



HOCHSCHULE FULDA
FACHBEREICH ANGEWANDTE INFORMATIK
STUDIENGANG WI (B.Sc.)

Low code Applikation Entwicklung mit SAP
AppGyver im Vergleich zu nativen Fiori Entwicklung

Bachelorarbeit von Fangfang Tan
Matrikelnummer: 1222047

Erstgutachter: Prof. Dr. Norbert Ketterer
Zweitgutachter: M. A. Mike Zaschka
Abgabetermin: 6. Februar 2023



Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben. Ich erkläre weiterhin, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Bad Hersfeld, den 30. Januar 2023

DEINE Unterschrift

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Bachelorarbeit unterstützt und motiviert haben.

Zunächst möchte ich mich bei meinem Betreuer Mike Zaschka für die hilfreichen Anregungen, konstruktive Kritik und vieles mehr während der Erstellung dieser Arbeit bedanken.

Besonderer Dank gebührt auch meinem Betreuer an der Hochschule Fulda, Prof. Dr. Norbert Ketterer, der mich richtungsweisend und mit viel Engagement während meiner Arbeit begleitet hat.

Ein herzliches „Dankeschön!“ geht auch an allen anderen Kollegen und Kolleginnen der Firma p36, die mich herzlich aufgenommen und mir während der Schreibphase meiner Bachelorarbeit wertvollen Unterstützungen gegeben haben.

Zusammenfassung

Die vorliegende Bachelorarbeit befasst sich mit der Low-Code Anwendungsentwicklung mit SAP AppGyver im Vergleich zur nativen SAP Fiori-Entwicklung, wobei der Fokus auf drei Technologien liegt: SAP AppGyver, Fiori Elements und SAPUI5. Anhand eines kundenorientierten Anwendungsfalles werden Anwendungen mit den drei Technologien entwickelt, um die drei Technologien zu betrachten und zu vergleichen. Eine Reihe von Funktionalitäten außerhalb des Anwendungsfalls wird ebenfalls untersucht, damit die drei Technologien eingehend bewertet werden können. Anschließend werden Bewertungsmatrizen definiert, Bewertungen durchgeführt und schließlich die Bewertungsergebnisse interpretiert und diskutiert. Mit Hilfe der Bewertungsergebnisse werden die Vor- und Nachteile der Technologie ermittelt.

Abstract

This bachelor thesis is dealing with low-code application development with SAP AppGyver compared to native SAP Fiori development, with the focus on three technologies: SAP AppGyver, Fiori Elements and SAPUI5. A customer-oriented use case will be used to develop applications with the three technologies for consideration and comparison. A set of functionalities outside of the use case will also be studied so that the three technologies can be evaluated in more depth. Then, evaluation matrices will be defined, evaluations will be carried out, and finally the evaluation results will be interpreted and discussed. With the evaluation results, the advantages and disadvantages of the technologies will be identified.

Inhaltsverzeichnis

| | |
|--|-------------|
| Abbildungsverzeichnis | VIII |
| Tabellenverzeichnis | X |
| Quelltextverzeichnis | XI |
| Abkürzungsverzeichnis | XII |
| 1 Einleitung | 1 |
| 1.1 Problemstellung und Motivation | 1 |
| 1.2 Ziele der Arbeit | 2 |
| 1.3 Beschreibung des Anwendungsfalls | 3 |
| 1.4 Aufbau der Arbeit | 4 |
| 2 Grundlagen | 6 |
| 2.1 Entwicklung mit Low-Code/No-Code | 6 |
| 2.2 Architektur von SAP-Anwendungen in der SAP Business Technology Plattform | 8 |
| 2.3 SAP Fiori | 10 |
| 2.4 SAP AppGyver | 11 |
| 2.4.1 Grundlage von AppGyver | 11 |
| 2.4.2 Entwicklungsumgebung: Composer Pro | 12 |
| 2.5 SAPUI5 | 16 |
| 2.5.1 Grundlage von SAPUI5 | 16 |
| 2.5.2 Entwicklungsumgebung: Visual Studio Code | 17 |
| 2.6 Fiori Elements | 19 |
| 2.6.1 Grundlage von Fiori Elements | 19 |
| 2.6.2 Entwicklungsumgebung: Business Application Studio . . . | 21 |
| 3 Implementierung des Anwendungsfalls | 22 |

| | | |
|----------|--|-----------|
| 3.1 | Fiori Elements | 23 |
| 3.1.1 | Dev Space erstellen | 23 |
| 3.1.2 | Datenentitäten und Service definieren | 24 |
| 3.1.3 | User Interface erstellen | 25 |
| 3.1.4 | Deployment des OData-Services | 27 |
| 3.2 | AppGyver | 29 |
| 3.2.1 | Projektaufbau | 29 |
| 3.2.2 | OData Integration | 29 |
| 3.2.3 | Listenansicht zur Anzeige aller Produkte | 30 |
| 3.2.4 | Einzelansicht für ein Produkt | 32 |
| 3.2.5 | Maske zum Erstellen eines einzelnen Produkts | 34 |
| 3.3 | SAPUI5 | 35 |
| 3.3.1 | Erstellung einer Anwendung mit dem UI5-Generator | 35 |
| 3.3.2 | OData Service einbinden | 37 |
| 3.3.3 | Listenansicht zur Anzeige aller Produkte | 38 |
| 3.3.4 | Einzelansicht für ein Produkt | 41 |
| 3.3.5 | Maske zum Pflegen eines einzelnen Produkts | 42 |
| 4 | Untersuchung weiteren Funktionen | 44 |
| 4.1 | Integration von Suchfilter | 44 |
| 4.2 | Integration von Paginierung | 46 |
| 4.3 | Integration von Bild- und PDF-Datei | 49 |
| 4.4 | Integration von Barcode-Scanner-Funktion | 51 |
| 4.5 | Nutzung nativer mobiler Funktionen | 53 |
| 4.6 | Möglichkeiten zum Deployment für unterschiedliche Endgeräte . | 53 |
| 4.7 | Freie Gestaltungsmöglichkeit | 54 |
| 5 | Bewertung der Technologien | 57 |
| 5.1 | Bewertungsmatrixen definieren | 57 |
| 5.1.1 | Bewertungsmatrix zur Implementierung der Funktionalität | 57 |
| 5.1.2 | Bewertungsmatrix für die Entwicklerperspektive | 60 |
| 5.2 | Durchführung der Bewertung | 62 |
| 5.2.1 | Bewertung zur Implementierung der Funktionalität | 62 |
| 5.2.2 | Bewertung von der Entwicklerperspektive | 65 |
| 5.3 | Interpretation und Diskussion | 68 |
| 5.3.1 | SAP Fiori Elements in Verbindung mit Business Application Studio | 68 |
| 5.3.2 | SAP AppGyver in Verbindung mit Composer Pro | 70 |
| 5.3.3 | SAPUI5 in Verbindung mit SAPUI5 | 72 |

| | |
|-------------------------------|-----------|
| 6 Schluss und Ausblick | 75 |
| Literaturverzeichnis | 79 |
| A JSON-Schema | 85 |

Abbildungsverzeichnis

| | | |
|------|--|----|
| 2.1 | Architektur einer SAP-Anwendung in der SAP BTP | 10 |
| 2.2 | Pages einer AppGyver-Anwendung | 13 |
| 2.3 | Toolbar in Composer Pro | 13 |
| 2.4 | View-Components in Composer Pro | 15 |
| 2.5 | Exemplarische Formel in AppGyver | 16 |
| 2.6 | List Report und Object Page (Quelle: SAP Fiori Design Guidelines. URL: https://experience.sap.com/fiori-design-web/smart-templates) | 19 |
| 2.7 | Overview Page (Quelle: SAP Fiori Design Guidelines. URL: https://experience.sap.com/fiori-design-web/smart-templates) . . | 20 |
| 3.1 | Umsetzungsarchitektur der Technologie | 23 |
| 3.2 | Create a New Dev Space | 23 |
| 3.3 | Grundlegenden Aufbau inkl. der Cards von BAS | 24 |
| 3.4 | Datenmodell | 24 |
| 3.5 | Service | 25 |
| 3.6 | Sample Data | 26 |
| 3.7 | UI-Konfiguration | 27 |
| 3.8 | Listenseite, Detailseite und „neues Objekt anlegen- Seite der Fiori-Elements-Anwendung | 28 |
| 3.9 | Cloud Foundry Sign In und Targets | 29 |
| 3.10 | Destination-Konfiguration in SAP BTP Cockpit | 30 |
| 3.11 | OData-Integration in AppGyver | 30 |
| 3.12 | Data-Variable Logik | 31 |
| 3.13 | Listenseite und List Item mit Repeat in AppGyver | 31 |
| 3.14 | Logische Ablauffunktion für List Items | 32 |
| 3.15 | Wert zuweisen für App-Variable <i>appVarRecord</i> in <i>Set app variable Ablauffunktionskomponente</i> | 33 |
| 3.16 | Logische Ablauffunktion für Edit-, Back- und Delete-Button . . | 33 |

| | |
|--|----|
| 3.17 Logische Ablauffunktion für Save-Button | 34 |
| 3.18 Listenseite, Detailseite und „Create new record“-Seite der AppGyver-Anwendung | 35 |
| 3.19 Aussehen der UI5 Demo-Anwendung | 37 |
| 3.20 Benutzeroberfläche der Listenansicht der Produkte | 39 |
| 3.21 Benutzeroberfläche der Einzelansicht | 41 |
| 3.22 Maske zum Pflegen eines einzelnen Produkts | 43 |
| 4.1 Suchfilter in AppGyver | 45 |
| 4.2 Seitenlogik von Paginierung in AppGyver | 47 |
| 4.3 Paging definieren in Ablauffunktion „Get record collection“ | 47 |
| 4.4 Logik für PREV- und NEXT-Button in AppGyver | 48 |
| 4.5 Benutzeroberfläche nach Paginierung in AppGyver | 48 |
| 4.6 Image Binding in AppGyver | 50 |
| 4.7 Logik für Load-PDF-Button in AppGyver | 50 |
| 4.8 Logik für Scan QR/Barcode-Button in AppGyver | 51 |
| 4.9 Gestaltung von Weather-App, Barcode-Scanner-App ²¹ und Product-List-App ²² | 56 |
| 5.1 Daten Modellierung in AppGyver | 62 |
| 5.2 Daten-Capabilities in AppGyver | 62 |
| 5.3 Bewertung von Umsetzung der Anforderungen mit Fiori Elements | 69 |
| 5.4 Bewertung von Fiori Elements mit BAS | 69 |
| 5.5 Bewertung von Umsetzung der Anforderungen in AppGyver | 71 |
| 5.6 Bewertung von AppGyver Composer Pro | 71 |
| 5.7 Bewertung von Umsetzung der Anforderungen in SAPUI5 | 73 |
| 5.8 Bewertung von SAPUI5 mit VS-Code | 73 |

Tabellenverzeichnis

| | | |
|------|--|----|
| 1.1 | Definition der Anwendungsfall | 3 |
| 2.1 | AppGyver-Baustein in die Schichten des MVC-Models | 14 |
| 3.1 | Umsetzungsanforderung der Technologie | 22 |
| 5.1 | Bewertungsmatrix zur Implementierung der Funktionalität | 58 |
| 5.2 | Erfüllung Kriterien der Bewertungsmatrix | 59 |
| 5.3 | Bewertungsmatrix aus der Entwickler-Perspektive | 60 |
| 5.4 | Erfüllungskriterien aus der Entwickler-Perspektive | 61 |
| 5.5 | Bewertung der Backend-Funktionen | 63 |
| 5.6 | Bewertung der Grundlistenfunktionen | 64 |
| 5.7 | Bewertung der weiteren Funktionen | 64 |
| 5.8 | Bewertungsmatrix aus der Entwicklerperspektive | 65 |
| 5.9 | Vor- und Nachteile von SAP Fiori Elements in Verbindung mit BAS | 70 |
| 5.10 | Vor-und Nachteile von AppGyver mit Composer Pro | 72 |
| 5.11 | Vor- und Nachteile von SAPUI5 in Kombination mit VS-Code . . | 74 |

Quelltextverzeichnis

| | | |
|------|--|----|
| 3.1 | Eingabe der Paramtern des UI5-Generators | 36 |
| 3.2 | Die grundlegende Projektstruktur eines SAPUI5-Projekts | 36 |
| 3.3 | Auszüge aus der Klasse <code>Component</code> | 38 |
| 3.4 | Auszüge aus der View <code>Main.view.xml</code> | 39 |
| 3.5 | Auszüge aus der Controller <code>Main.control.ts</code> | 40 |
| 3.6 | Auszüge aus der View <code>Main.view.xml</code> | 40 |
| 3.7 | Auszüge aus der Controller <code>Main.control.ts</code> | 40 |
| 3.8 | Auszüge aus der View <code>Detail.view.xml</code> | 41 |
| 3.9 | Auszüge aus der Fragment <code>createProductDialog.fragment.xml</code> . | 42 |
| 3.10 | Auszüge aus der Controller <code>Main.control.ts</code> | 42 |
| 4.1 | Implementierung von Suchfilter in der <code>Main.view.xml</code> | 46 |
| 4.2 | Implementation von Suchfilter in der <code>Main.controller.ts</code> | 46 |
| 4.3 | Implementation von Paginierung in der <code>Main.view.xml</code> | 48 |
| 4.4 | Implementierung Bild- und PDF Datei Integration in SAPUI5 . . | 51 |
| 4.5 | Implementierung Barcode-Scanner in View <code>Barcode.view.xml</code> . | 52 |
| 4.6 | Implementierung Barcode-Scanner in Controller <code>Barcode.control.ts</code> | 52 |
| 5.1 | Datenmodellierung in der <code>model.cds</code> | 63 |
| 5.2 | Service Bereitstellung in der <code>public_service.cds</code> | 63 |
| A.1 | JSON-Schema für die Mapping-Datei | 85 |

Abkürzungsverzeichnis

| | |
|--------------|---|
| API | Application Programming Interface – Programmierschnittstelle |
| BTP | Business Technology Plattform |
| BAS | Business Application Studio |
| CAP | SAP Cloud Application Programming Model |
| CDS | Core Data Service |
| CORS | Cross Origin Resource Sharing |
| CRM | Customer Relationship Management |
| CRUD | Create Read Update Delete – Vier fundamentale Optionen des Datenmanagement |
| CSS | Cascading Style Sheets |
| CSV | Comma-Separated-Values – Datenformat |
| GPS | Global Positioning System |
| HTML | HyperText Markup Language – Auszeichnungssprache |
| IDE | Integrierte Entwicklungsumgebung |
| JSON | JavaScript Object Notation – Datenserialisierungsformat |
| LCNC | Low-Code/No-Code |
| MVC | Model-View-Controller – Software Designmuster |
| OData | Open Data Protocol |
| NDC | Native Device Capabilities |

| | |
|----------------|---|
| NPM | Node Package Manager |
| PDF | Portable Document Format |
| PT | Personentage |
| SAP | Systemanalyse Programmentwicklung – Softwarekonzern |
| SAPUI5 | SAP UI Development Toolkit für HTML5 |
| SDK | Software Development Kit |
| UI | User Interface |
| URL | Uniform Resource Locator – Internetadresse |
| UX | User Experience |
| VS-Code | Visual Studio Code – Software Entwicklungstool |
| XML | Extensible Markup Language – erweiterbare Auszeichnungssprache |
| YAML | YAML Ain't Markup Language – Datenserialisierungsformat |

Kapitel 1

Einleitung

1.1 Problemstellung und Motivation

Seit die Branchenanalysten von Forrester Research im Jahr 2014 erstmals das Konzept von Low-Code erwähnten [RJR22], hat sich der zugehörige Markt rasant entwickelt. Immer mehr Unternehmen nutzen Low-Code-Plattformen, um Anwendungen schneller zu entwickeln und damit den digitalen Wandel zu beschleunigen. Die Analysten von Gartner erwarten, dass im Jahr 2025 rund 70% der von Unternehmen entwickelten Anwendungen auf Low-Code-Technologien basieren werden [HV22].

Im Bereich der Enterprise Software werden die meisten Low-Code-Plattformen verwendet, um eine bestimmte Anwendungsform in einem spezifischen Kontext zu entwickeln. In der Studie „No-Code/Low-Code 2022“ im Magazin COMPUTERWOCHE, gibt die Mehrheit der gefragten Unternehmen an, dass sie Low-Code hauptsächlich in den Bereichen CRM (34%) und ERP (31%) einsetzen. Speziell ausgerichtete Low-Code Plattformen werden ebenfalls im HR-Umfeld (19%) verwendet, sowie für die Erstellung digitaler Workflows und Verwaltungsprozesse (jeweils 16%). Nur 10% der Befragten nutzen eine universell einsetzbare Plattform, die sich für übergreifende und flexible Geschäftsprozesse eignet. Laut Jürgen Erbeldinger, einem Low-Code-Experten der Low-Code-Plattform ESCRIBA, fehlt den Plattformen hierfür die entsprechende Tiefe [AS22].

AppGyver betrachtet sich selbst als die weltweit erste professionelle Low-Code Plattform, die es ermöglicht, Anwendungen für unterschiedliche Geschäftsprozesse, Anwendungsszenarien und auch Endgeräte zu erstellen [SAP22b]. Im Februar

2021 wurde AppGyver von dem Marktführer im Bereich Enterprise Software SAP übernommen und wird seitdem in deren Entwicklungsportfolio rund um die SAP Business Technology Plattform eingegliedert [Cen21], [Com22b]. Seit dem 15. November 2022 wurde SAP AppGyver in SAP Build App umbenannt und ist nun Teil von SAP Build. AppGyver steht damit in Konkurrenz zu etablierten Tools und Frameworks zur Anwendungsentwicklung: SAPUI5 ist ein JavaScript-Framework und bildet die Grundlage nahezu aller heute entwickelten SAP-Oberflächen. Die Erstellung von SAPUI5-Anwendungen setzt jedoch ein tieferes technisches Verständnis voraus. Basierend auf SAPUI5 steht mit SAP Fiori Elements ein Framework zur Verfügung, welches durch Annotationen die einfache Erstellung von datengetriebenen Oberflächen erlaubt. Dank der guten Integration in die SAP eigenen Entwicklungsumgebungen, das SAP Business Application Studio, kann SAP Fiori Elements in Kombination mit dem SAP Application Programming Model auch im Bereich der Low-Code Entwicklung platziert werden [Ele22]. Für Unternehmen ergibt sich in Zukunft nun die Fragestellung, welche Plattform und Tools im SAP-Umfeld eingesetzt werden sollten, um Anwendungen zu entwickeln. Die Aufgabe dieser Bachelorarbeit besteht darin, den Entwicklungsprozess von SAP AppGyver, SAPUI5, sowie Fiori Elements in Kombination mit dem SAP Application Programming Model zu bewerten, Vor- und Nachteile der jeweiligen Lösung herauszuarbeiten und dadurch ein Entscheidungsmatrix zu entwerfen, welche die Wahl zwischen diesen drei Technologien vereinfacht.

Diese wissenschaftliche Arbeit wird dabei unterstützt durch die Firma p36 GmbH. P36, mit dem Sitz im Bad Hersfeld, wurde 2015 von Patrick Pfau und Robin Wennemuth gegründet. Das mit 27 Mitarbeitern noch recht kleine, aber stark wachsende Softwareunternehmen besitzt einen starken Fokus auf die Entwicklung von Cloud-basierten Anwendungen im SAP-Umfeld [Gmb22]. Bei der Umsetzung der Anwendungen setzt p36 überwiegend auf die sehr technische SAPUI5-Entwicklung und evaluiert derzeit den Einsatz von Low-Code-Plattformen. p36 stellt deswegen einen, sich an reellen Kundenanforderungen orientierenden, Anwendungsfall zur Verfügung, der im Rahmen der Arbeit als Grundlage der Evaluierung dienen soll.

1.2 Ziele der Arbeit

Die folgenden Fragen werden in der Bachelorarbeit behandelt werden:

- Was genau verbirgt sich hinter dem Begriff Low-Code und wie grenzt sich Low-Code von bisherigen Arten der Entwicklung ab?

- Wie wird eine benutzerspezifische Anwendung mit dem Low-Code/No-Code basierten Tool SAP AppGyver implementiert?
- Wie wird eine benutzerspezifische Anwendung mit SAPUI5 implementiert?
- Wie wird eine benutzerspezifische Anwendung mit Fiori Elements implementiert?
- Welche Vor- und Nachteile dieser drei Technologien lassen sich durch die exemplarische Umsetzung herausstellen?
- Welche Technologie eignet sich in Zukunft für welche Umsetzungsszenarien?

1.3 Beschreibung des Anwendungsfalls

In dieser Arbeit werden die drei genannten Technologien verwendet, um eine konkrete Anwendung zu entwickeln. Der Anwendungsfall definiert sich, wie folgt:

| | |
|-------------------------|--|
| Name: | Applikation zur Verwaltung von Produktinformationen |
| Umsetzung in: | <ul style="list-style-type: none"> • SAP Fiori Elements mit SAP Business Application Studio (Backend + UI) • SAP AppGyver (UI) • SAPUI5 (UI) |
| Anforderungen Backend: | <ul style="list-style-type: none"> • Bereitstellung einer Datenbank-Entität Products mit folgenden Eigenschaften: <ul style="list-style-type: none"> – ID; title; materialNumber; description; price; stock • Bereitstellung eines OData-Services zum Auslesen, Erstellen und Aktualisieren (CRUD) der Produkte |
| Anforderungen Frontend: | <ul style="list-style-type: none"> • Funktionen: <ul style="list-style-type: none"> – Listenansicht zur Anzeige aller Produkte – Einzelansicht für ein Produkt – Maske zum Pflegen eines einzelnen Produkts • Datenanbindung: <ul style="list-style-type: none"> – Anbindung an den OData-Service zum Auslesen und Schreiben von Produkten • Look and Feel: <ul style="list-style-type: none"> – Implementierung in Anlehnung an die SAP UX-Guideline SAP Fiori |

Tabelle 1.1: Definition der Anwendungsfall

Zusätzlich zu den umzusetzenden Funktionalitäten wird eine Reihe weiterer

Funktionen ohne praktische Umsetzung untersucht, um SAPUI5, Fiori Elements und AppGyver tiefergehend zu evaluieren. Diese Funktionen umfassen:

- Integration von Suchfilter und Paginierung
- Integration von Bild und PDF-Datei
- Integration von Barcode-Scanner-Funktionen
- Nutzung mobiler Funktionen wie Sensoren
- Möglichkeiten zum Deployment für unterschiedliche Endgeräte
- Freie Gestaltungsmöglichkeiten

1.4 Aufbau der Arbeit

Die vorliegende Bachelorarbeit gliedert sich in insgesamt sechs Kapitel. In der Einleitung werden die Problemstellung und Motivation, die Ziele der Arbeit und der umzusetzende Anwendungsfall vorgestellt.

Im 2. Kapitel werden zunächst die grundlegenden Konzepte erläutert. Der erste Abschnitt beschäftigt sich mit dem Low-Code/No-Code (LCNC) Konzept und liefert eine Definition von LCNC und einen Überblick über existierende LCNC-Plattformen. Kapitel 2 beinhaltet ebenfalls einen Überblick über die Architektur von SAP-Anwendungen in der SAP Business Technology Plattform, sowie, im dritten Abschnitt, die Grundlagen von SAP Fiori. Der vierte, fünfte und letzte Abschnitt dieses Kapitels beschreibt die Grundlagen von AppGyver, SAPUI5 und Fiori Elements, sowie die Entwicklungsumgebung, in denen der genannte Anwendungsfall entwickelt wird, nämlich SAP Business Application Studio, Composer Pro und Visual Studio Code.

Kapitel 3 bis 5 bilden den Hauptteil der Bachelorarbeit. Im dritten Kapitel wird der Umsetzungsprozess des Anwendungsfalls mit Fiori Elements, AppGyver und SAPUI5 beschrieben. Die zu beschreibenden Funktionen umfassen:

- Bereitstellung eines OData-Services zum Auslesen, Erstellen und Aktualisieren der Produkte
- Erstellung einer Listenansicht zur Anzeige aller Produkte
- Erstellung einer Einzelansicht für ein Produkt
- Erstellung einer Maske zum Pflegen eines einzelnen Produkts

Im 4. Kapitel werden weitere Funktionen, ohne technische Implementierung, untersucht, um Fiori Elements, AppGyver und SAPUI5 eingehender zu bewerten. Basierend auf Kapitel 3 und Kapitel 4 konzentriert sich Kapitel 5 auf die Gegenüberstellung und Bewertung der drei Tools. Hierfür werden die Be-

wertungsmatrizen definiert, die Bewertung durchgeführt und anschließend die Bewertungsergebnisse diskutiert und interpretiert. Kapitel 6, das letzte Kapitel der Bachelorarbeit, enthält abschließend ein Fazit und einen Ausblick auf die künftige Forschung.

Kapitel 2

Grundlagen

2.1 Entwicklung mit Low-Code/No-Code

Low-Code/No-Code ist ein Ansatz der Softwareentwicklung, bei dem Anwendungen mit wenig oder gar keinem selbst programmierten Code erstellt werden können. Pro-Code hingegen bezieht sich auf die klassische Entwicklung, bei der die Codezeilen von Hand geschrieben werden. Anstatt auf komplexe Programmiersprachen zurückzugreifen, kann LCNC-Entwicklung die Anwendungen durch visuelle Programmierung, also durch Anklicken, Ziehen und miteinander verbinden von Anwendungskomponenten, erstellen. Die LCNC-Plattformen bieten hierfür spezielle visuelle Programmierumgebungen an, die aus einer Reihe an vorgefertigten Code-Bausteinen und den Möglichkeiten diese in Form einer Anwendung zusammenzusetzen, bestehen. No-Code-Plattformen ersetzen die traditionelle codebasierte Entwicklungsumgebung dabei vollständig, während bei Low-Code-Plattformen möglicherweise Basis-Programmierkenntnisse erforderlich sind.

