Bachelor-Thesis

Analyse und Bewertung von verschiedenen Low-Code-Plattformen für die Zielgruppe des Citizen Developers

Vorgelegt am: 15.08.2023

Von: Ruben Gottfried Lohse
Thümmelstraße 40
04600 Altenburg

Studiengang: Wirtschaftsinformatik
Studienrichtung: Wirtschaft

Seminargruppe: WI20

Matrikelnummer: 4004054

Praxispartner: BE-terna GmbH
Bornaer Str. 19
04288 Leipzig

Gutachter: M.Sc. Florian Becker (BE-terna)
Prof. Dr. Andreas Munke (Staatliche Studienakademie Glauchau)

STUDIEREN IM MARKT

Themenblatt Bachelorthesis

Studiengang Wirtschaftsinformatik

Student:	**Ruben Gottfried Lohse**
Matrikelnummer:	**4004054**
Seminargruppe:	**4WI20-1**

Thema der Bachelorthesis

Analyse und Bewertung von verschiedenen Low-Code-Plattformen für die Zielgruppe des Citizen Developers

Gutachter/ Betreuer:	M.Sc. Florian Becker
Gutachter (Studienakademie):	Prof. Dr. Andreas Munke

Ausgabe des Themas:	**23.05.2023**
Abgabe der Arbeit an den SG am:	**15.08.2023, bis 14:00 Uhr**

Fröhlich

Prof. Dr. Nils Fröhlich
Vorsitzender des Prüfungsausschusses
Wirtschaft

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

D365FSCM - Microsoft Dynamics 365 Finance & Supply Chain Management
REST - Representational State Transfer
SOAP - Simple Object Access Protocol

1. Einleitung

1.1. Hinführung zum Thema

Aufgrund der rasanten technologischen Entwicklung und Digitalisierung sind Softwareanwendungen zu einem zentralen Bestandteil des beruflichen und persönlichen Lebens vieler, auch nicht in der IT ausgebildeter, Menschen geworden. Aus diesem Grund ist die Nachfrage nach maßgeschneiderten Anwendungen zur Bewältigung individueller Herausforderungen gestiegen. Traditionell erforderte die Entwicklung solcher Anwendungen allerdings umfassende Kenntnisse im Bereich der Programmiersprachen und -konzepte, was den Prozess zeitaufwendig und komplex gestaltete. Zudem herrscht vor allem in der IT-Branche ein hoher Fachkräftemangel mit ca. 137.000 unbesetzten Stellen in Deutschland im Jahr 2022, sodass die hohe Nachfrage nach Softwareanwendungen oft nicht bedarfsgerecht bedient werden kann[1]. In diesem Zusammenhang ist das Aufkommen von Low-Code-Plattformen eine mögliche Antwort auf die Herausforderungen der herkömmlichen Anwendungsentwicklung.

Low-Code-Plattformen stellen einen Paradigmenwechsel in der Softwareentwicklung dar, indem sie die Hürden für die Erstellung von Anwendungen drastisch senken. Die Plattformen versprechen die Entwicklung von komplexen Anwendungen ohne großen Aufwand und vor allem ohne die Notwendigkeit von umfassenden Programmierkenntnissen. Das Konzept hinter Low-Code besteht darin, eine intuitive Umgebung zu schaffen, in der Personen auch ohne Erfahrungen in der Softwareentwicklung schnell und einfach Anwendungen erstellen können, ohne sich um die zugrundeliegende Code-Implementierung kümmern zu müssen. Die Low-Code-Entwicklung erhöht somit nicht nur die Geschwindigkeit der Anwendungsentwicklung, sondern versetzt auch eine breitere Gruppe von Menschen in die Lage Softwarelösungen zu entwickeln. Dabei werden neben professionellen Softwareentwicklern auch die sogenannten Citizen Developer angesprochen.

Für den Begriff „Citizen Developer“ wurde in Punkt 2.1.2. eine Definition formuliert, die für den gesamten Rahmen der Arbeit gilt. Abgekürzt lässt sich sagen, dass ein Citizen Developer ein Endnutzer, ohne eine formale Ausbildung in der Softwareentwicklung, ist, der Anwendungen für seinen jeweiligen Fachbereich selbst entwickeln kann. Dafür nutzt er Low-Code-Plattformen, die ihm von der IT-Abteilung seines Unternehmens zur Verfügung gestellt werden.

[1] Vgl. (KAY, 2023)

1.2. Zielsetzung

In dieser Arbeit soll geklärt werden ob und inwieweit Low-Code-Plattformen ihr Versprechen einhalten können, dass auch Personen ohne Kenntnisse in der Softwareentwicklung eigene Anwendungen erstellen können. Zudem soll untersucht werden welche Low-Code-Plattformen für eine Nutzung durch einen Citizen Developer geeignet sind. Für die Beantwortung dieser Fragen ist es notwendig verschiedene Plattformen miteinander zu vergleichen und anhand allgemein gültiger Kriterien zu bewerten. Dafür soll ein einheitliches, realitätsnahes Szenario entwickelt werden, welches mit einer Auswahl an Low-Code-Plattformen umgesetzt wird. Anhand der Umsetzung sollen die Plattformen miteinander verglichen werden und hinsichtlich ihrer Eignung für den Citizen Developer bewertet werden.
Die Idee der Arbeit entstand bei der Entwicklung einer Anwendung mit der Low-Code-Plattform Microsoft Power Apps. Dabei wurde schon in einem sehr frühen Stadium der Entwicklung bemerkt, dass Entwickler ohne Programmierkenntnisse recht schnell an ihre Grenzen kommen können. Im Rahmen der Entwicklung kam es zu mehreren größeren Herausforderungen, die zwar bewältigt werden konnten, dafür allerdings größere Mengen an manuell geschriebenen Code notwendig waren, sodass die Entwicklung für einen Citizen Developer ab diesem Punkt sehr kompliziert geworden wäre. Es ist somit interessant zu untersuchen, inwiefern sich die Erfahrungen mit Microsoft Power Apps auf andere Low-Code-Plattformen übertragen lassen.

1.3. Methodik

Die vorliegende Bachelorarbeit verwendet einen Mixed-Methods-Ansatz, um die Eignung von Low-Code-Plattformen für die Zielgruppe der Citizen Developers zu analysieren und zu bewerten. Dieser Ansatz kombiniert sowohl qualitative Methoden in Form einer experimentellen Plattformanalyse als auch quantitative Methoden in Form einer Umfrage.

Für die Evaluierung der Eignung von Low-Code-Plattformen für den Citizen Developer wurde ein einheitliches, realitätsnahes Testszenario entwickelt und auf sechs ausgewählten Plattformen umgesetzt. Dieses Testszenario umfasst die Erstellung einer einfachen Anwendung unter Verwendung der Low-Code-Funktionen der Plattformen. Die Plattformanalyse wurden induktiv durchgeführt, indem die Umsetzung eines Testszenarios mit qualitativen und quantitativen Kriterien bewertet wurde und daraus eine Eignung der Plattformen für den Citizen Developer abgeleitet wurde. Die Ergebnisse wurden interpretativ ausgewertet, um die Stärken und Schwächen jeder Plattform im Kontext der Zielgruppe zu ermitteln.
Um die Plattformen hinsichtlich ihres Funktionsumfangs ausreichend bewerten zu können, wurde ein Testszenario entwickelt, dass sowohl die Datenverarbeitung als

auch die Dateneingabe und -ausgabe beinhaltet. Dafür sollte pro Plattform eine Anwendung entwickelt werden, die eine Excel-Datei importieren kann, in der mehrere E-Mail-Adressen aufgelistet sind. Anschließend sollten die enthaltenen E-Mail-Adressen so weit verarbeitet werden, dass für jede E-Mail-Adresse eine individuelle E-Mail versendet wird.
Bei der Auswahl der Bewertungskriterien wurde darauf geachtet, dass sowohl allgemeine Kriterien als auch Kriterien, die im Zusammenhang zur Eignung der Plattform für den Citizen Developer stehen, ausgewählt werden.
Folgende allgemeine Kriterien wurden ausgewählt und anhand einer Recherchearbeit bewertet:

- Vorteile & Nachteile
- Kosten
- Datenschutz & Sicherheit

Die folgenden Kriterien, die im Zusammenhang mit der Eignung der Plattform für den Citizen Developer stehen, wurden ausgewählt und anhand der Umsetzung des Testszenarios bewertet:

- Lernkurve & Nutzerfreundlichkeit
- Interoperabilität mit anderen Systemen und Datenquellen
- Umsetzungsdauer
- Dokumentation & Support
- Herausforderungen
- Low-Code Umsetzung

Für die Auswahl der Plattformen war die Größe und Bekanntheit der Plattform das Hauptkriterium, mit dem Hintergedanken, dass der Erfolg einer Plattform vor allem auf gute Erfahrungen anderer Nutzer mit dieser zurückzuführen ist, was die Chance auf eine erfolgreiche Umsetzung des Testszenarios tendenziell erhöht. Dabei wurden die Plattformen vor allem anhand ihrer Platzierung im „Gartner Magic Quadrant for Enterprise Low-Code Application Platforms“ ausgewählt[2]. Zudem sollten zum einen Plattformen ausgewählt werden, deren Anbieter auf Low-Code-Plattformen spezialisiert ist, und zum anderen Plattformen, deren Anbieter hauptsächlich in anderen Bereichen tätig ist.

Zusätzlich zur Plattformanalyse wurde eine Umfrage mit 125 Mitarbeitern der BE-terna GmbH durchgeführt, um Informationen über die Anzahl potenzieller Citizen Developer in einem IT-Unternehmen zu sammeln und ihre Wahrnehmung zum Begriff Citizen Developer zu erfassen. Die Umfrageergebnisse wurden deduktiv und quantitativ

[2] Vgl. (VINCENT & a., 2022)

analysiert, um die Anzahl der potenziellen Citizen Developer in Beziehung zur Anzahl der professionellen Softwareentwickler im Unternehmen zu setzen, sowie deren eigene Einschätzung zur Identifikation mit dem Begriff Citizen Developer zu bewerten.

Für die Umfrage wurden Thesen formuliert, welche mit den Umfrageergebnissen verifiziert werden sollten. Anhand dieser Thesen wurden die Fragen für die Umfrage aufgestellt. Dabei wurde zuerst der Tätigkeitsbereich innerhalb von BE-terna erfragt, um die Umfrageteilnehmer bestimmten Berufsgruppen zuordnen zu können. Anschließend wurde erfragt, ob die Umfrageteilnehmer Programmierkenntnisse besitzen und ob das Programmieren zudem ihre Haupttätigkeit darstellt. Damit wurden die Teilnehmer in 3 Gruppen eingeteilt, die jeweils andere Fragen im Verlauf der Umfrage gestellt bekommen sollten. Personen, die angaben, dass sie sowohl Programmierkenntnisse besitzen als auch hauptberuflich Programmieren wurden als professionelle Softwareentwickler eingestuft, die im Anschluss befragt wurden mit welcher Art von Programmiersprachen sie arbeiten. Damit sollte herausgefunden werden, wie verbreitet Low-Code-Plattformen unter professionellen Softwareentwicklern sind. Für diese Gruppe wurde die Umfrage nach dieser Frage beendet, da sie per Definition nicht als Citizen Developer gelten. Personen, die angaben, dass sie zwar Programmierkenntnisse besitzen, allerdings nicht hauptberuflich programmieren wurden als potenzielle Citizen Developer eingeteilt. In die Gruppe der Nicht-Entwickler wurden vorerst alle Personen eingeteilt, die keine Programmierkenntnisse besitzen und somit auch nicht hauptberuflich programmieren. Die Gruppen der potenziellen Citizen Developer und der Nicht-Entwickler wurde anschließend befragt, ob sie bereits Erfahrungen mit einer Low-Code-Plattform sammeln konnten. Zum Schluss sollten beide Gruppen anhand einer Definition zum Begriff „Citizen Developer“ auf einer Skala von 1 bis 5 einschätzen, wie stark sie sich mit diesem Begriff identifizieren können. Personen aus der Gruppe der Nicht-Entwickler, die angaben, dass sie sich mittel bis stark mit dem Begriff identifizieren können, also eine Punktzahl zwischen 3 und 5 gaben, wurden anschließend ebenfalls der Gruppe der potenziellen Citizen Developer zugeordnet.

Die Validität der Plattformanalyse wurde durch die einheitliche Umsetzung eines Testszenarios auf allen Plattformen gewährleistet. Mögliche Limitationen könnten allerdings in der begrenzten Anzahl der ausgewählten Plattformen und den spezifischen Kriterien der Plattformanalyse liegen. Die Umfrage wurde unter Verwendung klar formulierter Fragen und einer angemessenen Stichprobe durchgeführt, um die Validität und Reliabilität der quantitativen Daten sicherzustellen. Die Umfrage wurde unter Berücksichtigung ethischer Richtlinien durchgeführt, um die Privatsphäre und Anonymität der Teilnehmer zu schützen. Die Plattformanalyse wurde auf den Plattformen selbst durchgeführt, ohne sensible Daten einzubeziehen.

Die Kombination einer induktiven Bewertung der Plattformen mit qualitativen und quantitativen Kriterien und einer deduktiven und quantitativen Bewertung der Umfrageergebnisse ermöglichte eine umfassende Analyse der Eignung von Low-Code-Plattformen für die Zielgruppe der Citizen Developer.

