Abbildung einer komplexen Transaktion anhand der Richtlinien von SAP Fiori

Prototypische Abbildung einer individuellen Kundenanfrage

Marcel Möller
Hochschule Fulda
Leipziger Str 123
36037 Fulda
marcel.moeller@cs.hs-fulda.de

Prof. Dr. Norbert Ketterer Hochschule
Fulda
Leipziger Str 123
36037 Fulda
norbert.ketterer@cs.hs-fulda.de

ABSTRACT
In dieser Arbeit wird exemplarisch an Hand des Vorgehensmodells von SAP Fiori die Erstellung einer komplexen Transaktion zur Implementierung einer individuellen Kundenanfrage als Fiori-App dargestellt. Der Fokus der Darstellung liegt dabei an der direkten Gegenüberstellung der Entwicklungsschritte mit dem Vorgehensmodell für die Entwicklung einer Fiori-App. Diese Arbeit basiert auf einer Bachelorarbeit, die unter Betreuung der Hochschule Fulda und der STI Group durchgeführt wurde.

Keywords
SAP-ECC, SAP Fiori, SAP UI5, ODATA, Betriebliche Anwendungen

EINLEITUNG
Nach Englbrecht und Wegelin[1](S.26) betrachten die meisten Entwickler von Benutzeroberflächen die Anwendung oft aus technischer Sicht, sodass Oberflächen mit Funktionen überladen werden, die dann auch dem Endanwender angeboten werden, obwohl dieser die Funktionen oftmals gar nicht alle benötigt. Als Konsequenz werden dann oft Inhalte und Funktionen deaktiviert oder es tauchen gar Fehlermeldungen für fehlende Berechtigungen auf, die dieser Anwender weder besitzt noch benötigt. Die Verlagerung von Anwendungen in die browser-basierten Fioris kann einen Schritt in Richtung Verbesserung der Mensch-Maschine-Kommunikation darstellen, denn es entsteht eine neue Art, wie Anwendungen mit dem Anwender kommunizieren, um dessen Arbeit zu unterstützen.

Weitere Beweggründe für SAP Fiori sind nach Galer [2] zum einen die Attraktivität eines schnellen Zugriffs auf täglich wichtige Information, wodurch zum Beispiel ein Anwender auch außerhalb des Büros auf für ihn relevante Daten zugreifen kann. Zum Anderen sind Wartezeiten auf Genehmigungen somit verkürzbar, weil auch Vorgesetzte von unterwegs sofortigen Zugriff auf die nötigen Daten besitzen und Transaktionen über Fioris durchführen kann.

In dieser Arbeit wird das Vorgehensmodell vorgestellt, gegen die Infrastruktur eines Unternehmens abgeglichen und dann die Entwicklung einer komplexe Transaktion für das Unternehmen in einer UI5-/ Fiori-Plattform skizziert. Dies beinhaltet auch die Nutzung des MVC-Patterns, der spezifischen UI5-Libraries und die Frage der Mock-Daten.

PROBLEMDEFINITION
Es soll eine Applikation im Fiori-Umfeld erstellt werden, wobei eine komplexe Transaktion des Unternehmens abzulösen ist. Hierbei ist insbesondere auch eine Entscheidung über den technischen Transaktionstyp (wie Fiori-Master-Detail, UI5) zu treffen. Das Problem der Abbildung der Transaktion kann in die folgenden Teilprobleme gegliedert werden:

1. **Integration in die aktuelle Unternehmensinfrastruktur:** Zunächst müssen alle vorhandenen Architekturkomponenten aufgenommen und dann geprüft werden, ob eine Implementierung von SAP Fiori überhaupt möglich ist. Dies ist zwar ein notwendiger Schritt, der in dieser Veröffentlichung aber nicht weiter detailliert werden soll.
2. **Vorgehensmodell ableiten:** Ein wesentliches Problem dieser Arbeit stellt die Ableitung eines Vorgehensmodells mit anschließender Verifikation in Form eines Prototypens dar.
3. **Abbildung der aktuellen Kundenanfrage:** Zur Verifikation des Vorgehensmodells soll die Kundenanfrage implementiert werden. Es muss kritisch betrachtet werden, ob die entwickelte Vorgehensweise eine Abbildung der speziellen Kundenanfrage in aktueller Form als Fiori-Applikation überhaupt möglich ist. Die Einhaltung der Grundsätze und Richtlinien von SAP Fiori und SAP UI5 sind ein zentraler Punkt, aber es sind auch einige technische Fragen abzudecken, beispielsweise werden variable Felder benötigt, um das Produkt zu beschreiben.

ENTWICKLUNG VORGEHENSMODELL
Grundlegende Eigenschaften SAP Fiori
Das Designkonzept von SAP Fiori beruht auf einer Kombination von fünf Kernprinzipien, die eine überzeugende Benutzererfahrung garantieren sollen [3]. Dabei handelt es sich

um die Grundsätze *rollenbasiert, responsive Design, einfach, kohärent* und *ansprechend*.

Dieses Konzept resultiert dann in folgenden Prinzipien [1] (S. 29 f.):

- Herunterbrechen komplexer Transaktionen zu aufgabenorientierten Teileinheiten
- Es werden nur noch Funktionen angeboten, die auch wirklich für den Arbeitsschritt in der jeweiligen Rolle benötigt werden
- Einhaltung der Designrichtlinien sowohl auf mobilem Endgerät sowie auf traditionellem PC
- Bearbeitung genau eines Anwendungsfalls mit einer SAP-Fiori-Anwendung Anwendungsfall und das auf einem von drei möglichen Screens (PC, Tablet, Handy)
- Standardisierte Oberflächengestaltung, die durch definierte konzeptionelle Applikationstypen und Interaktionsmuster sichergestellt wird
- Layoutdefinition durch den konzeptionellen Applikationstyp
- Interaktionsmuster definieren Aufbau und das Layout eines Bildschirms, welcher die Daten erfasst
- Anwendungsprozess muss klar durch die Anwendung abgebildet werden
- Anwendung soll intuitiv bedienbar sein

Der strukturelle Aufbau ist somit prinzipiell vordefiniert - dies drückt sich in drei unterschiedlichen Applikationstypen aus [1] (S. 36 f.):

- **transaktionale Applikationen:** taskorientierte Arbeitsabläufe, die einzelne Arbeitsschritte in wenigen Screens abbilden und letztlich Transaktionen im Backend ausführen
- **Factsheets (Infoblätter):** diese beinhalten Daten und Fakten zu spezifischen Geschäftsvorgängen und erlauben eine Navigation zu diesen Geschäftsvorgängen, wodurch immer detailreichere Information erlangt werden können; diese erfordern eine HANA-Datenbank im Backend
- **analytische Applikationen:** diese Applikationen liefern dem Nutzer Information in Echtzeit und werden über virtuelle Datenmodelle (VDM's) auf dem SAP HANA XS und dem KPI Modeler auf dem ABAP-Frontend-Server konfiguriert, erfordern also eine komplette HANA-Umgebung