Auch wenn es bereits in der Vergangenheit Ansätze zur visuellen Programmierung gab, so ist die derzeitige LCNC-Entwicklung aufgrund des Reifegrades des Toolings eine ernsthafte Alternative zur Pro-Code-Entwicklung. Einige Experten glauben, dass LCNC die Zukunft der Softwareentwicklung ist, weil es einen schnelleren Entwicklungsprozess ermöglicht. Mit den einfach zu bedienenden visuellen Benutzeroberflächen, sowie den ausgereiften Entwicklungstoolkits ist man in der Entwicklung deutlich schneller, als wenn man Tausende von Codezeilen schreiben muss. Neben dem Zeitfaktor spielt auch die damit einzusparenden Kosten eine große Rolle [Con22].

Ein weiterer Vorteil der LCNC-Entwicklung ist, dass sie den Mangel an qualifizierten Entwicklern kompensiert. LCNC-Entwicklung eignet sich für Entwickler aller Niveaus. Die No-Code-Plattformen sind insbesondere für Citizen Developer sinnvoll, die möglicherweise überhaupt keine Programmierausbildung haben. Ein Citizen Developer ist ein Mitarbeiter, der mit zugelassenen Tools Anwendungen für eigene Nutzung oder die Nutzung durch andere erstellt [GG22]. Darüber hinaus bieten LCNC-Plattformen professionellen Entwicklern Unterstützung, um die aufwändigen zugrundeliegenden architektonischen und infrastrukturellen Aufgaben zu reduzieren.

Der Markt für LCNC-Plattformen ist in den letzten Jahren deutlich gewachsen. Nach Angabe von G2, eine der Website für Softwarelisten und Bewertungen, gibt es (Stand November 2022) 226 Low-Code Plattformen [Ove22a] und 288 No-Code Plattformen [Ove22b] auf dem Markt. Neben spezialisierten Unternehmen/Start-Ups, stellen auch größere Unternehmen LCNC-Platformen für ihr jeweiliges Ökosystem zur Verfügung. Dazu einige Beispiele:

App Engine von ServiceNow wurde im März 2021 veröffentlicht [doc22g]. Es ermöglicht großen Unternehmen Low-Code-Anwendungen zu erstellen und breitzustellen. ServiceNow stellt eine Reihe an Entwicklungsvorlagen für gängige Anwendungsfälle bereit, um die Erstellung der Anwendungen zu erleichtern [CVE22]. App Engine erlaubt den Nutzern allerdings auch, Code mit traditionellen Programmiersprachen wie HTML, Javascript sowie CSS zu schreiben und zu bearbeiten.

Salesforce, vom gleichnamigen Unternehmen, ist eine Plattform für Customer-Relationship-Management (CRM) und besitzt umfangreiche Funktionen einer App-Entwicklungsplattform, um die Standard-CRM-Funktionalitäten der Plattform zu erweitern. Mit Hilfe der visuellen Programmierung können Workflow-basierte Anwendungen schnell erstellt werden, um Geschäftsprozesse abzubilden oder Kunden Zugang zu wichtigen Informationen zu gewähren. Die Salesforce-Plattform bietet neben dem Low-Code-Ansatz auch eine vollständig angepasste Anwendungsentwicklung für unterschiedliche Programmiersprachen und ist daher auch geeignet für den Code-basierten Ansatz [G222].

OutSystems vom gleichnamigen deutschen Hersteller, ist ein Beispiel für eine spezialisierte LCNC-Plattform ohne direkte Einbindung in ein größeres Ökosystem. Auch hier steht die visuelle Full-Stack-Entwicklung im Vordergrund, mit der Benutzeroberflächen, Geschäftsprozesse, Logik und Datenmodelle aufgebaut und implementiert werden können. Auch bei OutSystems ist es jedoch möglich, eigenen Code für die Anwendungserstellung hinzufügen.

Der Wettbewerb im Trend-Thema LCNC ist heute sehr stark. Auf der einen Seite gibt es die großen Unternehmens wie ServiceNow und Salesforce, die über viele Ressourcen und Fachkräfte für die Entwicklung ihrer Plattformen verfügen und ihre Kunden mit der Bereitstellung von LCNC-Funktionalitäten weiter an die Plattform binden wollen. Die resultierenden Anwendungen sind zumeist plattformabhängig, haben jedoch den großen Vorteil, dass die Daten aus dem jeweiligen Ökosystem auch anwendungsbürgreifend wiederverwendet werden können. Auf der anderen Seite gibt es die spezialisierten LCNC-Anbieter, wie OutSystems und Mendix, mit denen die Benutzer unabhängige Anwendungen entwickeln können und weniger an eine Plattform oder einen Anbieter gebunden sind [CVE22].

Im weiteren Verlauf dieser Arbeit soll der Fokus auf der LCNC-Platform SAP AppGyver liegen. AppGyver ist einer der Top-Anbieter für die LCNC-Entwicklung und kann als hybride Plattform angesehen werden. Ursprüngliche unabhängig und spezialisiert, entwickelt sich AppGyver durch den Aufkauf durch SAP und der Integration in das SAP Ökosystem zu einer leistungsfähigen Plattform, die beide Welten miteinander verbindet. Auf AppGyver wird in Abschnitt 2.4 näher eingegangen.

2.2 Architektur von SAP-Anwendungen in der SAP Business Technology Plattform

Der praktische Teil dieser Arbeit befasst sich mit der Erstellung von Anwendungen in den drei gewählten Technologien: AppGyver, SAP Fiori Elements und SAPUI5. Die grundlegende Architektur der Anwendungen orientiert sich an der heutigen Referenzarchitektur von Anwendungen auf der SAP Business Technology Plattform. Frühere SAP-Standards zur Erstellung von web-basierten Anwendungen waren eng gekoppelt mit den Backendsystemen und wurden in den jeweiligen Programmiersprachen erstellt – wie z.B. WebDynpro für Java oder WebDynpro für ABAP. Der vollständige HTML-Code, inklusive der darzustellenden Daten, wurde serverseitig generiert und das Resultat an das Endgerät übermittelt und dort durch den Browser interpretiert [Eng20, S.46]. Aufgrund der Notwendigkeit des so genannten Server-Roundtrips hatte diese Technologie einige Nachteile:

- Enge Kopplung von Daten und Darstellung.
- Aufgrund der Komplexität musste der generierte HTML-Code server-seitig gerendert werden. Deshalb war es möglich, dass die Anwendung auf dem

Endgerät nicht optimal zur Darstellung kam.

- Es gab nur sehr eingeschränkte Möglichkeiten, die Benutzeroberfläche zu gestalten.
- WebDynpro unterstützt keine Gestensteuerung oder sonstige Technologien, die für mobile Endgeräte notwendig sind.
- Wenn die Bandbreite zwischen dem Endgerät und dem Server unzureichend ist, kann die Wartezeit sehr lang sein.

Seit 2012 verfolgt SAP einen neuen Ansatz für webbasierte Anwendungen. Die Daten und ihre Bereitstellung als OData-Service werden von der eigentlichen Darstellung im Browser getrennt. Der OData-Service dient als Kommunikator zwischen Backend und Frontend und wird von der UI konsumiert.

Der erste Schritt dieses Ansatzes wurde in der On-Premise-Welt mit SAP Netweaver Gateway realisiert. Danach wurde das Konzept auch in die Cloud überführt und bietet dort die Möglichkeit, eigene OData-Services bereitzustellen oder Services aus der On-Premise-Welt zu integrieren. Dieser Ansatz hat viele Vorteile:

- Durch die Trennung von Daten und Benutzeroberfläche kann die UI sehr flexibel gestaltet werden.
- Die einmal bereitgestellten Daten können von unterschiedlichen Frontend-Applikationen genutzt werden.
- Die UI kann in unterschiedlichen Technologien und auf unterschiedlichen Endgeräten erstellt werden und dort auch native (mobile) Funktionen unterstützen.
- Die reine Bereitstellung der Daten auf dem Server ist deutlich schneller und demnach sind die Wartezeiten kürzer.

Abbildung 2.1 zeigt die Architektur einer moderner SAP-Anwendung in der SAP Business Technology Plattform. Auf der mittleren Ebene befindet sich die SAP Business Technology Platform, welche die zentrale Plattform zur Bereitstellung von Daten für webbasierte Anwendungen ist. In der BTP lassen sich eigene Datenbanken halten und eigene Services zur Verfügung stellen. Es ist jedoch ebenfalls möglich, Daten aus der SAP On-Premise-Welt (via Cloud Connector) oder auch von Drittanbieter-Systemen zentral zu integrieren. Die Daten werden jeweils als REST oder OData-Services zur Verfügung gestellt und können von Anwendungen auf der Benutzeroberseite konsumiert werden.

Für diese Bachelorarbeit wird ein OData-Service auf der SAP Business Technology Plattform erstellt. Auf die Anbindung eines On-Premise-Systems oder

das eines Drittanbieters wird an dieser Stelle verzichtet. Für die Erstellung des Backend-Parts (Datenbank + OData-Service) wird auf Fiori Elements und das SAP Cloud Application Programming Model (kurz: SAP CAP) zurückgegriffen. Zwar stehen auf der BTP technologisch auch andere Backend-Frameworks zur Verfügung, der Entwicklungsansatz mit SAP CAP und Fiori Elements ist durch die native Integration in das Business Application Studio als Low-Code-Entwicklungsumgebung jedoch der Quasi-Standard.

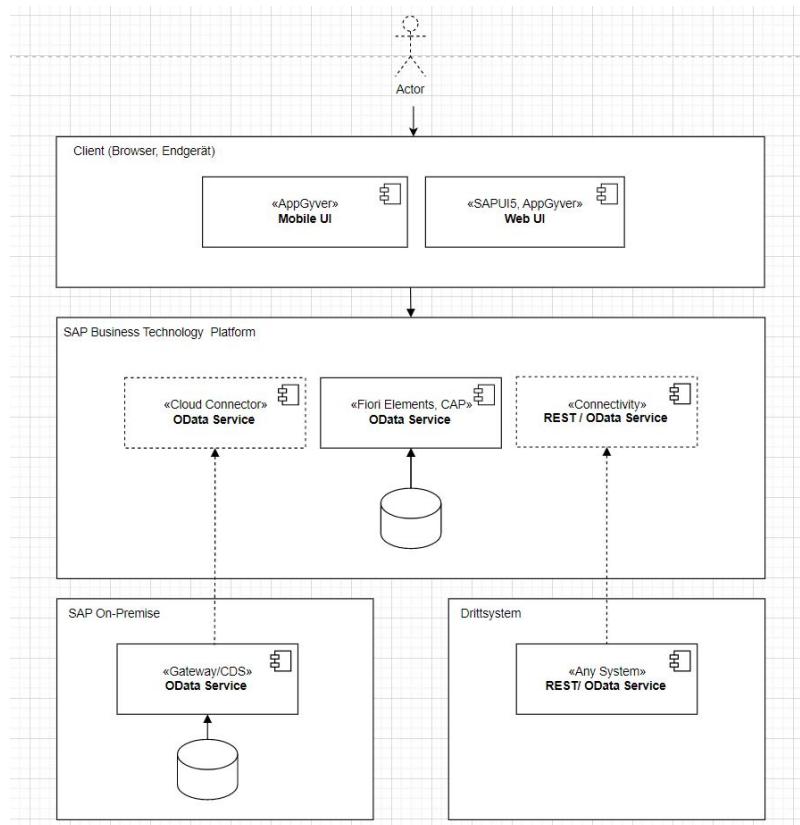


Abbildung 2.1: Architektur einer SAP-Anwendung in der SAP BTP

2.3 SAP Fiori

SAP Fiori wurde von der SAP als visuelle Leitlinie eingeführt, um User Interfaces über unterschiedliche Anwendungen hinweg zu standardisieren. Hinter dem Begriff verbergen sich jedoch ebenfalls technologische Aspekte, wie beispielsweise SAP Fiori Elements.

Das Grundkonzept von SAP Fiori ist es, Benutzeroberflächen so zu gestalten, dass Nutzer von Geschäftsanwendungen in ihrer täglichen Arbeit bestmöglich unterstützt werden, unabhängig davon, welche Endgeräte sie benutzen. Im Mit-

telpunkt stehen dabei Themen wie Usability, die Haptik und die User Experience der Anwendungen und folgende Grundsätze [Eng20, S.31]:

- Eine SAP-Fiori-App stellt dem Anwender nur die Funktionen zur Verfügung, die seiner Rolle entsprechen, sodass er nicht durch irrelevante Optionen abgelenkt wird.
- Mit SAP Fiori können Anwender sowohl auf mobilen Geräten als auch auf PCs arbeiten, wobei die Fiori-Anwendungen an das jeweilige Gerät angepasst werden müssen.
- SAP-Fiori-App statthen exakt die Funktionen des Anwendungsfalles aus. Die Funktionen, die nicht für Abarbeiten des Anwendungsfalles erforderlich sind, werden nicht in die Fiori-App ausgerichtet.
- SAP Fiori folgt einer einheitlichen Interaktion Designsprache und verfügt über ein standardisiertes Oberflächendesign.
- Die wesentlichen Funktionen der Fiori-Apps sollten für den Anwender intuitive bedienbar sein. Die Fiori-Apps sollen auch ansprechend sein [Eng20, S.34-35].

Der Fokus auf spezialisierte Applikationen macht es notwendig, diese zentral bereitzustellen. Das SAP Fiori Launchpad ist deswegen der zentrale Bereich, in welchem die SAP Fiori-Anwendungen zusammengeführt werden. Es stellt den Fiori-Apps Services wie Navigation und Anwendungskonfiguration zur Verfügung. Das Launchpad ist rollenbasiert und muss anwenderspezifisch angepasst werden. Die Rolle des Anwenders definiert somit, welche Fiori-App auf den Launchpad angezeigt werden [Gui22].

SAP Fiori als Design-Richtlinie wird im weiteren Verlauf nicht weiter betrachtet. Die Grundsätze finden sich jedoch in SAP Fiori Elements und auch in SAPUI5 wieder.

2.4 SAP AppGyver

2.4.1 Grundlage von AppGyver

AppGyver ist ein Pionier in der LCNC-Entwicklung. Das Unternehmen mit Hauptsitz in Helsinki wurde im Jahr 2010 gegründet und hat seit Gründung den Fokus auf der LCNC-Entwicklung [Lea22]. Mit Composer Pro stellt AppGyver eine zentrale Entwicklungsumgebung bereit, um Anwendungen für unterschiedliche Geschäftsprozesse und Anwendungsszenarien zu entwickeln, ohne eigenen Code zu schreiben. Diese Anwendungen können nicht nur als Webanwendungen, sondern auch als mobile Anwendungen eingesetzt werden. AppGyver unterstützt

sowohl iOS mit Bereitstellung der Anwendungen im App Store als auch Android Phone mit Bereitstellung in Google Play.

Im Februar 2021 wurde AppGyver von SAP übernommen und seitdem gibt es 2 Editionen: die Community Edition und die SAP Enterprise Edition. Die Community Edition basiert auf der initialen Version von AppGyver und bleibt zunächst unabhängig von SAP. Die Benutzer können es weiterhin kostenlos nutzen. Die SAP Enterprise Edition dagegen wird in das SAP-Ökosystem integriert und ist Bestandteil der SAP BTP. Zu den zusätzlichen Funktionen der Enterprise-Version zählen:

- Integration mit der SAP BTP Authentifizierung für Webanwendungen direkt in AppGyver.
- Erweiterte Integration von Daten aus anderen SAP-Systemen.
- Neue Enterprise-Funktionen, wie beispielsweise die Einführung einer Übersetzungsvariablen.
- Nutzer können das Projekt in Echtzeit mit anderen teilen

Am 15. November 2022, während der Anfertigung dieser Arbeit, erfolgte ein Rebranding von SAP AppGyver in SAP Build Apps. Zudem wird das Tool in Zukunft Teil einer Suite an Applikationen unter dem Label SAP Build sein, welche den Fokus auf die gesamtheitliche LCNC-Entwicklung von Anwendungen, die Automatisierung von Prozessen sowie das Design von Unternehmenswebsites legt. Neben SAP Build Apps sind auch Build Process Automation und Build Work Zone in SAP Build enthalten [Lea22]. Neben der reinen Integration, wird SAP Build App zudem in Zukunft um neue Funktionen erweitert, wie beispielsweise "Visual Cloud Functions", die die Speicherung von Daten in der Cloud und die Ausführung von Geschäftslogik ermöglichen [App22d]. Diese Erweiterungen werden jedoch im Rahmen dieser Thesis nicht weiter betrachtet.

2.4.2 Entwicklungsumgebung: Composer Pro

Eine Entwicklungsumgebung ist eine Zusammenstellung von Funktionen und Werkzeugen, die zur Entwicklung einer Anwendung notwendig sind. Werden diese gesamtheitlich und zentralisiert (via Internet) bereitgestellt, dann spricht man auch von einer Entwicklungsplattform. Composer Pro ist die zentrale Entwicklungsplattform von AppGyver. Dabei handelt es sich um eine spezialisierte LCNC-Plattform, mit der Anwendungen visuell und ohne Programmierung erstellt werden können. Der Aufbau der Plattform und die Funktionen sind dabei an die Zielgruppe, Citizen Developer ohne Programmiererfahrung, angepasst. Dennoch ist ein Verständnis des grundlegenden Aufbaus der Umgebung, sowie

der Prinzipien zur Entwicklung einer Anwendung notwendig.

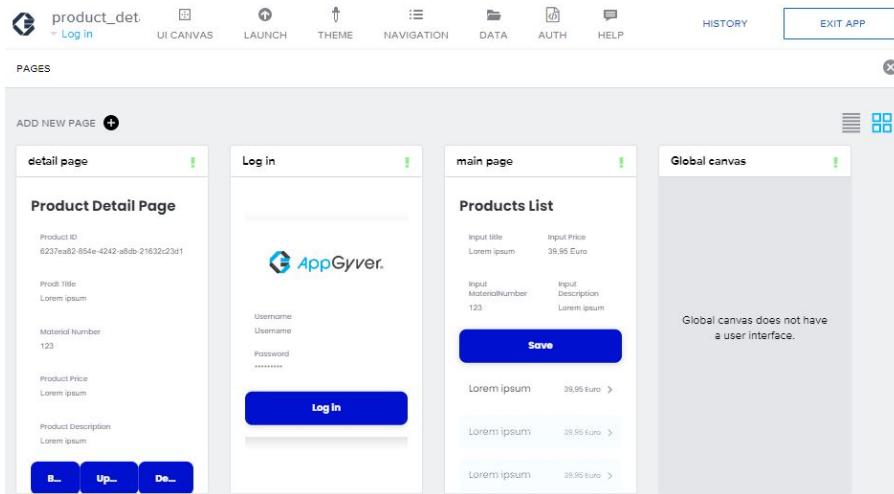


Abbildung 2.2: Pages einer AppGyver-Anwendung

Eine Anwendung in AppGyver ist grundsätzlich in mehrere Pages unterteilt. Jede Page besitzt einen eigenen Canvas, auf dem weitere Inhalte platziert werden können. Am oberen Rand der Benutzeroberfläche befindet sich die zentrale Toolbar, über die der Entwickler auf alle Ressourcen und Werkzeuge von AppGyver zugreifen kann.



Abbildung 2.3: Toolbar in Composer Pro

Dem LCNC-Ansatz folgend, finden sich in AppGyver keine programmatischen Bausteine (Code-Files, Klassen, etc.), sondern die diversen Funktionalitäten sind in eigenen, proprietären, Strukturen abgelegt. Dabei lassen sich diese grob in die Schichten des MVC-Modells einteilen. Das MVC-Paradigma strukturiert die Implementierung einer Anwendung in folgende drei Schichten:

- **M** steht für Model und repräsentiert das Datenmodell. Das Datenmodell stellt die relevanten Daten bereit.
- **V** bezieht sich auf View, d.h. die Präsentation. Diese Schicht ist zuständig für die Darstellung auf den Endgeräten und die Realisierung der Benutzerinteraktionen.
- **C** steht für Controller, also die Steuerung. Controller steuern und verwalten die Views. Der Controller kommuniziert mit dem Modell, wenn eine Benutzeraktion mit einer Datenänderung stattfindet.

| | |
|----------------------|---|
| Applikations-Schicht | AppGyver-Baustein |
| View | Page; View Component; Properties; Theme |
| Controller | Logic Flows; Formula Functions |
| Model | (Data) Variables; Data Resource |

Tabelle 2.1: AppGyver-Baustein in die Schichten des MVC-Models

View

Eine Anwendung in SAP AppGyver besteht aus mehreren Pages, d.h. mehreren eigenen Sichten. Diese werden aus vorgefertigten und konfigurierten View Components zusammengesetzt. Der Komponentenbibliothek in Composer Pro bietet einen Überblick über alle verfügbaren Komponenten. Diese sind in drei Registerkarten unterteilt. Unter CORE sind die Kernkomponenten verfügbar, die in den meisten Anwendungen verwendet werden. Dies sind beispielsweise Texte, Buttons oder Input-Felder. Unter BY ME sind die Komponenten aufgelistet, die der Entwickler selbst für diese Anwendung erstellt hat. Komponenten, die aus dem Marketplace für diese Anwendung hinzugefügt wurden, sind auf der Registerkarte *INSTALLED* zu finden [App22g]. Jede View Component besitzt spezifisch Eigenschaften (Properties), die sich in dem kontext-sensitiven Panels „Component Properties“ und „Style“ angepasst werden können. Der Layout-Tree, der sich unten in der rechten Seitenleiste befindet, zeigt die komplette Struktur der Komponenten in der Anwendung an und ermöglicht die direkte Auswahl. Weiterhin ist es möglich, einen grundlegenden Theme mit Styles bereitzustellen, der die grundlegenden Farben, Schriften, etc. für die Anwendung festlegt.

Model

AppGyver unterstützt in Bezug auf die Datenintegration zwei Szenarien: Es können Daten lokal im Projekt angelegt und abgespeichert werden. Diese Daten sind dann jedoch nur für die lokale Applikation gültig. Alternativ kann auf externe Datenendpunkte (REST, OData) zugegriffen werden. Die Daten werden dann über die externen Schnittstellen abgefragt und können in der Anwendung angezeigt werden. Zur Abbildung von Datenstrukturen und -flüssen gibt es in Composer Pro das Konzept der Variablen. Damit können ohne Programmierung verschiedene Arten von Strukturen festgelegt werden. „Page Variablen“ existieren nur für die aktuelle Seite und enthalten beliebige Werte, während „App Variablen“ über die

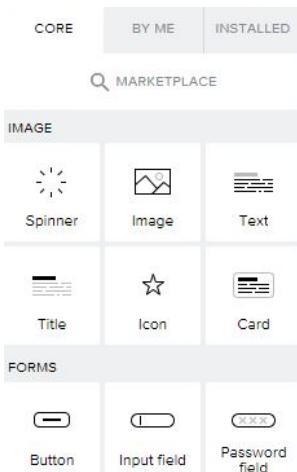


Abbildung 2.4: View-Components in Composer Pro

gesamte Anwendung hinweg existieren. „Data Variablen“ befinden sich ebenfalls nur auf der aktuellen Seite und beinhalten die Funktion, interne und externe Datenstrukturen zu mappen. Diese Variablen können an die spezifischen Eigenschaften (Properties) einer View Component gebunden werden (Data-Binding) und stellen somit das Bindeglied zwischen Model und View dar.

Weitere Variablen zur Ablage von Werten sind „Page Parameter“ (schreibgeschützte Textvariablen), die zur Übertragung von Daten zwischen den Seiten verwendet werden können und „Translation Variablen“, die für sprachabhängige Texte verwendet werden können.

Controller

Neben den Daten ist auch die Geschäftslogik ohne oder mit geringen Programmierkenntnissen umsetzbar. SAP AppGyver stellt hier Logische Ablauffunktionen bereit, mit denen per Drag&Drop, sowie einer Konfiguration, komplexere Prozesse definiert werden können. Der Logic-Canvas befindet sich in der unteren Hälfte der Benutzeroberfläche. Für jede Komponente gibt es einen eigenen Logic-Canvas-Kontext, der eine spezifische Konfiguration erlaubt. Damit auf Nutzereingaben reagiert werden kann, stellen die View Components Events zur Verfügung. Diese werden immer dann geworfen, wenn eine spezifische Interaktion auftritt.

Weiterhin existieren in AppGyver Formeln, die dazu genutzt werden können, um komplexere Logiken abzubilden. Dazu gibt es einen eigenen Formel-Editor, in dem die Logiken hinterlegt werden können. Tatsächlich stehen

dort Formel-Typen zur Verfügung, die sich nahe an der richtigen Programmierung bewegen (Logische Operatoren, IF-Statements, etc) [App22b].