2. Grundlagen

2.1. Definitionen und Konzepte

2.1.1. Low-Code

Der Begriff Low-Code bezieht sich nicht auf die Qualität des Codes, sondern auf die Code-Erstellung[3]. Er wurde vom Marktforschungsunternehmen Forrester Research geprägt durch einen Bericht über neue Entwicklungsplattformen für kundenbezogene Anwendungen. Low-Code bedeutet auf Deutsch so viel wie „wenig Code“ oder „wenig Programmieren“, was aussagen soll, dass bei der Low-Code-Entwicklung zu einem großen Teil auf das klassische, manuelle Programmieren verzichtet wird. Es wird stattdessen mit einer grafischen Benutzeroberfläche und vordefinierten, visuellen Bausteinen gearbeitet. Da nicht jedes Element aufwändig programmiert werden muss, wird die Erstellung einer Software erheblich vereinfacht und spart Zeit. Durch die grafische Benutzeroberfläche wird ein intuitives Arbeiten nach dem Baukasten-Prinzip ermöglicht. In der Regel können Nutzer Steuerelemente, wie Eingabefelder, Navigationsbuttons oder Dropdown-Felder, über eine Drag&Drop Funktion in die einzelnen Bildschirmansichten einfügen. Über vorgefertigte Funktionsbausteine können sich Bildschirmansichten und Steuerelemente miteinander verknüpfen lassen. Ebenso können Datenquellen meist über vordefinierte Schnittstellen in die Anwendung eingebunden werden.[4]

Wie der Begriff Low-Code bereits vermuten lässt, kommen Low-Code-Plattformen nicht komplett ohne das klassische Programmieren aus. Laut Ionos.de muss nur bei etwa 20 Prozent aller Entwicklungen auf eine manuelle Eingabe von Code zurückgegriffen werden. Plattformen, die keine Eingabe von Code erfordern, werden als No-Code-Plattformen bezeichnet. Eine No-Code-Plattform ist im Wesentlichen eine Low-Code-Plattform, mit der Anwendungen komplett ohne die Eingabe von Code entwickelt werden. Um einen Ersatz für manuell geschriebenen Code zu schaffen, arbeiten die Anbieter solcher Lösungen oft mit noch aufwendiger gestalteten, grafischen Oberflächen. Der Umgang mit diesen Plattformen ist recht einfach zu erlernen und sie bieten auch zu Beginn der Entwicklungszeit einen sehr schnellen Fortschritt. Allerdings stoßen diese Plattformen mit zunehmender Entwicklungszeit oft an ihre Grenzen, was im schlimmsten Fall ganze Projekte zum Stillstand bringen kann.[5]

[3] Vgl. (t2informatik, kein Datum)
[4] Vgl. (Ionos.de, 2020); Vgl. (ADRIAN, HINRICHSEN, SCHULZ, & VOß, 2020)
[5] Vgl. (Ionos.de, 2020); Vgl. (HELLER, 2023)

Low-Code-Plattformen versprechen viele verschiedene Vorteile gegenüber der traditionellen Softwareentwicklung. Die wichtigsten Vorteile sind in diesem Abschnitt zusammengefasst.

- **Höhere Entwicklungsgeschwindigkeit**: Da das händische Programmieren zu einem großen Teil entfällt, können Anwendungen viel schneller fertiggestellt werden. Vor allem professionelle Softwareentwickler können davon profitieren, da sie sich vollständig auf die Kernfunktionen konzentrieren können und nicht so viel Zeit damit verbringen müssen Fehler im Code zu suchen.[6]
- **Einfachheit**: Die Plattformen sind sehr einfach gehalten, wodurch eine schnelle Einarbeitung ohne viel Vorwissen möglich ist. Der normalerweise von den Entwicklern manuell geschriebene Quellcode wird durch die Anordnung von Elementen in der Entwicklungsumgebung von der Plattform automatisch generiert. Dadurch können die Plattformen auch von Nicht-Programmierern genutzt werden.[7]
- **Kostensenkung**: Aus der höheren Entwicklungsgeschwindigkeit resultiert auch direkt eine Kostensenkung, da weniger Zeit mit der Entwicklung verbracht werden muss. Außerdem kann zum einen auf meist teurere professionelle Softwareentwickler verzichtet werden und zum anderen sind durch die Einfachheit der Plattformen keine kostenintensiven Mitarbeiterschulungen notwendig.[8]
- **Flexibilität**: Zum einen besteht die Flexibilität der Low-Code-Plattformen darin, dass sie eine einfache Veröffentlichung der Anwendung mit flexiblen Bereitstellungstools ermöglichen und durch die einfache Bedienung eine hohe Anpassbarkeit möglich ist.[9] Es scheint außerdem naheliegend, dass die Wahl des Entwicklers sehr flexibel ist, da die Zahl der möglichen Entwickler deutlich höher ist als bei der traditionellen Softwareentwicklung.
- **Höhere Qualität**: Durch Low-Code-Plattformen kann die Qualität der daraus resultierenden Anwendungen gesteigert werden. Da auch Nicht-Programmierer an der Entwicklung einer Anwendung mitwirken können, kann abteilungsübergreifendes Fachwissen eingebracht werden. Das verhindert Silodenken und hilft dabei neue kreative Lösungsansätze zu finden.[10]

[6] Vgl. (Ionos.de, 2020)
[7] Vgl. (Ionos.de, 2020)
[8] Vgl. (Ionos.de, 2020)
[9] Vgl. (Ionos.de, 2020)
[10] Vgl. (Ionos.de, 2020)

Die Zielgruppe von Low-Code-Plattformen ist sehr weit gefasst. Laut Anbietern von Low-Code-Plattformen gehören zu den Zielgruppen, die ihre Produkte nutzen, beispielsweise Business-Analysten, Datenbankadministratoren und professionelle Programmierer. Für jeden Bereich gibt es zwischen den Plattformen unterschiedliche Ausprägungen der Tools, die für den jeweiligen Nutzer optimiert sind. Das Spektrum der Nutzer geht vom Endanwender bis hin zum IT-Profi. Die wichtigste Zielgruppe von Low-Code-Plattformen bewegt sich genau dazwischen. Das sind zum einen IT-affine Mitarbeiter der Fachbereiche und zum anderen IT-Spezialisten, die sich über die Jahre hinweg in einer Fachabteilung eingearbeitet haben und nicht mehr offiziell als Softwareentwickler tätig sind. Für diese Gruppe steht der Begriff „Citizen Developer“.[11]

2.1.2. Citizen Developer

Für die Arbeit wird folgende Definition für den Begriff Citizen Developer verwendet:

> *Der Begriff Citizen Developer beschreibt Mitarbeiter, die keine formale Ausbildung in der Softwareentwicklung, aber ein gewisses technisches Verständnis haben, wodurch sie in der Lage sind Anwendungen für ihren jeweiligen Fachbereich zu erstellen[12]. Citizen Developer schaffen eigene Lösungen für Anforderungen in ihrem Arbeitsalltag, da sie als Endnutzer und Experten ihres Fachbereichs die Prozesse innerhalb ihrer Abteilung sehr gut kennen.[13] Dafür nutzen sie meist Low-Code-Plattformen, in denen mit Drag&Drop und einfachen Skripten beispielsweise Prozesse automatisiert, Tätigkeiten zeitgesteuert oder Verbindungen zwischen bestehenden Programmen geschaffen werden.*

Der Begriff Citizen Developer bedeutet also so viel wie „Fachbereichsentwickler“. Darunter fallen vor allem technisch versierte Mitarbeiter ohne umfangreiche Programmierkenntnisse, die Anwendungen für ihre Abteilung erstellen. Die Entwicklung der Anwendungen stellt dabei allerdings nicht ihre Haupttätigkeit dar. Citizen Developer verkürzen den Weg von der Idee bis hin zur fertigen Anwendung, da der ganze Prozess in einer Hand liegt. Sie verstehen die Probleme, Aufgaben, Ziele und Prozesse ihrer Abteilung und können bei Bedarf sofort eigenständig Anwendungen erstellen.[14]

Der Begriff Citizen Developer entstand im Zuge des Aufkommens von Low-Code-Plattformen. Allerdings ist das Prinzip, dass Mitarbeiter aus den Fachbereichen eigene IT-Lösungen entwickeln nichts neues. Seit vielen Jahren schon werden von IT-affinen Mitarbeitern eigenständig Lösungen für Probleme in ihren Fachbereichen entwickelt,

[11] Vgl. (NOACK, 2018); Vgl. (HELLER, 2023)
[12] Vgl. (go-make-it.de, 2022)
[13] Vgl. (ETEMADIAN, 2023)
[14] Vgl. (ETEMADIAN, 2023)

für die von der IT-Abteilung keine offizielle Lösung zur Verfügung gestellt wird. Die IT-Abteilung hat in der Regel keine Kenntnisse von der eigenständigen Entwicklung, die somit oft nicht den organisatorischen Anforderungen hinsichtlich Kontrolle, Dokumentation, Sicherheit und Zuverlässigkeit entsprechen. In diesem Fall spricht man von der Schatten-IT.[15]

Der Unterschied zwischen dem Citizen Development und der Schatten-IT ist, dass Citizen Developer eine standardisierte Entwicklungsumgebung von der IT-Abteilung bereitgestellt bekommen, mit der sie arbeiten können. Somit können Anwendungen auf einer einheitlichen technologischen Basis entwickelt werden, die von Anfang an Teil der IT-Infrastruktur des Unternehmens sind. Somit kann das Unternehmen von der Eigeninitiative und Innovationen der Mitarbeiter profitieren, aber gleichzeitig eine zentrale Überwachung durch die IT-Abteilung ermöglichen, um die sicherheitsrelevanten Anforderungen sicherzustellen.[16]

2.2. Umfrage: Citizen Developer bei BE-terna

2.2.1. Ziel

Um einen Eindruck über die Anzahl potenzieller Citizen Developer in einem IT-Unternehmen zu bekommen, soll eine Umfrage mit allen Mitarbeitern der BE-terna GmbH in Deutschland durchgeführt werden. Es soll aufgezeigt werden, dass, vor allem in IT-Unternehmen, eine große Anzahl an Citizen Developern existiert, die dafür eingesetzt werden können, um Anwendungen mit Low-Code-Plattformen zu entwickeln.

Für die Umfrage werden folgende Thesen aufgestellt, die mit den Umfrageergebnissen verifiziert werden sollen:

These 1: Die potenziellen Citizen Developer stellen den größten Anteil in einem IT-Unternehmen dar, noch vor den professionellen Softwareentwicklern.

These 2: Viele Personen in einem IT-Unternehmen können sich mit dem Begriff „Citizen Developer“ identifizieren, vor allem wenn sie die entsprechenden technischen Kenntnisse besitzen.

These 3: Personen, die in Tätigkeitsbereichen arbeiten, in denen technische Kenntnisse notwendig sind, sind eher potenzielle Citizen Developer und können sich auch stärker mit dem Begriff identifizieren.

2.2.2. Aufbau

Um die in Punkt 2.2.1 formulierten Thesen zu verifizieren, wurde eine Umfrage mit dem Aufbau, wie es in Bild 1 zu sehen ist, erstellt. Dafür wird jedem Teilnehmer zuerst die Frage nach seinem Tätigkeitsbereich bei BE-terna gestellt, wobei aus allen relevanten

[15] Vgl. (HAUSCHILDT, kein Datum); Vgl. (ETEMADIAN, 2023)
[16] Vgl. (HAUSCHILDT, kein Datum); Vgl. (ETEMADIAN, 2023)

Bereichen einer ausgewählt werden kann. Darauf folgt die Frage, ob die Person Programmierkenntnisse besitzt, die mit „Ja“ oder „Nein“ beantwortet werden kann. Anschließend werden die Teilnehmer je nach Antwort an verschiedene Fragen weitergeleitet. Personen, die Programmierkenntnisse besitzen kommen zuerst zur Frage, ob das Programmieren auch ihre Haupttätigkeit sei. Wenn sie diese Frage ebenfalls mit „Ja“ beantworten, gelangen sie zu der Frage mit was sie programmieren, bei der aus den Möglichkeiten Skriptsprachen, Abfragesprachen, objektorientierten Sprachen und Low-Code-Plattformen mehrere ausgewählt werden können. Für Personen, die hauptberuflich programmieren ist die Umfrage danach beendet, da diese per Definition nicht mehr dem Begriff Citizen Developer zugeordnet werden können und somit die weiteren Fragen irrelevant wären. Personen, die dagegen keine Programmierkenntnisse besitzen oder jene besitzen, aber das Programmieren nicht ihre Haupttätigkeit ist, kommen zu der Frage, ob sie schon einmal mit einer der zur Auswahl stehenden Low-Code-Plattformen gearbeitet haben. Die zur Auswahl stehenden Plattformen sind jene, die in Punkt **Fehler! Verweisquelle konnte nicht g efunden werden.** genannt werden. Nach dieser Frage kommen die Teilnehmer zur abschließenden Frage, bei der sie sich anhand der Definition für den Begriff Citizen Developer aus Punkt 2.1.2 mit einer Bewertung von 1 bis 5 einschätzen sollen, wie stark diese Definition auf sie zutrifft. Das zugehörige Umfrageformular ist im Anhang dieser Arbeit zu finden.

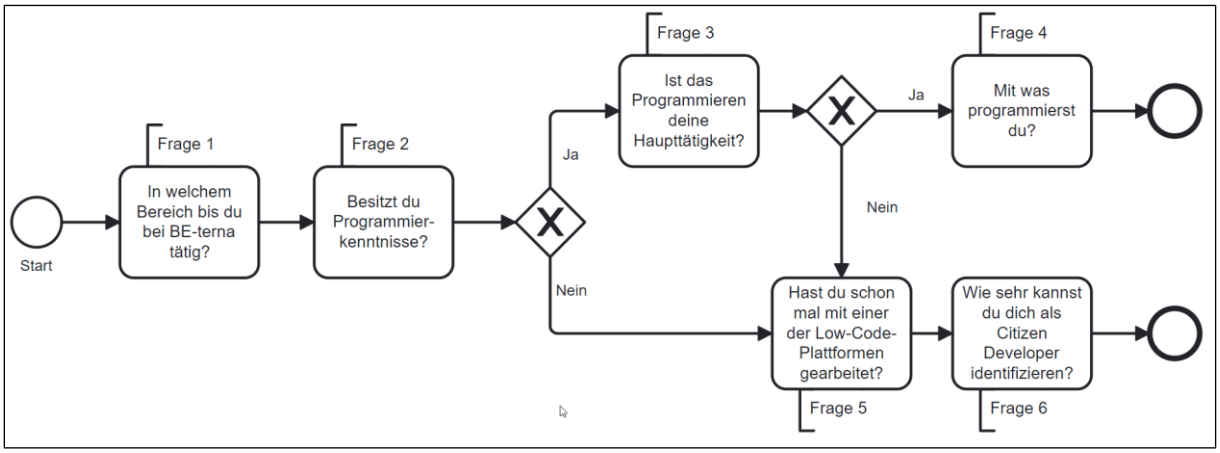

Bild 1:Ablaufbeschreibung der Umfrage zum Thema Citizen Developer

2.2.3. Auswertung

Die Ergebnisse der Umfrage sind im Anhang der Arbeit aufgelistet.

