_Entity, Delete_Entity). In jeder entsprechenden Klasse kann die benötigte Logik implementiert werden. Die Nutzung der CDS-Views erspart die Arbeit mit allen Read-Methoden. Die Methode Material_Update_Entity kriegt einen Übergabeparamater in Form eine Tabelle von Material-Entität, falls es mehrere Änderungen gibt. Nach der Erstellung des OData-Service und Einfügung der benötigten CDS-Views, können die Klassen generiert werden.

5.2 Frontend Implementierung

Nach jedem Schritt der Frontend-Implementierung wurde die neue Funktionalität manuell vom UI getestet und die Ergebnisse auf dem Backend verifiziert. SAP bietet die Web IDE für die Entwicklung der SAPUI5-Anwendungen. Nach der Erstellung der Anwendung werden Standard-Datei und Ordner automatisch generiert (Für eine Bildschirmaufnahme vom SAPUI5-Projekt, siehe Frontendprojekt). Die UI5-Manifestdatei (Manifest. JSON) beinhaltet Einstellungen für die Anwendung zum Beispiel das Model, Destination zum Backend durch einen URI-Link, SAPUI5-Version und Routing zwischen den Seiten.

5.2.1 Implementierung der Main View

Nach der Erstellung der SAPUI5-Anwendung wird eine View mit dem dazu gehörigen Controller generiert. Die View beinhaltet die XML-Tags und die Zeichenelemente, die die Hauptseite darstellt. Anhand der SAP-Entwicklungsrichtlinie und die SAPUI-Dokumentation, wurde ein GridList-Element für die Darstellung der Wettbewerber verwendet. Die GridList hat die Eigenschaft items, in der die Entität (Competitor) eingegeben werden kann. Die GridList ist ein sogenanntes Parent-Element vom GridListItem, die alle Wettbewerber aus dem Modell als Kacheln in der GridList darstellt, dabei können Aktionen und Einstellungen für die Kacheln durch Eigenschaften hinzugefügt werden.

5.2.2 Implementierung der Detail View

Bei Erstellung einer View generiert die Web IDE automatisch die jeweiligen Controller. Für die Detail-View wurde sich für eine SemanticPage entschieden. Dieses SAPUI5-Element bietet dynamische Funktionen mit sehr viel Flexibilität, zum Beispiel Footer, das kann andere Elemente beinhalten und durch eine Funktion aus-/eingeschaltet werden. Eine TabBar kann ebenfalls in eine SemanticPage integriert werden. Diese ermöglicht es, mehrere Tabs innerhalb einer Seite zu implementieren, die jeweils unterschiedliche Elemente haben. Durch Klicken auf das TabBar-Element kann zu in die entsprechenden Tabs gewechselt werden. In den beiden Tabs ist jeweils ein SAPUI5-Smartelement. Die Smartelemente sind einfach zu steuern, durch die Annotation, die in den CDS-Views

implementiert wurden. Die Definierung des Datenmodells in einem Smartelement erfolgt über die Eigenschaft EntitySet, diese hat das Datenmodell als Übergabeparamater.

5.2.3 Bindung die Wettbewerber und Materialien in der Detail-View

Das Routing zwischen den beiden Seiten ist eine durch die Routing-Definition in der Manifestdatei zu implementieren, diese geht über die Schlüsselfelder LocationID, Comp. Durch die definierte Assoziation in den CDS Views steht die Tabelle Materialien zu Wettbewerber als Many-To-Many. Das Data-Binding in einem Smartelement erfolgt über die Eigenschaft tableBindingPath und enableTableBinding, diese hat die Assoziation-Entity als Übergabeparamater.

5.2.4 Besonderheiten der Benutzeroberfläche

SAPUI5-Elemente können in dem dazugehörigen Controller über den gegebenen ID instanziiert werden, die eventuell durch ihre eigene Aggregation Funktionen beinhalten. Diese Funktionen bieten Flexibilität und dynamische Einstellungen an, zum Beispiel über der Instanz von der SemanticPage kann der Footer durch getFooterEnabled (Boolean) de-aktiviert werden. Diese ist bei Klicken auf die Edit-Drucktaste implementiert. (Für eine Bildschirmaufnahme, siehe Besonderheiten der UI) Ebenfalls wird die Save-Drucktaste nach einer Änderung in der Tabelle. de-aktiviert. Diese dynamischen Einstellungen sind sehr wichtige Punkte in den SAP-Anwendungen für die Benutzerführung.

5.2.5 Implementierung der Datenspeicherung

Die SmartTable bietet nach dem Instanziieren eine Funktion „submitChanges“. Diese Funktion gibt alle Änderungen ins Backend, in die Methode Material_ Update_Entity in den OData-Klassen. Dafür wurde eine Code-Logik mit ABAP implementiert, um die Daten in die History- und Preis-Datenbanktabelle zu speichern. (Für eine Bildschirmaufnahme vom Quellcode, siehe ABAP-Code)

6 Erstellung der Dokumentation

In diesem Kapitel werden Information für die Dokumentationen gegeben.

6.1 Entwicklerdokumentation

Die Entwicklerdokumentation befindet sich in dem Code in Form der Kommentare. Die Datenelemente, Datenbanktabellen und die bestimmten Klassen des OData-Service sind in Form ein Text in Implicos internen Tool „Confluence“ dokumentiert. Bei der Dokumentation handelt sich um eine detaillierte Beschreibung des Projektes wie die Begründungen, warum es auf eine CDS-View basierten backend entschieden wurde.