Zur Entwicklung einer Fiori-App existieren drei verschiedene Umsetzungsstrategien: **ADOPT**, **ADAPT** und **DEVELOP**, die sich bezüglich des Grades der Anpassung unterscheiden: Nutzung bestehender App, Nutzung eines Extension-Points einer bestehenden App oder Neuprogrammierung einer App ([1](S. 78 f.)). Eine Fiori-Applikation kann dabei direkt bei der Erstellung als Fiori-Applikation ausgewiesen werden und folgt dann einer eher fixen Struktur (Master-Detail oder Worklist) oder kann frei als SAP UI5-Applikation entwickelt werden. Im ersten Fall muss die Verbindung zur Datenquelle (inklusive Datenbindung) bei der Erstellung der Applikation a-priori vorgegeben werden, im zweiten Fall ist dies nicht notwendig, die Applikation wird frei nach MVC-Pattern entwickelt. Das entsprechende Template ist Anfangs auszuwählen (Abbildung 4).

Design Guidelines

Um eine durchgängige Geräteunabhängigkeit zu gewährleisten, sollten nur UI-Controls verwendet werden, die einen Responsive Charakter haben ([1](S. 87 f.)). Es wird das Grid-Layout verwendet, die Maßeinheit ist künstlich zu *rem* definiert, um eine Pixel-Unabhängigkeit zu erhalten. Eine Schaltfläche ist dann 3x3 *rem* groß. Schriftarten sind nichtserifal; es werden verschiedene Schriftarten verwendet, um mehrere Browser und Plattformen unterstützen zu können. Icons werden als Zeichen eingefügt - es existiert ein Icon-Explorer (https://sapui5. hana.ondemand.com /iconExplorer.html).

Über *Floorplans* wird der visuelle Aufbau der Fiori-Anwendung gesteuert, insbesondere wie die Anwendung aufgebaut, welche UI-Controls verwendet werden und wie die Navigation gestaltet werden soll (Abbildung 1).

Das Split-Screen-Layout kommt dabei bei Anwendungen zum Einsatz, bei denen zur Navigation Listen verwendet werden. Verwendet wird dabei in aller Regel das Master-List-Designparadigma, bei dem links eine Auswahlliste steht und rechts das ausgewählte Element detailliert wird. In der Dokumentation von SAP zu den Fiori Design Guidelines [4] wird eine Nichtempfehlung für dieses Layout für die folgenden Fälle ausgesprochen: Das Split-Screen-Layout sei nicht zu empfehlen, wenn man verschiedene Attribute eines Objektes aus der Liste sehen will und diese mit Attributen weiterer Objekte vergleichen möchte. Außerdem sollte bei einem Betrachten eines einzigen Objekts aus verschiedenen Sichten auf ein anderes Layout zurückgegriffen werden.

Das Full-Screen-Layout hingegen wird für die Anzeige einer größeren Menge an Information empfohlen, es bietet dem Entwickler durch den komplett freien Bildschirm eine gewisse Flexibilität, da UI-Aspekte relativ frei positioniert werden können. Zu beachten sei jedoch, dass dieses Layout nicht per se responsive sei, weshalb der Entwickler dafür sorgen müsse, dass je nach Endgerät alles korrekt dargestellt wird [1](S. 98 ff.).

Die Dokumentation der SAP beschreibt ergänzend, dass man das Layout nicht verwenden soll, wenn man nur wenig Information darstellen möchte oder wenn man nicht weiß, wie man die darzustellende Information strukturieren soll [5]. Im zweiten Schritt werden durch die speziellen Floorplans der SAP die Layouts mit Inhalt gefüllt, ein Beispiel typischer Fortführungen zeigt Abbildung 2. Der Floorplan wäre bei Auswahl einer Fiori-Applikation bei der „Template Selection“ fix definiert, bei einer SAP UI5-App ist er zu spezifizieren oder wird durch die gewählte API bereits definiert. Die direkte Definition erfolgt über „sap.ui.layout.form.Form“ oder „sap.ui.layout.form.SimpleForm“.

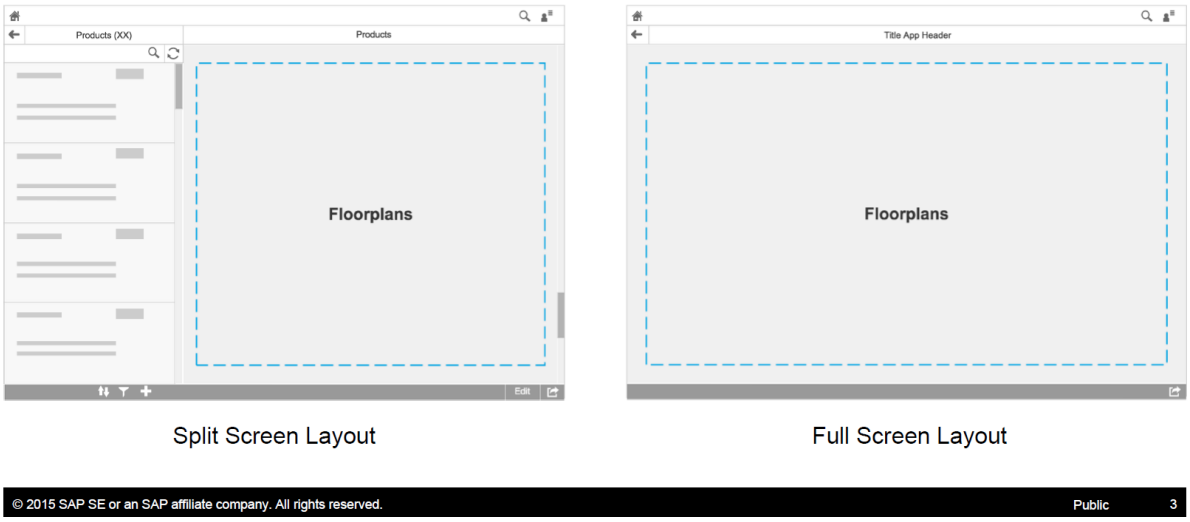

Abbildung 1: Seitenlayouts über Floorplans [6]