Um die Umfrage richtig auswerten zu können ist zuallererst eine Einteilung notwendig, welche Personen als potenzielle Citizen Developer betrachtet werden. Zu den potenziellen Citizen Developern gehören zum einen alle Personen, die angegeben haben, dass sie Programmierkenntnisse besitzen, aber nicht hauptberuflich programmieren, unabhängig davon, wie diese Personen sich selbst eingeschätzt haben. Bei diesen Personen kann durch die vorhandenen Programmierkenntnisse

davon ausgegangen werden, dass das notwendige technische Verständnis vorhanden ist um als Citizen Developer zu agieren. Zudem werden alle Personen als potenzielle Citizen Developer gezählt, die zwar keine Programmierkenntnisse besitzen, aber sich bei der Selbsteinschätzung mit den Punktzahlen 3 bis 5 bewertet haben. Bei diesen Personen kann ebenso davon ausgegangen werden, dass ein gewisses technisches Verständnis vorliegt, wenn auch keine expliziten Erfahrungen im Bereich Programmierung, und diese sich mit ihren Fähigkeiten eine Tätigkeit als Citizen Developer zutrauen. Im Folgenden werden die Punkt **Fehler! Verweisquelle konnte n icht gefunden werden.** formulierten Thesen aufgegriffen und bewertet.

These 1: Die potenziellen Citizen Developer stellen den größten Anteil in einem IT-Unternehmen dar, noch vor den professionellen Softwareentwicklern.
Die Umfrage wurde von insgesamt 125 Personen ausgefüllt. Von diesen 125 Personen zählen 52 als Citizen Developer, wobei 45 Personen Programmierkenntnisse besitzen und 7 Personen keine Programmierkenntnisse besitzen, aber sich selbst als Citizen Developer eingeschätzt haben. Dazu kommen 49 Personen, die angegeben haben, dass sie hauptberuflich programmieren und 24 Personen, die keiner der beiden Gruppen zugeordnet werden können und somit als Nicht-Entwickler zählen. Die Verteilung ist in Bild 2 dargestellt. Wie zu erkennen ist, stellen die potenziellen Citizen Developer den größten Anteil dar, noch vor den hauptberuflichen Softwareentwicklern. Die 1. These wurde somit verifiziert.

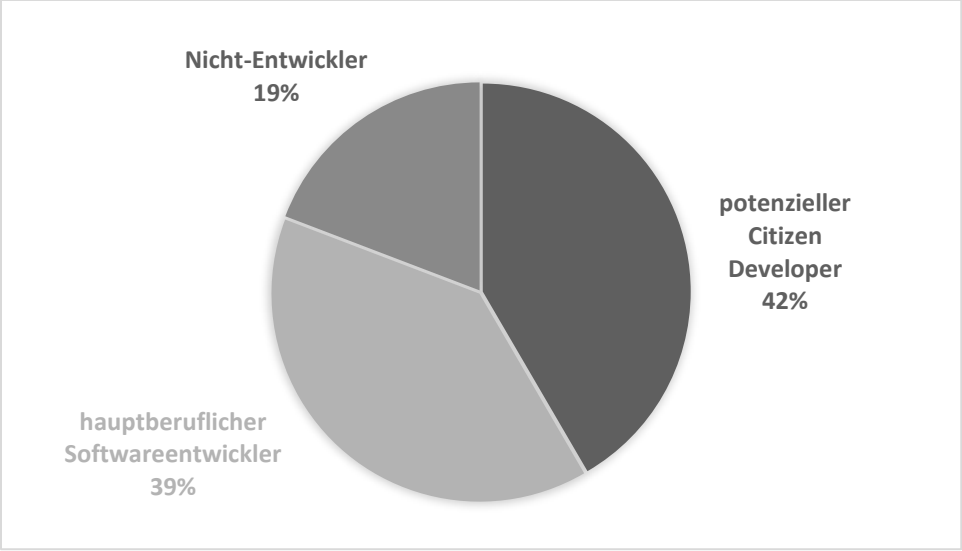

Bild 2: Einordnung der Umfrageteilnehmer

These 2: Viele Personen in einem IT-Unternehmen können sich mit dem Begriff „Citizen Developer“ identifizieren, vor allem wenn sie die entsprechenden technischen Kenntnisse besitzen.
In Bild 3 wird das Verhältnis zwischen der Selbsteinschätzung der Umfrageteilnehmer und deren Programmierkenntnissen abgebildet. Dabei ist ein klarer Trend zu

erkennen. Es ist auffällig, dass mit einem steigenden Verhältnis der Personen mit Programmierkenntnissen zu denen ohne, auch die Punktzahl bei der Selbsteinschätzung steigt. Bei den Personengruppen, die sich stärker mit dem Begriff Citizen Developer identifizieren können, also eine höhere Punktzahl gewählt haben, sind anteilig mehr Personen mit Programmierkenntnissen zu finden. Dieses Ergebnis stimmt insoweit mit der aufgestellten These überein, da technische Kenntnisse, zu denen auch vor allem Programmierkenntnisse zählen, in der Definition zum Citizen Developer explizit erwähnt wurden.
Ein eher unerwartetes Bild zeichnet sich jedoch bei den Personengruppen ab, die sich wenig bis gar nicht mit dem Citizen Developer identifizieren können. In der Gruppe, die sich bei der Selbsteinschätzung nur einen Punkt gegeben hat, geben trotzdem 46% der Personen an, dass sie Programmierkenntnisse besitzen. Also fast die Hälfte dieser Gruppe besitzt laut Definition zumindest einen Teil der notwendigen Kenntnisse und kann sich trotzdem gar nicht mit dem Begriff identifizieren. Auffällig ist auch, dass sich ganze 66% der Personen, die die Selbsteinschätzung abgegeben haben, wenig bis gar nicht mit dem Begriff Citizen identifizieren können, also die Punktzahlen 1 und 2 vergeben haben. Von insgesamt 76 Personen geben 26 die Punktzahl 1 an. Dagegen wird die Punktzahl 5 von nur 6 Personen vergeben. Eine höhere Identifikation wird also tendenziell von weniger Personen gewählt als eine niedrige Identifikation.
Die 2. These kann also nur teilweise verifiziert werden, da zwar eine höhere Identifikation mit dem Begriff Citizen Developer tendenziell eher bei Personen mit Programmierkenntnissen vorhanden ist, aber der Großteil der befragten sich generell weniger mit dem Begriff identifizieren kann.

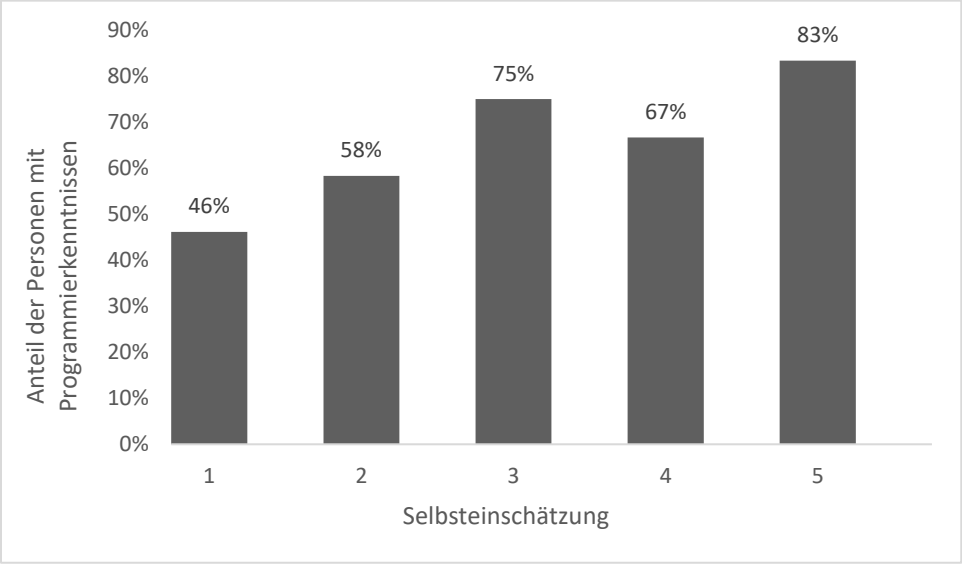

Bild 3: Verhältnis der Selbsteinschätzung zu den Programmierkenntnissen

These 3: Personen, die in Tätigkeitsbereichen arbeiten, in denen technische Kenntnisse notwendig sind, sind eher potenzielle Citizen Developer und können sich auch stärker mit diesem Begriff identifizieren.

In Bild 4 wird das Verhältnis zwischen dem Tätigkeitsbereich und den, nach Definition, potenziellen Citizen Developern, sowie auch den Personen, die sich selbst als Citizen Developer einschätzen, abgebildet. Dabei ist allerdings zu beachten, dass bei 6 der 9 Tätigkeitsbereiche jeweils nur maximal 3 Stimmen abgegeben wurden. Nur die Bereiche Consulting mit 44 Teilnehmern, Management mit 11 Teilnehmern und Sales/Marketing mit 8 Teilnehmern konnten eine größere Anzahl erreichen. Das ist vor allem dadurch begründet, dass sich die meisten Stimmen auf die beiden Bereiche Consulting und Development verteilen. Ganze 92 der 125 Teilnehmer sind in einem der beiden Bereiche tätig. Dazu kommt, dass alle Personen, die angegeben haben, dass sie hauptberuflich programmieren, nicht in der Übersicht beachtet wurden, da diese, laut Definition, keine Citizen Developer sein können und die Frage der Selbsteinschätzung nicht beantworten mussten. Diese Personen sind hauptsächlich im Bereich Development anzufinden, der dadurch ebenfalls in der Übersicht nur 3 Stimmen verzeichnen kann. Außer bei dem Bereich Consulting ist durch die geringe Teilnehmerzahl keine direkte Bewertung der einzelnen Bereiche möglich. Die in Bild 4 abgebildeten Ergebnisse sind also nicht repräsentativ und sollten mit Vorsicht betrachtet werden.

Trotzdem zeichnet sich bei dem Verhältnis zwischen den Tätigkeitsbereichen und den potenziellen Citizen Developern ein zu erwartendes Bild ab. Vor allem in den Bereichen, in denen IT- oder Programmierkenntnisse unabdingbar sind, wie beispielsweise Development oder IT, sind viele potenzielle Citizen Developer anzutreffen. In den Bereichen, die nicht direkt etwas mit IT zu tun haben, wie HR/Recruiting, Finance/Accounting und Administration/Controlling, sind dagegen keine potenziellen Citizen Developer zu finden.

Beim Verhältnis zwischen dem Tätigkeitsbereich und der Selbsteinschätzung der Teilnehmer ist allerdings wieder zu erkennen, dass viele Personen, die laut der Definition einen Citizen Developer darstellen könnten, sich nicht selbst diesem zuordnen würden. Vor allem stechen die Bereiche Development, Architect und IT heraus, bei denen sich keine der Personen, trotz vorhandenen Programmierkenntnissen, dem Begriff Citizen Developer zuordnen würde.

Auch die 3. These kann somit nur teilweise verifiziert werden, da vor allem in den technischen Tätigkeitsbereichen Personen tätig sind, die per Definition als potenzieller Citizen Developer gelten. Ein großer Teil dieser Personen kann sich allerdings wenig bis gar nicht mit dem Begriff „Citizen Developer“ identifizieren.

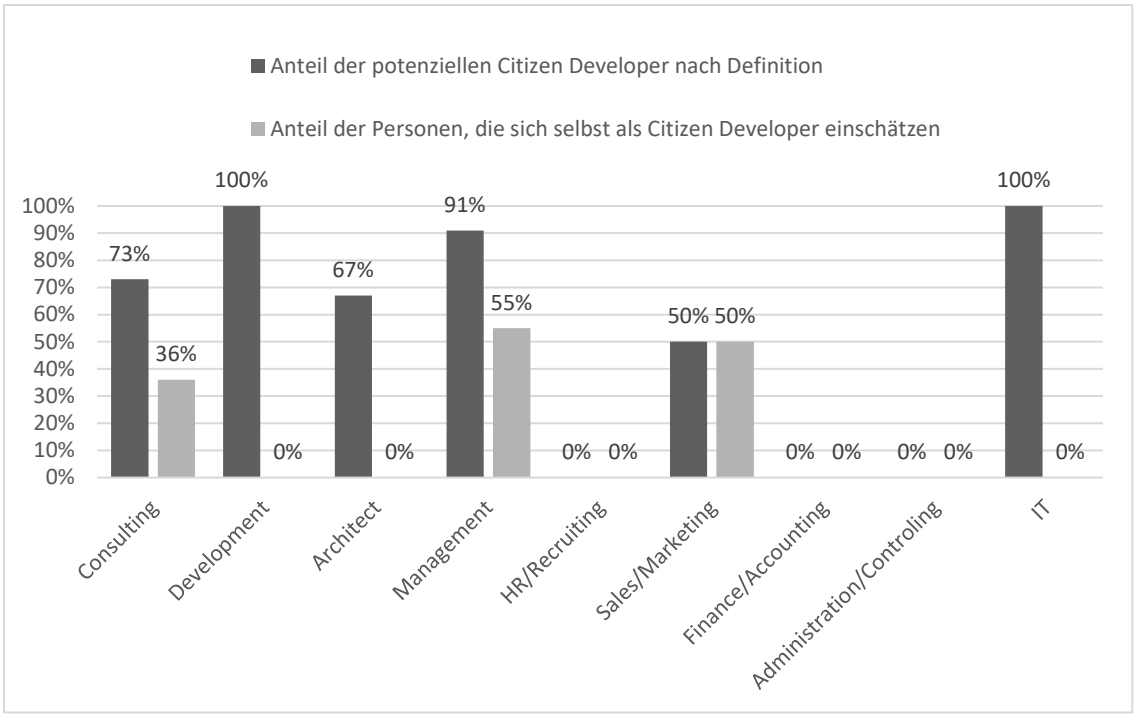

Bild 4: Verhältnis Citizen Developer zu Tätigkeitsbereich

Die hauptberuflichen Softwareentwickler wurden in der Auswertung nicht weiter betrachtet, da diese per Definition keine Citizen Developer sind und somit in Bezug auf den Begriff Citizen Developer nicht relevant sind. Man kann jedoch festhalten, dass zumindest 10 der 49 hauptberuflichen Softwareentwickler regelmäßig mit Low-Code-Plattformen arbeiten. Bei den potenziellen Citizen Developern und den Nicht-Entwicklern haben zusammen 47 der insgesamt 76 Teilnehmer angegeben zumindest schon einmal Erfahrungen mit einer Low-Code-Plattform sammeln zu können.