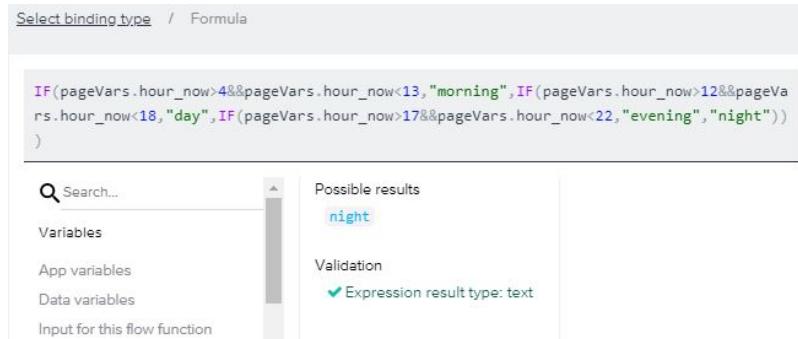


Abbildung 2.5: Exemplarische Formel in AppGyver

2.5 SAPUI5

2.5.1 Grundlage von SAPUI5

SAP Fiori beschreibt als Leitfaden die Entwicklungsrichtlinie zur Erstellung von Anwendungen im SAP-Umfeld. Für die Umsetzung sind jedoch auch technische Bausteine notwendig. Mit SAP WebDynpro, der vormals führenden UI-Technologie, ist SAP Fiori aber schwer umzusetzen, da der HTML-Code auf dem Server generiert und dann an das Endgerät übermittelt wird. In diesen Kontext ist die Entwicklung eines clientseitigen Ansatzes erforderlich. Bei einem clientseitigen Ansatz steht der Frontend-Server im Mittelpunkt der Kommunikation und lädt das UI-Framework, das die weiteren Verarbeitungsschritte übernimmt [Eng20, S.45-47].

SAPUI5 (kurz für: SAP UI Development Toolkit für HTML5) ist ein JavaScript-basiertes clientseitiges Framework und eine UI-Bibliothek, mit der SAP Fiori-Anwendungen sehr flexibel entwickelt und auf verschiedene Plattformen portiert werden können. Die SAPUI5-Bibliothek basiert auf modernen Web-Standards, wie JavaScript, HTML5 und CSS3 [Ant16, S.139]. SAPUI5 verfügt über mehr als 500 UI-Controls, die an den neuesten SAP Fiori Design Guidelines ausgerichtet sind, und bietet integrierte Unterstützung für Enterprise-Funktionen, wie Datenbindung, Routing, Message Handling, Mehrsprachigkeit, etc. Heute ist SAPUI5 der Standard für die Implementierung von Frontend-Anwendungen im SAP-Umfeld [Com22a].

Da es sich bei SAPUI5 nicht um eine LCNC-Technologie handelt, ist ein grundlegendes Programmierverständnis in der Sprache JavaScript notwendig. Zudem setzt SAPUI5 das Verständnis einiger Entwurfsmuster voraus. Das MVC-Paradigma beispielsweise strukturiert die Implementierung von SAPUI5-Anwendungen in drei Schichten, die sich so auch im Quellcode wiederfinden.

- **Model:** Hier stehen spezielle Klassen zur Verfügung, die den Zugriff auf unterschiedliche Arten von Daten abstrahieren (JSONModel, ODataModel, XMLModel).
- **View:** Die View beinhaltet den Aufbau des User Interfaces und SAPUI5 stellt hier UI Controls zur Verfügung, dafür genutzt werden. Diese lassen sich via JavaScript, HTML oder XML definieren und konfigurieren.
- **Controller:** Zu jeder View gibt es in SAPUI5 einen Controller, der eine Benutzeraktion über Events und zugehörige Callback Implementierungen steuert, sowie komplexere Geschäftslogik enthalten kann [Ant16, S.149].

Da es sich bei SAPUI5 um ein komplexeres Framework handelt, können an dieser Stelle nicht alle Facetten im Detail erklärt und beschrieben werden. In Kapitel 3.3 werden ihm Rahmen der Implementierung jedoch einige genutzte Dinge vorgestellt.

2.5.2 Entwicklungsumgebung: Visual Studio Code

Durch den freien Programmieransatz ist SAPUI5 nicht an eine Entwicklungsumgebung/plattform gebunden. Im Rahmen dieser Thesis wird Visual Studio Code (kurz: VS-Code) für die Umsetzung verwendet. VS-Code ist ein Quellcode-Editor von Microsoft, der im Jahr 2015 für verschiedene Betriebssysteme wie Windows, MacOS und Linux veröffentlicht wurde. Er unterstützt diverse Programmiersprachen und Frameworks (z.B. JavaScript, TypeScript und Node.js), kann jedoch über das Erweiterungskonzept um weitere Programmiersprachen, Laufzeiten und Funktionen ergänzt werden [Cod22a]. VS-Code ist heute aufgrund der ausgereiften Funktionen und der guten Bedienung ein Standard in der Entwicklung von Web-Anwendungen [Wik22c] und wird deswegen an dieser Stelle potenziellen anderen Entwicklungsumgebungen vorgezogen.

Der Arbeitsbereich des VS-Code besteht aus ein oder mehreren Verzeich-

nissen. Es vereinfacht die sprachübergreifende Anwendungsentwicklung in einer integrierten Entwicklungsumgebung. VS-Code integriert ein voll funktionsfähiges Terminal mit dem Editor, um Shell Skript und Kommanden durchzuführen. Das integrierte Terminal kann verschiedene Shells verwenden, die auf dem Rechner installiert sind [Cod22b].

Die lokale Implementierung der SAPUI5-Anwendung in VS-Code erfordert die zusätzliche Installation von weiteren Bibliotheken: das SAP Cloud Application Programming Model (kurz: SAP CAP) und des Easy UI5 Generators. Das SAP CAP ist ein Framework zur Erstellung von Backend-Anwendungen und kommt auch bei Fiori Elements zum Einsatz [SAP22a]. Mit Hilfe von Core Data Services als die universelle Modellierungssprache für Domänenmodelle und Servicedefinitionen, können in sehr kurzer Zeit Datenmodelle generiert und als OData-Service bereitgestellt werden [SAP22a]. Der Easy UI5 Generator enthält Vorlagen zur Erstellung einer SAPUI5-Anwendung mit aktuellen Best Practices. So lassen sich bereits die grundlegenden Strukturen einer Anwendung erstellen, auf deren Basis die Entwicklung dann stattfinden kann [SAP22c]

Um die UI5-Anwendung zu implementieren, muss die Entwicklungsumgebung wie folgt eingerichtet werden:

- Herunterladen und Installieren des aktuellen Node.js Installationspaket von <https://nodejs.org/en/download/> inklusive Runtime und Package Manager (npm). Node.js bietet eine asynchrone, ereignisgesteuerte Laufzeitumgebung für Javascript und wird für das lokale Betreiben eines Webservers verwendet [Wik22a].
- Installieren von yeoman und dem Easy-ui5 Generator. Yeoman ist ein Kommandozeilen-Scaffolding-Tool für Node.js, um das Gerüst für die weitere Entwicklung der SAPUI5-Anwendung zu generieren
- Installieren der SAP Cloud Application Programming Model-Module (@sap/cds-dk und @sap/cds).
- Herunterladen und Installieren der SQLite-Datenbank-Treiber. Als ein leicht eingebettetes Datenbanksystem, lässt sich SQLite direkt in entsprechende Anwendungen integrieren, ohne keine weitere Server-Software [Wik22b].
- Installation der VS Code Erweiterungen für SAP Fiori und SAP CAP CDS, zur Unterstützung bei der Entwicklung.

Das cds-dk und SQLite werden benötigt, um bei der lokalen Entwicklung

einen OData Mock Service zu erzeugen. Die VS-Code-Erweiterungen bieten Sprachunterstützung für die Core Data Services (CDS), welche die Grundlage des SAP Cloud Application Programming Model (CAP) bilden, sowie für SAPUI5 [Mar22].

2.6 Fiori Elements

2.6.1 Grundlage von Fiori Elements

SAP Fiori als gestalterische Richtlinie lässt sich mit SAPUI5 technisch umsetzen. Allerdings ist dort immer Programmieraufwand nötig. Als LCNC-Ansatz stellt die SAP jedoch mit Fiori Elements eine Technologie zur Verfügung, mit der UI Controls und sogar komplette Applikationen automatisch auf Basis von Metadaten zur Laufzeit generiert werden können. Die Metadaten werden dabei via OData-Annotationen beschrieben, die vom Backend-Service bereitgestellt werden müssen [Eng20, S.48]. SAP Fiori Elements basiert technologisch auf SAPUI5 und erweitert dieses um intelligente Komponenten und Views. SAP Fiori Elements umfasst sogenannte Floorplans, d.h. Grundrisse und UI-Patterns für gängige Anwendungsfälle. Fiori Elements verfügt über die folgenden vier grundlegenden Floorplans:

- List Report und Object Page

Ein List Report wird verwendet, wenn Elemente in einer Tabelle oder Liste dargestellt werden sollen. Mit dem List Report können die Objekte angezeigt, gefiltert und bearbeitet werden. Der List Report wird in der Regel in Verbindung mit der Objekt Page verwendet. Auf die Objekt Page kann der Nutzer durch Klicken auf ein einzelnes Element in der Liste gelangen und dort detaillierte Informationen über einzelne Objekte angezeigt bekommen und es bearbeiten.

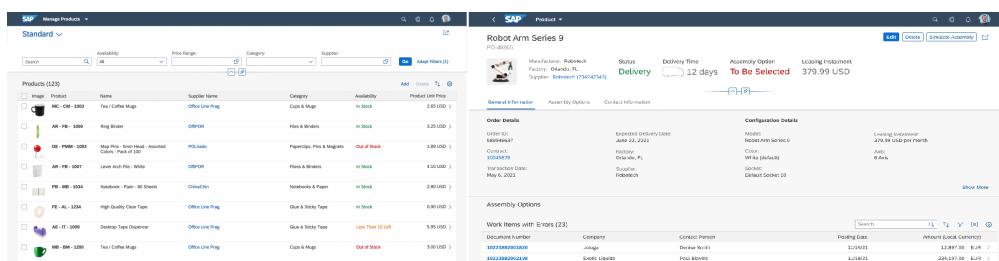


Abbildung 2.6: List Report und Object Page (Quelle: SAP Fiori Design Guidelines. URL: <https://experience.sap.com/fiori-design-web/smart-templates>)

- Worklist

Eine Worklist zeigt ebenfalls eine Liste von Elementen an, die von Benutzer bearbeitet werden sollen. In einer Worklist ist jedoch keine komplexe Filterung möglich und die Anzeige, bzw. das Editieren erfolgt ausschließlich in der Worklist-Ansicht [Doc22f].

- Overview Page

Ein Overview Page ermöglicht es, den Nutzern eine große Menge unterschiedlicher Informationen für einen Überblick bereitzustellen. Unterschiedliche Informationen werden in verschiedenen Cards visualisiert. Eine Card zeigt die Details zu einem bestimmten Geschäftsobjekt. Mit der Overview Page können Anzeigen, Filtern und Verarbeiten von Daten einfach und effizient gemacht werden.

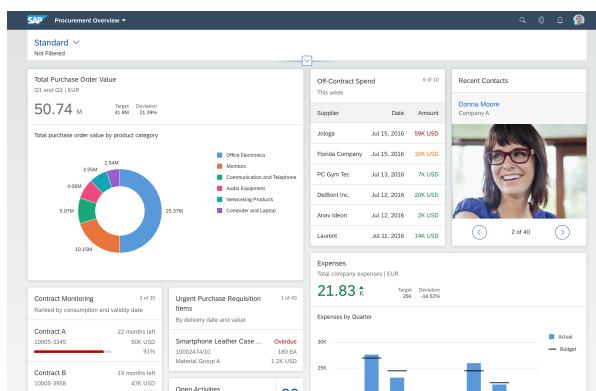


Abbildung 2.7: Overview Page (Quelle: SAP Fiori Design Guidelines. URL: <https://experience.sap.com/fiori-design-web/smart-templates>)

- Analytical List Page

Die Analytical List Page ist die Weiterentwicklung von List Report und verfügt über mehr Funktionen: Daten können in verschiedene Perspektiven analysiert werden, z.B. durch Drill-Downs zur Ursachenforschung [Doc22f].

Mit Hilfe von Extension Points in den Floor-Plans können in einer Fiori Element-Anwendung zusätzliche Funktionalitäten hinzugefügt werden. Dafür ist jedoch immer eine Programmierung erforderlich. SAP Fiori Elements ohne Extensions benötigt keinen Programmieraufwand, setzt jedoch einen OData-Service und eine Konfiguration der Annotationen voraus. Hierfür stehen potentiell unterschiedliche Technologien zur Verfügung. Dank der SAP eigenen Entwicklungsumgebung, SAP Business Application Studio,

können jedoch komplett SAP Fiori Elements-Anwendungen inklusive der Definition von Datenstrukturen und OData-Services und Annotationen visuell aufgebaut werden [Doc22e].

2.6.2 Entwicklungsumgebung: Business Application Studio

Grundsätzlich kann SAP Fiori Elements mit verschiedenen Entwicklungsumgebungen erstellt werden. Durch die sehr gute Integration und Bereitstellung einer LCNC-Umgebung, eignet sich das SAP Business Application Studio (BAS) jedoch sehr gut für das Erstellen einer Anwendung im Kontext dieser Thesis. BAS ist ein Service der SAP Business Technology Platform (SAP BTP), der seit Februar 2020 als Nachfolger der SAP Web IDE zur Verfügung steht. SAP BAS ist ein Cloud-basiertes Werkzeug, das nicht lokal installiert werden kann und im Browser des Nutzers ausgeführt wird und entspricht damit den Kriterien einer Entwicklungsplattform [Por22d].

Das Business Application Studio lässt sich via Dev Spaces für unterschiedliche Anwendungsarten konfigurieren. Die Entwicklungsszenarien umfassen zum Beispiel SAP Fiori, SAP S/4HANA Erweiterungen, Workflows und SAP HANA-Anwendungen. In jedem Dev Space werden je nach Auswahl eine Reihe von vordefinierten Erweiterungen installiert, die spezialisierte Funktionen bereitstellen [Por22d].

Neben der klassischen Code-basierten Entwicklung, stellt BAS auch die Möglichkeit bereit, „Low-Code basierte Full-Stack Anwendungen“ zu entwickeln. Dafür sind dann diverse Wizards und UI-Masken vorhanden, die im Hintergrund automatisch den notwendigen Quellcode interpretieren und erstellen. Eine Full-Stack-Anwendung besteht dabei aus dem Datenmodell und OData-Service des SAP Cloud Application Programming Models und SAP Fiori Elements als Frontend-Technologie. Alle notwendigen Parameter lassen sich via UI konfigurieren.

Im Rahmen dieser Arbeit wird ein Dev Space erstellt, der als Entwicklungsumgebung für die Low-Code-basierte Full-Stack-Anwendung ausgewählt wurde, um den in Kapitel 1, Abschnitt 1.3 beschriebenen Anwendungsfall mit Fiori-Elements zu entwickeln.

Kapitel 3

Implementierung des Anwendungsfalls

In Abschnitt 1.3 wurden die Anforderungen an die zu implementierende Anwendung bereits vorgestellt, in diesem Kapitel wird nun die Umsetzung in den einzelnen Technologien (Fiori Elements in Kombination mit SAP CAP, AppGyver und SAPUI5) näher beschrieben.

| Technologie | Frontend | Backend |
|--------------------------------|----------|---------|
| AppGyver | X | - |
| SAPUI5 | X | - |
| Fiori Elements (inkl. SAP CAP) | X | X |

Tabelle 3.1: Umsetzungsanforderung der Technologie

Dabei ist zu unterscheiden zwischen Backend- und Frontend-Funktionalität. SAPUI5 verfügt grundsätzlich über keine Backend-Funktionalität. Mit SAP AppGyver können lokale Datenstrukturen erzeugt werden. Diese Funktion wird jedoch nicht genutzt, sondern ein zentraler OData-Service inklusive Datenbackend an anderer Stelle bereitgestellt. Hierfür wird auf die „Full Stack“ -Anwendungen, bestehend aus Fiori Elements und SAP CAP, im SAP Application Studio zurückgegriffen. Das Fiori Elements-Frontend wird den erstellten OData-Service dann konsumieren. Das gleiche Datenmodell und der gleiche OData-Service werden dann ebenfalls von SAP AppGyver und SAPUI5 verwendet.

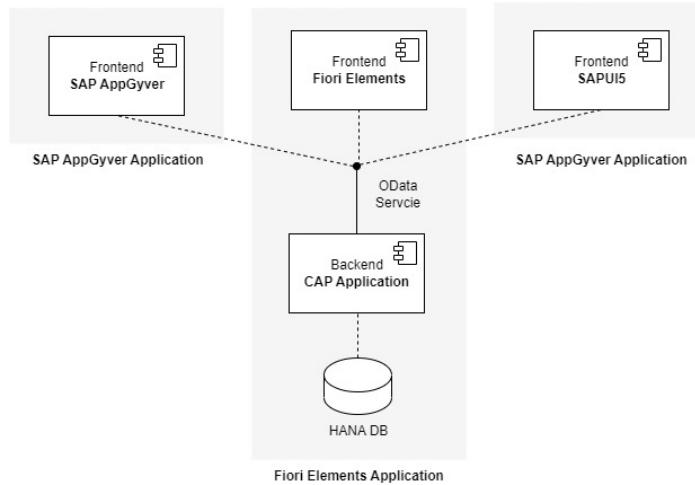


Abbildung 3.1: Umsetzungsarchitektur der Technologie

3.1 Fiori Elements

3.1.1 Dev Space erstellen

Für die Entwicklung mit Fiori Elements wird das Business Application Studio als Entwicklungsumgebung genutzt. Wie bereits in Abschnitt 2.6.2 erwähnt, stellt das BAS Dev Spaces für unterschiedliche Entwicklungsszenarien bereit. Die komplette Anwendung ließe sich auch als klassisches Entwicklungsprojekt auf- und umsetzen, im Sinne der Arbeit wird jedoch das “Low-Code-Based Full-Stack Cloud-Application” Preset für den Dev Space gewählt, um Zugriff auf die LCNC-Funktionalitäten zu bekommen.

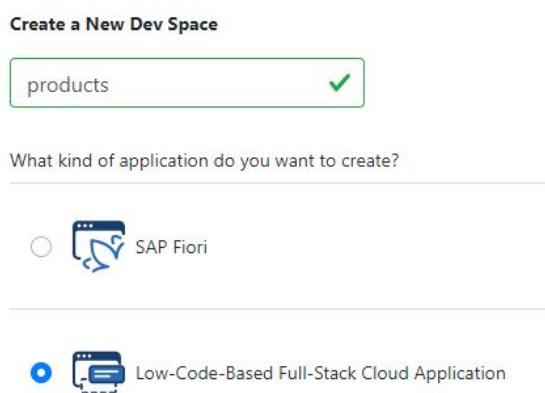


Abbildung 3.2: Create a New Dev Space

Nach der Erstellung des Dev Space wird die Entwicklungsumgebung für eine Low-Code-basierte Full-Stack Cloud-Anwendung bereitgestellt. In

dieser Umgebung stehen verschiedene „Cards“ zur Verfügung, um den auf Low-Code basierenden Implementierungsprozess durchzuführen.

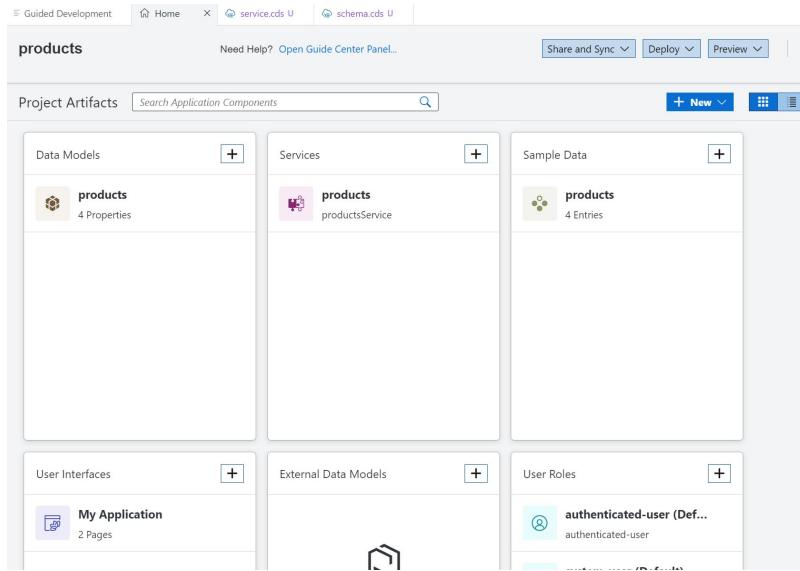


Abbildung 3.3: Grundlegenden Aufbau inkl. der Cards von BAS

3.1.2 Datenentitäten und Service definieren

Für den Anwendungsfall wird eine Datenentität mit sechs Properties festgelegt. Neue Entitäten können einfach in der Datenmodellkarte hinzugefügt und konfiguriert werden. Der Entitätsname wird als „Products“ definiert und die sechs Properties sind ID, Title, MaterialNumber, Description, Price und Stock. Um die Implementierung des Anwendungsfalls zu erleichtern, wird die ID hier als UUID und alle anderen Properties als String definiert.

| Products | | |
|----------|----------------|--------|
| | ID | UUID |
| Ab | Title | String |
| Ab | MaterialNumber | String |
| Ab | Description | String |
| Ab | Price | String |
| Ab | Stock | String |

Abbildung 3.4: Datenmodell

Beim Anlegen der Entität über die UI wird im Hintergrund eine Code-Datei erzeugt (model.cds) und die Definition dort abgelegt. Dies ist im LCNC-

Modus für den Entwickler nicht sichtbar und er muss sich um das Coding nicht weiter kümmern. Grundsätzlich ist es aber möglich, das Coding manuell anzupassen und die UI reagiert entsprechend auf die Änderungen im Coding. Es ist jedoch herauszustellen, dass die Wizards und UI-Masken von BAS nicht alle Funktionalitäten von SAP CAP unterstützen und so spezielle Dinge ggf. nur über das Coding hinzugefügt werden können. Für den gewählten Use-Case ist dies jedoch nicht von Belang.

Angelegte Datenbank-Entitäten können im Weiteren einfach als OData-Service exponiert werden. Ähnlich wie beim Datenmodell, gibt es dafür eine eigene Karte. Dort wird lediglich die Entität ausgewählt, Name, Namespace und Typ festgelegt, sowie einige weitere Properties konfiguriert und dann kann der Service erstellt werden.

| | ID | UUID |
|----|----------------|--------|
| Ab | Title | String |
| Ab | MaterialNumber | String |
| Ab | Description | String |
| Ab | Price | String |
| Ab | Stock | String |

Abbildung 3.5: Service

Unter der Sample-Data-Karte können Mock-Daten hinterlegt werden, damit man während der Entwicklung direkt Zugriff auf Daten hat. Da die Property-ID als UUID definiert ist, werden die IDs automatisch vom System vergeben. Die anderen Properties wie Title, MaterialNumber, Description, Price und Stock müssen manuell eingetragen werden.

Tatsächlich hat man durch die Verwendung der drei Cards mit diesen wenigen Klicks und Eingaben in sehr kurzer Zeit ein eigenes Datenmodell und einen OData-Service definiert, den man in der Preview im BAS auch direkt aufrufen und testen kann.

3.1.3 User Interface erstellen

Die Anforderungen an die Benutzeroberfläche bestehen darin, dass eine Listenansicht für die Anzeige aller Produkte und eine Einzelansicht für ein

| Products(15) | | | | | | | Mock Data: AN |
|--------------------------|-----|-----------------------|--------------------------|-------------|---------------------------|------------|-------------------------------|
| | ID* | TITLE | MATERIALNUMBER | DESCRIPTION | PRICE | STOCK | |
| <input type="checkbox"/> | 1 | 0165cdd0-f1aa-4c62... | Carpendo car seat co... | 252 | Black front seats and... | 40.19 Euro | STOCK2786 |
| <input type="checkbox"/> | 2 | f3e89493-d10a-4d32... | Carpendo Auto Spor... | 338 | Seat covers black-gr... | 46.99 Euro | STOCK5817 |
| <input type="checkbox"/> | 3 | b1200d4e-e514-47ef... | Woltu car seat covers | 111 | Black, with "Super D... | 26.99 Euro | STOCK9274 |
| <input type="checkbox"/> | 4 | e209033f-5993-4113... | Walser Comfort Car ... | 760 | Car seat cover S-Rac... | 16.95 Euro | STOCK256 |
| <input type="checkbox"/> | 5 | c27cb5cb-3ad2-42ec... | Walser car seat cover... | 949 | Universal seat cover ... | 74.95 Euro | STOCK4803 |
| <input type="checkbox"/> | 6 | f798ef12-f5a2-480b... | Flying Banner Car Se... | 872 | Universal Set with Ai... | 29.99 Euro | STOCK3350 |
| <input type="checkbox"/> | 7 | 87de15d1-221b-432f... | Flying Banner PVC C... | 848 | Complete set with ai... | 34.39 Euro | STOCK5088 |
| <input type="checkbox"/> | 8 | a31f73e0-be2c-47d3... | Flying Banner Univer... | 757 | Complete Cover Brea... | 34.99 Euro | STOCK5150 |
| <input type="checkbox"/> | 9 | 0094daed-09f7-4557... | Upgrade4Cars car se... | 641 | Seat covers complet... | 39.95 Euro | STOCK961 |
| <input type="checkbox"/> | 10 | 071d0a34-c4e4-427... | Upgrade4Cars car se... | 110 | Car seat cover for sp... | 13.95 Euro | STOCK778 |
| <input type="checkbox"/> | 11 | 31901267-a50f-49b4... | Fixcap PRO car seat... | 904 | Car seat cover for fro... | 15.90 Euro | STOCK4737 |
| <input type="checkbox"/> | 12 | 22407997-be2a-40b... | Fixcape Neoprene C... | 919 | Car seat cover for th... | 29.50 Euro | STOCK1005 |
| <input type="checkbox"/> | 13 | 658b32a1-2403-4a6... | Leader Accessories u... | 567 | Imitation Leather Car... | 24.99 Euro | STOCK9967 |

Abbildung 3.6: Sample Data

einzelnes Produkt, sowie eine Maske für die Pflege jedes einzelnen Produkts entworfen werden sollen. SAP Fiori Elements stellt hierfür bereits komplett vorgefertigte Floorplans zur Verfügung, die nahtlos in BAS integriert sind. Unter der User-Interfaces-Karte kann die Benutzeroberfläche der Anwendung definiert werden. Dies erfolgt in vier Schritten:

1. Eingabe der Details der UI-Anwendung wie Name und Namespace.
2. Auswahl des Anwendungstyps. In dem hier vorgestellten Anwendungsfall wird eine *Template-Based, Responsive Application* ausgesucht. Es wäre jedoch auch möglich, eine Freestyle SAPUI5-Anwendung zu erstellen.
3. Wie in Abschnitt 2.6.1 erwähnt, werden von SAP verschiedene Arten von Floorplans für Fiori Elements bereitgestellt. In diesem Fall wird *List Report Object Page* für UI Application Template verwendet.
4. Der letzte Schritt bei der Erstellung einer UI-Anwendung ist die Auswahl des richtigen Datenobjekts. Hier wird die Datenentität *Products* gewählt und es werden automatisch Tabellenspalten zur Listenseite und ein Abschnitt zur Objektseite eingeschaltet.