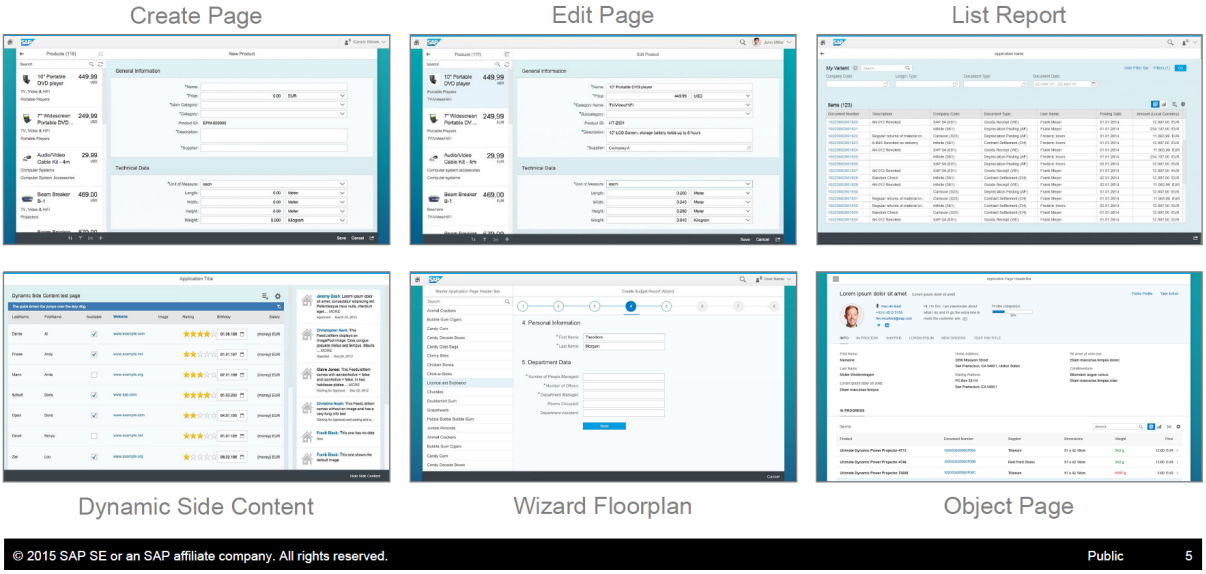

Abbildung 2: Beispiele von Floorplänen [6]

Applikationsentwicklung

Applikationsstruktur

Die Applikationsentwicklung erfolgt via MVC-Pattern, die Nutzung dieses Patterns zeigt Abbildung 3, wobei die Komponenten prinzipiell dem üblichen MVC-Konzept entsprechen. Nach dem SAPUI SDK - Demokit [7] stehen Views und Controller aber oftmals in einer 1:1 - Beziehung zueinander. Dennoch sei die Bildung sog. „Applikationscontroller“ möglich, wenn Controller keine UI besitzen. Ebenfalls können Views ohne eigenen Controller erstellt werden, wie dies auch hier in der Entwicklung der eigenen Applikation erfolgen wird.

Wird eine neue Applikation angelegt, ist ein neues Projekt anzulegen. Dieses kann auf einem Template basieren oder auch auf einer Beispielapplikation. Wenn es sich nur um eine Erweiterung handelt, wird dies als Erweiterung spezifiziert - es muss hierfür in der Plattform der Service der „Web-IDE“ aufgerufen werden.

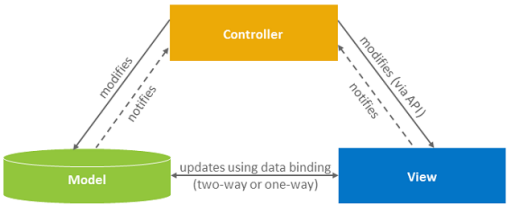

Abbildung 3: MVC Pattern in Fiori [7]

Soll ein Template verwendet werden, werden eine Reihe von Templates angeboten und es wird auch gezeigt, wie die Applikation später aussehen wird - außer es handelt sich um eine SAP UI5-Applikation. Ein Beispiel einer Auswahl des Templates und den prinzipiellen Preview für eine Fiori-Applikation zeigt Abbildung 4.

Danach wird die Datenverbindung angegeben, beispielsweise eine Service-URL oder auch nur eine Datei, je nachdem ob mit einem realen Backend oder nur mit Mock-Daten entwickelt werden soll. Nach Pflege von Applikationsparametern und insbesondere des „Data-Bindings“ wird die Codevorlage erstellt. Wäre statt eines Fiori nur ein SAP UI5-Template ausgewählt worden, wird nach Auswahl des View-Typs (XML, JSON, JavaScript, HTML) dann direkt die Codevorlage erstellt - empfohlen wird in [8] (S. 152 ff.) insbesondere der Viewtyp „XML“ - dieser Viewtyp erlaubt beispielsweise auch eine Nutzung des graphischen Layout Editors.

Soll statt einem Template eine Beispielapplikation verwendet werden, kann eine bestehende Applikation Basis für die Generierung der Codevorlage sein - dies bietet sich insbesondere an, wenn leichte Varianten einer Applikation erstellt werden sollen, da die Applikationen ja benutzerspezifischer zugeschnitten sein sollten. Hier wird dann direkt auf Basis der Vorlageapplikation die neue Codevorlage erstellt - siehe Abbildung 5.

Wird die Applikation ohne Vorlage erstellt, wird sofort die Codevorlage nach dem MVC-Pattern als Skelett erstellt und es kann auf dieser z.B. der Layout-Editor aufgerufen werden, um dann dort Elemente (wie Buttons) zu platzieren und der Rest des Skeletts ausgefüllt werden. In der Detailebene ist die Applikation dann wie in Abbildung 6 aufgebaut:

- **manifest.json:** App-Einstellungen und wichtige Information, die zum Starten einer Applikation notwendig sind (z.B. auch die zu instanziierenden Modelle sowie die Libraries) sowie zur Integration in das Launchpad.
- **root view (App.view.xml):** Link zu den Views
- **component.js:** Definition des Applikationssetups, die dort definierte *init*-Funktion wird automatisch beim Applikationsstart aufgerufen. Die Komponentenklasse (hier „UIComponents“) wird erweitert.

Datenbindung

Um Änderungen synchron zu halten, wird eine Datenbindung verwendet [8] (S. 210 ff.); man kann dabei zwischen einer One-Way-, Two-Way- und One-Time-Datenbindung unterscheiden. Die One-Way-Datenbindung propagiert Änderungen vom Modell zur angebundenen Benutzeroberfläche, die Two-Way-Datenbindung propagiert die Änderung in beiden Richtungen. Bei der One-Time-Datenbindung findet dagegen die Aktualisierung nur einmalig statt.