Aus den Ergebnissen der Umfrage kann entnommen werden, dass ein Großteil der befragten Mitarbeiter bei BE-terna die notwendigen Fähigkeiten mitbringt, um als Citizen Developer tätig werden zu können. Allerdings sieht sich nur ein kleiner Teil dieser Personen tatsächlich als potenzieller Citizen Developer. Das bedeutet, dass für den Einsatz von Low-Code-Plattformen viele potenzielle Entwickler vorhanden wären. Da die meisten Low-Code-Umsetzungen durch einen Citizen Developers eher durch Eigeninitiative von diesem angetrieben werden, müssen sich die potenziellen Entwickler dieses Potenzial erst selbst eingestehen, um wirklich davon profitieren zu können.

2.2.4. Kritische Bewertung und Ansätze zur Verbesserung

Bei der Auswertung der Umfrage sind im Nachhinein mehrere Dinge aufgefallen, die bei der Erstellung der Umfrage verbessert werden könnten. Der erste Punkt ist die Benennung der Tätigkeitsbereiche. Diese hätte spezifischer ausfallen können bzw.

sich noch näher an den tatsächlichen Rollenbezeichnungen orientieren können. Durch die Umfrageergebnisse, aber auch durch direktes Feedback einiger Umfrageteilnehmer, kam heraus, dass es Personen gibt, die sich nicht ausschließlich einem einzigen Bereich zuordnen können. Gerade in den Bereichen Consulting und Development ist es so, dass es viele Personen gibt, die in beiden Bereichen tätig sind. Um hier eine Eindeutigkeit zu schaffen, wäre die tatsächliche Rollenbezeichnung die bessere Wahl gewesen.
Ein weiterer Punkt ist die Definition des Citizen Developers. Diese hätte direkt zum Anfang der Umfrage für alle Teilnehmer einsehbar sein sollen, um sicherzustellen, dass das Thema der Umfrage für alle Personen erkennbar ist. Tatsächlich konnte die Definition nur von den Teilnehmern erreicht werden, die nicht angegeben haben, dass ihre Haupttätigkeit das Programmieren ist. Ebenso ist es denkbar, dass die Definition für die Selbsteinschätzung nicht optimal gewählt wurde. Gerade der 2. Teil der Definition lässt vermuten, dass ein Citizen Developer zwingend eine Low-Code-Plattform durch die IT-Abteilung zur Verfügung gestellt bekommen muss, obwohl dies nicht zwingend der Fall sein muss, sondern hauptsächlich die Fähigkeiten und Kenntnisse ausschlaggebend sind. Das könnte ein möglicher Punkt sein, der die Selbsteinschätzung negativ beeinflusst hat, sodass weniger Personen angegeben haben, sich gut mit dem Begriff Citizen Developer identifizieren zu können.

2.3. Erfahrungswerte aus dem Projekt „D365FSCM Item Gallery“

2.3.1. Projektbeschreibung

Durch das Projekt „D365FSCM Item Gallery“ konnten bereits im Jahr 2022 erste Erfahrungen mit der Low-Code-Plattform Microsoft Power Apps gesammelt werden. Dabei konnte die Erkenntnis erlangt werden, dass ein Citizen Developer bei der Entwicklung einer Anwendung mit einer Low-Code-Plattform recht schnell an seine Grenzen gelangen kann. Besonders dann, wenn sich der Komplexitätsgrad aufgrund von aufkommenden Herausforderungen bei der Entwicklung erhöht und somit umfangreichere Fähigkeiten erforderlich werden.
In dem entsprechenden Projekt sollte eine Lösung mit Microsoft Power Apps umgesetzt werden, um einen kreativen Prozess bei BE-terna-Kunden aus der Fashionbranche zu digitalisieren, der gewissen visuellen Anforderungen unterliegt und somit nicht direkt in einem ERP-System abgebildet werden kann. Bei dem Prozess ging es um die Bewertung und Einordnung von einzelnen Kleidungsstücken in Kollektionen oder Saisons. Dieser wurde bisher so umgesetzt, dass die notwendigen Artikeldaten aus dem ERP-System Microsoft Dynamics 365 Finance & Supply Chain Management, kurz D365FSCM, in eine PDF-Datei exportiert wurden. Die PDF-Datei wurde anschließend ausgedruckt, um sich einen Überblick über alle darin enthaltenen Artikel, inklusive Artikelbilder und Farbpaletten, verschaffen zu können. Um diesen

Prozess digitaler zu gestalten, sollte eine App entwickelt werden, die sich direkt aus dem ERP-System heraus mit einer selbst festgelegten Auswahl an Artikeln öffnen lasst und diese in einem vorgegebenen Layout übersichtlich anzeigt.

Es wurden folgende Anforderungen an die Lösung formuliert:

- **A1**: Ein Raster in einer Canvas-App als Layout für die Anzeige der Artikel
- **A2**: Kacheldesign, um einzelne Artikel und die dazugehörigen Informationen abzubilden
- **A3**: Daten aus dem D365FSCM in die Power App übertragen
- **A4**: Integration der Power App in das D365FSCM
- **A5**: Auswahl der Artikel direkt im D365FSCM mit anschließender Übergabe der Auswahl an die Power App
- **A6**: Anzeigen der Artikelbilder

Die Power App sollte also direkt mit einer Artikelauswahl aus dem D365FSCM heraus gestartet werden und in diesem eingebettet sein. Die vorgenommene Artikelauswahl sollte direkt zum Start der App an diese übergeben werden und inklusive der Artikelbilder in einem Raster mit je einem Artikel pro Kachel angezeigt werden.
Die Anwendung wurde mit einem iterativen Verfahren umgesetzt, was bedeutet, dass die Entwicklung in mehreren Schritten erfolgte und die App mit jedem Schritt weiter verbessert wurde. Dieses Vorgehen wurde allerdings nicht von Anfang an festgelegt, sondern wurde durch diverse Komplikationen notwendig. Dabei entstanden über den Entwicklungszeitraum hinweg verschiedene Versionen der Anwendung.
In Version 1 wurde das Layout für die Anzeige der Artikel umgesetzt, sodass diese ohne eine bestimmte Auswahl an die Anwendung übergeben und angezeigt werden konnte. Dabei wurde allerdings eine vorerst nur Standardtabelle im D365FSCM verwendet, sodass noch nicht alle benötigten Daten übertragen wurden.
Mit Version 2 wurde eine Filterseite in die Anwendung eingebaut, da die Auswahl der Artikel im D365FSCM nicht wie geplant umzusetzen war. Somit wurde die Auswahl der Artikel in die Anwendung verlagert, wodurch die Artikel auch nach anderen Kriterien als der Artikelnummer ausgewählt werden konnten.
In Version 3 musste die Filterlogik angepasst werden, da eine neue, individuelle Tabelle für den Datenexport an die Power App im D365FSCM entwickelt wurde, sodass alle benötigten Daten an die Anwendung übergeben werden konnten. Die in der Anwendung neu verfügbaren Daten mussten auf der Filterseite ebenfalls berücksichtigt werden, sodass eine Anpassung dieser Seite notwendig war.
In Version 4 mussten Performanceprobleme behoben werden, die vor allem aufgrund der umfassenderen Filterlogik entstanden sind. Somit mussten einige Filterfunktionen wieder entfernt werden, um eine ausreichende Performance der Anwendung sicherstellen zu können.

In Version 5 wurde die Schnittstelle zwischen der Power App und dem D365FSCM geändert, da über die Standardschnittstelle die Übertragung der Datensätze auf 2000 Stück limitiert war. Da diese Anzahl für den Kundeneinsatz nicht ausreichend war, musste eine Schnittstelle konfiguriert werden, bei der keine Limitierung der zu übertragenden Datensätze vorhanden ist.

2.3.2. Herausforderungen

Wie bereits angedeutet, gab es bei der Entwicklung der Power App Komplikationen, die eine Überarbeitung der App notwendig zur Folge hatten. Vor allem, dass es nicht möglich war eine Artikelauswahl im D365FSCM vorzunehmen und diese anschließend an die Power App zu übergeben, verursachte ein größeres Umdenken bezüglich der Anforderungen. Zur Lösung des Problems wurde entschieden, dass die Auswahl der Artikel erst in der Anwendung vorgenommen werden soll. Dafür wurde eine Filterseite in der Anwendung eingebaut, mit der die Artikel nach bestimmten Kategorien ausgewählt werden konnten. Für die Entwicklung des Filters wurde es allerdings notwendig Code zu schreiben. Hinzu kam, dass die Daten nicht nur gefiltert, sondern auch vorher aufbereitet werden mussten, da viele Datensätze mehrfach vorhanden waren und zuerst bereinigt werden musste. Das wurde durch die Zusammenstellung der Daten im D365FSCM bedingt. Beispielsweise kann ein Artikel in mehreren Kollektionen vorhanden sein, sodass für jede Kollektion ein eigener Datensatz existiert mit der gleichen Artikelnummer. Für den Filter war es aber notwendig, dass jede Artikelnummer nur einmal vorhanden ist. Somit musste die mehrfach vorhandenen Datensätze vor dem Filtervorgang entfernt werden, was ebenfalls nur durch Code zu lösen war. Nach dem Filtervorgang war es dann wiederum notwendig die bereinigten Daten wiederherzustellen und wieder einzufügen, um sicherstellen zu können, dass alle Informationen angezeigt werden.

Für die anfänglich geplante Variante der Power App wären Kenntnisse über Variablen ausreichend gewesen, um die Anwendung vollständig umzusetzen. Da diese Kenntnisse allein nicht besonders kompliziert sind und schnell zu erlernen sind, wäre die Umsetzung durch einen Citizen Developer auf jeden Fall möglich gewesen.

Durch die notwendigen Anpassungen der Anforderung und weiteren Herausforderungen ist es allerdings notwendig geworden Code zu schreiben, in dem verschachtelte Schleifen und If-Bedingungen verwenden werden. Das war vor allem durch die Filterlogik und die Bereinigung der mehrfach vorhandenen Datensätze notwendig. Für die Umsetzung der kompletten Filterlogik musste Code in der Programmiersprache „Power FX“ manuell geschrieben werden. Mit dem Code mussten Sammlungen erstellt und befüllt werden, die mit ineinander verschachtelten Schleifen mehrfach durchlaufen werden mussten. Innerhalb dieser Schleifen wurden If-Bedingungen verwendet, um für jeden Datensatz der aktuell zu durchlaufenden

Sammlungen zu überprüfen, ob der eingestellte Filter auf diesen zu trifft oder nicht. Ein Ausschnitt aus dem beschriebenen Code ist in **Fehler! Verweisquelle konnte nicht g efunden werden.** zu sehen. Der aufgezeigte Code prüft für die drei Filterkategorien Saison, Kollektion und Warenhierarchie, ob ein Filter gesetzt wurde und befüllt anschließend jeweils eine Sammlung pro Kategorie mit allen Artikeln, die auf die ausgewählten Filteroptionen zutreffen. Des Weiteren waren Kenntnisse über Datentypen notwendig, um auf die Sammlungen, die wie Tabellen aufgebaut sind, und einzelne Spalten dieser zugreifen zu können. An diesem Punkt wäre ein Citizen Developer ohne jegliche Vorkenntnisse im Bereich Programmierung höchstwahrscheinlich an seine Grenzen gekommen.
An diesem Beispiel ist gut zu erkennen, wie schnell sich die Bedingungen bei der Entwicklung einer Anwendung ändern können, sodass schnell deutlich umfassendere Fähigkeiten notwendig sind. Ein Citizen Developer, der die anfangs geplante Anwendung ohne größere Probleme hätte umsetzen können, wäre aufgrund der aufgetretenen Herausforderungen, die nicht von der Plattform selbst verursacht wurden, sehr früh im Projekt an seine Grenzen gekommen und hätte dieses mit hoher Wahrscheinlichkeit abbrechen müssen.

```
Concurrent(
    If(!IsEmpty(SammlungCBSaison);
        ForAll(SammlungCBSaison;
            Collect(SammlungSeasonFinal;
                Search('PDM_InventTableProductVariantsEntity_FN (mserp)'; Result; "mserp_seasoncode")
            )
        )
    );
    If(!IsEmpty(SammlungCBKollektion);
        ForAll(SammlungCBKollektion;
            Collect(SammlungCollectionFinal;
                Filter('PDM_InventTableProductVariantsEntity_FN (mserp)'; Collection = Result)
            )
        )
    );
    If(
        !IsEmpty(SammlungCBKategoriehierarchie);
        ForAll(SammlungCBKategoriehierarchie;
            Collect(SammlungCategoryFinal;
                Filter('PDM_InventTableProductVariantsEntity_FN (mserp)'; Name = Result)
            )
        )
    )
);;
```

Bild 5: Power FX Code zur Befüllung von Sammlungen nach Filteroptionen

3. Vergleich von Low-Code-Plattformen

3.1. Beschreibung Testszenario

Um herauszufinden, inwiefern sich Low-Code-Plattformen wirklich für die Nutzung durch einen Citizen Developer eignen, soll ein Vergleich zwischen mehreren Plattformen durchgeführt werden. Dabei soll untersucht werden, welche Kenntnisse für die Umsetzung einer Anwendung mit der jeweiligen Plattform notwendig sind und welche Plattformen gegebenenfalls besser geeignet sind für einen Citizen Developer.
An dem in Punkt 2.3. beschriebenen Beispiel wurde gezeigt, dass die Grenzen des Citizen Developers bei komplexeren Anforderungen schnell erreicht sein können. Das Testszenario sollte für einen umfassenden Vergleich allerdings möglichst mit mehreren Plattformen umsetzbar sein, wodurch ein ähnlich komplexes Szenario nicht sinnführend wäre. Darum wurde ein deutlich einfacheres, aber trotzdem realitätsnahes Szenario gewählt, das mit jeder der ausgewählten Plattformen umgesetzt werden soll. Die dabei entstehende Anwendung soll von Dateneingabe, über Datenverarbeitung bis hin zu Datenausgabe alles abdecken.
Die minimale Funktionalität, die eine Anwendung abdecken soll, ist in Bild 6 dargestellt. Zuerst sollen die Daten einer Excel-Datei in einer Liste oder einer ähnlichen Datensammlung gespeichert werden. Dabei ist es auch zulässig, wenn die Excel-Datei schon während der Entwicklung in der Anwendung hinterlegt wird und vom Nutzer nicht änderbar ist. Anschließend soll die Liste mit einer For-each-Schleife durchlaufen werden und für den jeweils aktuellen Datensatz soll eine E-Mail erstellt und versendet werden. Die Excel-Tabelle soll die Spalten „E-Mail-Adresse“ und „Name“ besitzen, deren Daten jeweils als Variable in eine E-Mail-Vorlage eingetragen werden sollen. Somit soll für jeden Datensatz der Excel-Tabelle eine E-Mail versendet werden, in der die E-Mail-Adresse und der Empfängername automatisch eingetragen werden. Die Plattformen sollen also in der Lage sein Excel-Dateien zu verarbeiten und E-Mails zu versenden.