6.2 Dokumentation für die Berater

Eine Dokumentation für die Berater, die in Kontakt mit den Kunden stehen, wurde parallel zum Schreiben dieses Dokuments erstellt. Sie beinhaltet eine kurze Beschreibung der Anwendung und Anleitungen mit Bildschirmaufnahmen für die verschiedene Funktionalitäten.

7 Planung der Einführung

Im nachfolgenden Kapitel werden Information zur Qualitätssicherungsphase gegeben.

7.1 Übertragung zum Implico GmbH Namensraum

SAP bietet ihren Kunden die Möglichkeit für sich mit Z-Namensraum zu entwickeln. Das Projekt wurde ebenfalls im Z-Namensraum des Autors entwickelt. Für die Einführung im Produkt muss es in den produktiven Namensraum übertragen werden.

7.2 Übertragung in Qualitätssicherungssystem

Alle Anwendungen werden auf einem getrennten System vom Entwicklungssystem getestet

7.3 Qualitätssicherungsabteilung

Qualitätssicherung ist sehr wichtig bei der Entwicklung neuer Applikationen. Die Tests wurden wie oben beschrieben von dem Entwickler durchgeführt. Die Qualitätssicherungsabteilung baut ein präziseres und mehr tiefgehenden manuellen und automatischen Test-Katalog auf. Wenn Fehler gefunden werden, werden sie an den Entwickler berichtet. Der Entwickler behebt den Fehler und gibt die Anwendung wird zurück zur Qualitätssicherung für den Nachtest.

7.4 Einführung bei den Kunden

Die Einführung und Anleitung der Benutzer werden entweder von den Implico Beratern, die im Kontakt mit den Kunden stehen, oder durch die SAP-Online-Help durchgeführt. Im Fall von Fragen ist der Entwickler verfügbar. Wenn Fehler gefunden werden, kann der Kunde einen Support-Ticket aufmachen und der Korrekturprozess wird ausgelöst.

8 Projektabschluss

In diesem Kapitel werden Information zum Zeitvergleich und Rückblick zur Realisierung des Projektes gegeben

8.1 Soll-Ist Zeitvergleich

In der Tabelle werden die Soll- als auch die tatsächlichen benötigten Ist-Stunden aufgelistet, welcher zur Füllung des Projektauftrags und der Erstellung der Dokumentation benötigt wurden.

8.2 Rückblick

Obwohl alle geplanten Anforderungen realisiert werden konnten, können noch andere Erweiterungsvorschläge in der Zukunft umgesetzt werden. Zum Beispiel der Authorization-Check und die Zuständigkeit auf die zugeordneten Tankstellen der User. Diese können durch die Umgebungsvariable im Backend in einer Erweiterung geschafft werden.

9 Fazit

Das Projekt hat dem Autor viele Möglichkeiten angeboten sich technisch zu entwickeln und die Fähigkeiten im Bereich Projektplanung zu verbessern. Die Koordination der Arbeit mit vielen Bereichen und Abteilungen war eine lehrreiche Herausforderung. Das Projekt ist zwar am Ende, aber die Entwicklung auf Basis der Ergebnisse dessen wird in der nächsten Version des Produkts weitergehen. Das zeigt auch die Wichtigkeit der Verbindung zwischen verschiedenen Projekten und der langfristigen Planung von neuen Anwendungen und Features.