Die SAP UI5-Basis innerhalb der Fiori-Entwicklung unterstützt bei der Implementierung der Datenbindung vier Modelltypen, wobei die Vorgehensweise bei der Implementierung des Data Bindings immer gleich ist: Nachdem Daten definiert worden sind, muss eine passende Instanz des passenden Modells erzeugt werden. Zuletzt müssen die Modell-Parameter an das UI-Control gebunden werden. Dabei kann die Datendefinition lokal erfolgen oder aus einem SAP-System stammen. Die vier Modelle haben verschiedene Standardbindungstypen: Das JSON Modell und das XML Modell nutzen das Two-Way-Binding, während das Resource Modell und das OData Modell den One-Way-Bindungstyp aufweisen. Je nach Wahl des Modells wird ein anderes Format für die Datei benötigt, sodass bspw. beim JSON Modell das JSON-Format verwendet wird.

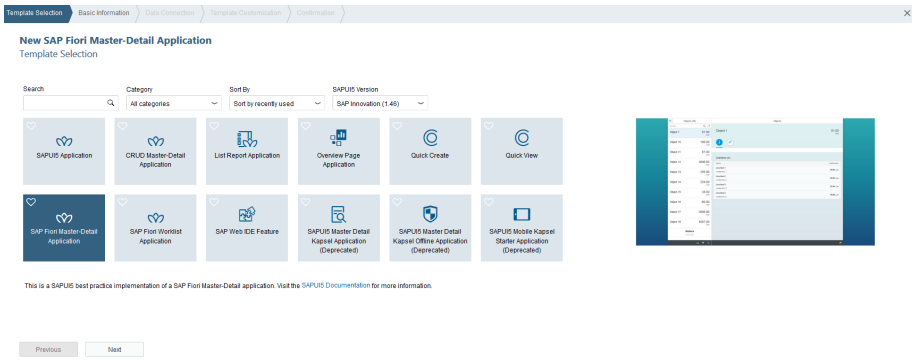

Abbildung 4: Auswahl Template in Fiori

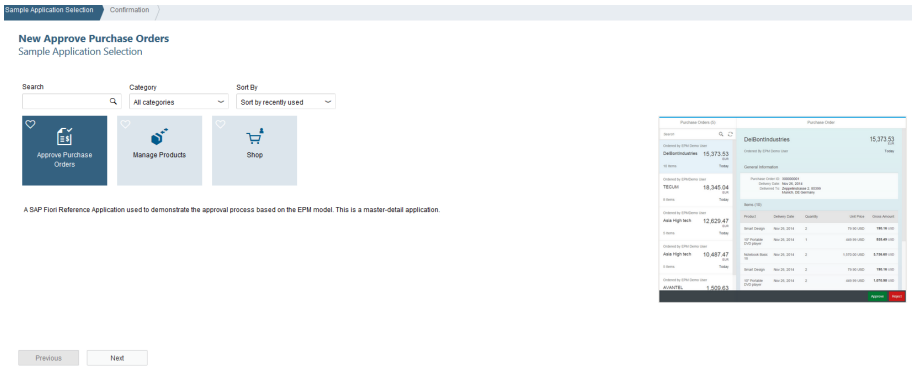

Abbildung 5: Erstellung einer Applikation auf Basis einer Vorlageapplikation

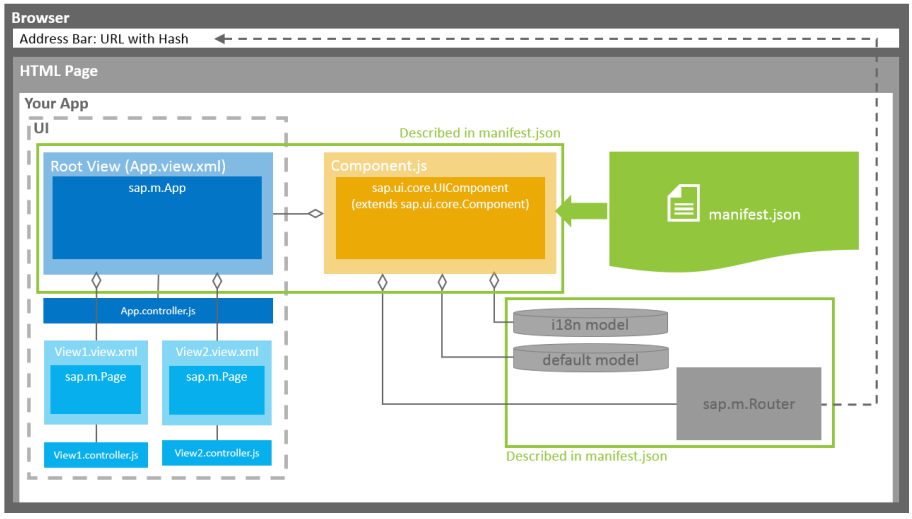

Abbildung 6: Struktur einer Applikation [9]

Bibliotheken

Das „UI5 Demo Kit - UI Development Toolkit for HTML5“ [10] informiert den Programmierer neben allgemeinen Dingen zur SAP UI5 - Entwicklung auch über die verschiedenen Bibliotheken, die während des Erstellens einer Applikation verwendet werden. Unter „API-References“ liefert insbesondere die Library „sap.m“ Responive Controls, wie sie zur Entwicklung auf touchbasierten Geräten und auch Desktop-Browsern benötigt werden. Für die APIs werden auch Beispiele gegeben sowie Codinggerüste angegeben, die diese APIs nutzen. Ein Beispiel der API „sap.m.List“ zeigt Abbildung 7, die Nutzung zu einer Gruppierung in einem Beispielcoding zeigt Abbildung 8.

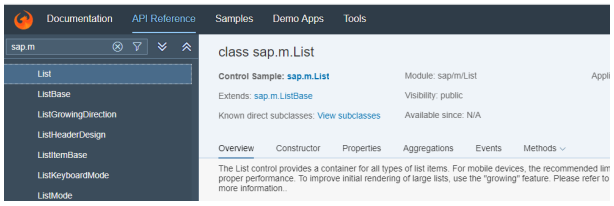

Abbildung 7: Beispiel des sap.m.list APIs

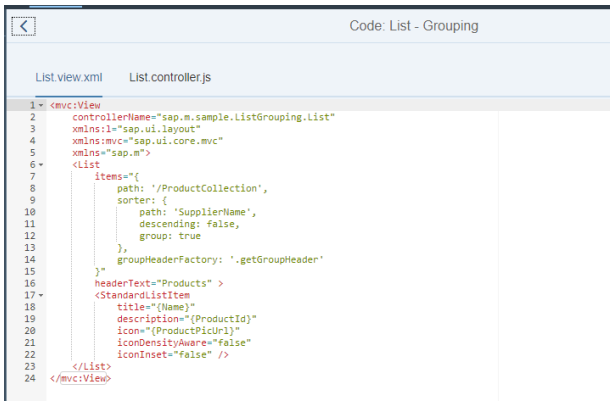

Abbildung 8: Beispiel des sap.m.list APIs

Layout-Editor

Neben einem Code-Editor stellt die SAP Web IDE über den Layout-Editor auch einen Editor zur Verfügung, in dem die View-Elemente für XML-basierte Views direkt graphisch auf dem View platziert (also ähnlich wie bereits bei Dynpros und Web-Dynpros) und für die verschiedenen Devices geprüft werden können. Das Coding des View wird dann generiert - Abbildung 9 zeigt die graphische Darstellung.