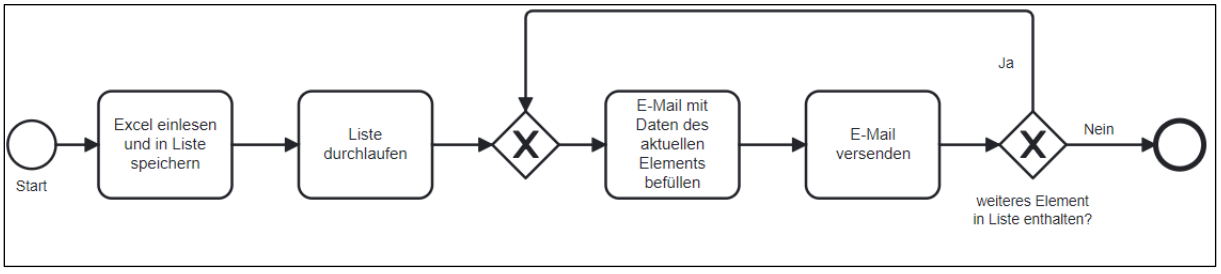

Bild 6: minimale erwartete Funktionalität der Anwendung

Die Umsetzung von weiteren Funktionalitäten ist möglich, aber nicht notwendig, damit das Testszenario als vollständig umgesetzt zählt. Beispielsweise könnten über eine Benutzeroberfläche, der Betreff und der Inhalt der E-Mail eingegeben werden. Somit

kann vom Endnutzer die E-Mail weiter angepasst werden. Außerdem wäre es möglich, dass der Endnutzer eine Absender-E-Mail-Adresse angeben kann, um die E-Mail von seinem persönlichen Account versenden zu können. Für den Excel-Import wäre es denkbar, dass der Endnutzer eine eigene Datei hochladen kann, die die Daten enthält, die er benötigt.
Für jede Plattform wird eine maximale Umsetzungszeit von 16 Stunden festgelegt. Damit stehen für die Umsetzung 2 volle Arbeitstage zur Verfügung. Dies wird dadurch begründet, dass bei einer längeren Umsetzungsdauer als 16 Stunden die angestrebte Zeitersparnis wegfällt. Wenn nach dieser Zeit das Testszenario nicht umgesetzt werden konnte, wird die Arbeit mit der Plattform beendet und folglich als ungeeignet für den Citizen Developer in diesem Szenario bewertet.

3.2. Auswahl der Bewertungskriterien

Für die Bewertung der Plattformen haben sich zwei verschiedene Kriterien-Kategorien ergeben. Die erste Kategorie enthält die Kriterien, die mit einer Recherche über die jeweilige Plattform beantwortet und vergleichen werden können. Diese Kriterien sind:

- Vorteile und Nachteile: Welche spezifischen Vor- und Nachteile bieten die Plattformen gegenüber anderen?
- Kosten: Wie viel kostet die Nutzung der Plattform pro App oder pro Nutzer?
- Datenschutz und Sicherheit: Wie gehen die Plattformen mit dem Datenschutz um und gibt es spezielle Vorkehrungen in diesem Bereich?

Die zweite Kategorie enthält Kriterien, für deren Einschätzung eine Nutzung der Plattform notwendig ist. Um die Plattformen anhand dieser Kriterien bewerten zu können ist somit zuerst eine Umsetzung des Testszenarios notwendig. Diese Kriterien sind:

- Lernkurve & Nutzerfreundlichkeit: Wie intuitiv kann die Plattform bedient werden und wie schnell ist der Lernfortschritt während der Nutzung?
- Interoperabilität mit anderen Systemen und Datenquellen: Gibt es standardmäßig integrierte Schnittstellen zu anderen Systemen und welche Datenquellen können angebunden werden?
- Umsetzungsdauer: Wie lange dauert es das Testszenario mit der Plattform umzusetzen?
- Dokumentation & Support: Werden Entwicklerdokumentationen und Supportforen bereitgestellt?
- Herausforderungen: An wie vielen Stellen sind Fehler aufgetreten, für die nicht sofort eine Lösung gefunden werden konnte?
- Low-Code – Umsetzung: Wie viel Code musste während der Umsetzung des Testszenarios selbst geschrieben werden und wie komplex war dieser Code?

3.3. Auswahl der Plattformen

Für den Vergleich sollten verschiedene Typen von Plattformen ausgewählt werden, um unterschiedliche Herangehensweisen an den Low-Code-Ansatz zu betrachten und zu vergleichen. Das Hauptkriterium für die Auswahl war, dass möglichst nur größere, etablierte Plattformen verglichen werden und keine kleinen Nischenlösungen. Das hatte den Grund, dass bei großen, erfolgreichen Plattformen davon ausgegangen werden kann, dass sie bereits viele Nutzer mit ihrer Funktionalität überzeugen konnten und sie somit tendenziell auch für den Citizen Developer gut geeignet sind. Bei kleinen Plattformen mit Fokus auf einen bestimmten Anwendungsbereich, kann im Fall, dass das Testszenario nicht umsetzbar ist, nicht sicher bewertet werden, ob die Plattform generell ungeeignet ist oder ob nur das Testszenario nicht zu dem Plattformfokus passt. Zudem wurde auch darauf geachtet, dass sowohl Plattformanbieter ausgewählt werden, deren Hauptprodukt die Low-Code-Plattform ist, als auch Anbieter, bei denen die Plattform nur einen kleinen Teil der Produktpalette ausmacht. In dem Fall ist es sehr interessant, ob man eine Tendenz erkennen kann, ob eine Art von Anbietern generell besser geeignet ist, oder ob nicht davon abhängig ist.

Für den Vergleich wurde folgende Plattformauswahl getroffen:

- **Microsoft Power Apps:** Power Apps stand von Anfang an als erster Vergleichskandidat fest, da durch die Nutzung der Plattform die Grundidee dieser Arbeit entstanden ist. Weil mit Power Apps schon deutlich komplexere Anforderungen umgesetzt wurden, wie in Punkt 2.3. beschrieben wird, kann die Plattform sehr gut als Referenz dafür dienen, wie sich Plattformen, die in diesem Vergleich ähnlich abschneiden, bei komplexeren Anforderungen schlagen würden.
- **Microsoft Power Automate:** Power Automate wurde als zweiter Kandidat aus dem Hause Microsoft ausgewählt, wobei es sich hierbei nicht um eine Low-Code-Plattform handelt und somit keine Anwendungen, sondern nur automatisierte Workflows erstellt werden können. Für den Vergleich ist Power Automate deshalb interessant, um zu sehen, ob für die Umsetzung des Testszenarios zwingend eine Low-Code-Plattform notwendig ist.
- **Oracle APEX:** Mit Oracle Application Express wurde eine dritte Plattform, neben den beiden Lösungen von Microsoft, ausgewählt, bei denen die Haupttätigkeit des Anbieters nicht im Bereich Low-Code-Plattformen liegt. APEX differenziert sich vor allem damit, dass es zwar als Stand-Alone-Plattform genutzt werden kann, aber seine eigentlichen Stärken erst in Verbindung mit einer Oracle Datenbank ausspielt.
- **Mendix:** Mendix ist aktuell der Marktführer im Bereich Low-Code-Plattformen und ist die erste ausgewählte Plattform, bei der der Anbieter hauptsächlich im

Bereich Low-Code-Plattformen tätig ist. Bei Mendix ist interessant, dass eine No-Code- und eine Low-Code-Plattform angeboten wird. Dadurch kann auch getestet werden, ob für das Testszenario auch eine Plattform ausreichend ist, bei der kein Code geschrieben werden muss.[17]

- **OutSystems:** OutSystems ist ebenfalls einer der Marktführer für Low-Code-Plattformen und ist neben Mendix der zweite Vergleichskandidat mit der Haupttätigkeit in diesem Bereich. OutSystems soll sich vor allem durch seine Schnelligkeit bei der Entwicklung auszeichnen.[18]
- **Appian:** Appian ist ein etwas kleinerer Anbieter von Low-Code-Plattformen, wobei es nach Gartners immer noch unter den Top 10 Anbietern zu finden ist. Appian ist ebenfalls hauptsächlich im Bereich Low-Code-Plattformen tätig und hat einen großen Fokus auf Automatisierung.[19]

3.4. Analyse der Plattformen

3.4.1. Microsoft Power Apps

Microsoft Power Apps ist Teil der Microsoft Power Plattform. Power Plattform ist eine integrierte Anwendungsplattform, die allen Unternehmen ermöglichen soll, die Produktivität und Effizienz ihrer Prozesse zu steigern. Diese beinhaltet außer Power Apps die Dienste Power Automate, Power BI und den Common Data Service.[20]

Microsoft Power Apps ist eine Low-Code-Plattform, mit der die Erstellung von Business-Anwendungen ohne umfassende Programmierkenntnisse möglich ist. Power Apps kann sowohl für den Aufbau einfacher mobiler Apps als auch für die Erstellung komplexer Business-Applikationen eingesetzt werden. Es werden zwei verschiedene Entwicklungsumgebung für unterschiedliche Anforderungen angeboten.[21]

Mit dem Power Apps Studio lassen sich sogenannte Canvas-Apps erstellen. Für die Erstellung von Canvas-Apps können verschiedene Steuerelemente auf einer anfänglich leeren Leinwand per Drag&Drop eingefügt und angeordnet werden. Der Aufbau erinnert dabei an PowerPoint, wodurch sich die Entwicklung wie die Erstellung von PowerPoint Slides anfühlt. Canvas-Apps sind sehr flexibel in der Gestaltung, da der Entwickler die volle Kontrolle über jeden Aspekt hat, wie z.B. die Größe, die Farbe und das Format der Komponenten.[22]

Mit dem App Designer lassen sich sogenannte Model-driven Apps erstellen. Model-driven Apps verfolgen den „Data first“-Ansatz und basieren immer auf einem

[17] Vgl. (FELDHERR, 2021a)
[18] Vgl. (FELDHERR, 2023a)
[19] Vgl. (VINCENT & a., 2022)
[20] Vgl. (WELSCH, 2023)
[21] Vgl. (FELDHERR, 2023b)
[22] Vgl. (FELDHERR, 2023b); Vgl. (LINKE, 2021)

Datenmodell. Das Datenmodell kann bereits in einer Datenquelle bestehen, aber auch neu erstellt werden. Auch hierbei wird das User Interface per Drag&Drop erstellt, wobei das Datenmodell das Layout weitestgehend bestimmt. Die Entwickler haben weniger Kontrolle über das Layout und die Funktionalität als mit Canvas-Apps.[23]

3.4.2. Microsoft Power Automate

Power Automate ist eine Cloud-Software, mit der Workflows und Aufgaben über mehrere Anwendungen und Dienste hinweg erstellt und automatisiert werden können. Um einen Workflow, auch „Flow“ genannt, zu erstellen muss der Nutzer eine Aktion angeben, die stattfinden soll, sobald ein bestimmtes Ereignis eintritt. Mit Power Automate lässt sich von einfachen Push-Benachrichtigungen bis hin zu komplexen Geschäftsprozessen mit definierten Schritten alles automatisieren. Am häufigsten wird es für das Auslösen einer Benachrichtigung über ein bestimmtes Ereignis verwendet.[24]

3.4.3. Oracle APEX

Orcale Application Express, kurz APEX, ist eine Low-Code-Plattform, mit der Entwickler webbasierte Anwendungen in einer Oracle Datenbank schnell erstellen können[25]. APEX kann bei Bedarf als Stand-Alone mit einer Drittanbieter-Datenbank verwendet werden, allerdings überwiegen die Vorteile bei einer Nutzung mit einer Oracle Datenbank. Oracle differenziert sich zum Wettbewerb durch die Verwendung seiner Oracle Autonomous Database, die Kunden Hochverfügbarkeit und Skalierbarkeit bieten soll, und der Verwendung von SQL. Da SQL-Kenntnisse vorteilhaft für viele Prozesse in APEX sind, bietet Oracle innerhalb der Plattform einen SQL-Workshop an.[26]

3.4.4. Mendix

Mendix ist nach den Marktforschungsunternehmen Gartner und Forrester einer der Marktführer im Bereich Low-Code-Entwicklung. Es nutzt die Anbindung an offene Standards wie BPMN (Business Process Model and Notation), UML (Unified Modeling Language) oder OData (Open Data) und ist keine abgekapselte Insellösung. Mendix unterstützt das Konzept des Multi-User-Developments, wodurch mehrere Entwickler gleichzeitig transparent in einer App arbeiten und sich abstimmen können. Mit Mendix können Apps entwickelt, bereitgestellt, getestet und verwaltet werden, die sowohl auf iOS- als auch auf Android-Geräten ausgeführt werden können. Laut Mendix selbst sind sie die schnellste und einfachste Low-Code-Plattform am Markt, mit der eine

[23] Vgl. (FELDHERR, 2023b); Vgl. (LINKE, 2021)
[24] Vgl. (ComputerWeekly.de, 2020)
[25] Vgl. (AMBATI, 2018)
[26] Vgl. (FELDHERR, 2023a); Vgl. (VASKE, 2022)

Bereitstellung von Apps innerhalb weniger Tage möglich ist. Für die Entwicklung werden zwei verschiedene Plattformen für unterschiedliche Professionalitätsgrade angeboten.[27]