Anhang

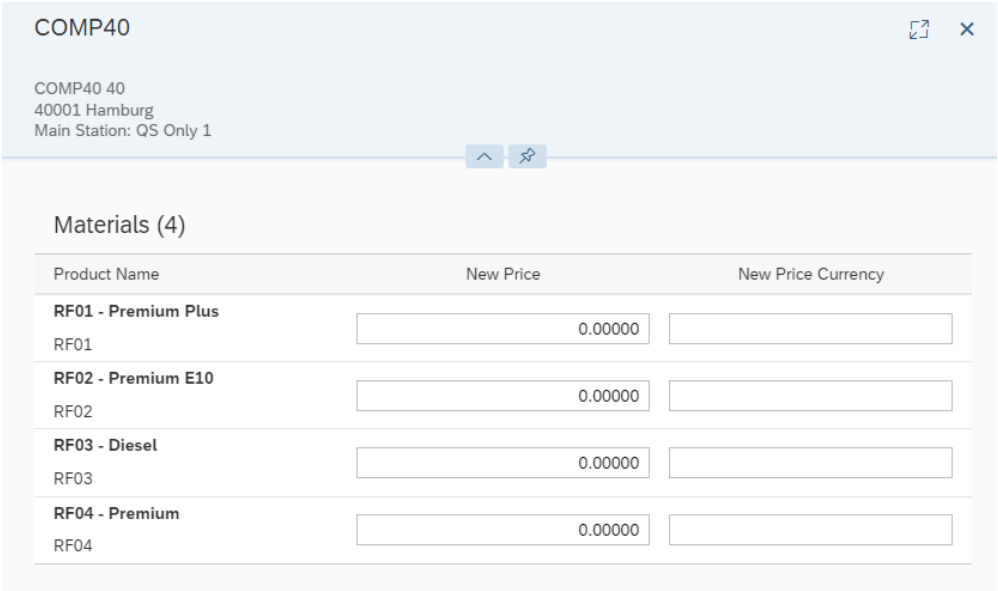
A1 Detaillierte Zeitplanung

Phase	Soll	Ist
Analyse	**5**	**5**
Analyse, welche Datenbanktabellen existieren	2	1
Die Beziehungen zwischen den wichtigen Tabellen analysiert	1.5	0.5
Die alte Benutzeroberfläche getestet	0.5	0.5
ABAP-Logik analysiert	1	2
Design der neuen Datenbaktabellen	1	1
Mockup	6	4
Richtige UI-Elemente gesucht	2	1
Erstes Mockup erstellt	2	1
Gespräch mit Consulting	1	1
Änderung auf das Mockup nach dem Gespräch	1	1
Implementierung	39	43
Backend: Datenbanktabellen erstellt	2	1
Backend: Basic/Consumption CDS Views erstellt	9	11
Backend: OData-Service erstellt und regiestriert	3	2
Backend: ABAP-Logik zur Speicherung der Daten implementiert	4	4
Backend: Testdaten erstellt	3	1
Backend: Annotation in den CDS-Views implementiert	2	4
Frontend: Git- und Web IDE Projekt erstellt	2	2
Frontend: Verknüpfung zum OData-Service erstellt	2	1
Frontend: UI-Elemente in Hauptseite platziert und verknüpft	3	5
Frontend: Routing implementiert	3	2
Frontend: Objektseite erstellt	0.5	0.5

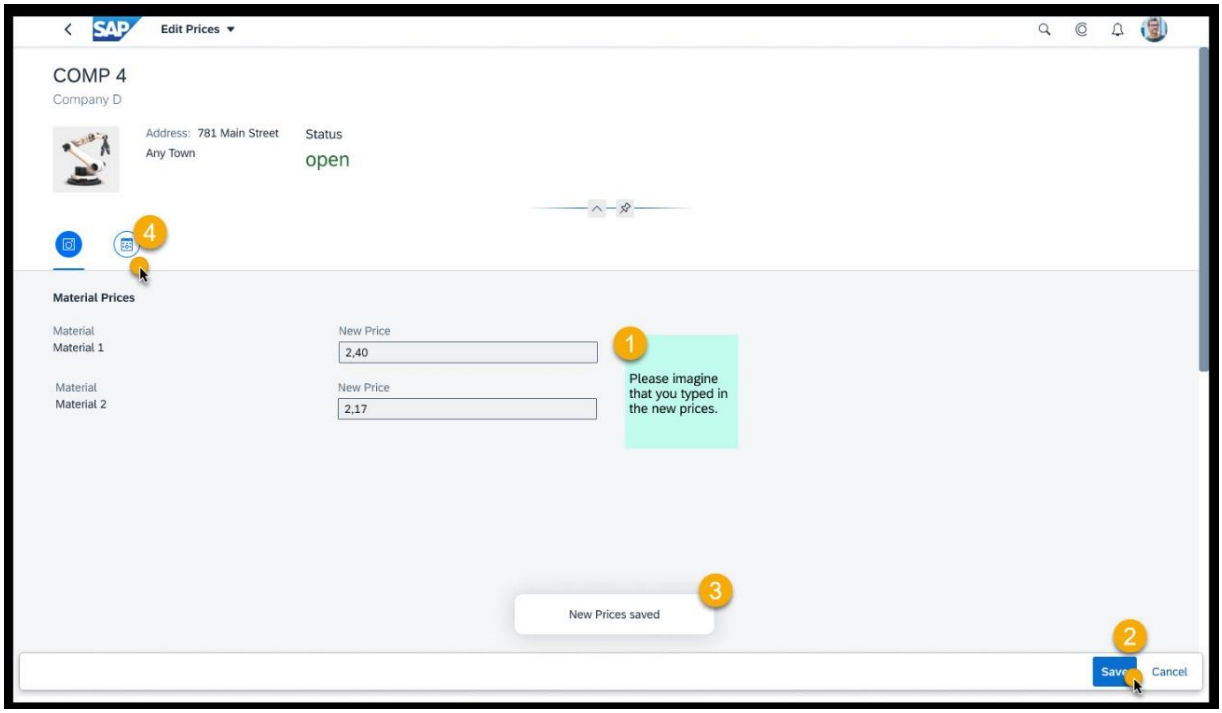
Frontend: SmartTable in Objektseite platziert und verknüpft	2.5	2
Frontend: SmartChart in Objektseite platziert und verknüpft	2.5	2.5
Frontend: Verbesserung der UI	0.5	1
Test	7	4
Entwicklung und Implementierung von Tests	4	2
Test durchgeführt	2	1
Fehleranalyse und -behebung	1	1
Abschlussphase	13	14
Projektdokumentation erstellt	11	13
Übergabe und Einweisung	2	1