Nach dem Klick auf den „Finish“ -Button wird die UI-Application automatisch generiert. Auch hier wird im Hintergrund Quellcode abgelegt: Zum einen werden die OData-Annotationen in weiteren cds-Dateien abgelegt. Das Weiteren wird das Grundgerüst einer SAPUI5-Anwendung angelegt und mit der entsprechenden Konfiguration für SAP Fiori Elements ver-

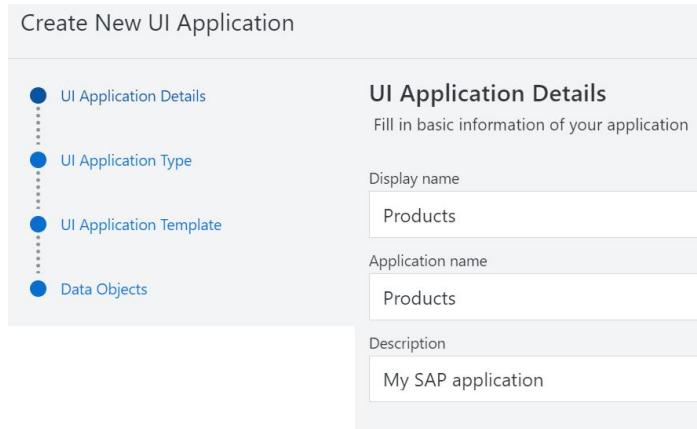


Abbildung 3.7: UI-Konfiguration

sehen. Auch dies ist für den Entwickler nicht direkt in BAS einsehbar, bzw. es ist nicht notwendig sich mit dem Code auseinander zu setzen, es sei denn, man möchte gezielt Funktionen nutzen, die von den Wizards in BAS nicht unterstützt werden. Die finale Anwendung besteht aus einer Listenseite und einer Detailseite für jedes Produkt. Die Listenseite zeigt eine Liste aller Produkte. Wird auf ein Listenelement geklickt, so wird die Detailseite für das jeweilige Produkt geöffnet. Zudem kann ein neues Produkt hinzugefügt, ein Produkt gelöscht oder bearbeitet werden. Während man über die Konfiguration einen Einfluss auf die Tabellenspalten und Inhalte der Einzelansicht nehmen kann, funktionieren Dinge wie Navigation, Daten laden, Datenbindung und auch das Speichern von neuen Datensätzen einfach so, ohne, dass vom Entwickler irgendeine komplexere Konfiguration vorgenommen werden muss. Allerdings kann man von den dem Standardverhalten auch nicht abweichen und jede Anwendung folgt dem grundlegenden Aufbau und Verhalten des ausgewählten Floorplans. Mithilfe der Page Map in BAS kann die UI-Anwendung angepasst werden. Zum Beispiel können für die hier vorgestellte Anwendung das initiale Laden aktiviert werden und auch die Properties, die in der Liste angezeigt werden, können je nach Anforderung geändert werden. Dabei sind alle Parameter über UI-Masken konfigurierbar und an keiner Stelle muss ein eigenes Coding erfolgen.

3.1.4 Deployment des OData-Services

Um auf den OData-Service auch aus den anderen Applikationen zuzugreifen und die UI-Anwendung bereitzustellen, muss ein Deployment aus

The screenshot displays three pages of a Fiori Elements application:

- List Page:** Shows a table with columns: Title, MaterialNumber, Description, Price, and Stock. Three items are listed:

| Title | MaterialNumber | Description | Price | Stock |
|-----------------------------------|----------------|---|------------|-----------|
| Upgraded4Cars car seat covers set | 641 | Seat covers complete set for the front seats & back seat with zipper and airbag opening. Black and Gray | 39,95 Euro | STOCK961 |
| Carpendo car seat covers | 252 | Black front seats and rear seats with airbag system, not for vehicles with sport seats and integrated headrests | 40,19 Euro | STOCK2786 |
| Upgraded4Cars car seat covers | 110 | Car seat cover for sports work workshop etc. car seat covers front seats universal, black | 13,95 Euro | STOCK778 |
- Detail Page:** Shows "General Information" for the item with MaterialNumber 641. It includes fields for Title, Description, Stock, MaterialNumber, and Price.
- New Object Page:** A form titled "Neues Objekt" for creating a new object. It has sections for "General Information" and "Material Number". Fields include Title, Description, Stock, MaterialNumber, and Price. Buttons at the bottom include "Anlegen" and "Entwurf verwerfen".

Abbildung 3.8: Listenseite, Detailseite und „neues Objekt anlegen- Seite der Fiori-Elements-Anwendung

BAS heraus erfolgen. Das Deployment erfolgt direkt auf die SAP BTP. Hierfür muss ein sogenannter „Space“ auf der SAP Business Technology Plattform vorliegen, sowie eine SAP HANA Cloud-Instanz bereitgestellt werden. Nachdem die Anwendungsbenutzeroberfläche in Business Application Studio erstellt worden ist, kann diese direkt aus BAS heraus deployed werden, wiederum ohne notwendiges Coding. Dazu muss man in BAS bei Cloud Foundry anmelden und das Cloud Foundry-Ziel angeben. Abbildung 3.9 zeigt, wie dies geht.

Danach lässt sich das Deployment starten und es werden während des Vorgangs alle notwendigen Komponenten (Applikation und notwendige BTP-Servie-Instanzen) erstellt. Nach dem Deployment stehen OData-Service, sowie UI-Applikation dann unter einer URL zur Verfügung. Auch das Deployment erfolgt vollständig als No-Code-Ansatz. Die notwendige Konfigurationsdatei (mta.yml) wird im Hintergrund des Projekts abgelegt und beim Deployment ausgelesen. Während der Implementierung gab es jedoch gelegentlich Probleme bei dem Deployment, sodass eine tiefergehende Analyse erforderlich wurde. Insbesondere in solchen Fällen musste die LCNC-Umgebung verlassen werden, damit in den Log-Dateien der Fehler gefunden werden konnte. Zudem waren die Probleme meist sehr technischer

The screenshot displays the 'Cloud Foundry Sign In and Targets' interface. On the left, the 'Cloud Foundry Sign In' section contains fields for the Cloud Foundry Endpoint (set to https://api.cf.eu20.hana.ondemand.com), authentication method (set to Credentials), and user credentials (username fangfang.tan@p36.io and password). On the right, the 'Cloud Foundry Target' section contains fields for the Cloud Foundry Organization (set to 7458bcbtrial) and Space (set to dev), with an 'Apply' button.

Abbildung 3.9: Cloud Foundry Sign In und Targets

Natur und erforderten ein tiefgreifenderes Verständnis der verwendeten Technologien (SAP CAP, SAP BTP).

3.2 AppGyver

3.2.1 Projektaufbau

Für die Umsetzung des Projekts wird SAP AppGyver Enterprise auf der SAP BTP verwendet. Die notwendigen Services können einfach in der SAP BTP aktiviert werden und nach Zuweisung der entsprechenden Rollen steht die Composer Pro-Entwicklungsplattform bereit.

3.2.2 OData Integration

Seit dem Kauf durch SAP wird SAP AppGyver immer mehr in das Ökosystem der SAP integriert. So ist es in der Enterprise-Edition mittlerweile möglich, Funktionen der SAP BTP zu nutzen, um Daten in AppGyver bereitzustellen. Um die Daten des OData-Services in AppGyver nutzen zu können, muss eine „Destination“ im SAP Business Technology Plattform Cockpit angelegt werden. Der SAP BTP Connectivity Service stellt via Destinations Reverse Proxy- und Authentifizierungsfunktionen zur Verfügung, sodass unter Verwendung der Destination ein einfacherer Zugriff auf einen Remote-Endpunkt erfolgen kann. Die Verbindungsdetails können via Eingabemaske gepflegt werden [Por22c]. Anzugeben sind Name, eine URL, Authentifizierungsdetails sowie weitere spezifische Konfigurationsparameter. Die Destination wird hier *products-on-trial_simple* genannt. Die URL

bezieht sich auf die OData-Service-Adresse. Die Authentifizierung wird dann als „OAuth Client Credentials Authentication“ definiert. Wichtig ist, dass die „Additional Properties“ gesetzt werden: *AppgyverEnabled* sowie *HTML5.DynamicDestination* müssen auf *true* gesetzt werden, damit der OData-Service in AppGyver eingebunden werden kann.

Destination Configuration

| | | | |
|------------------------------|---|--|------|
| Name:* | products-on-trial_simple | Additional Properties | |
| Type: | HTTP | AppgyverEnabl... | true |
| Description: | | HTML5.Dynam... | true |
| URL:* | https://7458bcbtrial-dev-products-srv.cfapps.us10.hana... | <input checked="" type="checkbox"/> Use default JDK truststore | |
| Proxy Type: | Internet | | |
| Authentication: | OAuth2ClientCredentials | | |
| Use mTLS for token retrieval | <input type="checkbox"/> | | |
| Client ID:* | sb-Products-dev!t97003 | | |
| Client Secret: | ***** | | |
| Token Service URL Type:* | Dedicated | Common | |
| Token Service URL:* | https://7458bcbtrial.authentication.us10.hana.ondeman... | | |
| Token Service User: | | | |
| Token Service Password: | | | |

Abbildung 3.10: Destination-Konfiguration in SAP BTP Cockpit

Nachdem die Destination erstellt wurde, ist der Service *products-on-trial_simple* in SAP AppGyver unter dem Menüpunkt Data – SAP Systems verfügbar und kann importiert werden. AppGyver verfügt über eine Funktionalität zum Auslesen der Metadaten des OData-Services und listet die vorhandenen Entitäten in der UI auf. Die Entität „Products“ kann dann einfach aktiviert werden und steht dann in der Anwendung zur Verfügung. Bereits im Integrations-Wizard lassen sich Dateninhalte und Werte des OData-Services ansehen, bzw. sogar verändern.

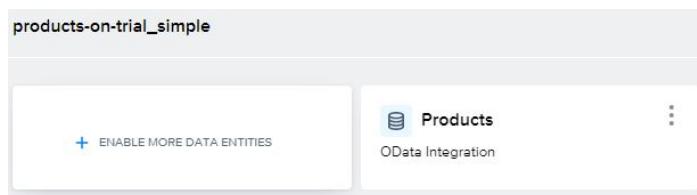


Abbildung 3.11: OData-Integration in AppGyver

3.2.3 Listenansicht zur Anzeige aller Produkte

Um eine Liste aller Produkte anzeigen zu können, muss zunächst eine Listenseite mit dem Namen „The List of Products“ erstellt werden. Diese Seite besteht aus vier Elementen: Einem Titel mit dem Text „Products“, einem Button zum Erstellen eines neuen Produkt-Datensatzes und einem

weiteren Button zum Einblenden eines Suchfeldes, sowie einem List Item. Die Erstellung eines Suchfeldes wird in Kapitel 4 näher betrachtet werden. Eine Data-Variable mit dem Namen „Product1“ wird erstellt, um die Daten von OData-Service abzurufen. Der Typ der Data-Variablen ist auf „Collection of data records“ eingestellt, wodurch eine Sammlung von Datensätzen geliefert wird. Die Data-Variable verfügt über eine Default-Logik, die bestimmt, wie sie sich mit Daten aus dem Backend füllt. Die Daten werden aus dem Backend über den Ablauffunktionskomponente „Get record collection“ bezogen, wobei die Daten hier nur als ein Ausgangsargument des Knotens existieren. Dann wird die „Set data variable“ -Ablauffunktionskomponente verwendet, damit die Daten in die Data-Variable setzen können. Nun steht die Data-Variable *Product1* für die Verwendung in View-Componente Binding zur Verfügung.



Abbildung 3.12: Data-Variable Logik

Zur Anzeige der Produkteliste muss die View-Component „List item“ mit dem erstellten Data-Variable *Product1* verbunden und wiederholt werden. Das Primary und das Secondary Label bestimmen, welcher Property-Wert in der Liste angezeigt werden soll. In dieser AppGyver-Anwendung sollen der Produkttitle und der Produktpreis angezeigt werden. Deshalb sollten das Primary Label mit dem Wert *current.Title* und das Secondary Label mit *current.Price* verbunden werden. Nach dem Speichern wird dann die Liste mit allen Produkten angezeigt.

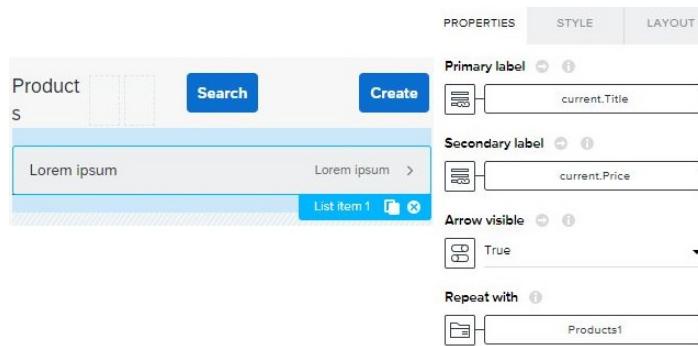


Abbildung 3.13: Listenseite und List Item mit Repeat in AppGyver

3.2.4 Einzelansicht für ein Produkt

Die Einzelansicht enthält alle Properties zu einem Produkt, nämlich den Produkttitel, die Beschreibung, die Materialnummer, den Preis, den Bestand und die Produkt-ID. Für die Darstellung der Informationen muss eine Detailseite mit dem Namen „Product Details“ erstellt werden. Eine App-Variable *appVarRecord* mit sechs Properties wird erstellt, damit die Daten an verschiedene Seiten der Anwendung übertragen werden können.

Um von der Listenseite zur Detail-Seite zu navigieren, muss die logische Ablauffunktion für das List Item in der Listenseite aufgebaut werden, dabei geht es um ein Component tap Event. Das Event wird ausgelöst, wenn das List Item vom Benutzer angeklickt wird. Der Aufbau der logischen Ablauffunktion kann Abbildung 3.14 entnommen werden.



Abbildung 3.14: Logische Ablauffunktion für List Items

Eine „Open page“-Ablauffunktionskomponente ist erforderlich, um die Detailseite zu öffnen. Die Ausgangspunkte der Ablauffunktionskomponente „Open page“ sind mit der Ablauffunktionskomponente „Set app variable“ verknüpft. Mit dieser Komponente werden die Daten in dem List Item mit der neu angelegten App-Variable verbunden. Die Werte der Properties von der App-Variable werden mit den jeweiligen Property-Werten in Wiederholung von List Item gesetzt. Dabei handelt es sich um die Werte der Data-Variablen *Products1*. Da die Werte der Date-Variablen *Product1* mit dem OData Service verbunden wurden, wird der Wert aus dem Backend in die App-Variable eingesetzt.

Auf der Detailseite werden sechs Input-Felder erstellt. Die Werte der Input-Felder sind mit dem entsprechenden Property-Wert der App-Variablen *appVarRecord* verbunden, nämlich dem Titel, der Materialnummer, dem Preis, der Produkt-ID und der Beschreibung sowie dem Bestand. Außerdem werden drei Buttons für 3 Component tap Events auf der Detailseite erstellt: Der „Back“-Button wird verwendet, um zurück zur Listenseite zu navigieren. Um dies zu implementieren, muss lediglich eine Logikablauffunktion mit der Komponente „Navigation back“ erstellt werden (vgl. Abb. 3.16). Der „Edit“-Button wird zum Aktualisieren der Pro-

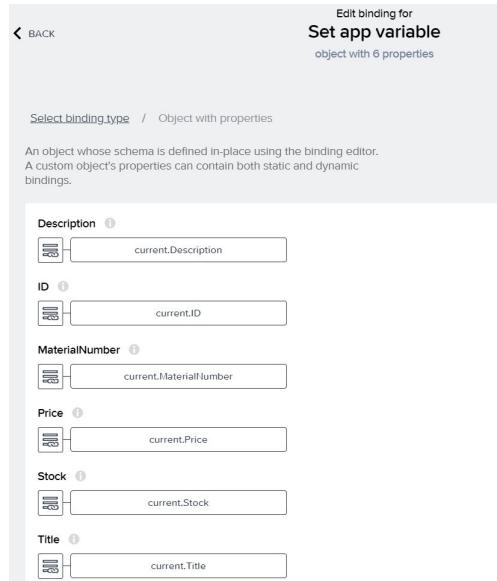


Abbildung 3.15: Wert zuweisen für App-Variablen *appVarRecord* in *Set app variable* Ablauffunktionskomponente

duktinformationen genutzt. Dazu ist die in Abbildung 3.16 dargestellte Ablauffunktion mit der Komponente „Update record“ erforderlich. Die Ablauffunktionskomponente ist mit der Datenressource Products (vgl. Abb. 3.11) und der App-Variablen *appVarRecord* verknüpft. Die „Update record“-Komponente hat zwei Ausgänge: Der obere ist mit einer „Alert“-Ablauffunktionskomponente verlinkt, die zur Anzeige von „Update Record ID“ dient, der untere ist mit einer „Toast“-Ablauffunktion verbunden, die nur benutzt wird, wenn die Aktualisierung des Datensatzes nicht erfolgreich abgeschlossen werden kann. Der „Delete“-Button in Kombination mit der „Delete record“-Komponente wird zum Löschen eines Produkts eingesetzt.

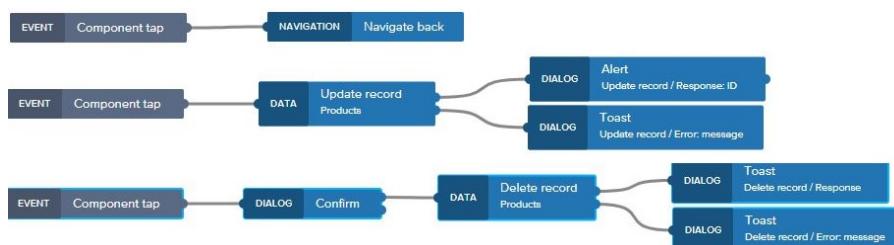


Abbildung 3.16: Logische Ablauffunktion für Edit-, Back- und Delete-Button

3.2.5 Maske zum Erstellen eines einzelnen Produkts

Zunächst muss eine Seite mit dem Namen „Create New Record“ erstellt werden. Diese Seite besteht aus fünf Input-Feldern, einem „Save“-Button sowie einem „Back“-Button. Eine neue Data-Variable mit dem Variablen-typ „New data record“ wird erstellt, die ebenfalls mit der Data-Ressource Products verbunden wird. Data-Variable mit dem Typ „New data record“ initialisiert eine leere Data-Variable, die beim Erstellen eines Formulars zum Anlegen von Datensätzen im Backend zu verwenden ist. Die Werte der Input-Felder werden mit den entsprechenden Property-Werten der Data-Variablen verknüpft. Da die Produkt-ID als UUID definiert ist und vom System generiert wird, muss das Input-Feld dafür nicht erstellt werden.

Um den Datensatz speichern zu können, wird eine Ablauffunktion mit der Komponente „Create record“ für den „Save“-Button benötigt. Ähnlich wie die Ablauffunktionskomponente „Update record“ ist die Komponente „Create record“ mit der Datenressource Products und der App-Variablen appVarRecord verbunden. Sie hat ebenfalls zwei Ausgänge: einen für die Bestätigung der erfolgreichen Erstellung von Datensätzen und einen anderen für die Anzeige des Fehlers (vgl. Abb. 3.17). Der „Back“-Button kann für die Navigation zurück zur Listenseite genutzt werden.



Abbildung 3.17: Logische Ablauffunktion für Save-Button

Wie bereits in Kapitel 2.4.2 vorgestellt, lassen sich die AppGyver-Baustein in 3 Schichten des MVC-Modells unterteilen, nämlich in View-, Controller- und Model-Schichten. Auf der View-Schicht wurden für den Anwendungsfall insgesamt 3 Seiten- und 21 View-Komponenten erstellt, darunter 3 Seitentitel, ein List Item, 11 Input-Felder sowie 7 Buttons. Damit die Buttons und das List Item dynamisch funktionieren, wurden in der Controller-Schicht 8 logische Ablauffunktionen mit Componente tap Event erstellt. In der Model-schicht wurden 2 Daten-Variablen und eine App-Variable erstellt, um die Daten vom Odata-Service zu holen, die Daten zu aktualisieren und sie an verschiedene Seiten zu übertragen.

Ein wichtiges Konzept im AppGyver ist die Binding von Komponenten an Daten, die in Variablen gespeichert sind. Mit nur wenigen Klicks kann Bei-

spielweise eine View-Component Property direkt mit einer Data-Variable in Binding treten. Die Binding wird automatisch aufgelöst, wenn neue Daten aus dem Backend abgerufen und in der Date-Variable gespeichert werden. Die View-Component Property wird aktualisiert und die neuen Daten wird in View angezeigt [App22c].

The screenshot displays three views of an application:

- The List of Products:** A table with three rows. The first row shows "Upgrade4Cars car seat covers set 2000" with a price of "150,95 Euro". The second row shows "Carpendo car seat covers" with a price of "40,19 Euro". The third row shows "Upgrade4Cars car seat covers" with a price of "13,95 Euro".
- Product Details:** A detailed view for the first product. It includes fields for Product Details (Product Title: Upgrade4Cars car seat covers set 2000, Product description: Seat covers complete set for the front seats & back seat with zipper and airbag opening, Black and Grey, Material number: 644), Price (150,95 Euro), Stock (STOCK991), and Product ID (00044eac-0971-4557-994b-1bae0e0cb8fe). Buttons at the bottom are Back, Edit, and Delete.
- Create New Record:** A form for creating new records. It has sections for "Create a new record" (Title, Type here..., Material Number, Type here..., Description, Type here...) and "Price" (Type here..., Stock, Type here...). Buttons at the bottom are Save and Back.

Abbildung 3.18: Listenseite, Detailseite und „Create new record“ -Seite der AppGyver-Anwendung

3.3 SAPUI5

3.3.1 Erstellung einer Anwendung mit dem UI5-Generator

In Kapitel 2.5 wurden SAPUI5 und die Einrichtung der Entwicklungsumgebung für die Implementierung von SAPUI5-Anwendungen kurz vorgestellt. In diesem Abschnitt wird nun die Erstellung der SAPUI5-Applikation näher beschrieben. Mit Hilfe des easy-ui5-Generators lässt sich ein grundlegendes SAPUI5-Anwendungsgerüst erstellen. Da die Entwicklung der Applikation auf dem neuesten Stand der Technik erfolgen soll, wird TypeScript für das Coding verwendet. TypeScript ist eine Design-Time-Programmiersprache, die JavaScript um eine starke Typisierung erweitert. Das UI5-TypeScript-Tooling nutzt den Babel-Compiler (JavaScript Compiler), um den TypeScript-Code in ausführbaren JavaScript-Code umzuwandeln [Mue22]. Dabei läuft ein Prozess im Hintergrund, der die Quellen im src-Ordner überwacht und die transpilierten Dateien in den webapp-Ordner kopiert.

Zur Erzeugung des Applikations-Gerüsts wird der Befehl `yo easy-ui5 ts-app` in der Command-Prompt-Konsole ausgeführt. Dieser Befehl führt

dazu, dass eine UI5-TypeScript-Anwendung basierend auf den folgenden Parametern erstellt wird:

```
? How do you want to name this application? products
? Which namespace do you want use? fanfan.products
? Which framework do you want to use? SAPUI5
? Which framework version do you want to use? 1.108.4
? Who is the author of the application? Tan Fangfang
? Would you like to create a new directory for the application? Yes
```

Quelltext 3.1: Eingabe der Parameter des UI5-Generators

Nach Eingabe der Parameter müssen noch alle externen Bibliotheken via npm install installiert werden und dann kann die Entwicklung beginnen.

Der Generator erzeugt die grundlegende Projektstruktur eines SAPUI5-Projekts. Das MVC-Pattern findet sich auch im Dateisystem wieder. So liegen die Controller im Verzeichnis controller, die Views im gleichnamigen Verzeichnis und es ist ebenfalls ein model-Verzeichnis vorgeesehen. Dies wird jedoch im Projekt nicht verwendet, sondern es werden auf Standard Odata- und JSONModel zurückgegriffen [Doc22a]. Daneben sind noch eine Reihe an Konfigurationsdateien vorhanden, wie package.json oder manifest.json:

```
project-root
|— src
  |— controller
    |— App.controller.ts
    |— BaseController.ts
    |— Main.controller.ts
  |— i18n
  |— model
  |— view
    |— App.view.xml
    |— Main.view.xml
  |— Component.ts
  |— manifest.json
  |— index.html
  |— [...]
  |— package.json
  |— README.md
  |— ui5.yaml
```

Quelltext 3.2: Die grundlegende Projektstruktur eines SAPUI5-Projekts

Die Datei „manifest.json“ enthält die Grundkonfiguration der Anwendung, die für die Ausführung benötigt wird. Alle Modelle, Bibliotheken und Komponenten, die in der „manifest.json“-Datei konfiguriert sind, werden beim Starten der Anwendung automatisch geladen und instanziert. Die

„rootView“- und „routing“-Konfigurationen definieren die Hauptansicht der Anwendung, die nach Öffnen im Browser angezeigt wird, und die Navigation zwischen den Ansichten.