Mendix Studio ist eine No-Code-Plattform und richtet sich an Citizen Developer. Für die Entwicklung ist somit kein manuelles Schreiben von Code notwendig. Die Entwicklungsumgebung ist dafür sehr intuitiv und visuell ausgerichtet. Die damit entwickelten Apps können zum einen tatsächlich genutzt werden oder zum anderen als Grundlage für professionelle Entwickler dienen, die daran weiterarbeiten können. Die erstellten Seiten besitzen ein responsives Design, wodurch sie sich automatisch an den jeweiligen Bildschirm anpassen. Außerdem wird mit Mendix Assist ein mit künstlicher Intelligenz ausgestatteter Agent mitgeliefert, der dem Entwickler bei der Einrichtung von Microflows helfen soll. Microflows können Anweisungen ausführen, wie z.B. das Erstellen oder Verändern von Objekten oder das Anzeigen von Seiten. Sie ersetzen somit den normalerweise notwendigen Code.[28]

Mendix Studio Pro ist eine Low-Code-Plattform für professionelle Entwickler. Es ermöglicht die Entwicklung von komplexeren Anwendungen mit wenig Code, wobei der Codeanteil auch deutlich ausgeweitet werden kann, um die Anwendungen an individuelle Anforderungen anzupassen und zu erweitern. In Mendix Studio Pro ist ebenfalls Mendix Assist integriert, der hier allerdings eher als Reviewer fungiert, um Logikabläufe zu überprüfen und Fehler aufzudecken. Zwischen einzelnen Entwicklungsschritten kann zwischen Mendix Studio und Mendix Studio Pro hin und her gewechselt werden, wodurch eine direkte Zusammenarbeit zwischen Citizen Developern und professionellen Softwareentwicklern möglich ist.[29]

3.4.5. Outsystems

Outsystems ist einer der Marktführer für Low-Code-Plattformen und sticht vor allem durch seine Schnelligkeit in der Entwicklung heraus. Die Plattform zielt auf eine schnelle Anpassung und Weiterentwicklung von bestehenden Applikationen ab, sodass bei einer Änderung der geschäftlichen Anforderungen sehr schnell reagiert werden kann. Es bietet KI- und Automatisierungstools, die bei der Entwicklung von Anwendungen zur Seite stehen. Die wichtigste Funktion der Plattform ist allerdings die visuelle Full-Stack-Entwicklung für Web-Anwendungen. Dabei wird nicht wie herkömmlich zwischen Frontend und Backend getrennt, sondern die vollständige Entwicklung findet innerhalb der Plattform statt. Außerdem unterstützt die Plattform eine schnelle Bereitstellung von neuen oder angepassten Applikationen und bietet eine Folgeabschätzung, um mögliche Probleme bei der Bereitstellung zu erkennen.[30]

[27] Vgl. (FELDHERR, 2021a); Vgl. (STAAR, 2022)
[28] Vgl. (FELDHERR, 2021a)
[29] Vgl. (FELDHERR, 2021a)
[30] Vgl. (FELDHERR, 2023a); Vgl. (FELDHERR, 2021b)

3.4.6. Appian

Appian bietet eine Low-Code-Plattform mit Fokus auf komplexes Case Management und Automatisierung. Das bedeutet, dass mit der Plattform vor allem unterstützende Anwendungen für Managementprozesse von multidisziplinären Arbeitsaufgaben innerhalb einer Organisation erstellt werden können. Dafür werden verschiedene Funktionen angeboten, wie zum Beispiel einem Process Modeler, mit dem BPMN-konforme Visualisierungen von Prozessen erstellt werden können, oder auch einem Automation Planner für Processmining und Automatic-Life-Cycle-Management. Außerdem besteht mit dem App Market ein Marktplatz, auf dem bestehende Anwendungen vertrieben und erworben werden können.[31]

3.5. Vergleichende Bewertung der Plattformen

3.5.1. Vorteile und Nachteile

Um einen Überblick über die Vorteile und Nachteile der einzelnen Plattformen zu bekommen, werden diese in Tabelle 1 abgebildet. Dabei wird in der ersten Spalte die Plattform benannt und anschließend werden die Vorteile dieser Plattform in der 2. Spalte und die Nachteile in der 3. Spalte aufgelistet. Die Quellenangaben für die jeweiligen Vor- und Nachteile sind über die Fußnote am Plattformnamen in der 1. Spalte zu finden.

[31] Vgl. (FELDHERR, 2023a)

Plattform	Vorteile	Nachteile
Microsoft Power Apps[32]	• Mögliche Integration von Machine Learning • Codesharing über Github • Synergieeffekte mit anderen Microsoft Diensten • KI-gestützte Entwicklung und Open APIs	• Viele Informationen nur auf Englisch verfügbar • Komplexe Struktur mit unterschiedlichen Tools • Anwendungen sind nicht offline-fähig
Microsoft Power Automate[33]	• Auslöser basierende Workflows • Viele Konnektoren	• Nicht für komplexe Anwendungen geeignet • Unterstützt nur sequenzielle Workflows
Oracle APEX[34]	• Ohne komplizierte Deployment-Schritte • Inklusive intelligenter Assistenten • Einfache Integration in verschiedene Datenquellen • Inklusive Advisor, der die Qualität der Anwendung beurteilt • Einfache Erstellung von APIs	• Viele Informationen nur auf Englisch • Hosting nur auf einer Oracle-Datenbank • Keine eingebaute Versionskontrolle • Nur für Entwickler und Administratoren, die mit PL/SQL vertraut sind
Mendix[35]	• Anbindung an offene Standards wie BPMN, UML oder oData • Interoperabilität mit anderen Systemen • Multi-User-Development • IoT App-Entwicklung • KI gestützte Entwicklung und Fehler-Assistent • Deckt auch Anforderungen an sehr komplexe Business Apps ab	• Konzentration ausschließlich auf das User-Interface • Nicht für Entwicklung der gesamten Bandbreite von Anwendungen geeignet
Outsystems[36]	• KI-gestützte App-Entwicklung • Schnelle Entwicklung • Continous Integration und Delivery • Hohe Datensicherheit	• Nicht für komplexe Apps geeignet • Viele Informationen nur in Englisch verfügbar
Appian[37]	• Vereint Process Mining, Workflow und Automation • Abdeckung unterschiedlicher Entwickler Personas • Hohe Sicherheitsstandards und Security Alerts	• Einige Updates nur für die Appian-Cloud verfügbar • Komplexe Strukturen innerhalb der Plattform

Tabelle 1: Vorteile und Nachteile der ausgewählten Low-Code-Plattformen

[32] Vgl. (STAAR, 2022); Vgl. (FELDHERR, 2023a); Vgl. (FELDHERR, 2023b)
[33] Vgl. (sharepoint360.de, 2019)
[34] Vgl. (STAAR, 2022); Vgl. (FELDHERR, 2023a)
[35] Vgl. (STAAR, 2022); Vgl. (FELDHERR, 2023a)
[36] Vgl. (STAAR, 2022); Vgl. (FELDHERR, 2023a)
[37] Vgl. (STAAR, 2022); Vgl. (FELDHERR, 2023a)

Wie in Tabelle 1 zu sehen ist, bieten alle Plattformen verschiedene Vorteile und Nachteile, wobei nicht alle von diesen für die Umsetzung des Testszenarios relevant sind. Besonders interessant könnten Vorteile wie Automatisierung, KI-gestützter Entwicklung oder Integration von anderen Systemen in die Anwendung sein.
Im Bereich Automatisierung haben Power Automate und Appian ihre Vorteile. Diese könnten vor allem bei der Verarbeitung der Daten aus der Excel-Datei und dem Einfügen dieser in eine E-Mail-Vorlage per Variablen nützlich sein, sodass diese Funktion ohne großen Entwicklungsaufwand mit den beiden Plattformen umgesetzt werden könnte. Im Gegenzug dazu bringen die anderen vier Plattformen eine KI-gestützte Entwicklung oder andere Assistenten mit, die bei der Umsetzung helfend zur Seite stehen sollen. Diese könnten die Nachteile der fehlenden Automatisierungsfunktionen ausgleichen und die Entwicklung ebenso vereinfachen.
Mit einer großen Auswahl an integrierbaren Systemen punkten die Plattformen von Microsoft, sowie Mendix und Oracle APEX. Diese sind zwingend notwendig, um Zugriff auf Excel-Dateien und E-Mail-Dienste zu bekommen, wobei diese wahrscheinlich eher zum Standard der Plattformen gehören. Somit kann daraus in Bezug auf das Testszenario kein besonderer Vorteil gezogen werden.
Weitere Vorteile wie Code-Verwaltung und Zusammenarbeitsmöglichkeit oder auch einfache Deployment-Schritte sind für das Testszenario weniger von Bedeutung, da es nur um die Umsetzung geht. Außerdem wären wahrscheinlich im realen Umfeld mögliche Deployment-Aufgaben nicht vom Citizen Developer zu bearbeiten, sondern würden von der betreuenden IT-Abteilung übernommen werden.
Somit kann aus den aufgelisteten Vorteilen der einzelnen Plattformen kein besonderer Nutzen einer einzelnen Plattform für das Testszenario gezogen werden. Die meisten Vorteile wären eher bei der Auswahl einer Plattform für einen spezifischer Anforderungen oder die Arbeit in Teams von Bedeutung.
Bei den Nachteilen stechen vor allem Oracle APEX und Mendix heraus, bei denen aufgrund bestimmter Punkte unter Umständen Schwierigkeiten bei der Entwicklung auftreten könnten. Bei Oracle APEX ist der entscheidende Nachteil, dass PL/SQL-Kenntnisse notwendig sind, wodurch die Entwicklung deutlich erschwert werden könnte. Auch bei einer erfolgreichen Umsetzung des Testszenarios wäre es denkbar, dass diese Plattform aufgrund der notwendigen Kenntnisse als ungeeignet für den Citizen Developer eingestuft wird. Bei Mendix könnte dagegen eine starke Konzentration auf das User-Interface im Testszenario zu Problemen führen. Für das Testszenario ist nicht zwingend ein User-Interface notwendig, sondern eher die Verarbeitung der Daten, die möglichst einfach umzusetzen sein sollte.

3.5.2. Kosten

Die Kosten der einzelnen Plattformen sind in **Fehler! Verweisquelle konnte nicht g efunden werden.** aufgelistet. Die Kosten sind dabei in zwei verschiedene Kategorien eingeteilt. Zum einen die Kosten für die Erstellung einer einzigen App und zum anderen die Kosten für ein Modell bei dem es keine Limitierung in der Anzahl der zu erstellenden Apps gibt. Der jeweilige Plattformname ist der ersten Spalte der Tabelle zu entnehmen, an dem außerdem die Quellenangabe für die einzelnen Informationen zu finden ist. Die Kosten für eine App werden in der zweiten Spalte aufgelistet und die Kosten für das unlimitierte Modell in der dritten Spalte.

Plattform	Kosten für eine App	Kosten für unbegrenzte Anzahl an Apps
Microsoft Power Apps[38]		• 18,70€/Nutzer/Monat
Microsoft Power Automate[39]		• 14€/Nutzer/Monat
Oracle APEX[40]		• 333,33 € inklusive „Autonomous Database“ • Kostenlos bei bereits vorhandener Oracle-Datenbank
Mendix[41]	• Basic: ab 50€/Monat (inkl. 5 Nutzer) • Standard: ab 800€/Monat • + 10€/Nutzer/Monat	• Standard: ab 2000€/Monat • + 25€/Nutzer/Monat
OutSystems[42]		• Ab 1250€/Monat
Appian[43]	• Standard: 75$/Nutzer/Monat • Infrequent: 9$/Nutzer/Monat • Input-Only: 2$/Nutzer/Monat	• Individuelles Angebot

Tabelle 2: Kostenübersicht

[38] Vgl. (powerapps.microsoft.com, kein Datum)
[39] Vgl. (powerautomate.mircosoft.com, kein Datum)
[40] Vgl. (oracle.com, kein Datum), Vgl. (apex.oracle.com, kein Datum)
[41] Vgl. (mendix.com, kein Datum)
[42] Vgl. (outsystems.com, kein Datum)
[43] Vgl. (appian.com, kein Datum)

Wie in **Fehler! Verweisquelle konnte nicht gefunden werden.** zu erkennen ist, b ieten die Plattformen Mendix und Appian Pläne für eine einzelne App an. Beide Plattformen bieten ebenfalls verschiedene Abstufungen, um dem Kunden passend auf seine Bedürfnisse das beste Angebot zu bieten. Mendix differenziert seine verschiedenen Pläne durch die angebotenen Funktionen. Somit gibt es eine Basic-Variante mit etwas weniger Funktionalität für 50€/Monat inklusive 5 Nutzern und eine Standard-Variante für 800€/Monat. Bei der Standard-Variante wird keine Anzahl der Nutzer genannt, die bereits inklusive sind. Für beide Varianten kostet ein weiterer Nutzer 10€/Monat. Appian dagegen bietet verschiedene Varianten für unterschiedliche Nutzer an. Ein Standard-Nutzer kostet bei Appian 75$/Nutzer/Monat. Für Nutzer, die zwar an der App mitarbeiten, aber seltener aktiv sind, gibt es den Infrequent-Nutzer. Dieser kostet 9$/Nutzer/Monat. Für Nutzer, die nicht an der App entwickeln, sondern nur die fertige Anwendung nutzen gibt es den Input-Only-Nutzer, für den 2$/Nutzer/Monat bezahlt werden muss. Bei Appian gibt es allerdings eine Mindestmenge von 100 Nutzern, die frei auf die drei Varianten verteilt werden können. Somit würde man für den 1-App-Plan bei Appian mindesten 273$/Monat bezahlen, da mindestens ein Standard-Nutzer vorhanden sein muss, um eine App entwickeln zu können und der Rest mit 99 Input-Only-Nutzern aufgefüllt werden kann. Im Vergleich ist Mendix in den meisten Fällen bei einem 1-App-Plan flexibler und günstiger. Eine Ausnahme bilden Fälle, in denen bei Appian wenige Standard-Nutzer und sonst nur Input-Only-Nutzer notwendig sind, während bei Mendix die Standard-Variante genutzt wird.[44]