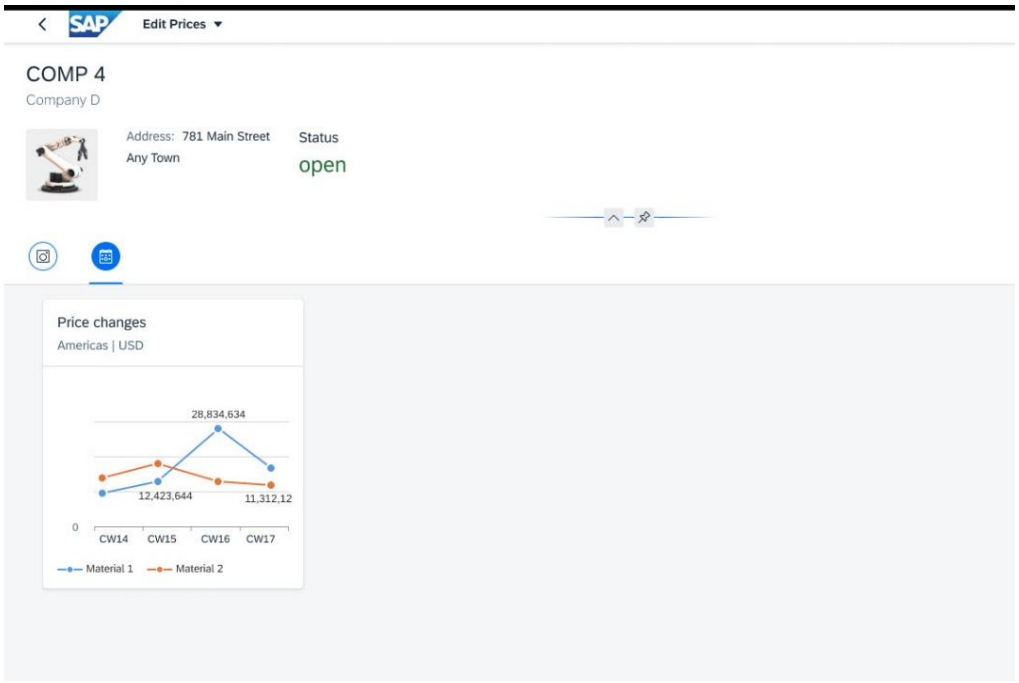
Tabelle 4: Detaillierte Zeitplanung mit Ist-Soll Vergleich zur Realisierung des Projekts