In der Datei „App.view.xml“ wird die Benutzeroberfläche der Hauptansicht der Anwendung definiert. SAPUI5 unterstützt mehrere View-Typen (XML, HTML, JavaScript, JSON). In diesem Projekt wird das XML-Format verwendet. Für jede Ansicht, die in der Anwendung verwendet wird, soll eine eigene View-Datei erstellt werden, und jede View wird von einem separaten Controller gesteuert.

Das Öffnen der Webanwendung erfolgt durch Aufrufen der index.html-Seite über einen Webserver im Browser. Das UI5-Tooling bringt bereits einen lokalen Webserver mit, der via npm run start gestartet werden kann. Abbildung 3.19 zeigt das Aussehen der erzeugten Anwendung, die bereits erste Inhalte hat, die angepasst und ersetzt werden müssen. Während die Anwendung läuft, kann der Code in VS-Code geändert und die Änderungen direkt danach direkt im Browser angesehen werden.

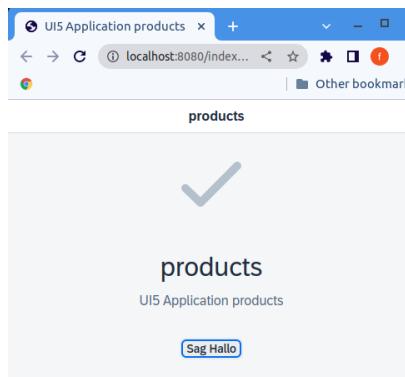


Abbildung 3.19: Aussehen der UI5 Demo-Anwendung

In den nächsten Abschnitten wird die Implementierung der UI5-Anwendung nun im Detail beschrieben.

3.3.2 OData Service einbinden

In Kapitel 3.2.2 wurde die Bereitstellung der OData-Services und der Destination in der SAP-BTP-Umgebung vorgestellt. Die SAPUI5-Anwendung muss diese Destination ebenfalls integrieren, um die OData-Services hinter der Destination konsumieren zu können.

Leider lässt sich der OData-Server lokal nicht einfach im Browser einbin-

den, da dies zu einem Cross-Origin-Resource-Sharing (CORS)-Problem führt. Die Anwendung läuft unter der Domäne „localhost“ und moderne Browser erlauben es nicht, Daten via XMLHttpRequest von einer anderen Domäne zu laden (es sei denn, diese sendet entsprechende http-Header mit) [Doc22d]. Um dieses Problem zu vermeiden, wird der Request über einen lokalen Reverse-Proxy (SAP Approuter) geleitet, der über das UI5-Tooling in das Projekt integriert werden muss. Dies ist zwar in wenigen Konfigurationschritten durchführbar, es zeigt aber bereits, dass auch gleich funktionierende Dinge (Zugriff auf die OData-Services via Destination) in SAPUI5 aufwendiger zu implementieren sind und die beiden LCNC-Toolings (Fiori Elements mit CAP und SAP AppGyver) derlei Dinge vom Entwickler fernhalten.

Der Zugriff auf die Daten in SAPUI5 erfolgt auf Basis eines ODataModels. Zwar ließen sich die Daten auch manuell als JSON abrufen, die Nutzung des ODataModels bietet jedoch den Vorteil, dass dies die http-Kommunikation komplett vom Entwickler fernhält. Zudem bietet das ODataModel weitere Features wie Databinding an, mit dem Daten sehr einfach an View-Controls gebunden und dargestellt werden können.

Das ODataModel wird in der Component.ts-Datei erstellt und damit beim Starten der Anwendung initial in allen Controllern und Views bekannt gemacht.

```
const oData = new ODataModel({
  serviceUrl: "public/",
  synchronizationMode: "None",
  operationMode: "Server",
});
this.setModel(oData);
```

Quelltext 3.3: Auszüge aus der Klasse **Component**

3.3.3 Listenansicht zur Anzeige aller Produkte

Die Listenansicht soll alle Produkte in einer Liste darstellen und es dem User ermöglichen, in die Einzelansicht eines Produkts zu springen. Da die Applikation nach dem Erstellen bereits über eine View verfügt, wird dies in der Main-View realisiert. Zentrales Element bildet die List-Control, welche Daten in einer Liste darstellen kann. Abbildung 3.20 zeigt die Benutzeroberfläche der Listenansicht der Produkte.

SAPUI5-Controls sind UI-Elemente, die das Aussehen und Verhalten kap-

| Products Information | | | | |
|---|------|--------|---------------|-----|
| The list of the Products | Scan | Create | SearchProduct | Q |
| Upgrade4Cars car seat covers | | | 13,95 Euro | > X |
| Leader Accessories universal car seat cover | | | 24,99 Euro | > X |
| Woltu car seat covers | | | 26,99 Euro | > X |
| Carpendo car seat covers | | | 40,19 Euro | > X |
| Fixcape Neoprene Car Seat Covers | | | 29,50 Euro | > X |
| Flying Banner PVC Car Seat Covers | | | 34,39 Euro | > X |
| Upgrade4Cars car seat covers set | | | 39,95 Euro | > X |
| Walser car seat cover Rey | | | 74,95 Euro | > X |

Weitere

Abbildung 3.20: Benutzeroberfläche der Listenansicht der Produkte

seln. In SAPUI5 stehen mehr als 100 Controls wie z.B. Buttons, Inputs, Texte, Tabellen, Messages, Dialoge, Listen usw. zur Verfügung. Eine Control kann andere Controls aggregieren und diese dient dann als Container. Die List-Control ist ein Container für entsprechende Listenelemente. Die Steuerung der View wird im entsprechenden Controller *Main.controller.ts* implementiert.

```

<List mode="Delete" delete="onDeletePress" items="{path:'/Products'}" id="productsList">
  <headerToolbar>
    <OverflowToolbar width="100%">
      <Title text="The list of the Products" />
      <ToolbarSpacer />
      <Button icon="sap-icon://camera" text="Scan" press="onScanNavi"/>
      <Button icon="sap-icon://add" text="Create" ariaHasPopup="Dialog" press="pressAddInList" />
      <SearchField width="200px" placeholder="SearchProduct" search="mySearchProduct" />
    </OverflowToolbar>
  </headerToolbar>
  <items>
    <StandardListItem title="{title}" type="Navigation" info="{price}" press="onItemPress"/>
  </items>
</List>
```

Quelltext 3.4: Auszüge aus der View *Main.view.xml*

Das Coding zeigt den Ausschnitt der Listendefinition. Jede Control verfügbar über bereitgestellte Eigenschaften (Properties) und Events, die im XML angegeben werden können. Mit der Eigenschaft *mode="Delete"* wird beispielsweise automatisch ein Lösch-Button in der Liste hinzugefügt. Das Event *delete= "onDeletePress "* verweist dahingehend dann auf die Callback-Funktion *onDeletePress* in der Klasse *Main.controller.ts*, welche

das Löschen dann programmatisch vornehmen muss:

```
public onDeletePress(event: UI5Event) {
    const listItem = event.getParameter("listItem") as ColumnListItem;
    const context = listItem.getBindingContext() as Context;
    context.delete();
}
```

Quelltext 3.5: Auszüge aus der Controller `Main.controller.ts`

In dieser Methode zeigt sich, dass in SAPUI5 durch die Models und das Databinding weder die Daten direkt aus den UI-Controls abrufen noch direkt mit dem OData-Service kommuniziert werden muss. Es wird an dieser Stelle lediglich der BindingContext (die Information zum Binding selber) benötigt und diese aus dem Model gelöscht. Das sorgt dann automatisch dafür, dass das Model einen DELETE-Request zum OData-Service sendet und das Produkt aus der Datenbank entfernt wird.

Gleiches gilt auch für das initiale Setzen des Bindings. In obigem Coding zur Listenansicht ist zu sehen, dass in der Liste lediglich ein StandardListItem definiert wird. Dies dient dabei als Template für alle Produkte, die aus dem OData-Service ausgelesen werden. Über das Aggregation-Binding der Aggregation items (`items="{path:'/Products'}"`) werden alle Produkte, sich in unter dem Pfad abrufen lassen gebunden und für jedes Produkt wird ein *StandardListItem* erzeugt, das den Titel und Preis anzeigt [Doc22c].

```
<StandardListItem title="{title}" type="Navigation" info="{price}" press="onItemPress"/>
```

Quelltext 3.6: Auszüge aus der View `Main.view.xml`

Ein weiteres wichtiges Event ist das press-Event eines StandardListItems. Klickt ein User auf ein Produkt, so soll er auf die Detailseite weitergeleitet werden. Das zugehörige Coding zeigt, dass an dieser Stelle auf den Router zugegriffen wird. In SAPUI5 sorgt der Router dafür, dass URL-Pfade auf Views/Controller gemappt werden und ermöglicht es, dass das Framework im Hintergrund die notwendigen Controller/Views instanziert.

```
public onItemPress(event: UI5Event){
    var item = event.getSource();
    var context = item.getBindingContext();
    var path = context.getPath().substr(1);
    this.navTo("detail", {path: path});
}
```

Quelltext 3.7: Auszüge aus der Controller `Main.controller.ts`

Auch hier wird auf das Binding zurückgegriffen, um Informationen zum Produkt auszulesen, die an die Detailseite übergeben werden.

3.3.4 Einzelansicht für ein Produkt

Damit die Navigation funktioniert, müssen für die Einzelansicht ein View *Detail.view.xml* und ein Controller erstellt werden. In der View werden alle Merkmale, wie der Name, die Beschreibung, die Materialnummer, der Preis und der Bestand eines Produkts in den Input Controls angezeigt.

```
<Label text="Title" />
<Input value="{title}" width="1000px" />
<Label text="Description" />
<Input value="{description}" width="1000px" />
<Label text="Material Number" />
<Input value="{materialNumber}" width="1000px" />
<Label text="Price" />
<Input value="{price}" width="1000px" />
<Label text="Stock" />
<Input value="{stock}" width="1000px" />
```

Quelltext 3.8: Auszüge aus der View *Detail.view.xml*

Damit die Werte angezeigt werden können, wird in der View wieder ein ElementBinding verwendet, dass ein komplettes Produkt an die View bindet. Dann ist es ausreichend, lediglich die Attribute zu benennen und das ODataModel lädt die relevanten Daten automatisch. Abbildung 3.21 kann die finale Benutzeroberfläche der Einzelansicht für ein Produkt entnommen werden.

| < Product Detail Page | |
|-----------------------|---|
| Title | Upgrade4Cars car seat covers |
| Description | Car seat cover for sports work workshop etc, car seat covers front seats universal, black |
| Material Number | 110 |
| Price | 13,95 Euro |
| Stock | Stock110 |

Abbildung 3.21: Benutzeroberfläche der Einzelansicht

3.3.5 Maske zum Pflegen eines einzelnen Produkts

Für die Zusammenstellung der Maske zur Pflege eines Produkts werden Label-, Input- und Button-Controls und das Element *Fragment* verwendet. Fragmente sind leichtgewichtige UI-Komponenten, die wiederverwendbar sind, aber keinen Controller haben [Doc22b]. Daher kann ein Fragment als ein Container für andere UI5-Controls betrachtet werden. Das UI-Design dieses Fragments ist in der Datei *createProductDialog.fragment.xml* gespeichert.

```
<content>
  <VBox class="sapUiMediumMargin">
    <Label text="Title" />
    <Input value="{listInputModel>/recipient/title}" width="200px" />
    <Label text="Material Number" />
    <Input value="{listInputModel>/recipient/materialNumber}" width="200px" />
    <Label text="Description" />
    <Input value="{listInputModel>/recipient/description}" width="200px" />
    <Label text="Price" />
    <Input value="{listInputModel>/recipient/price}" width="200px" />
    <Label text="Stock" />
    <Input value="{listInputModel>/recipient/stock}" width="200px" />
  </Vbox>
</content>
<beginButton>
  <Button text="Cancel" ariaHasPopup="Dialog" press="closeDialog" />
</beginButton>
<endButton>
  <Button text="Add" ariaHasPopup="Dialog" press="pressAddInProductsList" />
</endButton>
```

Quelltext 3.9: Auszüge aus der Fragment *createProductDialog.fragment.xml*

In der Fragment-Implementierungsdatei werden mehrere Eingabefelder und zwei Buttons, *Cancel* und *Add* erstellt. Wird das press-Event des *Add*-Buttons geworfen, so wird die Methode *pressAddInProductsList* aufgerufen, um den Datensatz, der in das Eingabefeld eingegeben wurden, in den OData-Service zu übernehmen. Die Eingabewerte aller Eingabefelder werden zunächst via Binding an ein lokales JSON-Model übergeben. In der *pressAddInProductsList*-Methode werden die Werte aus dem lokalen Model ausgelesen und ein neues OData-Binding erzeugt, dass die Daten automatisch an den OData-Endpunkt sendet.

```
public pressAddInProductsList() {
  var title = this.getView().getModel("listInputModel").getProperty("/recipient/title");
  var id = this.getView().getModel("listInputModel").getProperty("/recipient/id");
  var description = this.getView().getModel("listInputModel").getProperty("/recipient/
    description");
  var materialNumber = this.getView().getModel("listInputModel").getProperty("/recipient/
    materialNumber");
```

```

var price = this.getView().getModel("listInputModel").getProperty("/recipient/price");
var stock = this.getView().getModel("listInputModel").getProperty("/recipient/stock");

const context = (<ODataListBinding>(
    this.byId("productsList").getBinding("items")
)).create({
    title: title,
    id: id,
    description: description,
    materialNumber: parseInt(materialNumber),
    price: price,
    stock: stock,
});
this.closeDialog();
}

```

Quelltext 3.10: Auszüge aus der Controller `Main.controller.ts`

Da das Formular zum Anlegen in einem Dialog geöffnet wird, muss dieser nach dem Absenden dann auch wieder geschlossen werden. Abbildung 3.22 zeigt das Dialogfenster zum Einfügen eines neuen Produkts.

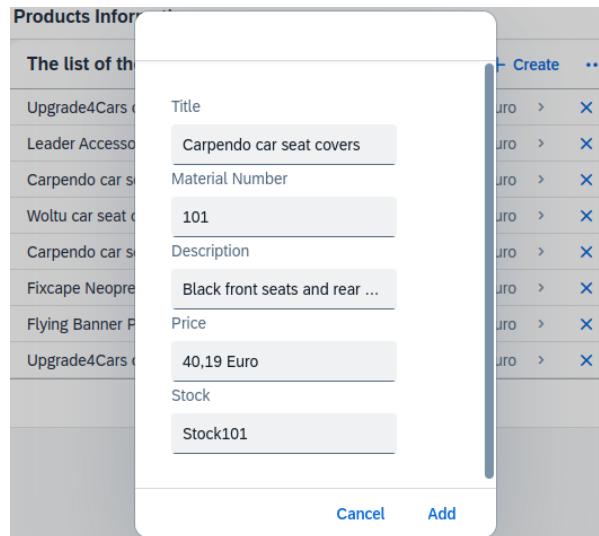


Abbildung 3.22: Maske zum Pflegen eines einzelnen Produkts

Kapitel 4

Untersuchung weiteren Funktionen

In Kapitel 3 wurden Funktionen beschrieben, die im Rahmen dieser Arbeit technisch umgesetzt wurden. Dieses Kapitel betrachtet weitere Standard-Anforderungen an Webanwendungen im SAP-Kontext und Funktionalitäten von Fiori Elements, SAP AppGyver und SAPUI5, ohne praktische Umsetzung. Diese Funktionen umfassen:

- Integration von Suchfiltern und Paginierung.
- Integration von Bildern und PDF-Dateien.
- Integration einer Barcode-Scanner-Funktionen.
- Nutzung weiterer mobiler Funktionen.
- Möglichkeiten zum Deployment für unterschiedliche Endgeräte
- Freie Gestaltungsmöglichkeiten in der UI

4.1 Integration von Suchfilter

In Fiori Elements ist es nicht notwendig, die Suchfilter-Funktion zusätzlich zu implementieren, da sie bereits in der List Report Vorlage vorhanden ist. Das heißt, wenn die Benutzeroberfläche in Fiori Elements mit dem List Report Floorplan erstellt wird, ist die Suchfilter-Funktion automatisch integriert. Über die Annotationen ist es möglich, Suchfilter genauer zu beschreiben oder gezielt ein- und auszublenden. Die Erstellung der Benutzeroberfläche wurde bereit in Abschnitt 3.1.3 dargestellt.

In AppGyver ist die Integration eines Suchfilters durch das manuelle Hinzufügen von Bausteinen möglich. Dabei ist jedoch zu erwähnen, dass der Filter komplett client-seitig funktioniert und zunächst alle Daten vom Backend geladen werden. Das sorgt dafür, dass bei großen Datenmengen längere Ladezeiten entstehen können. Die Implementierung in AppGyver ist wie folgt:

Ein List Item wurde bereits in Kapitel 3.2.3 mit der angelegten Daten-Variable *Product1* verbunden und wiederholt. Mit der Ablauffunktion-komponente „set app Variable“ wurde der List Item ebenfalls mit die in Abschnitt 3.2.3 erstellte App-Variable eingebunden. Für den Filter wird ein Input-Field per Drag and Drop in der Benutzeroberfläche integriert. Da die Suche über den Produkttitel erfolgen soll, wird der Wert des Input-Fields an die App-Variable Property *title* gebunden. In den Advanced Properties des List Items befindet sich ein Visible-Flag, mit dem die Sichtbarkeit des List Items eingestellt werden kann. Der Schalter kann auf true oder false gesetzt werden. Bei false wird das List Item nicht gerendert. Über dieses Flag kann der Filter implementiert werden: Eine Contains-Formel wird an die Visible-Eigenschaft gehangen, die *true* zurückgibt, wenn der Text den angegebenen Textabschnitt enthält. Andernfalls gibt die Formel *false* zurück und das Produkt wird ausgeblendet. Um die Groß- und Kleinschreibung nicht zu berücksichtigen, können alle Zeichenketten in Kleinbuchstaben umgewandelt werden (lowercase-Formel):

```
CONTAINS(LOWER CASE(repeated.current.Title), LOWER CASE(app Vars.app VarRecord.Title))
```

Abbildung 4.1 zeigt eine beispielhafte Suche nach dem Begriff *fix*.

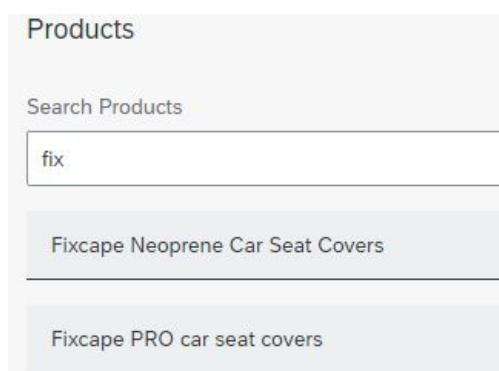


Abbildung 4.1: Suchfilter in AppGyver

In SAPUI5 ist die Integration eines Suchfilters durch das Hinzufügen

von weiteren Controls und einem Filter auf dem AggregationBinding der Liste möglich. Das SearchField-Control verfügt bereits über das notwendige Verhalten und die zu konfigurierenden Eigenschaften und Events. Das Search-Event wird ausgelöst, wenn der Benutzer Werte in das Suchfeld eingegeben hat und dies per Button-Druck oder Drücken der Enter-Taste bestätigt. (Siehe Quelltext 4.1)

```
<List items="{path:'/Products'}" id="productsList">
  <headerToolbar>
    <OverflowToolbar width="100%">
      <SearchField width="200px"
        placeholder="SearchProduct"
        search="mySearchProduct" />
    </OverflowToolbar>
  </headerToolbar>
</List>
```

Quelltext 4.1: Implementierung von Suchfilter in der `Main.view.xml`

Das Search-Event wird im Controller in der Methode `mySearchProduct` implementiert. Über das UI5Event-Object kann dort der Suchwert als Parameter abgerufen werden. Mittels manuellem Zugriff auf die Tabelle kann das ODataListBinding angesprochen und durch die Verwendung von Filtern auf den Suchwert gefiltert werden. Dem Coding ist zu entnehmen, dass wir nur auf die Title-Eigenschaft filtern. Direkt beim Setzen des Filters wird das ODataModel das Binding aktualisieren, die gefilterten Daten vom OData-Backend anfragen und die Liste an Produkten in der UI aktualisieren.

```
public mySearchProduct(event:UI5Event){
  const query = event.getParameter("query");
  MessageBox.show(query);
  const table = this.getView().byId("productsList");
  const binding = table.getBinding("items") as ODataListBinding;
  const filters = [];
  if(query){
    filters.push(new Filter("title", FilterOperator.Contains, query))
  }
  binding.filter(filters);
}
```

Quelltext 4.2: Implementation von Suchfilter in der `Main.controller.ts`

4.2 Integration von Paginierung

Die Paginierung in Fiori Elements ist bereits als Standard-Funktionalität vorhanden. Per Default werden in einer Fiori-Elements-Anwendung 30

Listeinträge auf einer Seite angezeigt. Sind weitere Einträge vorhanden, werden erst beim Scrollen automatisch weitere 30 Listeinträge geladen, bis alle Datensätzen abgerufen werden. Die Standardeinstellungen können in Low-Code Kontext nicht geändert werden.

SAP AppGyver bietet für Listenansichten keine native Unterstützung einer Paginierung. Das Verhalten kann jedoch nachgebaut werden. Hierfür werden eine List-Component und zwei Buttons zur Steuerung der Paginierung benötigt. Zusätzlich sind zwei App-Variablen notwendig, eine zum Speichern der Seitennummer (pageNumber) und die andere zum Speichern der Gesamtzahl der Einträge (recordCount). Die Logik wird dann in einer Ablauffunktion (siehe Abbildung 4.2) implementiert. In Ablauffunktion „Get record collection“ werden zunächst die Daten von Backend geholt. Danach wird das Paging mit einem Custom Object definiert, in welchem die Page size (wie viele Einträge auf einer Seite angezeigt werden) und Page number (aktuelle Seite) abgelegt werden. (siehe Abbildung 4.3).



Abbildung 4.2: Seitenlogik von Paginierung in AppGyver



Abbildung 4.3: Paging definieren in Ablauffunktion „Get record collection“

Damit die Paginierung funktioniert, muss die Logik der Buttons noch hinzugefügt werden. Für beide Buttons gibt es eine eigene Ablauffunktion. Abbildung 4.4 zeigt die Logik der Buttons. Für PREV-Button wird den Wert der App-Variable „pageNumber“ mit folgender Formel ermittelt:

$SUBTRACT(appVars.pageNumber, 1)$

Für NEXT-Button wird antstatt der SUBSTRACT-Function die ADD-Function verwendet:

$ADD(appVars.pageNumber, 1)$

Damit der PREV-Button auf Seite 1 und der NEXT-Button auf der Letzen Seite ausgeblendet werden, muss die Sichtbarkeit der Buttons über das Visible-Flag gesteuert werden. Hier wird jeweils auf eine Formel verwiesen. Die Formel für PREV-Button lautet:

NOT(IS_EQUAL(appVars.pageNumber, 1))

Die Formel für NEXT-Button sieht wie folgt aus:

MULTIPLY(appVars.pageNumber, 4) <= appVars.recordCount



Abbildung 4.4: Logik für PREV- und NEXT-Button in AppGyver

Abbildung 4.5 zeigt die Benutzeroberfläche in der AppGyver-Anwendung, nachdem die Paginierung implementiert wurde.

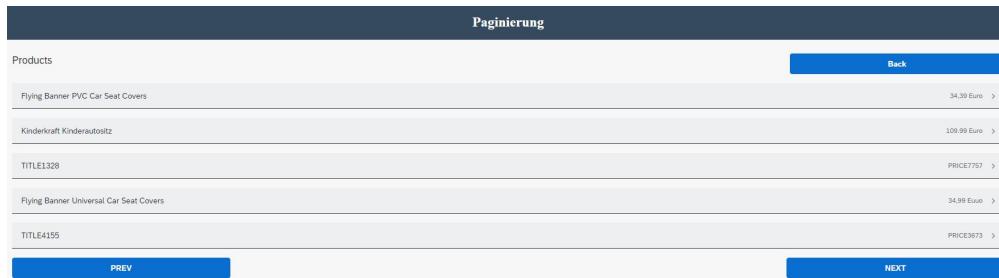


Abbildung 4.5: Benuteroberfläche nach Paginierung in AppGyver

Die Integration der Paginierung in SAPUI5 ist sehr einfach, denn die entsprechenden List-Controls haben das Verhalten bereits implementiert. Setzt man die Eigenschaft *growing* auf *true*, dann wird das Nachladeverhalten bei Scrollen aktiviert. Eine weitere Property *growingThreshold* kann hinzugefügt werden, um die Anzahl der Listeneinträge auf einer Seite zu bestimmen. Quelltext 4.3 zeigt nur mit 2 Zeile Code kann der Paginierung in SAPUI5 implementiert werden.

```
<List growing="true" growingThreshold="8">
</List>
```

Quelltext 4.3: Implementation von Paginierung in der `Main.view.xml`

4.3 Integration von Bild- und PDF-Datei

Die Integration von Bild und PDF-Dateien in Fiori Elements ist im Low-Code Kontext nicht ohne weiteres möglich, da es in Fiori Elements derzeit noch keine Annotationen gibt, Datei-Formate auszuweisen. Grundsätzlich unterstützt das verwendete SAP Cloud Application Programming Model (CAP) die Bereitstellung von Mediendaten[CAP22], aber diese können nicht automatisiert in der UI dargestellt werden.