Open Data Protocol

Der direkte Zugriff auf die SAP-Backend-Services geschieht durch das SAP Gateway auf Basis des standardisierten OData-Protokolls. Das Open Data Protocol (kurz OData) ist ein auf HTTP basierender, offener Standard. Es ermöglicht die Erstellung von REST-basierten (Representational State Transfer) Datendiensten aufgrund der Standards des Atom Publishing und der Atom Syndication. Diese beiden Formate bieten die Möglichkeit, Webinhalte in XML zu bearbeiten und zu verbreiten [8] (S. 327/328). Zusätzlich existieren noch SAP-Annotationen, die mit dem OData-Protokoll mit Hilfe

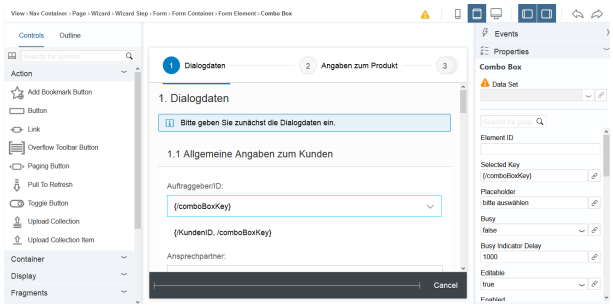

Abbildung 9: Beispiel des Layout-Editors

des SAP Gateway versendet werden können, woraus sich folgende Struktur eines OData-Services innerhalb von SAP ergibt [8] (S. 329) - insgesamt setzt sich ein OData-Service damit aus den folgenden Komponenten zusammen, die in Abbildung 10 dargestellt werden. Mit $metadata kann ein Servicedokument zu einem OData-Service aufgerufen werden, welches Information über den Aufbau des Services, dann welche Entitäten und Entitätstypen angeboten werden und welche Navigationsmöglichkeiten innerhalb des Services möglich sind.

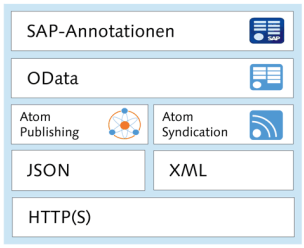

Abbildung 10: Aufbau eines OData-Protokolls innerhalb einer SAP-Umgebung

Verwendung von Mockdaten

Da die Entwicklungen in Backend und Frontend eher von zwei Personenkreisen durchgeführt werden, gibt es in SAP UI5 die Möglichkeit über den Mockserver Mockdaten zu erzeugen, auf dessen Datenbasis mit der Implementierung der Oberfläche begonnen werden kann; ein Frontend-Entwickler muss demnach nicht auf das fertige Backend warten. Es ist hierzu ein Mockdaten-Verzeichnis anzulegen, in welches die EDMX-Datei importiert werden kann. Danach können die Mockdaten editiert werden/ es können Zufallswerte gemäß der EDMX-Datei erzeugt werden, wobei allerdings die Referenzen zwischen Master und Detail-Daten nicht beachtet werden. Die IDE generiert dann daraus eine JSON-Datei mit den eingegebenen Beispieldaten. In der `OnInit`-Methode des Controllers muss der Mockserver instanziiert werden.

Verwendung des Fiori Launchpads

Nachdem eine Registrierung in der SAP Fiori Cloud Demo [11] durchgeführt wurde, wird dem Benutzer die SAP Fiori Homepage angezeigt. Dort können je nach Berechtigungen Applikationen aus einem Applikationskatalog hinzugefügt werden. Dabei können Gruppen angelegt werden, sodass dort die Applikationen hinzugefügt werden können. Es kann die Oberfläche personalisiert werden, die Benutzer-

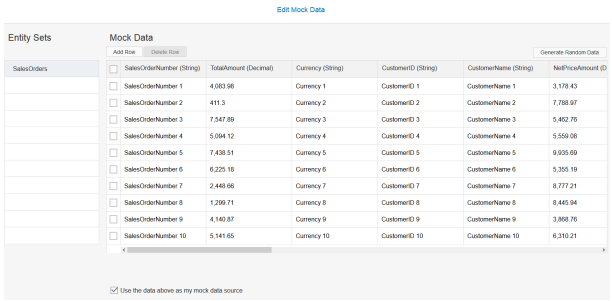

Abbildung 11: Mockdaten bearbeiten

angaben eingesehen und bearbeitet sowie das Fiori Configuration Cockpit über den Menüpunkt „Website verwalten“ aufgerufen werden. In diesem können sämtliche Optionen des SAP Fiori Launchpads (Abbildung 12) bearbeitet, verwaltet und überwacht werden.

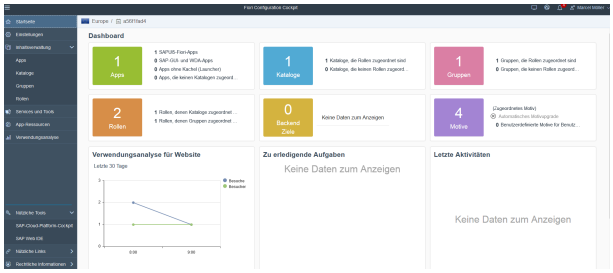

Abbildung 12: Fiori Launchpad

Neben dieser Startseite, die u.a. einen Überblick über aktive Applikationen und die Benutzerverwaltung gibt, können Services und Tools wie z.B. die SAP Web IDE oder das SAP-Cloud-Platform-Cockpit aufgerufen werden. Über die Inhaltsverwaltung können dort alle Benutzer, Gruppen und Rollen verwaltet und zugeordnet werden. Wählt der Nutzer unter Services das Tool SAP Web IDE, so kann dieser beginnen, eine SAP Fiori Applikation zu entwickeln. Ist eine Applikation für einen ersten Test fertiggestellt, kann diese dem SAP Fiori Launchpad hinzugefügt werden. Dafür wählt der Nutzer den root-Ordner der Applikation und navigiert mit einem Rechtsklick auf DEPLOY und dann auf DEPLOY TO SAP CLOUD PLATFORM. Eine Applikation muss immer zunächst in der SAP Cloud Platform registriert werden, damit diese dann in einem zweiten Schritt dem SAP Fiori Launchpad hinzugefügt werden kann. Nun müssen noch Angaben zum Account, Projekt und Version der Applikation gemacht werden um das Hinzufügen abzuschließen. Dabei füllt der Wizard schon die Felder standardmäßig aus.