A2 Bildschirmaufnahme von der alten Benutzeroberfläche

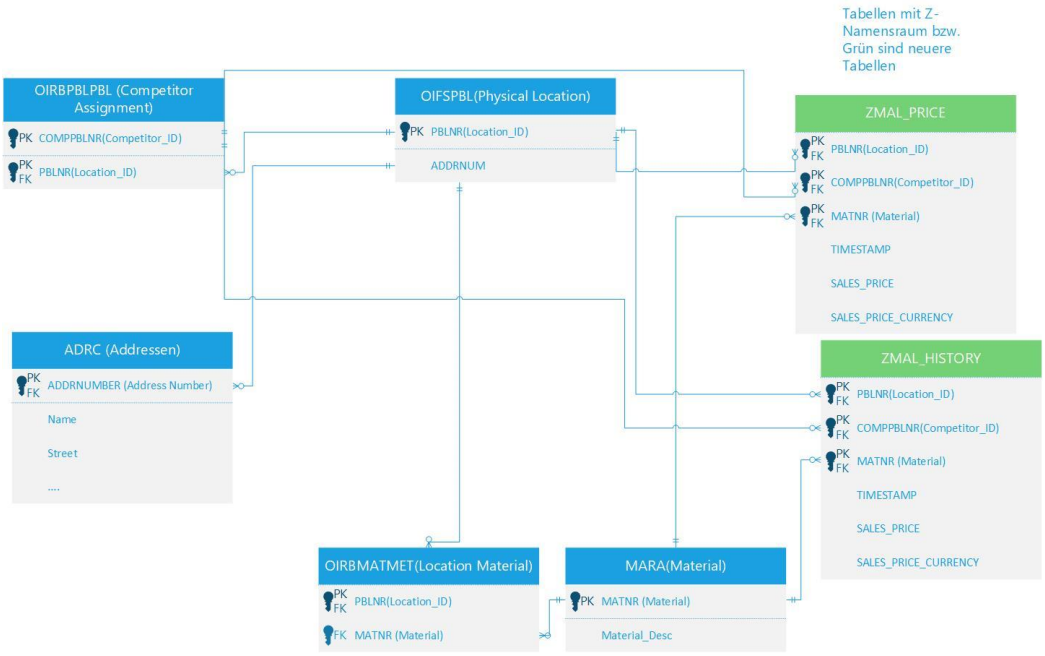


A3 Bildschirmaufnahmen vom UI Prototyp

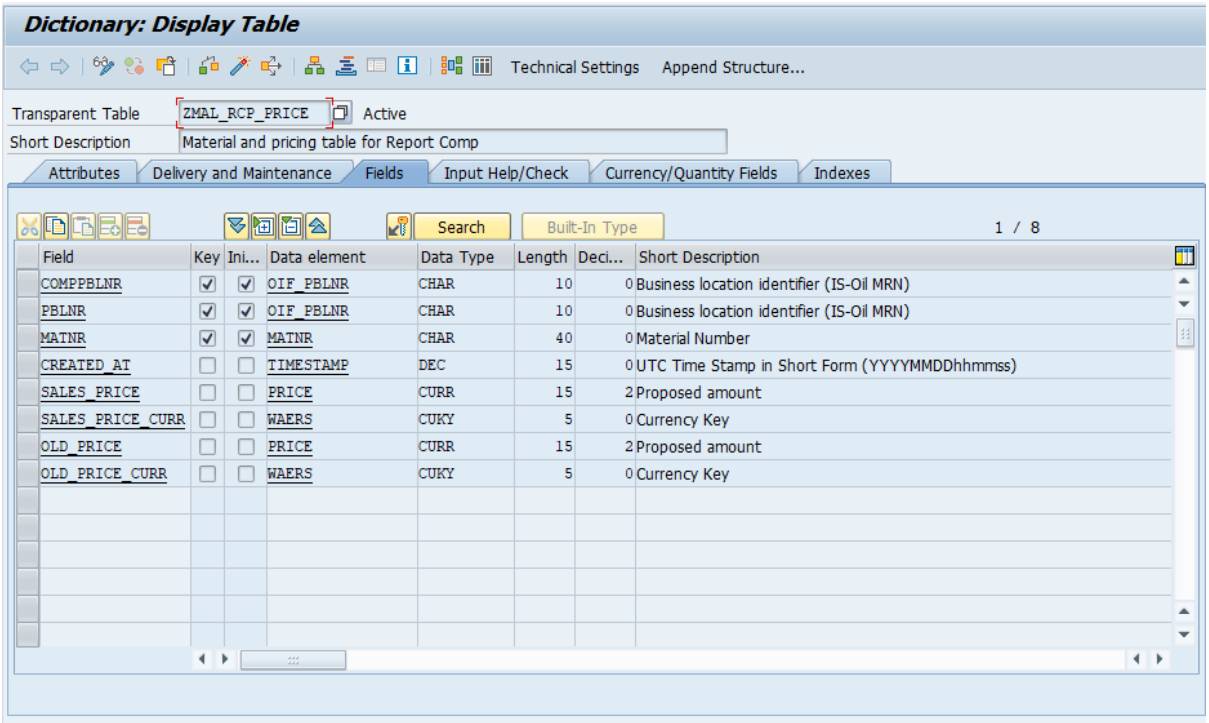




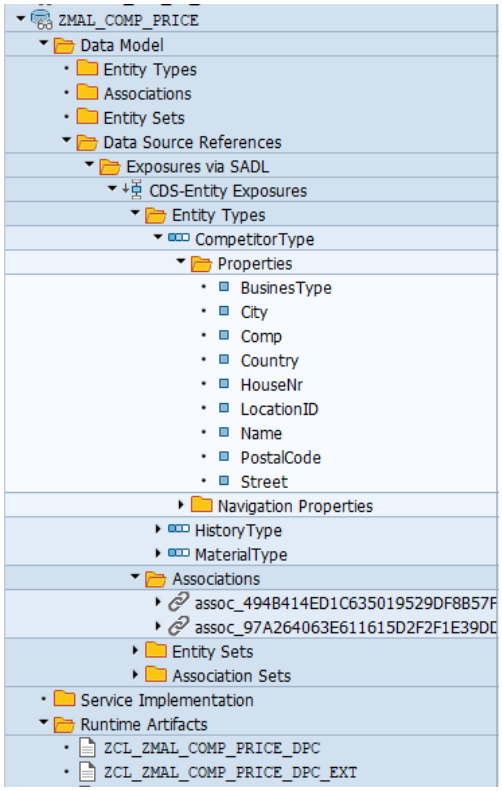
A4 UML Datenbanktabellen



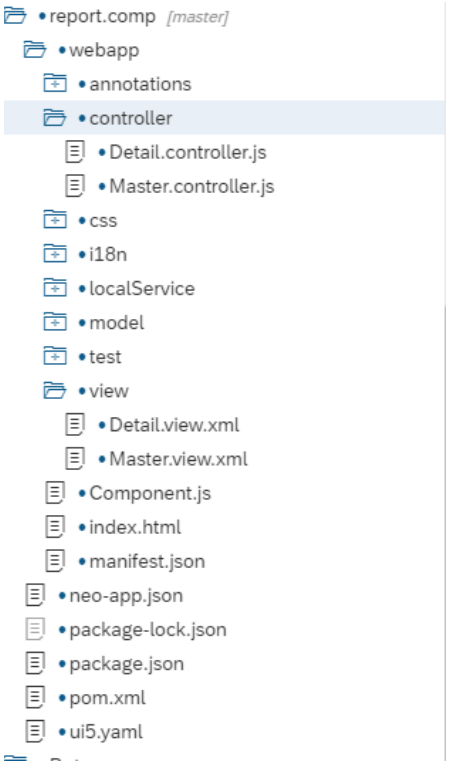
A5 ABAP Dictionary Oberfläche bei Erstellung einer SAP-Datenbanktabelle