Ein Plan für eine unlimitierte Anzahl an Apps wird dagegen von allen Plattformen angeboten. Eine Besonderheit stellt dabei Oracle APEX dar. Dieses ist komplett kostenlos, wenn man bereits Nutzer einer Oracle-Datenbank ist[45]. Falls keine Oracle-Datenbank genutzt wird, kostet APEX 333,33€ im Monat inklusive einer „Autonomous Database“[46]. Bei Appian wird bei einem Multi-App-Plan kein Preis genannt, sondern es muss ein individuelles Angebot eingeholt werden[47]. Bei OutSystems startet der Multi-App-Plan ab 1250€/Monat[48]. Bei Mendix ist der Plan ab 2000€/Monat erhältlich und jeder zusätzliche Nutzer kostet 25€/Monat[49]. Bei beiden Plattformen wird allerdings nicht erwähnt wie viele Nutzer in den Paketen jeweils inklusive sind, sodass diese Preise schlecht vergleichbar sind. Microsoft verlangt für Power Apps 18,70€/Nutzer/Monat und für Power Automate 14€/Nutzer/Monat[50]. Da die Plattformen unterschiedliche Vorgehen in der Preisgestaltung aufweisen, können diese nur bedingt

[44] Vgl. (mendix.com, kein Datum); Vgl. (appian.com, kein Datum)
[45] Vgl. (apex.oracle.com, kein Datum)
[46] Vgl. (oracle.com, kein Datum)
[47] Vgl. (appian.com, kein Datum)
[48] Vgl. (outsystems.com, kein Datum)
[49] Vgl. (mendix.com, kein Datum)
[50] Vgl. (powerapps.microsoft.com, kein Datum); Vgl. (powerautomate.mircosoft.com, kein Datum)

miteinander verglichen werden. Auf den ersten Blick könnten allerdings die Microsoft-Plattformen als vermeintlicher Preissieger gewertet werden, da im Vergleich zu Mendix der Preis/Nutzer/Monat geringer ausfällt und keine Mindestanzahl an Nutzern angegeben wird.[51]

3.5.3. Datenschutz & Sicherheit

Im Bereich Datenschutz gibt es bis auf Mendix keine Besonderheiten der Plattformen. Bei jeder Plattform ist auf der Website eine Datenschutzerklärung zu finden. Weitere Besonderheiten im Bereich Datenschutz werden allerdings nur von Mendix aufgeführt. Mendix besitzt den höchsten globalen Standard für Cloud-Sicherheit und Datenschutz und das PCI-Payment Application Data Security Standard Level 1, was die höchste Sicherheitsstufe zur Verarbeitung von Kreditkartentransaktionen ist. Ebenso ist Mendix auch für sensible Anwendungsfälle im Finanz- und Gesundheitswesen geeignet. Für alle anderen Plattformen kann aufgrund der fehlenden Informationen keine genauere Aussage getroffen werden, wodurch es nicht möglich ist diesen Punkt zu bewerten.[52]

[52] Vgl. (mendix.com, 2020)

4. Umsetzung eines Testszenarios

4.1. Beschreibung der Umsetzung

4.1.1. Microsoft Power Apps

Für Power Apps musste keine Testversion beantragt werden, da diese bei BE-terna für jeden Mitarbeiter zur Verfügung gestellt wird. Die Umsetzung des Testszenarios mit Power Apps verlief ohne größere Herausforderungen und dauerte ca. 1 Stunde. Bei der Dauer sollte allerdings beachtet werden, dass bereits Kenntnisse mit dem Umgang mit Power Apps vorhanden waren und somit keine Einarbeitungszeit notwendig war.
Die Anwendung ist so aufgebaut, dass eine Excel-Datei fest hinterlegt ist, die von der App eingelesen wird. Ebenso ist die E-Mail-Adresse des Absenders fest hinterlegt. Der Nutzer kann über Inputfelder den Betreff und den Inhalt der E-Mail festlegen und über einen Button die E-Mails versenden. Die dazugehörige Nutzeroberfläche mit den Inputfeldern und dem Button zum Versenden der E-Mail ist in Bild 7 zu erkennen.

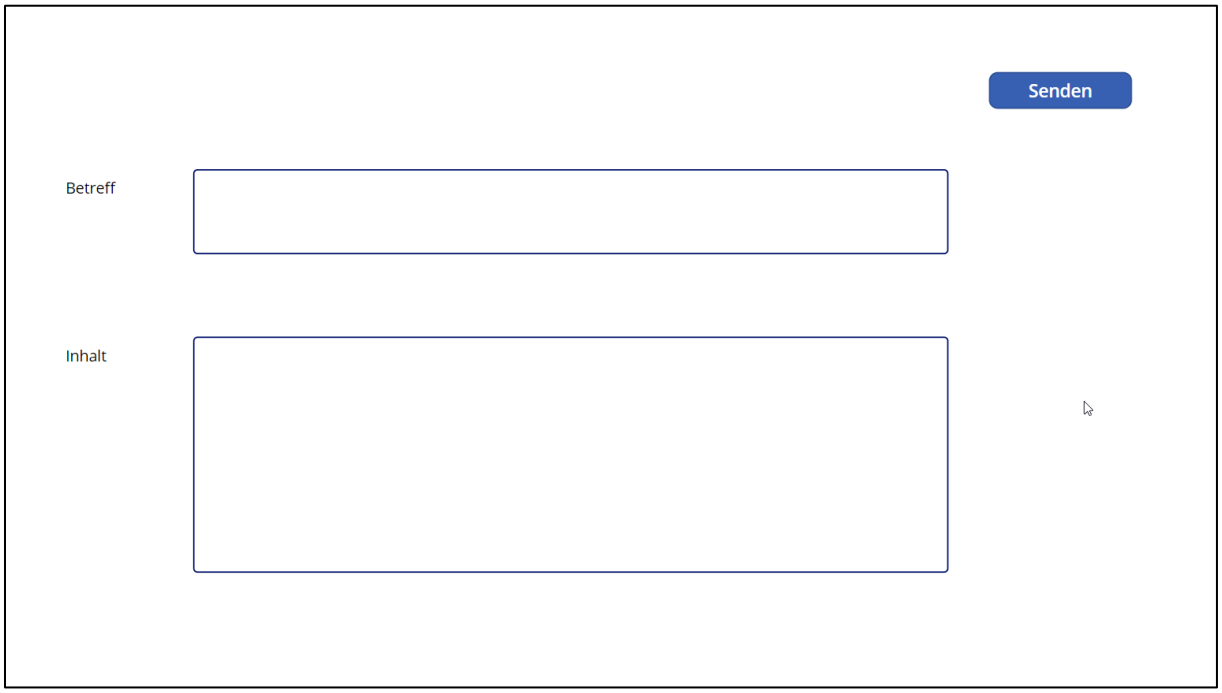

Bild 7: Benutzeroberfläche einer Power App Anwendung zum Versand von E-Mails

Bei der Umsetzung wurden zuerst die Konnektoren zu Excel und Outlook eingerichtet, was relativ einfach ging, da für beide als Microsoft-System ein standardmäßiger Konnektor vorhanden ist. Für die Einrichtung musste sich für beide Systeme mit dem Microsoft-Account eingeloggt werden und bei der Excel-Verbindung die Datei und der Speicherort dieser ausgewählt werden. Durch dieses Vorgehen sind sowohl die Excel-Datei als auch die E-Mail-Adresse fest in der App hinterlegt und können vom Nutzer nicht geändert werden. Für die Verarbeitung der Daten ist in Power Apps manueller Code notwendig. Dieser ist in Bild 8 zu sehen und wird durch die Betätigung des Buttons ausgeführt. Der Code wurde in der Programmiersprache „Power FX“

geschrieben. Durch die Anweisung in der ersten Zeile wird die hinterlegte Excel-Datei aktualisiert, um Änderungen außerhalb der Power App beachten zu können. Die beiden darauffolgenden Anweisungen speichern die Daten aus der Excel-Datei in einer Sammlung ab. Dafür wird die Sammlung zuerst geleert, um doppelte Datensätze zu verhindern und anschließend wird die Excel-Tabelle zeilenweise durchlaufen und der aktuelle Datensatz wird in der Sammlung gespeichert. Die letzte Anweisung steuert den E-Mail-Versand, indem es die erstellte Sammlung durchläuft und für jeden enthaltenen Datensatz eine individuelle E-Mail versendet.
Theoretisch ist es möglich die E-Mail-Adresse zum Versenden vom Nutzer individuell eingeben zu lassen, allerdings sind dafür weitere Einstellungen in Outlook notwendig. Dabei müsste festgelegt werden, dass über die E-Mail-Adresse, die bei der Entwicklung eingerichtet wurde, andere E-Mail-Adressen E-Mails versenden dürfen. Diese E-Mail-Adressen müssen dann in dem Outlook-Account hinterlegt werden, wodurch dieses Vorgehen nur für eine begrenzte und bekannte Anzahl an Nutzer sinnvoll erscheint. Mit der aktuellen Umsetzung wäre eine Nutzung durch mehrere Mitarbeiter mit einer neutralen E-Mail-Adresse eines Unternehmens eine sinnvolle Einsatzmöglichkeit.
Für die Umsetzung des Testszenarios mit Power Apps waren Kenntnisse von Variablen, Schleifen und Datentypen von Vorteil. Besonders da manueller Code geschrieben werden musste, wenn auch nur vorgefertigte Funktionen benutzt wurden, zu denen Hinweise angezeigt werden, welche Eingaben erwartet werden.

```
Refresh(Tabelle1);;

Clear(ExcelSammlung);;

ForAll(Tabelle1; Collect(ExcelSammlung;ThisRecord));;

ForAll(ExcelSammlung; Office365Outlook.SendEmailV2(Emailadresse; EmailSubject.Text; "Hallo " & Name & ", <br/>" & EmailBody.Text));;
```

Bild 8: Power FX Code zur Umsetzung des Testszenarios

4.1.2. Microsoft Power Automate

Für Power Automate wurde ebenfalls keine kostenlose Testversion benötigt, da bei BE-terna jeder Mitarbeiter eine Lizenz besitzt. Die Arbeit mit Microsoft Power Automate dauerte 1 Stunde, wobei die eigentliche Umsetzung nur ca. 10 Minuten in Anspruch nahm. Wie bei der Umsetzung mit Power Apps ist sowohl die Excel-Datei als auch die E-Mail-Adresse des Absenders fest hinterlegt. Außerdem ist der Betreff und der Inhalt der zu versendenden E-Mail vordefiniert und kann vom Nutzer nicht verändert werden, da bei einem Flow keine Benutzeroberfläche vorhanden ist.
Die Umsetzung funktioniert ähnlich zu der Umsetzung mit Power Apps, nur mit der Besonderheit, dass keine Benutzeroberfläche erstellt und kein Code manuell geschrieben werden muss. Bei Power Automate werden sogenannte „Flows“ genutzt,

um die Funktionalität abzubilden. Ein „Flow“ ist eine Automatisierung[53]. Dafür werden verschiedene Aktionen in einer bestimmten Reihenfolge angeordnet, die anschließend sequenziell ausgeführt wird. Ein Flow startet dabei immer mit einem Auslöser. Für das Testszenario wurde ein manueller Auslöser gewählt, also eine Buttonbetätigung. Anschließend wurde der Excel-Konnektor mit der Funktion „In Tabelle vorhandene Zeilen auflisten“ ausgewählt und eingerichtet, um alle Zeilen der Excel-Tabelle einzulesen und zu speichern. Dafür ist ebenfalls eine Anmeldung im Microsoft-Account und die Auswahl der Excel-Datei notwendig. Danach wurde mit der Funktion „Auf alle anwenden“ eine Schleife gebildet, die alle Datensätze durchläuft, die aus der Excel-Datei ausgelesen wurde. Innerhalb der Funktion wurde ein Outlook-Konnektor eingefügt, mit dem in jedem Schleifendurchlauf eine E-Mail für den aktuellen Datensatz gesendet wird. Dafür ist die Anmeldung in Outlook und das Einfügen der Daten in die E-Mail-Vorlage über automatisch erstellte, auswählbare Variablen notwendig. Der vollständige Flow ist in Bild 9 zu erkennen. Für die Umsetzung waren Kenntnisse von Variablen und Schleifen notwendig, jedoch musste nichts davon selbst erstellt werden.

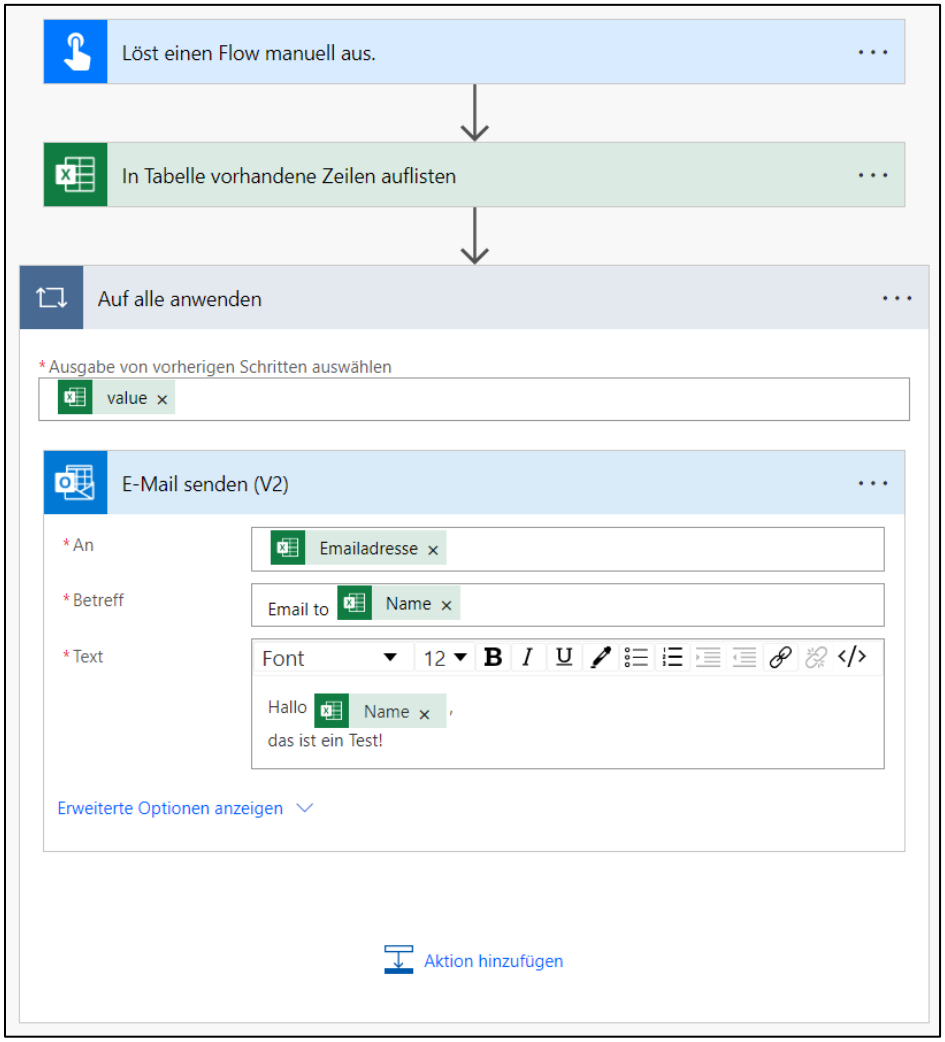

Bild 9: Power Automate Flow zur Umsetzung des Testszenarios

[53] Vgl. (learn.microsoft.com, 2023)

4.1.3. Oracle APEX

Eine kostenlose Testversion von Oracle APEX konnte schnell zur Verfügung gestellt werden, sodass innerhalb von 10 Minuten der Zugriff auf die Plattform möglich war. Allerdings verfügt die Testversion nicht über alle Funktionen, da APEX vor allem für die Nutzung in Verbindung einer Oracle Datenbank konzipiert ist, die in der APEX-Testversion nicht enthalten ist. Somit konnte das Testszenario nicht umgesetzt werden, da keine SMTP-Server-Einstellungen möglich waren, die für den E-Mail-Versand notwendig sind.