Auf der Seite von SAP CAP-Dokumentation kann man das Datenmodell wie folgt annotieren, um darauf hinzuweisen, dass die Entität Mediendaten enthält:

- *@Core.Media Type* gibt an, dass das Element Mediendaten enthält.
- *@Core.IsMediaType* gibt an, dass das Element einen MIME-Typ enthält. MIME ist die Abkürzung für Multipurpose Internet Mail Extensions und bedeutet Internet Media Type[Pag22].
- *@Core.IsURL@Core.MediaType* gibt an, dass das Element eine URL enthält, die auf die Mediendaten verweist.
- *@Core.ContentDisposition.Filename* gibt an, dass das Element als Anhang angezeigt werden soll, der heruntergeladen und lokal gespeichert wird.
- *@Core.ContentDisposition.Type* kann verwendet werden, um den Browser anzuweisen, das Element inline anzuzeigen.

Um die Medienressourcen zu lesen, sollte die GET-Anfrage in der Form */Entity/mediaProperty* verwendet werden. Um eine Medienressource zu erstellen, müssen wir zunächst eine Entität ohne Mediendaten erstellen, indem wir eine POST-Anfrage an die Entität stellen. Nach der Erstellung der Entität kann die Medien-Property mit der PUT-Methode eingefügt werden[CAP22].

Die Integration von Bild-Dateien in AppGyver ist relativ einfach zu implementieren. Hierzu gibt es in AppGyver ein Standard-View-Component Image. Mit Drag and Drop kann diese Komponente zur Benutzeroberfläche hinzugefügt werden.

Es gibt verschiedene Möglichkeiten, Bilder an die Benutzeroberfläche zu binden. (Abbildung 4.6) Mit „static image“ lässt sich ein lokales Bild auswählen und direkt hochladen. Wenn die Datei unter einer externen URL

erreichbar ist, kann diese per „Formula“ eingegeben werden. Weiterhin ist es möglich, auch Bilder aus Datenstrukturen auszulesen. Zudem gibt es im Marketplace verschiedene Ablauffunktionen, um die Kamera eines (mobilen) Endgeräts zu öffnen und ein Foto zu machen („Take photo“) oder ein Bild aus der Bibliothek des Geräts auszuwählen („Pick image from library“).

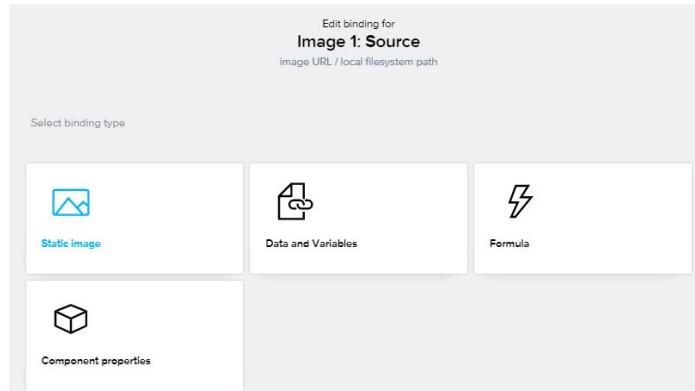


Abbildung 4.6: Image Binding in AppGyver

Die Integration von PDF-Dateien in AppGyver erfordert ebenfalls nur einen geringen Implementierungsaufwand. Im Marketplace existiert eine Ablaufkomponente „Preview PDF“, welche an einen Button angehangen werden kann. In den Eigenschaften der Ablaukomponente kann dann dynamic oder statisch die URL der PDF-Datei angegeben werden. Abbildung 4.7 zeigt die Logik eines Flows mit der integrierten Ablaukomponente.



Abbildung 4.7: Logik für Load-PDF-Button in AppGyver

Für die Integration von Bild- und PDF-Dateien in UI5 stehen Standard-Controls zur Verfügung, die mit geringem Konfigurationsaufwand einsetzbar sind. Für Bilder kann in SAPUI5 die Control „Image“ der sap.m-Library verwendet werden. Via „src“-Property kann ein relativer oder absoluter Pfad zur URL hinterlegt werden. Zudem kann man auch die Größe („width“-Eigenschaft) oder Höhe („height“-Eigenschaft) bestimmen. Mit der Standard Control „PDFViewer“ kann man PDF-Dateien integrieren. Hier gibt es eine „source“-Eigenschaft, die den Pfad zu der anzuzeigenden PDF-Datei enthält. Quelltext 4.4 zeigt die Implementierung von Bild- und PDF-Datei Integration in UI5 auf Basis einer XML-View.

```
<Image src="https://prod.pictures.autoscout24.net/listing-images/0693be87-aaad-4dc8-99c5-a69b3bd48b81_1934e390-16f2-4e47-b248-02e9cc9b746b.jpg"
       width="200px" height="150px"/>
<PDFViewer source="https://www.uni-muenster.de/imperia/md/content/zns/dokumente/beispielepos_kowi_empirisch.pdf"/>
```

Quelltext 4.4: Implementierung Bild- und PDF Datei Integration in SAPUI5

4.4 Integration von Barcode-Scanner-Funktion

Die native Integration einer Barcode-Scanner-Funktion in Fiori-Elements ist nicht vorgesehen. Über eine Extension könnte ein Button in der UI platziert werden, die eine solche Funktion bereitstellt. Die Logik dafür entspricht der SAPUI5-Implementierung (siehe unten).

In AppGyver existiert eine Ablauffunktionskomponente „Scan QR/Barcode“, welche die direkte Integration einer Barcode-Scan-Funktion ermöglicht. Das triggern der Scan-Funktion kann über einen Button angestoßen werden. Abbildung 4.6 zeigt die Logik für einen Scan-Button. Die Scan QR/barcode Ablauffunktionskomponente kann dann mit dem Event „Component tap“ verbunden werden. Die Ablauffunktion hat mehrere Ausgänge: Es ist möglich, den Textinhalt des gescannten QR-Barcodes zu erhalten. Weiterhin kann man ebenfalls eventuell auftretende Fehler erhalten. Nachfolgende Abbildung zeigt die exemplarische Integration und Ausgabe von gefundenen Barcodes oder Fehlern in einem Dialog.



Abbildung 4.8: Logik für Scan QR/Barcode-Button in AppGyver

Die AppGyver-Anwendung kann sowohl als Webanwendung, als auch als mobile Anwendung eingesetzt werden. Die Barcode-Scanner-Funktion wird von der AppGyver allerdings in einer Desktop-Webanwendung nicht unterstützt, sondern funktioniert nur auf mobilen Endgeräten.

In SAPUI5 existieren mehrere Barcode-Scanner-Controls. Der Barcode Scanner Button (Namespace: sap.ndc) ist nur für die Nutzung auf mobilen Endgeräten vorgesehen (*ndc* steht für *native device capabilities*) und kann nur in Applikationen genutzt werden, die in der SAP Fiori Client-App geöffnet werden. Die neuere Komponente Barcode Scanner Dialog (Name-

space: sap.ui.webc.fiori) kann dagegen auf allen Endgeräten im Browser eingesetzt werden.

Die Integration in eine Anwendung erfolgt durch Integration der jeweiligen Control in die View. In folgendem Beispiel wird ein BarcodeScannerButton integriert. Die Control stellt dann entsprechende Events bereit (*scanSuccess* und *scanFail*), die nach erfolgreicher oder fehlerhaftem Scan-Vorgang geworden werden.

```
<mvc:View controllerName="tff.Products.controller.Barcode" displayBlock="true" xmlns:ndc="sap.ndc" xmlns="sap.m" xmlns:mvc="sap.ui.core.mvc">
<VBox class="sapUiSmallMargin">
<ndc:BarcodeScannerButton
  id="button"
  scanSuccess="onScanSuccess"
  scanFail="onScanError"
  dialogTitle="Barcode Scanner Button Sample"
/>
<HBox class="sapUiTinyMarginTop">
  <Label text="Scan Result:"/>
  <Text id="result" text="" class="sapUiTinyMarginBegin"/>
</HBox>
</VBox>
</mvc:View>
```

Quelltext 4.5: Implementierung Barcode-Scanner in View **Barcode.view.xml**

Im Controller kann die Callback-Funktion dann den Wert des Barcodes oder den Fehler aus den Event-Parametern auslesen. In folgendem Beispiel werden die Informationen auf dem Bildschirm in Form eines MessageToasts ausgegeben.

```
public onScanSuccess(oEvent) : void {
  if (oEvent.getParameter("cancelled")) {
    MessageToast.show("Scan cancelled", { duration:1000 });
  } else {
    if (oEvent.getParameter("text")) {
      oScanResultText.setText(oEvent.getParameter("text"));
    } else {
      oScanResultText.setText("No barcode detected");
    }
  }
}

public onScanError(oEvent) : void {
  MessageToast.show("Scan failed: " + oEvent, { duration:1000 });
}
```

Quelltext 4.6: Implementierung Barcode-Scanner in Controller **Barcode.control.ts**

4.5 Nutzung nativer mobiler Funktionen

Der Zugriff auf native mobile Funktionen, wie z.B. Sensor-Informationen des Smartphones, wird heute von modernen Browsern unterstützt und ermöglicht die Nutzung in Applikationen.

SAP Fiori Elements-Apps greifen standardmäßig nicht auf native Funktionen zurück. Hier können über Extensions Funktionalitäten hinzugefügt werden, was jedoch einen Programmieraufwand mit sich bringt (siehe SAPUI5 unten).

SAP AppGyver unterstützt den Zugriff auf Sensordaten durch die Bereitstellung von Flow-Funktionen, sowie spezifischen Sensor-Variablen. Diese können bei der Erstellung einer Anwendung berücksichtigt werden. Folgende Sensoren werden unterstützt:

- Accelerometer
- Barometer
- Kompass
- Geolocation (GPS)
- Gyroscope
- Magnetometer

SAPUI5 bietet im Namespace *sap.ui.Device* grundlegende Informationen wie den Namen des Browsers oder der Plattform des Endgeräts, auf dem die Anwendung läuft. Tiefergreifende Funktionen zum Zugriff auf Sensordaten sind jedoch nicht verfügbar. Hierfür lässt sich im Coding jedoch auf die APIs der Browser zugreifen und die entsprechenden Informationen in den UI5-Anwendungen nutzen.

4.6 Möglichkeiten zum Deployment für unterschiedliche Endgeräte

Standardmäßig können Fiori-Elements-Anwendungen und SAPUI5-Anwendungen nur als Webanwendung deployed werden. Es gibt dennoch unterschiedliche Möglichkeiten, eine Anwendung als App auf ein mobiles Endgerät zu bekommen. Die SAP bietet mit dem SAP BTP SDK für iOS eine native Implementierung von SAP Fiori an, die für diese Zwecke genutzt werden sollte. Quellcode oder Applikationslogik muss hier allerdings nativ in Apple Xcode und den iOS-Technologien implementiert werden. Daneben

steht mit der SAP Fiori Client-App eine spezielle App zur Verfügung, mit der SAPUI5-Anwendungen auf einem mobilen Endgerät geöffnet werden können. Die Applikation wird allerdings weiter als Webapplikation deployed.

In AppGyver gibt es mehrere Möglichkeiten, die erstellte Anwendung zu deployen. Hierfür steht der SAP AppGyver Build Service zur Verfügung, der die Anwendungen Verpacken und auch direkt Verteilen kann. Die Deployment-Optionen sind folgende:

- **Android Builds**

Eine Anwendung kann via Android SDK in eine native Android-App verpackt werden. Nach dem Bauen der Anwendung, kann diese mit einem USB-Kabel auf dem Gerät installiert werden oder als Direkt-Download-Link weitergegeben werden. Es ist ebenfalls möglich, die App über den Google Play Store oder andere Android-App-Management-Lösungen zu veröffentlichen. Die Dateigröße von Android kann sehr groß sein, da die Binärdatei optimierte Unterstützung für mehrere verschiedene Prozessoren enthält[App22a].

- **iOS Builds**

Das Bauen als native iOS-App erfordert zunächst die Mitgliedschaft im Apple Developer Programm. Über das Apple Developer Portal können dann die benötigten Anlagen generiert, konfiguriert und die App gebaut werden. Nach der Generierung lässt sich die App an die Mitglieder der Apple Developer-Organisation verteilen. Ebenso ist es möglich, die App zur Review an Apple senden. Nachdem die App angenommen wurde, ist sie im öffentlichen App Store verfügbar[App22e].

- **Web Builds**

Bei der Veröffentlichung als Webanwendung mit dem Build Service, wird die Anwendung in der AppGyver Cloud unter der von uns definierten und konfigurierten Subdomain bereitgestellt.

4.7 Freie Gestaltungsmöglichkeit

Fiori-Elements basieren auf den SAP Fiori Design Guidelines und den fest definierten Floorplans (siehe Kapitel 2.6.1). Zwar sind durch die Extension-Points und dem Hinzufügen eigener Controls geringe Abweichungen im Layout möglich, die Gestaltungsmöglichkeiten bleiben jedoch sehr beschränkt. Durch die Auswahl eines eigenen SAPUI5-Themes oder dem

Überschreiben von CSS-Anweisungen könnte das Aussehen weiterhin beeinflusst werden. Die Grundidee hinter Fiori Elements, leicht zu erstellende sowie gleich aussehende und funktionierende Standard-Apps, steht jedoch generell im Widerspruch zur Idee der freien Gestaltungsmöglichkeiten.

In AppGyver ist es möglich, die Anwendungen nach dem Designvorlieben eines Nutzers zu entwickeln. Der Themen-Editor in Composer Pro ermöglicht die Auswahl, sowie die Anpassung von Themen für die Anwendungen. Ein Theme bezeichnet dabei das grundlegende visuelle Erscheinungsbild einer Anwendung und definiert Schriftgröße und -art, Farben der UI-Elemente sowie Hintergrundbilder[Por22a]. Themes besitzen zudem Inhaltspaletten und semantischen Farben, mit denen eine Anwendung gezielt Formattierungen bereitstellen kann[App22f]. Als grundlegende Themes stehen in AppGyver zwei Themes bereit:

- Das universelle Theme ist ein modernes Allround-Theme, das sich einfach als Basis für ein individuale Theme adaptieren lässt.
- Das SAP-Fiori-Theme hilft bei der Implementierung des SAP Fiori Designsystems in AppGyver-Anwendungen[Sar21].

Im AppGyver Composer Pro haben alle View-Components Style-Klassen und Layout-Parameter. Die Style-Klassen definieren, wie eine Komponente in der UI aussieht, einschließlich Schriftart, Farben, Rahmen usw. Damit lassen sich allgemeine Werte aus den Themes überschreiben. Der Layout-Parameter bestimmt, wie eine bestimmte Komponenten auf der aktuellen Seite angeordnet wird (Abstand, Breite und Höhe und Position). Die Style und Layout können je nach Präferenz des Benutzers angepasst werden. Abbildung 4.9 zeigt 3 verschiedene Gestaltungen von AppGyver Anwendungen.

1 2

SAPUI5 orientiert sich zunächst ebenfalls an den SAP Fiori Guidelines. Anders als in Fiori Elements ist man jedoch beim Aufbau einer Anwendung in „freestyle“ -Apps komplett frei. Auch das Design lässt sich anpassen, hier gibt es zwei Möglichkeiten:

¹ Weather-App wurde nach dem Youtube AppGyver-Tutorial von Iyad Helwani erstellt.
URL: <https://www.youtube.com/watch?v=ktb8so0mKmI&t=1819s>

² Barcode-Scanner-App wurde nach SAP Tutorial von Daniel Wroblewski erstellt. URL:
<https://developers.sap.com/tutorials/appgyver-create-application.html>



Abbildung 4.9: Gestaltung von Weather-App, Barcode-Scanner-App¹ und Product-List-App²

- **Gestaltung einzelner UI-Elemente mit CSS**

Einzelne Controls in SAPUI5 können durch die Vergabe von speziellen CSS-Klassen mit eigenen Style-Definition umgesetzt werden[Ant16, S.261]. Zudem ist es möglich, gezielt existierende Styles zu überschreiben.

- **Erstellung eigenständiger Themes mit Theme Designer**

SAPUI5 unterstützt ebenfalls Themes. Ein benutzerdefiniertes Theme kann mit dem UI-Theme-Designer erstellt werden. In einem Theme werden verschiedene Parameter (Schriften, Farben, etc.), Bilder und andere Ressourcen konfiguriert, die dann einen Einfluss auf alle Controls einer Anwendung haben[Por22b].

Kapitel 5

Bewertung der Technologien

Eines der Ziele dieser Bachelorarbeit ist es, die Vor- und Nachteile von SAP AppGyver, SAP Fiori Elements und SAPUI5 zu identifizieren und herauszufinden, welche Technologie für welche Implementierungsszenarien geeignet ist. In Kapitel 3 und 4 wurden die technische Umsetzung und weitere Funktionen beschrieben. In diesem Kapitel werden nun zwei Bewertungsmatrizen definiert, diese mit Bewertungen gefüllt und anschließend die Bewertungsergebnisse interpretiert und diskutiert.

Die Bewertungsmatrizen beziehen sich zum einen auf die Bewertung der Funktionen des jeweiligen Tools, sowie die Möglichkeit zur Abbildung der definierten Anforderungen. Eine zweite Matrix wird dann für die Entwickler-Perspektive angefertigt.

5.1 Bewertungsmatrizen definieren

Die Bewertungsmatrizen besitzen die Form einer Tabelle [Wik21] und jeweils einen Tabellenkopf mit den Spaltenbezeichnungen, eine Tabellenvorspalte mit der Beschreibung der Funktionen und die Bewertung als eigentlichen Tabelleninhalt. Der Tabelleninhalt zeigt so die Zusammenhänge zwischen den Zeilen und Spalten. [Tab21]

5.1.1 Bewertungsmatrix zur Implementierung der Funktionalität

In Kapitel 3 und 4 wurde die technische Umsetzung der Anforderungen oder eventuell vorhandene Umsetzungsansätze im Detail beschrieben. Dabei

lassen sich diese grundsätzlich in zwei Bereiche aufteilen: Backend- und Frontend-Funktionalität. Die Backend-Funktionen umfassen:

- Möglichkeiten zur Datenmodellierung.
- Möglichkeiten zur Datenerstellung, -speicherung, -bearbeitung und -löschung.
- Möglichkeiten zur zentralen Datenbereitstellung.

Es wurde bereits darauf hingewiesen, dass nicht alle Tools über diese Art von Backend-Funktionalitäten verfügen. In Kapitel 3 und Kapitel 4 wurden weiterhin diverse Frontend-Funktionen implementiert und untersucht. Tabelle 5.1 zeigt die volle Bewertungsmatrix der Funktionen.

| Tools Funktionen | Fiori Elements mit CAP | AppGyver | SAPUI5 |
|--|------------------------|----------|--------|
| Datenmodellierung | | | |
| Datenerstellung, -speicherung, -bearbeitung und -löschung | | | |
| Servicebereitstellung | | | |
| Listansicht zur Anzeige aller Produkte | | | |
| Einzelansicht für ein Produkt | | | |
| Maske zum Pflegen eines einzelnen Produkts | | | |
| Integration von Suchfiltern | | | |
| Integration von Paginierung | | | |
| Integration von Bild und PDF-Dateien | | | |
| Integration einer Barcode Scanner Funktionen | | | |
| Nutzung mobiler Funktionen | | | |
| Deployment für unterschiedliche Endgeräte | | | |
| Freie Gestaltungsmöglichkeiten | | | |

Tabelle 5.1: Bewertungsmatrix zur Implementierung der Funktionalität

Der Inhalt der Matrix wird mit Punktwerten belegt. Die Bewertungsmatrix umfasst zwei grundsätzliche Bewertungskriterien: die Umsetzbarkeit der zu bewertenden Funktionalitäten und den Implementierungsaufwand. Die Erfüllung der Kriterien wird auf einer Skala von 0 bis 3 eingestuft. Ein Wert von 0 bedeutet, dass die bewertete Funktion nicht umgesetzt werden kann. Der Aufwand wird im Rahmen dieser Bachelorarbeit wie folgt definiert: Der Aufwand wird gleichgesetzt mit der zeitlichen Dauer der Umsetzung, gemessen in Personentagen. Die Dauer umfasst dabei lediglich die reine technische Implementierungszeit und keine weiteren Aktivitäten, wie Konzeption, Testen oder Dokumentation. Die zeitliche Bewertung folgt dabei dem Wissenstand der Autorin in der jeweiligen Rolle:

- Citizen Developer: SAP AppGyver und SAP Fiori Elements.
- Pro Developer: SAPUI5.

Ein Wert von 1 bedeutet, dass die Funktion zwar implementiert werden kann, aber nur mit einem hohen Aufwand, also einer Umsetzungsdauer von mehr als 1 Personentag. Ein Wert von 2 bedeutet, dass die Umsetzung zwischen 0,5 bis 1 Personentage in Anspruch nimmt. Ein Wert von 3 bedeutet, dass die Funktion schnell umgesetzt werden kann, in wenigen Minuten bis zu einem halben Personentag.

| Erfüllungskriterien: 0 bis 3 |
|------------------------------|
| 0 = nicht umsetzbar |
| 1 = mehr als 1 PT |
| 2 = 0,5 bis 1 PT |
| 3 = weniger als 0,5 PT |

Tabelle 5.2: Erfüllung Kriterien der Bewertungsmatrix

5.1.2 Bewertungsmatrix für die Entwicklerperspektive

Für die Entwicklerperspektive wird eine Matrix zur Bewertung der Entwicklungsumgebungen und Entwicklungstools für die drei Technologien definiert. Folgende Tabelle liefert eine Auflistung aller Bewertungskriterien:

| Tools Entwickle- perspektive | Fiori Elements mit CAP | AppGyver | SAPUI5 |
|--|------------------------|----------|--------|
| Notwendiges technisches Verständnis | | | |
| Bedienbarkeit der Entwicklungsumgebung | | | |
| Vollständigkeit der Funktionen der Entwicklungsumgebung | | | |
| Spezialisierung der Entwicklungsumgebung | | | |
| Einrichtungsaufwand der Entwicklungsumgebung | | | |
| Unabhängigkeit der Entwicklungsumgebung von Betriebssystem | | | |
| Stabilität der Entwicklungsumgebung | | | |
| Dokumentation für Programmierer | | | |
| Debugger | | | |
| Versionsverwaltung | | | |

Tabelle 5.3: Bewertungsmatrix aus der Entwickler-Perspektive

Der Inhalt der Matrix wird auch hier mit Punktwerten belegt, wobei jede Tabellenvorspalte eigene Bewertungskriterien umfasst. Die Erfüllung der Kriterien wird auf einer Skala von 1 (grundsätzlich negativ) bis 3 (grundsätzlich positiv) eingestuft. Die Erfüllungskriterien werden in Tabelle 5.4 detailliert vorgestellt.

| Erfüllungs-kriterien Entwickle-perspektive | 1 | 2 | 3 |
|---|--|--|--|
| Notwendiges technisches Verständnis | Programmierausbildung notwendig | Grundlegende IT-Prinzipien notwendig | Kein IT-Verständnis notwendig |
| Bedienbarkeit der Entwicklungsumgebung | Komplex und nur mit mehrfacher Anleitung/nach Schulung bedienbar | Komplexer, aber nach erstmaliger Nutzung verständlich | Einfach und selbsterklärend |
| Vollständigkeit der Funktionen der Entwicklungsumgebung | Wichtige Funktionen fehlen | Funktionen fehlen, können aber über Erweiterungen installiert werden | Alle notwendigen Funktionen integriert |
| Spezialisierung der Entwicklungsumgebung | exklusiv und nicht für andere Dinge nutzbar | Unterstützung weiterer Programmiersprachen und Anwendungstypen | Offen und Unterstützung vieler Anwendungsfälle |
| Einrichtungsaufwand der Entwicklungsumgebung | Hoch (Manuell, mehrstufig, Dauer > 1 Stunde) | Mittel (Manuell, Konfiguration notwendig, Dauer > 10 Minuten < 1 Stunde) | Niedrig (Automatisch, nur wenig Konfiguration, Dauer < 10 Minuten) |
| Plattformunabhängigkeit der Entwicklungsumgebung | Spezifisch | Teilweise abhängig | Systemunabhängig |
| Stabilität der Entwicklungsumgebung | Instabil inklusive Datenverlust | Gelegentliche Instabilität | Stabil |
| Dokumentation für Programmierer | Nicht vorhanden | Unvollständig | Vollständig |
| Debugger | Nicht vorhanden | Komplexe Nutzung | Einfache Nutzung |
| Versionsverwaltung | Nicht integriert | Integrierbar | Integriert |

Tabelle 5.4: Erfüllungskriterien aus der Entwickler-Perspektive

5.2 Durchführung der Bewertung

5.2.1 Bewertung zur Implementierung der Funktionalität

An dieser Stelle sollte darauf hingewiesen werden, dass die Bewertung für den Aufwand subjektiv ist und aus der Sicht anderer, insbesondere erfahrener Entwickler, abweichen kann. Die Backend-Funktionalitäten wie Datenmodellierung, die Unterstützung von CRUD-Funktionalitäten, sowie die Servicebereitstellung mit Fiori Elements in Verbindung mit CAP wurden bereits in Kapitel 3.1.2 vorgestellt. Durch die sehr gute Integration in die Entwicklungsumgebung, können dort alle Dinge mit geringem Aufwand über Eingabemasken realisiert werden. SAP AppGyver bietet ebenfalls Funktionalitäten an, Datenstrukturen anzulegen und Daten zu speichern. Zur Datenmodellierung gibt es ein spezielles Data-Tool. (siehe Abbildung 5.1).