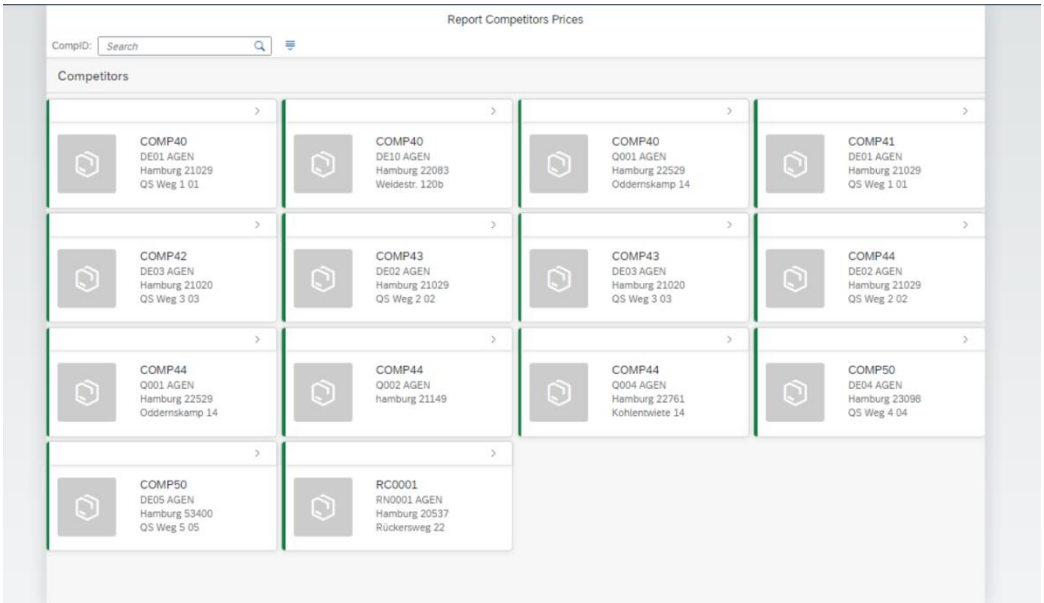
A6 OData-Projekt

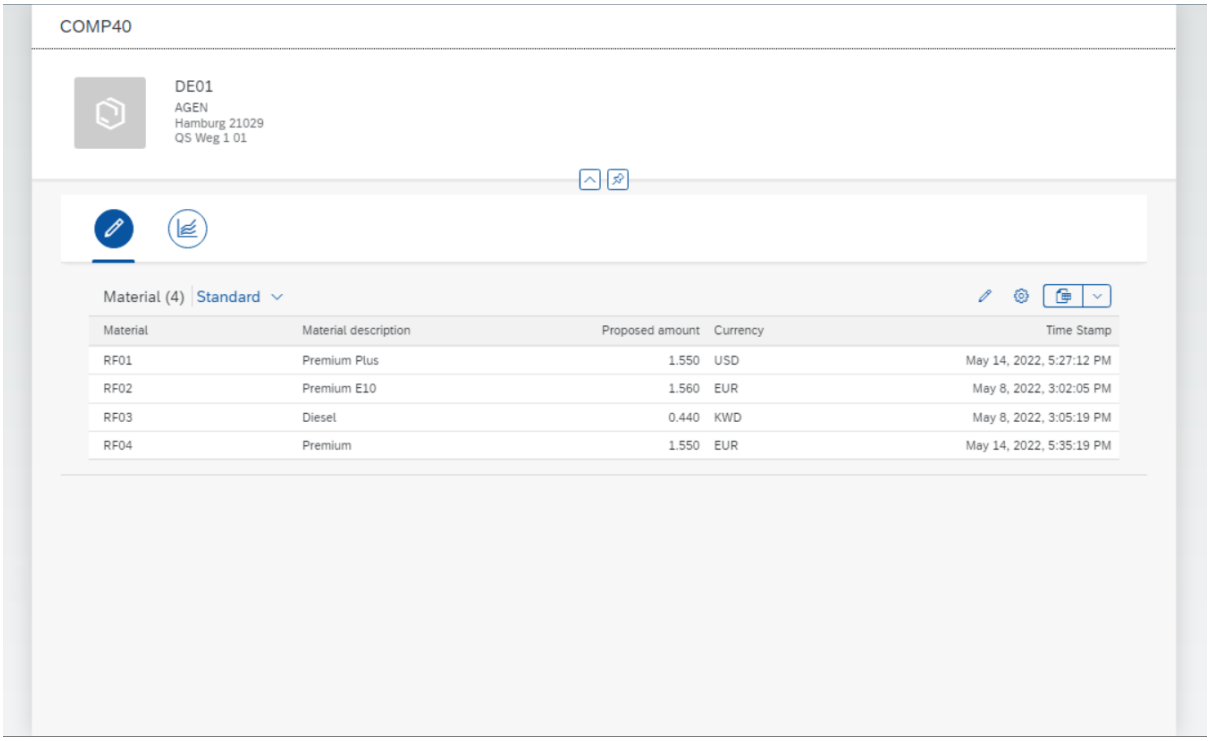


A7 Frontend-Projekt in der Web IDE

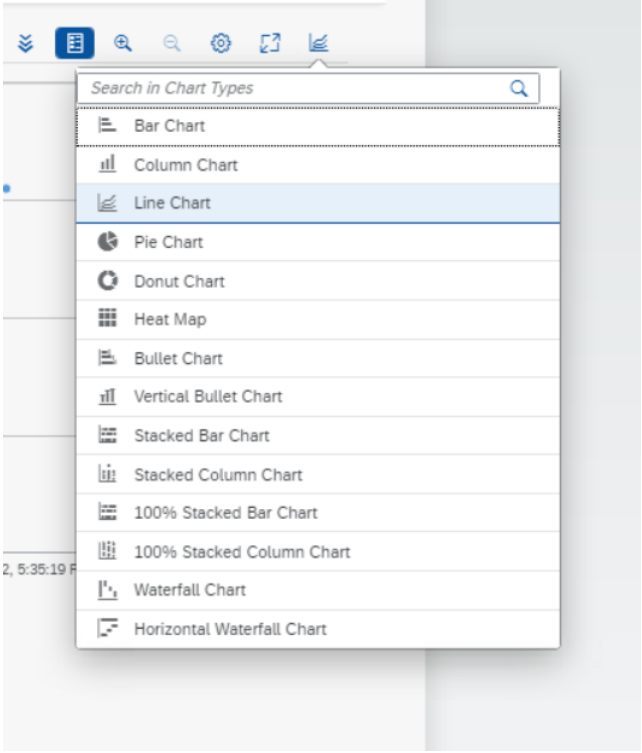


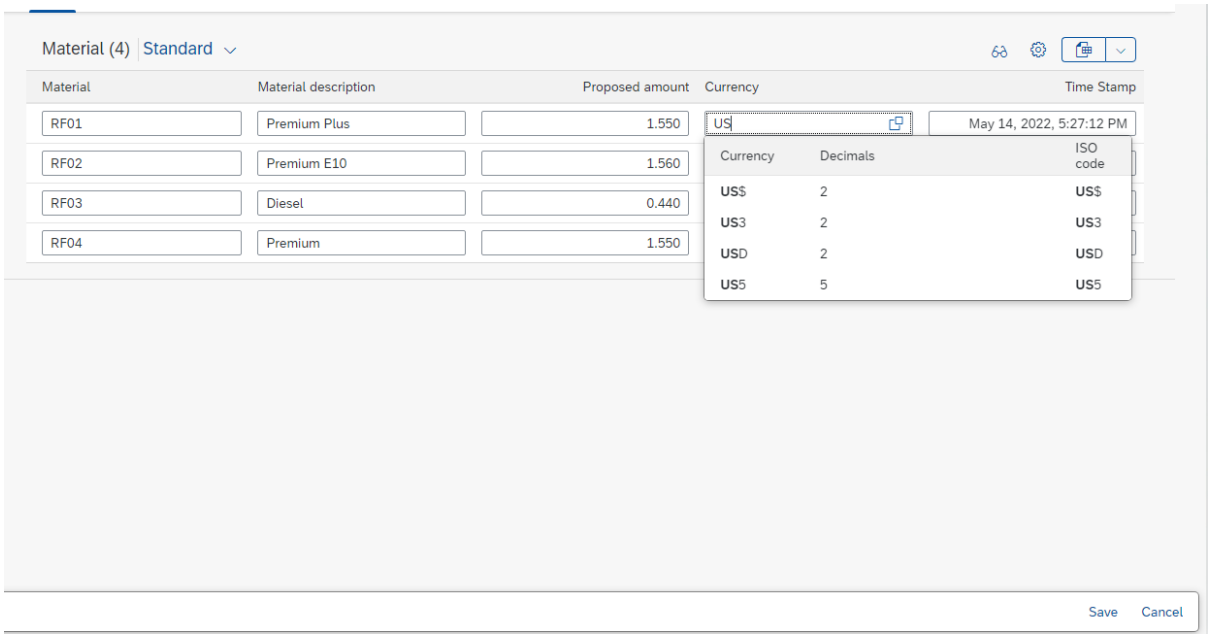
A8 Bildschirmaufnahmen von der neuen Benutzeroberfläche





A9 Besonderheiten der Benutzeroberfläche





A10 Basic CDS-View von Material

```
@AbapCatalog.sqlViewName: 'ZMALQ_MAT'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Materials'
define view zmal_mat as select from oirbmatmet as blm
        inner join oifspbl as oif on blm.pblnr = oif.pblnr
        inner join oirbpblpbl as oir on oif.pblnr = oir.pblnr
        inner join makt as makt on blm.matnr = makt.matnr
                                and makt.spras = $session.system_language
         left outer join zmal_rcp_price as price on blm.matnr = price.matnr
                                        and oir.comppblnr = price.comppblnr
                                            and oif.pblnr = price.pblnr {
  key oir.comppblnr as Comp,
  key oif.pblnr as LocationID,
  key blm.matnr as MaterialID,
  makt.maktx as Descrip,
  price.sales_price as SalesPrice,
  price.sales_price_curr as Currency,
  price.created_at as CreatedAt,
  price.old_price as OldSalesPrice,
  price.old_price_curr as OldCurrency }
```

A11 Consumption CDS-View von Material

```
@AbapCatalog.sqlViewName: 'ZMALQ_CMAT'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@VDM.viewType: #CONSUMPTION
@OData.entitySet.name: 'Material'
@EndUserText.label: 'Consumption View for Materials'
define view zmal_cmat as select from zmal_mat
  association [1..1] to zmal_ccomps as _ccomps on  $projection.Comp = _ccomps.Comp
        and $projection.LocationID = _ccomps.LocationID
        {