ENTWICKLUNG DER TRANSAKTION

Die Fiori-Transaktion soll die aktuelle Kundenanfrage ablösen, die klassisch als Dynpro-Transaktion als digitale Aktenlösung realisiert ist.

Aktuelle Kundenanfrage

Die aktuelle Kundenanfrage soll hier nicht detailliert dargestellt werden, jedoch sind die wesentlichen Dinge, die die Fiori-Entwicklung betreffen, festzuhalten:

- Es werden auf einem Einstiegsdynpro Angaben zum Kunden sowie zur Art der Verpackung festgelegt, speziell auch, ob die Verpackung Lebensmittelkontakt besitzt
- Danach wird ein Word-Dokument erzeugt, in welchem bereits einige Daten vorbelegt sind und in dem weitere Daten ausgefüllt werden können
- Wichtig ist, dass sich die auszufüllenden Daten abhängig von der ausgewählten Verpackungsart ändern

Voraussetzungen

Für SAP Fiori Applikationen sind neuere Produktversionen am Backend nötig, die Details hängen dann vom Typ der Fiori-Applikation ab.

Transaktionale Apps

Transaktionale Applikationen benötigen keine besonderen Softwarevoraussetzungen, lediglich irgendeine betriebsfähige Datenbank wird benötigt sowie ein aktueller Netweaver Stack (entweder 7.4 oder 7.31, wenn Komponenten hinzuinstalliert werden).

Factsheets(Infoblätter)

Als Voraussetzungen werden in Bezug auf die Softwarekomponenten die gleichen Komponenten benötigt (SAP Netweaver 7.4, ...). Der einzige aber entscheidende Unterschied liegt in der Wahl der Datenbank: Es bedarf einer SAP-HANA-Datenbank, mindestens die Platform Edition 1.0 SPS 8. Es wird von SAP empfohlen, den Dispatcher als Reverse-Proxy zu verwenden, da sämtliche HTTP-Requests für die UI's und des Backends mit einer Internetadresse kommunizieren, sodass der Reverse-Proxy der einzige Einstiegspunkt für HTTP-Requests darstellt, sonst wäre die „Same-Origin“-Policy von Standardbrowsern verletzt.

Analytische Applikationen

Für die analytischen Applikationen wird neben des SAP-HANA-Systems noch die SAP HANA XS - Erweiterung benötigt. SAP HANA XS stellt nach [1] (S. 183) einen leichtgewichtigen Anwendungsserver dar, „der die Anwendungsdaten in Form von virtuellen Datenmodellen (VDMs) für die Fiori-Apps in vereinfachter und optimierter Form zur Verfügung stellt“. Erneut gilt als Voraussetzung mindestens die Installation der SAP HANA Platform Edition 1.0 SPS 8. Ein Beispiel einer solchen Landschaft zeigt Abbildung 13. Für die übrigen Fälle entfällt dann die HANA XS Komponente bzw. auch die HANA-Datenbank und der Dispatcher.

In dem vorliegenden Fall war eine transaktionale Applikation zu entwickeln, denn es soll ein Formular ausgefüllt und an den Backend zurück geschickt werden, so dass die Systemvoraussetzungen keine HANA-Datenbank implizieren.

Skizze der Verwendung des Vorgehensmodells

Die Applikation und deren Entwicklung soll hier nicht zu detailliert dargestellt werden, es soll jedoch skizziert wer-

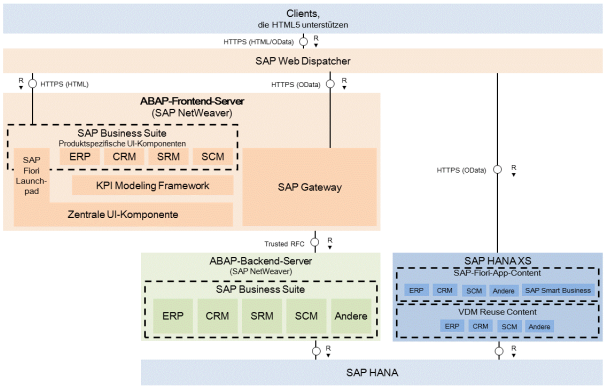

Abbildung 13: minimale Systemvoraussetzung für analytische Applikationen [12]

den, wie die wesentlichen Punkte des Modells Anwendung fanden.

Wahl des Templates

Es musste entschieden werden, welches Template in der Entwicklung verwendet werden soll. Es wird eine einzelne Anfrage angelegt, so dass Master-Detail-Ansichten/ Splitscreens nicht geeignet scheinen, zudem sollen eine Reihe von Daten eingebbar sein. Es ist somit eine SAP UI5-Applikation zu wählen, als Viewtyp wurde aus den weiter oben genannten Gründen der Typ `XML` verwendet.

Implementierung der Datenverbindung

Wie oben erläutert, gibt es verschiedene Möglichkeiten, eine Datenbindung herzustellen. Sinnvoll ist dabei ein Modell, welches eine Two-Way-Datenbindung unterstützt, damit eingegebene Daten angezeigt und aktualisiert werden. Testweise wurde als Modell im Sinne des MVC-Patterns das JSON-Modell verwendet - insbesondere, da es einen übersichtlicheren Eindruck macht, als das XML-Modell. Die Implementierung in die Applikation kann in einem ersten Schritt mit dem Erstellen der Model-Datei: `kunden.json` erfolgen. Diese liegt dann in dem „Model-Ordner“ der Applikation und es können dann selbst ausgedachte Test-Kunden eingepflegt werden. Es wurde hierzu der Pfad `KundenCollection` angelegt (Abbildung 14).