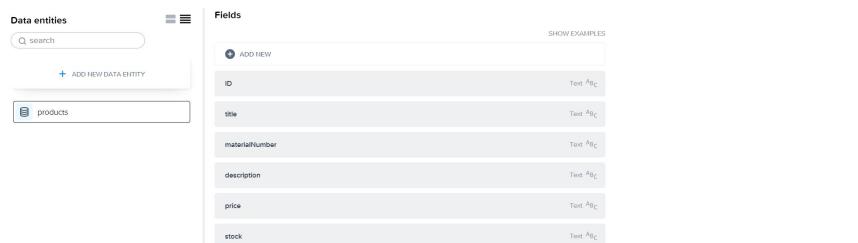


Abbildung 5.1: Daten Modellierung in AppGyver

Unter „Capabilities“ sieht man, dass die Standard-Operationen auf die Daten direkt in eine AppGyver-Anwendung integriert werden (Abbildung 5.2). Grundsätzlich werden die Daten jedoch direkt und nur in der Anwendung gespeichert. Eine zentrale Bereitstellung als OData-Service ist mit AppGyver nicht möglich.

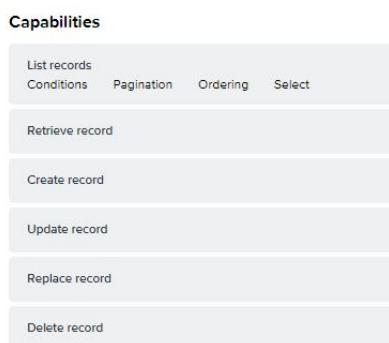


Abbildung 5.2: Daten-Capabilities in AppGyver

SAPUI5 unterstützt als reine Frontend-Technologie nicht die Erstellung von Backend-Datenstrukturen und OData-Services. Für die Integration eines Backend-Layers müsste analog zu SAP Fiori Elements das Cloud Application Programming-Model (oder eine andere Backend-Technologie) hinzugenommen werden. Da es hier jedoch keine einfache Integration in die Entwicklungsumgebung (VS-Code) gibt, müsste der Backend-Part manuell implementiert werden. Dies ist zwar in Sachen Implementierungsaufwand überschaubar (Quelltext 5.1 zeigt die Definition der Datenstrukturen, Quelltext 5.2 das Exponieren als OData-Service), allerdings muss der Entwickler neben dem SAPUI5- auch das CAP-Programmiermodell verstehen.

```
using { cuid } from '@sap/cds/common';
namespace db;
entity Products: cuid{
    title: String(1000);
    description: String(1000);
    materialNumber: Integer;
    price: String(100);
    stock: String(100);
}
```

Quelltext 5.1: Datenmodellierung in der `model.cds`

```
using db from './db/model';
service PublicService{
    entity Products as projection on db.Products;
}
```

Quelltext 5.2: Service Bereitstellung in der `public_service.cds`

Tabelle 5.5 stellt die Bewertung des Implementierungsaufwands zur Dateimodellierung, Datenerstellung, -speicherung, -bearbeitung und -löschung, sowie die Servicebereitstellung dar.

| Funktionen \ Tools | Fiori Elements mit CAP | AppGyver | SAPUI5 |
|--|------------------------|----------|--------|
| Datenmodellierung | 3 | 3 | 0 |
| Datenerstellung, -speicherung, -bearbeitung und -löschung | 3 | 3 | 0 |
| Servicebereitstellung | 3 | 0 | 0 |

Tabelle 5.5: Bewertung der Backend-Funktionen

In Kapitel 3 wurde eine Anwendung implementiert, die eine Listenansicht, eine Einzelansicht und eine Maske zur Pflege eines einzelnen Produkts

enthält. Tabelle 5.6 zeigt die Bewertungsergebnisse des Implementierungsaufwands dieser Funktionalitäten:

| Tools Funktionen | Fiori Elements mit CAP | AppGyver | SAPUI5 |
|--|------------------------|----------|--------|
| Listansicht zur Anzeige aller Produkte | 3 | 3 | 2 |
| Einzelansicht für ein Produkt | 3 | 2 | 2 |
| Maske zum Pflegen eines einzelnen Produkts | 3 | 2 | 1 |

Tabelle 5.6: Bewertung der Grundlistenfunktionen

Hier lässt sich herausstellen, dass Fiori Elements diese Sichten in Form eines Floorplans direkt bereitstellt und der Implementierungsaufwand entsprechend gering ist. In AppGyver und SAPUI5 müssen diese Dinge manuell nachgebaut werden, was einen größeren Aufwand bedeutet. AppGyver ist durch die Möglichkeit zum Drag&Drop hier jedoch deutlich schneller. In Kapitel 4 wurden weitere Funktionen untersucht, um Fiori Elements, AppGyver und SAPUI5 eingehender zu betrachten. Der geschätzte Implementierungsaufwand und Umsetzbarkeit dieser Funktionen werden in Tabelle 5.7 dargestellt.

| Tools Funktionen | Fiori Elements mit CAP | AppGyver | SAPUI5 |
|--|------------------------|----------|--------|
| Integration von Suchfiltern | 3 | 3 | 2 |
| Integration von Paginierung | 3 | 1 | 3 |
| Integration von Bild und PDF-Dateien | 1 | 3 | 3 |
| Integration einer Barcode Scanner Funktionen | 1 | 3 | 1 |
| Nutzung mobiler Funktionen | 1 | 2 | 1 |
| Deployment für unterschiedliche Endgeräte | 1 | 3 | 1 |
| Freie Gestaltungsmöglichkeiten | 1 | 3 | 3 |

Tabelle 5.7: Bewertung der weiteren Funktionen

Suchfilter und Paginierung lassen sich sehr einfach in Fiori Elements umsetzen, da sie Teil des Floorplans sind. Alle weiteren Dinge sind jedoch mit hohem Aufwand umzusetzen, da sie aufwändig über Erweiterungen hinzuprogrammiert werden müssten. In AppGyver lassen sich alle weiteren Funktionen umsetzen, die meisten mit geringem Aufwand. Lediglich die Paginierung erfordert eine komplexere Umsetzung. Mit SAPUI5 können ebenfalls alle Funktionen abgebildet werden. Allerdings ist hier der Implementierungsaufwand stets höher durch das notwendige Coding.

5.2.2 Bewertung von der Entwicklerperspektive

Wie auch die Bewertung der Funktionen, ist die Beurteilung der Entwicklerperspektive mit subjektiven Eindrücken durch die Erfahrung der Autorin geprägt. Auch hier wurden die Grundkenntnisse während der Auffertigung der Thesis erarbeitet und können von den Ergebnissen erfahrener Anwender abweichen. Tabelle 5.8 zeigt die Bewertungsergebnisse aus der

| Tools Entwickler- perspektive | Fiori Elements mit CAP | AppGyver | SAPUI5 |
|--|------------------------|----------|--------|
| Notwendiges technisches Verständnis | 3 | 2 | 1 |
| Bedienbarkeit der Entwicklungsumgebung | 3 | 2 | 1 |
| Vollständigkeit der Funktionen der Entwicklungsumgebung | 2 | 3 | 3 |
| Spezialisierung der Entwicklungsumgebung | 2 | 1 | 3 |
| Einrichtungsaufwand der Entwicklungsumgebung | 3 | 3 | 1 |
| Unabhängigkeit der Entwicklungsumgebung von Betriebssystem | 3 | 3 | 2 |
| Stabilität der Entwicklungsumgebung | 2 | 3 | 3 |
| Dokumentation für Programmierer | 2 | 2 | 1 |
| Debugger | 3 | 1 | 3 |
| Versionsverwaltung | 3 | 2 | 3 |

Tabelle 5.8: Bewertungsmatrix aus der Entwicklerperspektive

Entwicklerperspektive. Begründete Erklärungen finden sich nachfolgend.

Das notwendige technische Verständnis für die Umsetzung von Applikationen ist nicht einfach zu bewerten. Tatsächlich verlangt die Umsetzung von Fiori Elements in BAS lediglich das Ausfüllen von Formular-Dialogen und ist per Definition No-Code. Möchte man jedoch an irgendeiner Stelle vom Standardverhalten abweichen, dann ist das tiefgreifende Verständnis von JavaScript, SAPUI5 und Co notwendig. Da die Anforderungen jedoch mit der Standardfunktionalität umgestzt werden konnten, ist die Bewertung hier entsprechend gut (Bewertung 3). SAP AppGyver verlangt das grundlegende Verständnis von Kernprinzipien in der IT, wie technische Schnittstellen (REST, OData), UI-Events, (komplexe) Formeln, Data-Binding und Daten-, bzw. Logikflüsse. Eine Anwendung mit geringen IT-Kenntnissen wird hier bei der Erstellung deutlich überfordert sein. Für SAPUI5 ist definitiv eine Programmierausbildung notwendig und wird deswegen mit 1 bewertet.

Bei BAS handelt es sich zwar eine flexible Entwicklungsumgebung, die in vielen Bereichen ähnlich zu bedienen ist wie VS-Code, die Dialoge für die Fiori Elements-Entwicklung sind jedoch gezielt einfach gehalten und verstecken die Komplexität. Composer Pro für SAP AppGyver ist eine spezialisierte Umgebung, die versucht, die Funktionalitäten möglichst einfach abzubilden. Durch die Fülle an Funktionen sind manche Dinge jedoch komplex zu bedienen (Bewertung 2). Bei VS-Code handelt es sich um ein sehr mächtiges Werkzeug, das zwar Wert auf eine einfache Bedienung legt, viele Funktionen aufgrund der Komplexität jedoch erst nachgeschlagen werden müssen (Bewertung 1).

Der Einrichtungsaufwand von BAS und AppGyver Composer Pro (Enterprise-Version) ist mit wenigen Mausklicks durchführbar, da es sich hierbei um Software-as-a-Service-Umgebungen handelt. Beide müssen nur vom SAP BTP Service Marketplace zum eigenen SAP BTP-Subaccount hinzugefügt werden. Danach stehen sie für die Entwicklung bereit. Der Einrichtungsaufwand für die AppGyver Community-Version ist sogar noch geringer, da man sich lediglich bei AppGyver Composer Pro registrieren und anmelden muss. Der Einrichtungsaufwand in VS-Code für die Entwicklung der SAPUI5-Anwendung ist deutlich höher, da hier die Umgebung lokal auf dem Rechner installiert wird und noch weitere Tools, Generatoren und Extensions heruntergeladen und installiert werden müssen. Deswegen wird

auch hier die geringste Punktzahl vergeben.

Grundsätzlich sind alle drei Entwicklungsumgebungen unabhängig vom Betriebssystem. Bei VS-Code müssen jedoch alle Dinge lokal installiert werden, mitunter auch betriebssystemspezifisch. Im Kontext der Arbeit gab es an wenigen Stellen Unterschiede zwischen Windows und MacOS, weswegen VS-Code (SAPUI5) mit 2 bewertet wurde.

Ein Nachteil von web-basierten-Umgebungen ist die Tatsache, dass man abhängig von der Verfügbarkeit des Anbieters ist. Während des Anfertigens der Thesis stürzte das BAS zweimal und war danach für mehrere Stunden nicht verfügbar. Deswegen wird BAS (Fiori Elements) hier nur mit 2 Punkten bewertet. Composer Pro und VS-Code liefen komplett stabil.

Ressourcen für die Unterstützung von Entwicklern sind Dokumentationen, Online-Tutorials, Online-Foren, Bücher und E-Books. Eine Stärke von SAPUI5 ist die umfangreiche Dokumentation mit Control- und Code-Beispielen und einer vollständigen API-Dokumentation. Deswegen erhält SAPUI5 hier die volle Punktzahl. Für AppGyver und Fiori Elements sind deutlich weniger Ressourcen verfügbar, weswegen sie hier schlechter bewertet werden.

Das Debugging ist während der Entwicklung einer Anwendung sehr wichtig, um die Logik zu überprüfen. Zum Debuggen der UI5-Webanwendung können die Browser-Tools (Google DevTools oder Microsoft Edge DevTools) verwendet werden, die sehr bedienungsfreundlich sind. Mit dem Debugger von AppGyver Community Edition kann der Entwickler die Mobile App zwar diagnostizieren, aber die Bedienung des Debuggers ist sehr kompliziert. Außerdem ist die Verbindung zwischen Mobile App und AppGyver Debugger nicht immer stabil. Der Debugger ist derzeit für die SAP Enterprise-Edition sogar gar nicht verfügbar, weswegen die Bewertung hier bei 1 Punkt liegt. Die Entwicklung mit Fiori Elements erfordert kein Debugging.

Obwohl es sich bei der Erstellung der Fiori-Anwendung um No-Code handelt, wird im Hintergrund automatisch Quellcode generiert. Zur Verwaltung des Quellcodes in Fiori Elements und SAPUI5 gibt es Tools wie Git. Basierend auf dem Quellcode kann der Entwickler mit git mehrere Code-Linien (Branches) gleichzeitig, sowie Entwicklungs-Snapshots (Commit) verwalten. In BAS und VS-Code ist git nativ integriert und dementsprechend wird

hier die volle Punktzahl vergeben. Das “Release Management” in der AppGyver Enterprise-Edition bietet eine einfache Snapshot und Rollback Funktion für die Versionsverwaltung [Moh22], aber keine vollumfängliche Funktionalität wie git. So kann man beispielsweise nur eine einzelne Release-Linie verwalten. In der AppGyver Community-Edition ist das „Release Management“ gar nicht verfügbar.

5.3 Interpretation und Diskussion

Zwar wurden die einzelnen Funktionen und Aspekte der Entwicklungsumgebungen im Vorherigen punktuell bewertet, es wird jedoch an dieser Stelle keine Summe und damit Rangfolge gebildet, sondern jedes Tool für sich einzeln analysiert und im Hinblick auf die Fragestellungen der Arbeit bewertet:

- Welche Vor- und Nachteile dieser drei Technologien lassen sich durch die exemplarische Umsetzung herausstellen?
- Welche Technologie eignet sich in Zukunft für welche Umsetzungsszenarien?

Zur besseren visuellen Interpretation kommen Netzdiagramme zum Einsatz. Bei diesen werden die einzelnen Kategorien in Form eines Kreises angeordnet und die Werte auf der Basis von 0 bis 3 gezeichnet und verbunden.

5.3.1 SAP Fiori Elements in Verbindung mit Business Application Studio

Das Netz-Diagramm zur Umsetzung der Anforderungen mit Fiori Elements zeigt sehr gut, dass sich die unterstützten Standardfunktionalitäten sehr gut und schnell umsetzen lassen. Deutlich schneller als mit den beiden anderen betrachteten Werkzeugen. Sobald man jedoch von der Standard-Funktionalität der Floorplans abweicht, wird die Umsetzung aufwendig. So wird dann auch aus dem No-Code-Ansatz direkt ein Pro-Code, der eine komplexere Programmierung erfordert. Dies stellt auch eine Frage an die potenzielle Zielgruppe für Fiori Elements: Zwar kann ein geschulter Citizen Developer schnell Ergebnisse erzielen, er muss jedoch bei Erweiterungen entweder auch ein Pro-Developer sein oder auf einen solchen zurückgreifen.

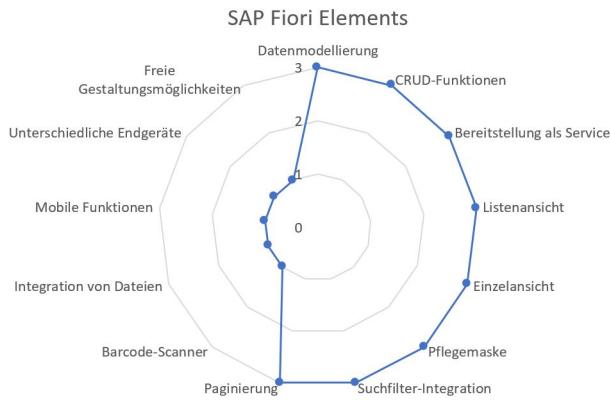


Abbildung 5.3: Bewertung von Umsetzung der Anforderungen mit Fiori Elements

Die Fragestellung stellt sich ebenfalls bei der Betrachtung des Business Application Studios. Hierbei handelt es sich um eine volle Entwicklungs-umgebung, die jedoch spezielle No-Code-Sichten für Fiori Elements und CAP bereitstellt. Der Sprung von No-Code zu Pro-Code kann somit innerhalb von wenigen Mausklicks erfolgen. No-Code- und Pro-Code-Entwickler müssten deswegen in der gleichen Entwicklungsumgebung arbeiten. Dies legt nahe, dass die Zielgruppe eigentlich Pro-Code-Entwickler sind, die für Standard-Anwendungen mit Fiori Elements sehr schnell Ergebnisse erzielen und diese bei Bedarf erweitern können.

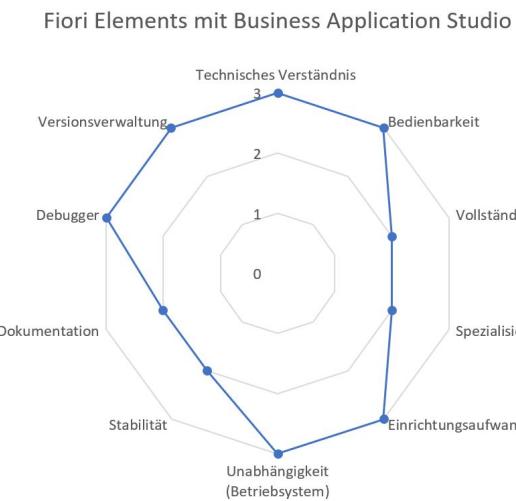


Abbildung 5.4: Bewertung von Fiori Elements mit BAS

Die Vor- und Nachteile von SAP Fiori Elements in Verbindung mit Business Application Studio lassen sich wie folgt zusammenfassen:

| Vorteile | Nachteile |
|---|---|
| <ul style="list-style-type: none"> • Sehr schnelle Umsetzung von Backend- und Frontend-Apps, basierend auf fest definierten Floorplans • Komfortable Entwicklungsumgebung • Integration von fortschrittlichen Entwickler-Tools (git) | <ul style="list-style-type: none"> • Großer Aufwand für Funktionalitäten außerhalb des Floorplans • Keine Freiheitsgrade bei der Gestaltung der Apps • Probleme mit Stabilität von BAS • Wenig Dokumentation für Entwickler |

Tabelle 5.9: Vor- und Nachteile von SAP Fiori Elements in Verbindung mit BAS

Basierend auf den Vor- und Nachteilen, sowie den diskutierten Details, lassen sich folgende Umsetzungsszenarien ableiten:

- Einfache Standardanwendungen zur Darstellung von Daten in Listen oder einfache CRUD-Anwendungen mit fest vorgegebenem Design in Teams von No-Code-Entwicklern
- Standardanwendungen mit sehr geringer funktioneller Abweichung, die von Pro-Entwicklern hinzugefügt werden müssen in gemischten Teams oder in einem Team aus Pro-Code-Entwicklern

5.3.2 SAP AppGyver in Verbindung mit Composer Pro

Das Netzdiagramm zur Umsetzung der Anforderungen mit AppGyver zeigt, dass fast alle der untersuchten Funktionalitäten mit AppGyver umgesetzt werden können und die meisten Funktionen einen sehr geringen Aufwand erfordern. Lediglich die Bereitstellung der Daten als zentrale Service ist nicht möglich.

Im Vergleich zu Fiori Elements dauert die Umsetzung länger und erfordert an diversen Stellen manuellen Low-Code-Entwicklungsaufwand, da die Funktionalitäten nicht per Default bereitstehen. Durch die einfache Bedienung ist die Umsetzung jedoch mit geringem oder mittlerem Zeitaufwand durchzuführen. Und die Vorteile gegenüber Fiori Elements sind auch klar zu benennen: Zusätzliche Funktionen können von der gleichen Entwickler-Gruppe (Low-Code Developer) hinzugefügt werden.

Im Vergleich zu SAPUI5 kann gesagt werden, dass sich beide in vielen

Konzepte wie Data Binding, View Controls/Components oder Events ähnlich sind. Die Umsetzung in AppGyver ist aber im Allgemeinen schneller durchzuführen, die die App über spezialisierte Dialoge visuell umgesetzt werden kann. Ein interessanter Aspekt an den gemeinsamen Konzepten ist, dass ein Citizen Developer damit auch ein grundlegendes Verständnis der Konzepte von SAPUI5 besitzt und dies einen potentiellen Entwicklungspfad aufweist.

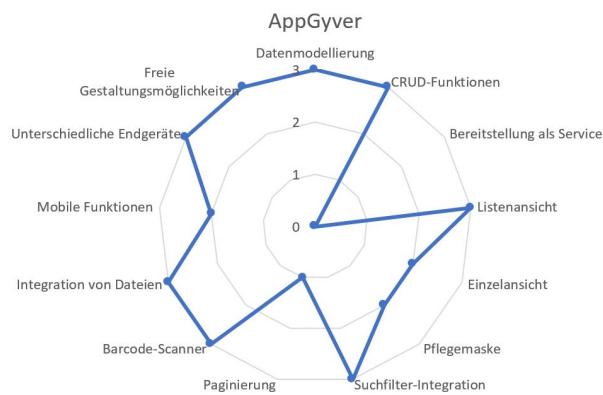


Abbildung 5.5: Bewertung von Umsetzung der Anforderungen in AppGyver

In Sachen Entwicklungsumgebung ist Composer Pro eine spezialisierte LCNC-Entwicklungsumgebung speziell für AppGyver, die alle notwendigen Funktionen in einer Oberfläche bündelt. Er ist zwar für den Citizen Devel-

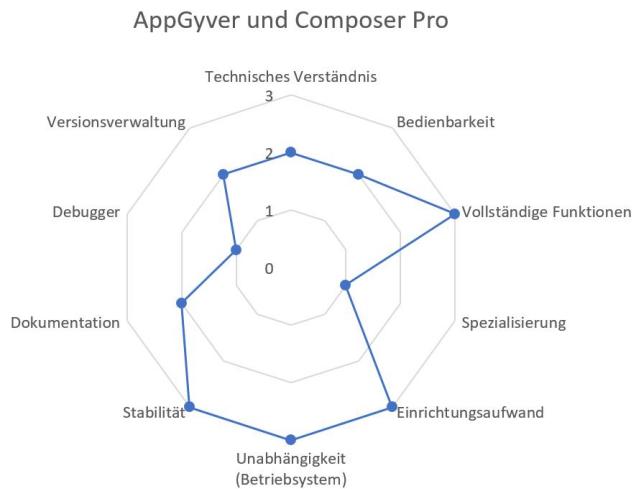


Abbildung 5.6: Bewertung von AppGyver Composer Pro

oper konzipiert, jedoch sind durch die komplexeren Konzepte auch grundlegende IT-Kenntnisse für die Bedienung des Composer Pro erforderlich. Die

Schwächen der Entwicklungsumgebung fallen insbesondere im Vergleich zu Pro-Umgebungen auf: die rudimentäre Quellcode-Versionierung und die mangelhafte Debugger-Integration.

Die Vor-und Nachteile von AppGyver in Kombination mit Composer Pro:

| Vorteile | Nachteile |
|--|---|
| <ul style="list-style-type: none"> • Geringer Aufwand für die Implementierung der meisten Backend- und Frontendfunktionalitäten • Flexibilität im App-Design • Geringer Einrichtungsaufwand der Entwicklungsumgebung • Unabhängigkeit der Entwicklungsumgebung von Betriebssystem • Stabilität von Composer Pro | <ul style="list-style-type: none"> • Wenig Ressourcen für Entwickler • Spezifische Entwicklungsumgebung • Debugger in Enterprise-Edition nicht verfügbar • Quellcode-Versionierung in Community-Edition nicht verfügbar |

Tabelle 5.10: Vor-und Nachteile von AppGyver mit Composer Pro

Aus den Vor- und Nachteilen kann folgendes Umsetzungsszenario abgeleitet werden:

- Durch die verfügbaren Funktionen, die flexible und schnelle Entwicklung, sowie die gute Integration von externen Services, lassen sich sehr viele Anwendungsszenarien umsetzen.
- Die Umsetzung kann dabei von einem Team aus reinen LCNC-Entwicklern erfolgen, ohne Zuarbeit von Pro-Entwicklern. Die Citizen Developer müssen jedoch über grundlegende IT- und spezielle AppGyver-Kenntnisse verfügen.

5.3.3 SAPUI5 in Verbindung mit SAPUI5

SAPUI5 ist eine mächtige und sehr flexible Frontend-Technologie. Das Netzdiagramm zeigt, dass bis auf die Backend-Funktionalitäten alle untersuchten Frontend-Funktionalitäten in SAPUI5 umsetzbar sind. Allerdings sind einige der Funktionalitäten mit hohem Programmieraufwand verbunden, wie z.B. die Pflegemaske eines einzelnen Produkts, die Nutzung mobiler Funktionalitäten und die Deployment für verschiedene Endgeräte.

SAPUI5 abstrahiert zwar bereits viel Logik in den Controls und es ist relativ einfach möglich, aus vielen Controls in einer View eine komplexere Anwendung zu erstellen, die Notwendigkeit die Ablauf-Logik aus zu programmieren (Events, etc.) ist jedoch ein Aufwandstreiber.