  @ObjectModel.foreignKey.association: '_ccomps'
  key Comp,
  @ObjectModel.foreignKey.association: '_ccomps'
  key LocationID,
@UI.lineItem: [{ importance: #HIGH, value: 'MaterialID', type: #STANDARD,
position: 1 }]
@ObjectModel.readOnly: true
  key MaterialID,
 @UI.lineItem: [{ importance: #HIGH, value: 'Descrip', type: #STANDARD, position:
2 }]
@ObjectModel.readOnly: true
Descrip,
@UI.lineItem: [{ importance: #HIGH, value: 'SalesPrice', type: #STANDARD,
position: 3 }]
  SalesPrice,
@UI.lineItem: [{ importance: #HIGH, value: 'Currency', type: #STANDARD, position:
4 }]
@Consumption.valueHelpDefinition: [{entity:{name: 'ZMAL_CURRENCY', element:
'Currency' }}]
  Currency,
@UI.lineItem: [{ importance: #HIGH, value: 'CreatedAt', type: #STANDARD, position:
5 }]
  CreatedAt,


  _ccomps
}
```

A12 Consumption CDS-View von Smart Chart mit Annotation

```
@AbapCatalog.sqlViewName: 'ZMALQ_CHISTORY'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@VDM.viewType: #CONSUMPTION
@OData.entitySet.name: 'History'
@EndUserText.label: 'Consumption View for pricing history'

@UI.chart:
[{ qualifier: 'History', chartType: #LINE,
dimensions: [ 'CreatedAt' ], measures: [ 'SalesPrice' ],
dimensionAttributes: [{ dimension: 'SalesPrice', role: #SERIES}],
```

```
measureAttributes: [{ measure: 'CreatedAt',role: #AXIS_1}]}]

define view zmal_chistory as select from zmal_history
association [1..1] to zmal_ccomps as _ccomps on  $projection.Comp = _ccomps.Comp
        and $projection.LocationID = _ccomps.LocationID
{
@ObjectModel.foreignKey.association: '_ccomps'
  key Comp,
@ObjectModel.foreignKey.association: '_ccomps'
  key LocationID,
  key MaterialID,
  key CreatedAt,
  Descrip,
  SalesPrice,
  SalesPriceCurr,

  _ccomps
} where CreatedAt <> 0
```