Zur Instanziierung muss das Modell noch in den Controller eingebunden werden (Abbildung 15). Um das synchrone Laden einzustellen, wird der Parameter bAsync auf „false“ gesetzt was aus der API-Referenz zum JSON-Modell hervorgeht. Ursprünglich war geplant, die Datenverbindung mit einem MockServer zu simulieren. Im Rahmen des Prototypens hat dies aber leider nicht korrekt funktioniert. Das Erstellen der xml-Datei (metadata.xml) im Model-Ordner sowie auch das Erzeugen der Mockdaten war erfolgreich, jedoch ist beim Starten der Applikation eine Fehlermeldung aufgetreten, dass keine metadata.xml-Datei gefunden werden könne. Für dieses Problem wurde keine funktionsfähige Lösung gefunden, sodass die Datenanbindung mit dem JSON-Modell hergestellt wurde. Es muss in Zukunft noch untersucht werden, welches Problem mit dem Mock-Server bestand.

```json
{
    "KundenCollection": [{
        "KundenID": "001",
        "Name": "Testfirma1",
        "PLZOrt": "11111 Testenhausen",
        "Kundenklassifikation": "Intressent",
        "Telefon": "0123123",
        "Email": "abc@def.de",
        "Projektname":"Projekt N",
        "Bonusvereinbarung":"Ja",
        "Planzeit": "8",
        "Vertriebsorganisation":"74"

    }, {
        "KundenID": "003",
        "Name": "Heinz GmbH",
        "PLZOrt": "22223 Ballen",
        "Kundenklassifikation": "Intressent & Kunde",
        "Telefon": "0512312",
        "Email": "ageabc@dvef.de",
        "Projektname":"Projekt X",
        "Bonusvereinbarung":"Nach Erfolg",
        "Planzeit": "5",
        "Vertriebsorganisation":"21"
    },{
```

Abbildung 14: Befüllung kunden.json

```
//Instanz des JSON-Model erzeugen
this.model = new sap.ui.model.json.JSONModel();
//JSON Model laden
this.model.loadData("model/kunden.json", null, false);
//JSON Model an View binden
this.getView().setModel(this.model);
```

Abbildung 15: Einbindung JSON-Modell „kunden.json“ im Controller

Verwendung des UI5 Demo Kit

Es sollen für verschiedene Arten von Verpackungen spezifische beschreibende Daten eingegeben werden, die sich je nach Verpackungstyp unterscheiden. Hierzu soll der Wizard auf dem Demo Kit verwendet werden. Der Wizard hat nämlich den Vorteil, dass bis zu acht Schritte implementiert werden können, sodass zu Übersichtlichkeitszwecken eine kleinschrittige Einteilung der relativ umfassenden Kundenanfrage vorgenommen werden kann. Am Ende des Wizards erscheint ein zusammenfassender Bildschirm, der alle zuvor eingegebenen Daten anzeigt. Für die Entität „Wizard“ [10] sind neben einer kurzen Beschreibung drei Code-Beispiele angegeben. Die Dokumentation beschreibt den Wizard als geeignet bei der Vervollständigung und Erfüllung einer komplexen Nicht-Standardanwendung, indem es die Aufgabe in Teileinheiten gliedert und so den Nutzer durch die Abschnitte führt. Das Code-Beispiel Wizard `standard use case` schien für die Kundenanfrage besonders gut geeignet, sodass die drei Code-Dateien (view, controller und ReviewPage.fragment.xml) übernommen und angepasst werden können. Ein Beispiel einer Anwendung des Wizards direkt aus dem UI5 Demo Kit heraus zeigt Abbildung 16 (Bemerkung: der Wizard `branching use case` scheint nun für die Aufgabe besser geeignet, jedoch war er bei Erstellung der Arbeit nicht verfügbar).

Die „View-Datei“ bzw. die „Controller-Datei“ des Wizards sind in den „View“ bzw. „Controller“-Ordner zu kopieren und anzupassen. Die „ReviewPage.fragment.xml“ des Wizards ist die abschließende Seite, in der die zuvor eingegebenen Daten überprüft und ggf. durch Betätigen des Edit-Buttons nochmal geändert werden können, bevor sie an das SAP-System abgeschickt werden. Fragments stellen eigentlich auch eine

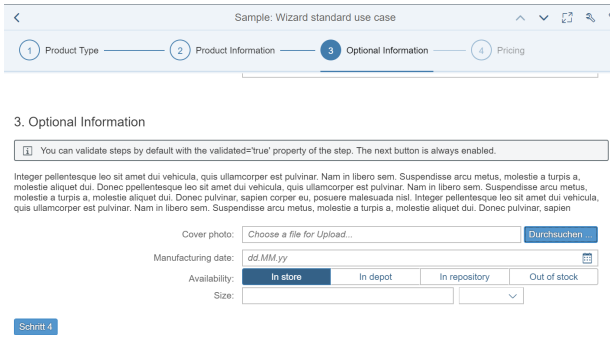

Abbildung 16: Wizard - direkt in UI5-Demo Kit

„View-Datei“ dar, besitzen jedoch aber keinen eigenen Controller, sondern den der anderen „View-Datei“.

Ein Beispiel der „View-Datei“, direkt aus dem Demo-Kit für den Wizard zeigt Abbildung 18.

```
<mvc:View
        height="100%"
        controllerName="sap.m.sample.Wizard.C"
        xmlns:form="sap.ui.layout.form"
        xmlns:core="sap.ui.core"
        xmlns:u="sap.ui.unified"
        xmlns:mvc="sap.ui.core.mvc"
        xmlns="sap.m">
        <NavContainer id="wizardNavContainer">
            <pages>
                <Page
                id="wizardContentPage"
                showHeader="false">
                    <content>
                        <Wizard id="CreateProductWizard"
                                complete="wizardCompletedHandler">
                            <WizardStep id="ProductTypeStep"
                                        title="Product Type"
                                        validated="true">
                                <MessageStrip class="sapUiSmallMarginBottom"
                                        text="The Wizard control is supposed to
                                            the user to work with."
                                        showIcon="true"/>
                                <Text class="sapUiSmallMarginBottom"
                                        text="Sed fermentum, mi et tristique ul
                                            nibh lorem malesuada diam. Nulla qu
                                            Nam vitae ante posuere, molestie ne
                                            lorem. Mauris vitae elementum mi, s
```

Abbildung 17: Wizard - View direkt aus UI5-Demo Kit

Beide „View-Dateien“ müssen im Controller (Abbildung 18) verarbeitet werden. Es wird der Controller des UI5-Kits angepasst, hier soll aber lediglich wieder der Code des unangepassten Controllers des UI5-Kits gezeigt werden. Es wäre in Zeile 10 die eigene App einzutragen, in Zeile 12 die View für die Kundenanfrage zu referenzieren. Zeile 13 und 14 beziehen sich auf unveränderte Teile der View und können übernommen werden, Zeile 15 müsste sich dann auf die Review-Page der Kundenanfrage beziehen. Selbstverständlich enthält der angepasste Controller auch die benötigten Feldvalidierungen.