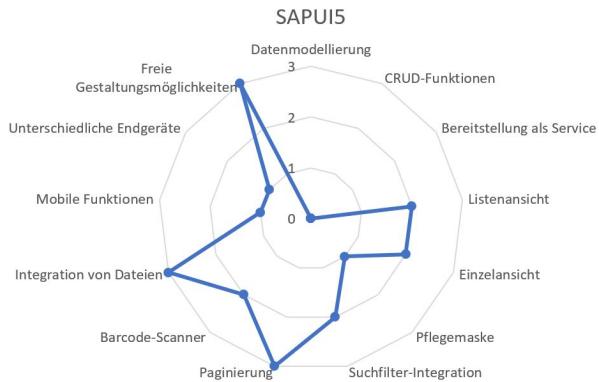


Abbildung 5.7: Bewertung von Umsetzung der Anforderungen in SAPUI5

Visual Studio Code ist eine offene Entwicklungsumgebung und es ist möglich, diese nach Belieben anzupassen. So sind auch alle notwendigen Funktionen für die SAPUI5-Entwicklung integrierbar und insbesondere die IT-Funktionalitäten sehr gut. Allerdings ist der Einrichtungsaufwand hoch

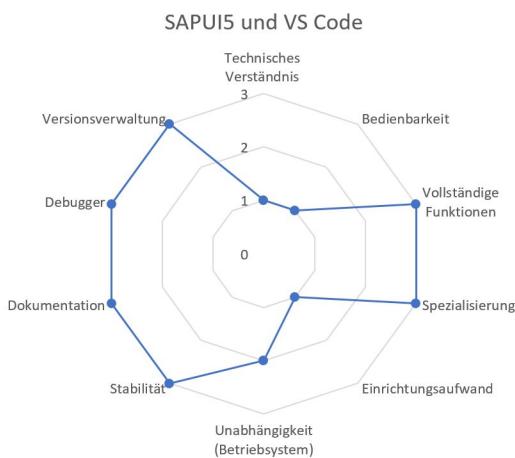


Abbildung 5.8: Bewertung von SAPUI5 mit VS-Code

und die Bedienung von VS-Code als klassische Programmierumgebung komplex. Daher ist SAPUI5 in Verbindung mit VS-Code nur für Pro-Code-Entwickler geeignet.

Die Vor- und Nachteile von SAPUI5 in Kombination mit VS-Code:

| Vorteile | Nachteile |
|--|---|
| <ul style="list-style-type: none"> • Sehr flexible Umsetzung möglich • Weit verbreitete Entwicklungsumgebung • Viele Hilfe-Ressourcen für Entwickler • Sehr gute Entwicklungsumgebung mit fortgeschrittenen Integrationen von Debugger-Funktionalitäten und Versionsverwaltung Tools | <ul style="list-style-type: none"> • Großer Umsetzungsaufwand für verschiedene Funktionalitäten • Großer Einrichtungsaufwand in VS Code • Betriebssystem spezifische Entwicklungsumgebung • Großer Lernaufwand für Anfänger |

Tabelle 5.11: Vor- und Nachteile von SAPUI5 in Kombination mit VS-Code

Anhand der Vor- und Nachteile, sowie der diskutierten Details, lassen sich folgende Umsetzungsszenarien von SAPUI5 benennen:

- Entwicklung von komplexen Frontend-Applikationen, die von der Standard-Funktionalität (Fiori Element-Floorplans) abweichen in einem Team von Pro-Code-Entwicklern.

Kapitel 6

Schluss und Ausblick

Die Ziele dieser Bachelorarbeit waren es, einen konkreten, kundenorientierten Anwendungsfall mit Fiori Elements (in Verbindung mit CAP), SAP AppGyver und SAPUI5 zu entwickeln und dadurch die drei genannten Technologien im Hinblick auf die Vor- und Nachteile zu evaluieren, sowie eine Empfehlung für die Auswahl der Technologien zu liefern.

In Kapitel 2 wurden einige Grundkonzepte für das Verständnis der Arbeit vorgestellt. Hier wurde zunächst der Fokus auf den No Code/Low Code-Ansatz, sowie die Architektur von SAP-Anwendungen auf SAP BTP gelegt. Danach erfolgte die Vorstellung einiger Grundlagen zu Fiori Elements, AppGyver und SAPUI5, sowie den gewählten Entwicklungsumgebungen .

Um die Frage „Wie implementiert man eine benutzerspezifische Anwendung mit Fiori Elements, SAP AppGyver und SAPUI5?“ zu beantworten, wurde in Kapitel 3 der konkrete Entwicklungsprozess des Anwendungsfalls mit den drei benannten Tools beschrieben. Die Implementierungsschritte umfassten die Bereitstellung eines OData-Services mit Fiori Elements in Verbindung mit CAP im BAS, die Erstellung einer Listenansicht, einer Detailansicht sowie einer Maske zur Pflege eines einzelnen Produktes. Um die drei Technologien eingehender zu bewerten, wurden in Kapitel 4 weitere Funktionen untersucht, die über sich aus gängigen Kundenanforderungen ableiten lassen. Beispiele dafür sind die Integration von Suchfiltern, eine Paginierung, der Umgang mit Bild- und PDF-Dateien, sowie Barcode-Scanner-Funktionen. Die Möglichkeiten zum Deployment für unterschiedliche Endgeräte und die freien Gestaltungsmöglichkeiten wurden ebenfalls in Kapitel 4 untersucht.

Über die jeweiligen Ergebnisse hinaus konnte während der Implementierung festgestellt werden, dass AppGyver als LCNC-Werkzeug und SAPUI5 für die Pro-Entwicklung, diverse Architektur- und Anwendungskonzepte teilen:

- SAPUI5 verwendet das MVC-Modell zur Aufteilung der Anwendungskomponenten. AppGyver benennt dies nicht explizit, die Komponenten lassen sich jedoch ebenfalls dieses Schichtenmodell unterteilen.
- Viele Komponenten einer Anwendung finden sich in beiden Welten wieder:
 - UI Control (SAPUI5) – UI Component (AppGyver)
 - Events (SAPUI5) – Events (AppGyver)
 - Callback-Funktion im Controller (SAPUI5) – Ablauffunktion (AppGyver)
- Sowohl in SAPUI5, als auch in AppGyver wird das Konzept des Data Bindings verwendet, um Daten an UI-Elemente zu binden und diese automatisch zu aktualisieren.

In Kapitel 5 erfolgte eine Bewertung der verwendeten Tools. Hierfür wurden zwei Bewertungsmatrizen erstellt. Eine Matrix diente dazu, die Umsetzung der Funktionalitäten zu bewerten. Dazu wurden zwei Kriterien herangezogen: die Umsetzbarkeit und der Umsetzungsaufwand. Die andere Matrix wurde verwendet, um die Technologie aus der Entwickler-Perspektive zu bewerten. Nach Durchführung der Bewertung erfolgte die Diskussion und Ableitung der Vor- und Nachteile der Technologien, sowie die Beantwortung der Frage, welche Technologie sich für welche Einsatzszenarien eignen.

Die Vorteile von Fiori Elements in Kombination mit BAS können wie folgt zusammengefasst werden:

- Sehr schnelle Umsetzung von Backend- und Frontend-Apps, basierend auf fest definierten Floorplans
- Spezifische und komfortable Entwicklungsumgebung
- Integration von fortschrittlichen Entwickler-Tools (git)

Die Nachteile umfassen:

- Großer Aufwand für Funktionalitäten außerhalb des Floorplans
- Keine Freiheitsgrade bei der Gestaltung der Apps
- Probleme mit Stabilität von BAS
- Wenig Dokumentation für Entwickler

Aus den Vor- und Nachteilen können Umsetzungsszenarien abgeleitet werden: SAP Fiori Elements eignet sich für einfache Standardanwendungen zur Darstellung von Daten in Listen oder einfachen CRUD-Anwendungen mit fest vorgegebenem Design in Teams von No-Code-Entwicklern. Es eignet sich aber auch für Standardanwendungen mit sehr geringer funktionaler Abweichung, die von Pro-Entwicklern hinzugefügt werden müssen in gemischten Teams oder in einem Team aus reinen Pro-Code-Entwicklern.

Die Vorteile von AppGyver in Kombination mit Composer Pro lassen sich wie folgt zusammenfassen:

- Geringer Aufwand für die Implementierung von meisten Backend- und Frontendfunktionalitäten
- Flexibel im App-Design
- Geringer Einrichtungsaufwand der Entwicklungsumgebung
- Unabhängigkeit der Entwicklungsumgebung von Betriebssystem
- Stabilität von Composer Pro

Die Nachteile sind:

- Bereitstellung von Services nicht möglich
- Geringe Anzahl an Ressourcen für Entwickler
- Spezifische Entwicklungsumgebung
- Debugger in Enterprise-Edition nicht verfügbar
- Version Management Tool in Community-Edition nicht verfügbar

SAP AppGyver bietet einen flexiblen Rahmen, mit dem sich fast alle Funktionalitäten mit relativ geringem Aufwand umsetzen lassen. Daher ist es für fast alle Anwendungsszenarien in Teams von (LCNC-)Entwicklern mit grundlegenden IT-Grundkenntnissen geeignet.

SAPUI5 in Kombination mit Visual Studio Code dagegen richtet sich ausschließlich an Pro-Developer. Es besitzt folgende Stärken:

- Flexibel im App-Design
- Weit verbreitete Entwicklungsumgebung
- Vielfältige Dokumentation für Entwickler

Die Nachteile sind:

- Größerer Aufwand in der Umsetzung für viele Funktionalitäten
- Großer Einrichtungsaufwand in V- Code

- Hoher Lernaufwand für Anfänger

SAPUI5 ist damit eine sehr flexible Frontend-Technologie, jedoch ist der Implementierungsaufwand im Vergleich zu SAP AppGyver höher. Sie eignet sich demnach für komplexe Frontend-Anwendungsszenarien in Teams von Pro-Code-Entwicklern.

Im Rahmen dieser Arbeit wurden nur ausgewählte Funktion einer Analyse und Bewertung unterzogen. Hier kann als Ausblick der Fokus auf zusätzliche Funktionen gelegt werden. Zum Beispiel sind die Authentifizierung und die Autorisierung sehr wichtige Themen für Geschäftsanwendungen, diese wurden im Rahmen der Thesis jedoch nicht berücksichtigt. Eine Analyse dieser Themen würde eine noch feinere Bewertung zulassen. Gleichfalls ließen sich auch die Anwendungsfälle weiter zeichnen. In Fiori Elements wurden für die Entwicklung der Anwendung nur die beiden Floorplans für Listenansicht und Detailseite verwendet. Mit der Overview Page und der Analytical List Page stehen jedoch auch Floorplan bereit, um analytische Anwendungsszenarien abzudecken. Zudem wurden im Kontext von SAP AppGyver und SAP Fiori Elements nur die LCNC-Funktionalitäten untersucht. Um SAP AppGyver noch flexibler zu machen, können komplett neue und eigene Komponenten vom Benutzer erstellt werden. Dies erfordert jedoch einen Programmieraufwand und ist nur von Pro Developern zu entwickeln. Auch für Fiori Elements wurde zwar auf die Erweiterungsfähigkeit hingewiesen, wie diese zu nutzen ist, wurde jedoch nicht weiter untersucht. Hier würde es sich anbieten, weitere Analysen und Bewertungen zu erarbeiten.

Literaturverzeichnis

- [Ant16] ANTOLOVIC, Miroslav: *Einführung in SAPUI5*, 2. Auflage. Rheinwerk Verlag, 2016. – ISBN: 978-3-8362-8901-6
- [App22a] APPGYVER: *Android builds*. 2022. – URL: <https://docs.appgyver.com/docs/android-builds> [abgerufen am 22.12.2022]
- [App22b] APPGYVER: *App logic*. 2022. – URL: <https://docs.appgyver.com/docs/app-logic> [abgerufen am 29.11.2022]
- [App22c] APPGYVER: *Binding data*. 2022. – URL: <https://docs.appgyver.com/docs/en/binding-data?highlight=data%20binding> [abgerufen am 13.01.2023]
- [App22d] APPGYVER: *Community Announcement*. 2022. – URL: <https://forums.appgyver.com/t/community-announcement/19453> [abgerufen am 22.11.2022]
- [App22e] APPGYVER: *IOS builds*. 2022. – URL: <https://docs.appgyver.com/docs/ios-builds> [abgerufen am 22.12.2022]
- [App22f] APPGYVER: *Theme*. 2022. – URL: <https://docs.appgyver.com/docs/theme?highlight=theme> [abgerufen am 26.12.2022]
- [App22g] APPGYVER: *View Components*. 2022. – URL: <https://docs.appgyver.com/docs/view-components> [abgerufen am 29.11.2022]
- [AS22] ANDREAS SCHAFFRY, Directorate-COMPUTERWOCHE: *Studie No-Code/Low-Code 2022 Licht und Schatten*. 2022. – URL: <https://www.computerwoche.de/a/licht-und-schatten,3553554> [abgerufen am 11.10.2022]

- [CAP22] CAP, SAP: *Serving Media Data*. 2022. – URL: <https://cap.cloud.sap/docs/guides/media-data> [abgerufen am 20.12.2022]
- [Cen21] CENTER, SAP N.: *SAP übernimmt No-Code-Pionier AppGyver*. 2021. – URL: <https://news.sap.com/germany/2021/02/sap-uebernimmt-no-code-pionier-appgyver/> [abgerufen am 11.10.2022]
- [Cod22a] CODE, Visual S.: *Overview*. 2022. – URL: <https://code.visualstudio.com/docs> [abgerufen am 01.12.2022]
- [Cod22b] CODE, Visual S.: *Terminal Basics*. 2022. – URL: <https://code.visualstudio.com/docs/terminal/basics> [abgerufen am 01.12.2022]
- [Com22a] COMMUNITY, SAP: *SAPUI5*. 2022. – URL: <https://community.sap.com/topics/ui5> [abgerufen am 25.11.2022]
- [Com22b] COMMUNITY, SAP: *Sneak Peek on SAP AppGyver Integration Into SAP BTP*. 2022. – URL: <https://groups.community.sap.com/t5/devtoberfest/sneak-peek-on-sap-appgyver-integration-into-sap-btp/ec-p/8958#M17> [abgerufen am 13.10.2022]
- [Con22] CONSULTING, Hecker: *Die Zukunft der Software Entwicklung: Low-Code-Plattformen*. 2022. – URL: <https://www.hco.de/blog/die-zukunft-der-software-entwicklung-low-code-plattformen> [abgerufen am 11.11.2022]
- [CVE22] COEN VAN EENBERGEN, Directorate-TECHZINE: *ServiceNow fully enters low-code market with App Engine Studio*. 2022. – URL: <https://www.techzine.eu/blogs/applications/56875/servicenow-fully-enters-low-code-market-with-app-engine-studio/> [abgerufen am 17.11.2022]
- [Doc22a] DOCUMENTATION, SAP U.: *App Overview: The Basic Files of Your App*. 2022. – URL: <https://sapui5.hana.ondemand.com/#/topic/28b59ca857044a7890a22aec8cf1fee9> [abgerufen am 24.12.2022]
- [Doc22b] DOCUMENTATION, SAP U.: *Step 16: Dialogs and Fragments*. 2022. – URL: <https://sapui5.hana.ondemand.com/sdk/#>

- /topic/4da72985139b4b83b5f1c1e0c0d2ed5a [abgerufen am 28.12.2022]
- [Doc22c] DOCUMENTATION, SAP U.: *Step 19: Aggregation Binding.* 2022.
– URL: <https://sapui5.hana.ondemand.com/sdk/#/topic/bf71375454654b44af01379a3c3a6273> [abgerufen am 28.12.2022]
- [Doc22d] DOCUMENTATION, SAP U.: *Step 25: Remote OData Service.* 2022. – URL: <https://sapui5.hana.ondemand.com/#/topic/44062441f3bd4c67a4f665ae362d1109> [abgerufen am 27.12.2022]
- [Doc22e] DOCUMENTATION, SAPUI5: *Developing Apps with SAP Fiori Elements.* 2022. – URL: <https://sapui5.hana.ondemand.com/#/topic/03265b0408e2432c9571d6b3feb6b1fd> [abgerufen am 28.11.2022]
- [Doc22f] DOCUMENTATION, SAPUI5: *Using SAP Fiori Elements Floor-plans.* 2022. – URL: <https://sapui5.hana.ondemand.com/#/topic/03265b0408e2432c9571d6b3feb6b1fd> [abgerufen am 28.11.2022]
- [doc22g] DOCUMENTATION, ServiceNow P.: *App Engine Studio release notes.* 2022. – URL: <https://docs.servicenow.com/bundle/store-release-notes/page/release-notes/store/platform-app-engine/store-rn-plat-app-engine-aes.html> [abgerufen am 17.11.2022]
- [Ele22] ELEMENTS, SAP F.: *SAP Fiori Design Guidelines.* 2022. – URL: <https://experience.sap.com/fiori-design-web/smart-templates/> [abgerufen am 13.10.2022]
- [Eng20] ENGLBRECHT, Michael: *SAP Fiori Implementierung und Entwicklung,* 3. Auflage. Rheinwerk Verlag, 2020. – ISBN: 978-3-8362-7268-1
- [G222] G2: *Salesforce Platform Reviews and Product Details.* 2022. – URL: <https://www.g2.com/products/salesforce-platform/reviews> [abgerufen am 18.11.2022]

- [GG22] GARTNER GLOSSARY, Directorate-Gartner: *Citizen Developer*. 2022. – URL: <https://www.gartner.com/en/information-technology/glossary/citizen-developer> [abgerufen am 11.11.2022]
- [Gmb22] GMBH p36: *p36 Überblick*. 2022. – URL: <https://p36.io/professional-services/sap-consulting/?lang=de> [abgerufen am 11.10.2022]
- [Gui22] GUIDELINES, SAP Fiori D.: *SAP Fiori Launchpad*. 2022. – URL: <https://experience.sap.com/fiori-design-web/launchpad/> [abgerufen am 24.11.2022]
- [HV22] HEINRICH VASKE, Directorate-COMPUTERWOCHE: *Low-Code-Plattformen auf einen Blick*. 2022. – URL: <https://www.computerwoche.de/a/low-code-plattformen-auf-einen-blick,3544905> [abgerufen am 10.10.2022]
- [Lea22] LEARNING, SAP: *Getting Started With Low-Code/No-Code and SAP Build*. 2022. – URL: https://learning.sap.com/learning-journey/utilize-sap-build-for-low-code-no-code-applications-and-automations-for-citizen-developers/getting-started-with-low-code-no-code-and-sap-build_afaa85b8-93eb-4607-94b8-221c80aa6479 [abgerufen am 18.11.2022]
- [Mar22] MARKETPLACE, Visual S.: *SAP CDS Language Support*. 2022. – URL: <https://marketplace.visualstudio.com/items?itemName=SAPSE.vscode-cds> [abgerufen am 19.12.2022]
- [Mue22] MUESSIG, Peter: *Getting Started with TypeScript for UI5 Application Development*. 2022. – URL: <https://blogs.sap.com/2021/07/01/getting-started-with-typescript-for-ui5-application-development/> [abgerufen am 24.12.2022]
- [Ove22a] OVERVIEW, G2: *228 Listings in Low-Code Development Platforms Available*. 2022. – URL: <https://www.g2.com/categories/low-code-development-platforms> [abgerufen am 11.11.2022]
- [Ove22b] OVERVIEW, G2: *289 Listings in No-Code Development Platforms Available*. 2022. – URL: <a href="https://www.g2.com/cat

- egories/no-code-development-platforms [abgerufen am 11.11.2022]
- [Pag22] PAGERANGERS: *MIME-Typ*. 2022. – URL: <https://pagerangers.com/seo-handbuch/onpage/allgemein/was-ist-ein-mime-typ/> [abgerufen am 20.12.2022]
- [Por22a] PORTAL, SAP H.: *Basics of Theming*. 2022. – URL: https://help.sap.com/docs/SAP_NETWEAVER_750/ab06dedc873746eaba1c041200c068e0/91ebfe2a14204244bd052a03018d6781.html?version=7.5.6&locale=en-US [abgerufen am 26.12.2022]
- [Por22b] PORTAL, SAP H.: *Basics of Theming*. 2022. – URL: https://help.sap.com/docs/SAP_NETWEAVER_750/ab06dedc873746eaba1c041200c068e0/91ebfe2a14204244bd052a03018d6781.html?version=7.5.6&locale=en-US [abgerufen am 26.12.2022]
- [Por22c] PORTAL, SAP H.: *Destinations*. 2022. – URL: https://help.sap.com/docs/SAP_BUILD_CLASSIC/acb8ee53fa0e448f9b4a4125591f5d5a/4be99e779fcb4ff9a8e5179b74bd72cc.html?locale=en-US [abgerufen am 09.12.2022]
- [Por22d] PORTAL, SAP H.: *What Is SAP Business Application Studio*. 2022. – URL: <https://help.sap.com/docs/SAP%20Business%20Application%20Studio/9d1db9835307451daa8c930fb9ab264/8f46c6e6f86641cc900871c903761fd4.html> [abgerufen am 29.11.2022]
- [RJR22] RICHARDSON, Clay; JOHN RYMER, Directorate-FORRESTER: *New Development Platforms Emerge For Customer-Facing Applications*. 2022. – URL: <https://www.forrester.com/report/New-Development-Platforms-Emerge-For-CustomerFacing-Applications/RES113411> [abgerufen am 10.10.2022]
- [SAP22a] SAP: *About CAP*. 2022. – URL: <https://cap.cloud.sap/docs/about/> [abgerufen am 01.12.2022]
- [SAP22b] SAP: *AppGyv*. 2022. – URL: <https://www.appgyver.com/> [abgerufen am 10.10.2022]

- [SAP22c] SAP: *Generator-easy-ui5*. 2022. – URL: <https://github.com/SAP/generator-easy-ui5> [abgerufen am 01.12.2022]
- [Sar21] SARSA, Harri: *New style and theme features + SAP Fiori Theme*. 2021. – URL: <https://blog.appgyver.com/new-style-and-theme-features-sap-fiori-theme-68667796e6fb> [abgerufen am 25.12.2022]
- [Tab21] TABELLE: *Matrix*. 2021. – URL: <https://de.wikipedia.org/wiki/Tabelle> [abgerufen am 28.12.2022]
- [Wik21] WIKIPEDIA: *Matrix*. 2021. – URL: <https://de.wikipedia.org/wiki/Matrix> [abgerufen am 28.12.2022]
- [Wik22a] WIKIPEDIA: *Node.js*. 2022. – URL: <https://en.wikipedia.org/wiki/Node.js> [abgerufen am 14.12.2022]
- [Wik22b] WIKIPEDIA: *SQLite*. 2022. – URL: <https://de.wikipedia.org/wiki/SQLite> [abgerufen am 14.12.2022]
- [Wik22c] WIKIPEDIA: *Visual Studio Code*. 2022. – URL: https://de.wikipedia.org/wiki/Visual_Studio_Code [abgerufen am 01.12.2022]

Anhang A

JSON-Schema

JSON-Schema wird verwendet, um die Struktur von JSON-Dateien zu validieren. Bei der Transformationsengine wird dieses Prinzip bei der Mapping-Datei aus Kapitel ?? angewendet – besonders in Abschnitt ?? wird detailliert auf JSON-Schema eingegangen. Das verwendete Schema ist in der Datei `MappingJsonSchema.json` gespeichert und wie folgt definiert:

```
{
  "$schema": "https://json-schema.org/draft-06/schema#",
  "title": "jsonataExcelMapping",
  "description": "A mapping for JSONata input into excel template",
  "type": "object",
  "properties": {
    "SheetMappings": {
      "description": "mappings for the sheets in the excel template",
      "type": "object",
      "additionalProperties": {
        "description": "different sheetNames",
        "type": "object",
        "properties": {
          "row": {
            "description": "start row in excel sheet where the device
                           data shall be written",
            "type": "integer",
            "minimum": 1
          },
          "category": {
            "description": "category of the sheet (complex data element
                           key)",
            "type": "string"
          },
          "columns": {
            "description": "different columns to be filled",
            "type": "array"
          }
        }
      }
    }
  }
}
```

```
        "type": "object",
        "patternProperties": {
            "^[a-zA-Z]{1,2}$": {
                "description": "the key is the column letter in excel
                                format, the value is the JSONata expression to
                                filter the right device data for this column",
                "type": "string"
            }
        },
        "additionalProperties": false,
        "minProperties": 1
    },
    "mandatoryColumns": {
        "type": "array",
        "description": "a list of those columns that are required
                        to be filled",
        "default": [],
        "items": {
            "type": "string",
            "pattern": "^[a-zA-Z]{1,2}$"
        }
    },
    "required": ["row", "columns"]
},
{
    "ElementMappings": {
        "description": "mappings for booleans, date formats and/or
                        specific data elements",
        "type": "object",
        "properties": {
            "Boolean": {
                "description": "mapping for boolean values in this template",
                "type": "object",
                "properties": {
                    "true": {
                        "description": "mapping of true in this agency",
                        "type": ["number", "string", "boolean"]
                    },
                    "false": {
                        "description": "mapping of false in this agency",
                        "type": ["number", "string", "boolean"]
                    }
                },
                "additionalProperties": false,
                "required": ["false", "true"]
            },
            "Date": {
                "description": "mapping for date or time formats",
                "type": "object",
                "additionalProperties": {
                    "description": "key: date format for p36 / value: date
                                    format in excel for agency",
                    "type": "string"
                }
            }
        }
    }
}
```

```
        "type": "string"
    }
},
"additionalProperties": {
    "description": "data element key",
    "type": "object",
    "additionalProperties": {
        "description": "key: element value for p36 / value: element
                        value for agency",
        "type": ["number", "string", "boolean"]
    }
}
},
"required": ["SheetMappings"]
}
```

Quelltext A.1: JSON-Schema für die Mapping-Datei