A13 Klassendefinition von OData-Standardklasse in ABAP

```
class ZCL_ZMAL_COMP_PRICE_DPC_EXT definition

protected section.
data: m_price type ZCL_ZMAL_COMP_PRICE_MPC=>TS_MATERIALTYPE,
mv_time type timestamp.

  methods MATERIAL_UPDATE_ENTITY
    redefinition .
private section.
    methods: INSERT_KEYS,
    insert_into_history.
endclass.
```

A14 Klassenimplementierung von Methode MATERIAL_UPDATE_ENTITY

```
METHOD MATERIAL_UPDATE_ETNTITY.

DATA: LS_PRICEINFO LIKE ER_ENTITY,
        ls_price type zmal_rcp_price.

    IO_DATA_PROVIDER->READ_ENTRY_DATA( IMPORTING ES_DATA = LS_PRICEINFO ).

m_price = ls_priceinfo.
me->insert_keys( ).
me->insert_into_history(  ).

get time stamp field mv_time.
update zmal_rcp_Price set created_at = @mv_time,
 sales_price = @ls_priceinfo-salesprice,
 sales_price_curr = @ls_priceinfo-currency
where comppblnr = @ls_priceinfo-comp
```

```
and matnr = @ls_priceinfo-materialid
and pblnr = @ls_priceinfo-locationid.
ENDMETHOD.


METHOD INSERT_KEYS.
 select single * from zmal_rcp_price
 where comppblnr = @m_price-comp and
 matnr = @m_price-materialid and
 pblnr = @m_price-locationid INTO @Data(ls_data).

if not sy-subrc = 0.
ls_data-comppblnr = m_price-comp.
ls_data-matnr = m_price-materialid.
ls_data-pblnr = m_price-locationid.
insert zmal_rcp_price from ls_data.
endif.
ENDMETHOD.
```

A15 Detailseite in XML

```
<mvc:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m" controllerName="ico.report.comp.controller.Detail"
    xmlns:f="sap.f" xmlns:layout="sap.ui.layout" xmlns:semantic="sap.f.semantic" xmlns:smartTable="sap.ui.comp.smarttable"
    xmlns:smartchart="sap.ui.comp.smartchart" xmlns:app="http://schemas.sap.com/sapui5/extension/sap.ui.core.CustomData/1">
    <semantic:SemanticPage id="semanticPage" title="Title">
        <semantic:titleHeading>
            <Title text="{Comp}" level="H2"/>
        </semantic:titleHeading>
        <semantic:headerContent>
            <HBox>
                <VBox class="sapUiSmallMargin" width="auto">
                    <Avatar displaySize="L" backgroundColor="Placeholder" displayShape="Square"/>
                </VBox>
                <VBox class="sapUiSmallMargin">
                    <Title id="addressTitle" titleStyle="H4" text="{LocationID}"/>
                    <Text id="cityDetails" text="{BusinesType}"/>
                    <ObjectAttribute text="{City} {PostalCode}"/>
                    <ObjectAttribute text="{Street} {HouseNr}"/>
                </VBox>
            </HBox>
        </semantic:headerContent>
        <semantic:content>
            <IconTabBar id="iconTabBar" class="sapUiResponsiveContentPadding">
                <items>
                    <IconTabFilter icon="sap-icon://edit" key="info">
                        <content>
                            <smartTable:SmartTable id="SmartTable" smartFilterId="smartFilterBar" entitySet="Material" tableType="ResponsiveTable"
                                tableBindingPath="to_cmat" useExportToExcel="true" useVariantManagement="true" useTablePersonalisation="true" header="Material"
                                showRowCount="true" persistencyKey="SmartTableAnalytical_Explored" enableAutoBinding="true" demandPopin="true"
                                class="sapUiResponsiveContentPadding" editTogglable="true" editToggled=".showFooter" fieldChange=".onFieldChange" app:useSmartToggle="true"/>
                        </content>
                    </IconTabFilter>
                    <IconTabFilter icon="sap-icon://line-chart" key="attachments">
                        <content>
                            <smartchart:SmartChart id="SmartChart" enableAutoBinding="true" entitySet="History" chartBindingPath="to_chistory"
                                useVariantManagement="true" persistencyKey="PKeyChartExample9" useChartPersonalisation="true" header="" showFullScreenButton="true"
                                selectionMode="Multi" showChartTooltip="true" showDrillBreadcrumbs="false" showDetailsButton="false" showDrillButtons="true"
                                app:chartQualifier="History"/>
                        </content>
                    </IconTabFilter>
                </items>
            </IconTabBar>
        </semantic:content>
        <semantic:footerCustomActions>
            <Button id="saveButton" text="Save" enabled="false" press="onSave"/>
            <Button id="cancelButton" text="Cancel" press="onCancel"/>
        </semantic:footerCustomActions>
    </semantic:SemanticPage>
</mvc:View>
```

A16 Quellenverzeichnis

- https://developers4sap.blog/odata-services-mit-cds-views-erstellen-teil-1/
  abgerufen am 13.05.2022
- https://sapui5.hana.ondemand.com/#/api
  abgerufen am 14.04.2022