Als zweite Entität wird die „ComboBox“ vorgestellt. Die Besonderheit hierbei ist, dass es in der Kundenanfrage „ComboBoxen“ gibt, die sich eigentlich der Daten aus dem Backend bedienen bzw. in diesem Fall aus dem JSON-Modell. Als Beispiel wird dafür die Auswahl des Auftraggebers verwendet, welches das erste Textfeld der Kundenanfrage darstellt. In diesem Feld sollen Daten aus dem SAP-System übernommen werden. Gelöst wurde das Problem mit einer doppelten ComboBox, die neben dem Namen auch die ID des Kundens sucht, einliest und abbildet.

```
sap.ui.define([
    'jquery.sap.global',
    'sap/ui/core/mvc/Controller',
    'sap/ui/model/json/JSONModel',
    "sap/m/MessageToast",
    "sap/m/MessageBox"
], function(jQuery, Controller, JSONModel, MessageToast, MessageBox) {
    "use strict";

    var WizardController = Controller.extend("sap.m.sample.Wizard.C", {
        onInit: function () {
            this._wizard = this.getView().byId("CreateProductWizard");
            this._oNavContainer = this.getView().byId("wizardNavContainer");
            this._oWizardContentPage = this.getView().byId("wizardContentPage");
            this._oWizardReviewPage = sap.ui.xmlfragment("sap.m.sample.Wizard.ReviewPage", this);

            this._oNavContainer.addPage(this._oWizardReviewPage);
            this.model = new sap.ui.model.json.JSONModel();
            this.model.setData({
                productNameState:"Error",
                productWeightState:"Error"
            });
```

Abbildung 18: Wizard - Controller direkt aus UI5-Demo Kit

Eine Vereinheitlichung von für den Nutzer sichtbaren Textelementen bietet den Vorteil, dass diese direkt übersetzt werden, wenn die Sprache, in der der Browser gerade ausgeführt wird, von der Applikation unterstützt wird. Für jede Sprache wird deshalb eine eigene i18n-Datei erstellt. Im Fall dieser Applikation wurde die Basisdatei i18n erstellt und dort beispielhaft die Überschriften sowie die Werte des ersten Schritts des Wizards verknüpfend implementiert (es werden beispielsweise der Titel der Applikation gepflegt, die Schritte des Wizards, die Überschriften der einzelnen Steps sowie die Texte des Layouts).

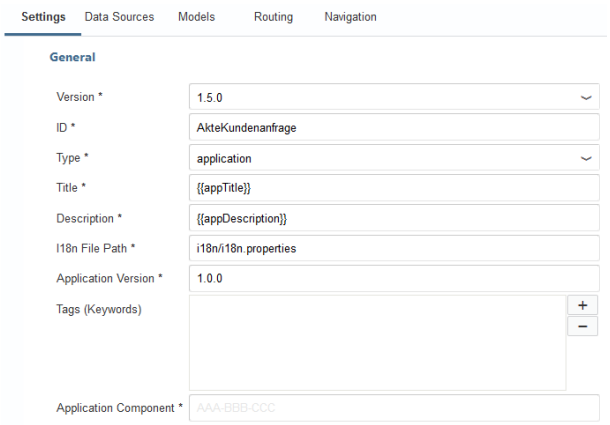

Abbildung 19: Wizard - Controller direkt aus UI5-Demo Kit

Anpassung der Manifest-Datei

Wie weiter oben ausgeführt, dient die `manifest.json`-Datei dazu, App- Einstellungen sowie Konfigurationseinstellungen vorzunehmen. Neben Models werden auch Data-Sources und Einstellungen zum User-Interface vorgenommen. Dazu bietet die SAP Web IDE neben dem Programmieren im Editor auch einen Descriptor Editor an - ein Beispiel für beide Optionen zeigen die Abbildungen 19 und 20.

Resume und Ausblick

Die wesentliche Idee dieser Arbeit bestand darin, das Entwicklungsmodell für Fiori und SAP UI5-Apps zu zeigen und exemplarisch über die Entwicklung einer Anwendung unter der Hana Cloud-Plattform zu dokumentieren. Die Hauptproblematik der App bestand in der Abbildung einer dynamischen Dateneingabe, was technisch über den Wizard der UI5-Bibliothek gelöst wurde.

Abbildung 20: Wizard - Controller direkt aus UI5-Demo Kit

Das technische Hauptproblem bestand darin, dass die Datenanbindung zum ERP-Backend nicht betrachtet werden sollte und somit mit Mock-Daten gearbeitet werden musste, wobei die Mock-Daten nicht direkt über die Mock-Daten-Funktion der Cloud-Plattform, sondern über eine JSON-Datei im Model der Applikation bereitgestellt wurden. Als nächstes ist zu untersuchen, wie die Probleme mit dem Mock-Datenserver behoben werden können bzw. nun die Einbindung direkt über einen OData-Service im Backend zu realisieren.

LITERATUR

[1] Michael Englbrecht and Michael Wegelin. *SAP Fiori - Implementierung und Entwicklung*. Rheinwerk Verlag GmbH, Bonn, 1. aufl. edition, 2016.

[2] Susan Galer. Was sap fiori bringt, 2013. zuletzt eingesehen am 22.06.2017.

[3] SAP SE. Benutzererfahrung (ux), 2017. zuletzt eingesehen am 07.07.2017.

[4] SAP SE. Split-screen layout | sap fiori design guidelines, 2016. zuletzt eingesehen am 12.07.2017.

[5] SAP SE. Full screen layout | sap fiori design guidelines, 2016. zuletzt eingesehen am 12.07.2017.

[6] SAP SE. Anatomy of sap fiori apps (week3, unit1), 2015. zuletzt eingesehen am 12.07.2017.

[7] SAP SE. Model view controller (mvc). zuletzt eingesehen am 20.07.2017.

[8] Miroslav Antolovic. *Einführung in SAPUI5*. Rheinwerk Verlag GmbH, Bonn, 2. aufl. edition, 2016.

[9] SAP SE. App overview: The basic files of your app. zuletzt eingesehen am 20.07.2017.

[10] SAP SE. Ui5 demo kit - ui development toolkit for html5 - “about“, 2017. zuletzt eingesehen am 04.09.2017.

[11] SAP SE. Sap fiori cloud demo, 2017. zuletzt eingesehen am 21.08.2017.

[12] SAP SE. Sap dokumentation || einrichten der sap-fiori-systemlandschaft mit sap hana xs, 2015. zuletzt eingesehen am 26.07.2017.