Für die Arbeit mit der Plattform wurden trotzdem etwa 7 Stunden aufgewendet. Davon entfallen 2 Stunden auf die Einarbeitung. Das war notwendig, da die Plattform recht kompliziert ist und man ohne Hilfe nicht wirklich weiß, wo man anfangen soll. Weitere 2 Stunden wurden für den Excel-Import aufgebracht, wobei allerdings die meiste Zeit davon aus Recherche nach einer sinnvollen Umsetzung bestand. Die Umsetzung des Excel-Imports ging am Ende recht schnell und war nach ca. 15 Minuten abgeschlossen. Dazu wurde eine neue Seite vom Typ *Data Loading* erstellt. Darin musste anschließend ein Template für die erwartete Excel-Datei erstellt werden, um die Datei am Ende richtig verarbeiten zu können. Damit war die Umsetzung des Excel-Imports schon abgeschlossen und es konnten Dateien hochgeladen werden, die allerdings der erstellten Vorlage entsprechen mussten.

Für den E-Mail-Versand wurden 3 Stunden aufgewendet bis herausgefunden wurde, dass die Umsetzung mit der kostenlosen Testversion nicht möglich ist. Die notwendigen Einstellungs-Menüs waren in der Testversion nicht vorhanden, sodass kein SMTP-Server für den E-Mail-Versand konfiguriert werden konnte. Des Weiteren sind Einstellungen an der Datenbank notwendig, die bei der kostenlosen Testversion nicht mit bereitgestellt wird. Die Verarbeitung der Daten aus der Excel-Datei, um diese per Variablen in eine E-Mail-Vorlage einfügen zu können, wurde nicht mehr bearbeitet.

4.1.4. Mendix

Bei Mendix war die kostenlose Testversion in 5 Minuten verfügbar, allerdings war für die Anmeldung zwingend eine Geschäfts-E-Mail-Adresse notwendig. Es konnte keine lauffähige Anwendung fertiggestellt werden, da die Umsetzung nach den vorgegebenen 16 Stunden beendet wurde.

Zu Beginn der Umsetzung war ca. 1 Stunde Einarbeitung notwendig, um die Plattform kennenzulernen. Als nächstes wurde der Excel-Import umgesetzt, wofür 3 Stunden aufgewendet wurden. Dafür musste zuerst eine Entität erstellt werden, in der die Excel-Datei später gespeichert werden kann. Danach wurde das Modul „Excel Importer“ heruntergeladen und installiert und weitere Module, zu denen das Modul Abhängigkeiten besitzt. Die Excel-Datei wird erst während der Laufzeit der Anwendung hochgeladen, über eine Data-Import-Seite. Auf dieser Seite muss zuerst ein Template

für die erwartete Excel-Datei erstellt werden und anschließend die Entität ausgewählt werden, in der die Daten gespeichert werden sollen. Dabei muss allerdings beachtet werden, dass das Template und die Entität übereinstimmen. Im Anschluss kann die Excel-Datei hochgeladen werden.
Der nächste Schritt war die Einrichtung des E-Mail-Versands. Dafür wurden ca. 4 Stunden benötigt. Dafür musste zuerst das Modul „Email Connector“ und ebenfalls weitere Module mit Abhängigkeiten installiert werden. Anschließend musste ein darin enthaltener „Microflow“ mit einem Button auf der Startseite verknüpft werden, um diesen starten zu können. Mit „Microflows“ kann die Logik der Anwendung erstellt werden[54]. Dazu können verschiedene Aktionen per Drag&Drop angeordnet und verknüpft werden, die dann sequenziell ausgeführt werden. Danach musste die Anwendung gestartet werden und der „Microflow“ mit dem Button ausgelöst werden. Dadurch wurde zu einer Seite navigiert, in der die Konfiguration für einen SMTP-Server vorgenommen werden musste. Da als SMTP-Server Gmail verwendet wurde, waren weitere Einstellungen direkt im Google-Konto notwendig.
Zum Schluss musste ein „Microflow“ erstellt werden, die Excel-Daten zu verarbeiten und für jeden Datensatz eine E-Mail zu versenden. Bei der Umsetzung des „Microflows“ kam es zu diversen Herausforderungen, wodurch 5,5 Stunden aufgewendet werden mussten. Der „Microflow“ konnte in dieser Zeit allerdings nicht fertiggestellt werden. Zuerst wurde eine Liste mit den Daten der Excel-Datei abgerufen, die mit einer Schleife durchlaufen werden konnte. Anschließend wurde eine Aktion zum Versenden von E-Mails hinzugefügt, die allerdings verschiedene Objekte als Eingabetypen benötigte, die ebenfalls erstellt werden mussten. Diese Objekte musste dann mit Variablen befüllt werden, wofür zum einen die Daten aus der Excel-Liste verwendet werden konnten. Zum anderen waren allerdings weitere Nutzereingaben auf der Oberfläche für den Betreff und den Inhalt der E-Mail notwendig. Dafür wurden zwei Eingabefelder auf der Oberfläche hinzugefügt, um beides eintragen zu können. Im Rahmen der Zeitvorgabe war es allerdings nicht mehr möglich eine Lösung zu finden, um diese Eingaben als Parameter an den „Microflow“ zu übergeben. Des Weiteren hätte der „Microflow“ immer mindestens zwei E-Mails für einen Datensatz versendet. Das hatte den Grund, dass zwei ineinanderlegende Schleifen notwendig waren, um alle Variablen der E-Mail-Versand-Aktion zuweisen zu können, da die Variablen immer nur innerhalb einer Schleife verfügbar sind. Somit konnten die beiden Schleifen nicht nacheinander angeordnet werden.
Weitere 2,5 Stunden wurden über den Umsetzungszeitraum verteilt aufgewendet, um verschiedene Error-Meldungen zu beheben. Diese traten vor allem durch den Download der verschiedenen Module auf, da z.B. Abhängigkeiten zwischen den Modulen bestanden, oder auch Label nicht in der eingestellten Standardsprache

[54] Vgl. (docs.mendix.com, 2023)

deutsch vorhanden waren, sondern nur in Englisch, was allerdings der Error-Meldung nicht immer direkt zu entnehmen war.
Ein Großteil der erwarteten Funktionalität konnte innerhalb der vorgegebenen 16 Stunden umgesetzt werden. Für die bestehenden Herausforderungen hätte allerdings noch weitere Zeit aufgewendet werden müssen. Da dafür keine spezifischen Informationen gefunden werden konnten, hätte die weitere Umsetzung mit ausprobieren fortgesetzt werden müssen, wodurch keine genaue Aussage zur benötigten Zeit getroffen werden kann.

4.1.5. OutSystems

Die Testversion wurde von OutSystems sehr schnell bereitgestellt und konnte innerhalb von 10 Minuten genutzt werden. Mit OutSystems hat die Umsetzung des Testszenarios 8,5 Stunden in Anspruch genommen. Davon wurden allerdings 5 Stunden für Fehlersuche und -behebung für einen Fehler außerhalb von OutSystems aufgewendet. Die Entwicklung innerhalb war nach etwa 3 Stunden abgeschlossen, aber aufgrund der aufgetretenen Fehler wurden einige Funktionen zurückgebaut. Diese musste nach der Behebung der Fehler erneut umgesetzt werden, was noch einmal eine halbe Stunde in Anspruch nahm.
Die entstandene Anwendung enthält eine Benutzeroberfläche, auf der der Betreff und der Inhalt der zu versendenden E-Mail in Inputfeldern eingetragen werden kann, ein Feld, um eine Datei hochzuladen, und ein Button, um den Versandvorgang auszuführen. Die Benutzeroberfläche ist in Bild 10 zu erkennen.

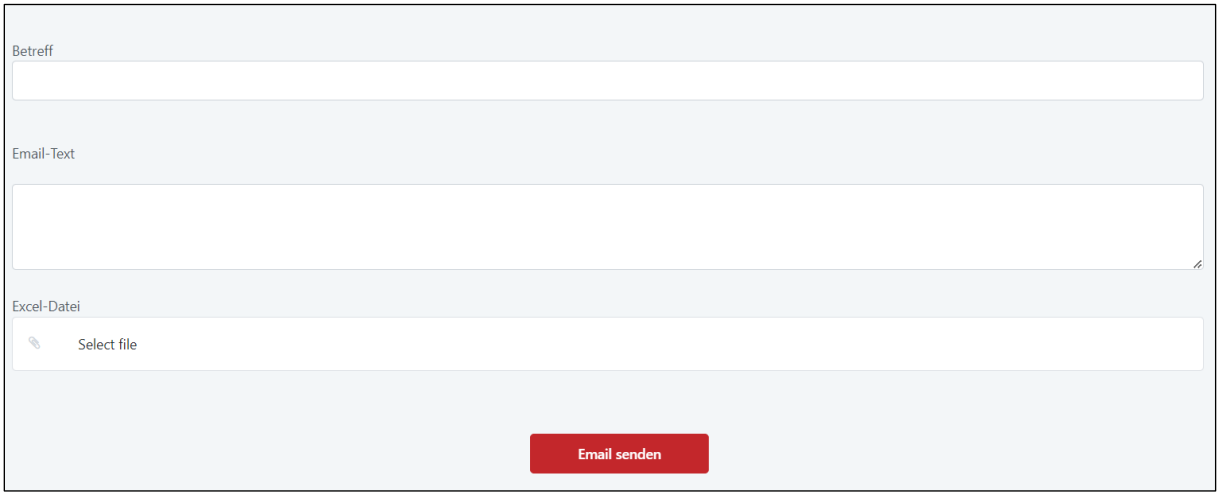

Bild 10: Benutzeroberfläche einer OutSystems-Anwendung zum E-Mail-Versand

Zuerst wurde die Benutzeroberfläche erstellt, indem die beschriebenen Felder per Drag&Drop angeordnet wurden. Für die Eingabefelder und das Upload-Feld mussten zudem Variablen erstellt werden, in denen die eingegebenen Werte gespeichert werden. Anschließend wurde eine „Client Action“ erstellt, die durch das Betätigen des Buttons ausgelöst wird. Eine „Client Action“ führt die Logik der Anwendung auf dem

Gerät des Nutzers aus und wird vom Benutzer von der Oberfläche aus gestartet[55]. Die „Client Action“ kann verschiedene Aktionen sequenziell ausführen, welche per Drag&Drop angeordnet und verknüpft werden können. In der „Client Action“ wurde eine „Server Action“ eingefügt, die ebenfalls die Logik der Anwendung ausführt, allerdings serverseitig[56]. In der „Server Action“, die in Bild 11 zu sehen ist, wurde die vollständige Logik für die Excel-Verarbeitung und den anschließenden E-Mail-Versand konfiguriert. Dafür wurde zuerst eine Aktion hinzugefügt, die eine ausgewählte Excel-Datei in eine Liste speichert. Anschließend wurde eine For-Each-Schleife hinzugefügt, die alle Datensätze der entstandenen Liste durchläuft. Innerhalb der Schleife wurde eine E-Mail-Versand-Aktion hinzugefügt, die für jeden Durchlauf für den aktuellen Datensatz eine E-Mail versendet. Um die auf der Benutzeroberfläche eingegebenen Werte in die E-Mail eintragen zu können mussten sowohl in der „Client Action“ als auch in der „Server Action“ Variablen definiert werden, über die die Werte von der Oberfläche bis zur E-Mail weitergegeben werden können. Außerdem musste eine E-Mail-Vorlage erstellt werden, um den Inhalt der E-Mail und Felder für die Werte der Variablen vorzugeben. In der E-Mail-Versand-Aktion musste zum Schluss nur noch die erstellte E-Mail-Vorlage ausgewählt werden, wodurch alle benötigten Felder der E-Mail eingeblendet wurden und mit den Variablen verknüpft werden konnten.

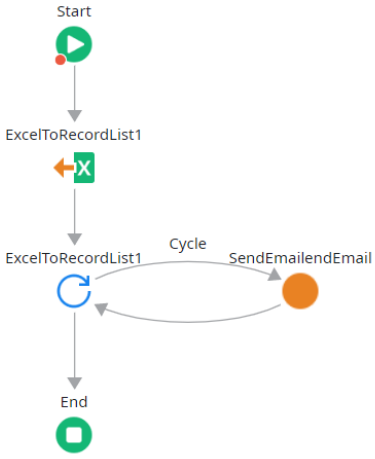

Bild 11: OutSystems Server Action zur Umsetzung des Testszenarios

Da für den Versand der E-Mail kein spezifischer Provider direkt in der Aktion bereitgestellt wird, muss eine SMTP-Server-Konfiguration für einen beliebigen E-Mail-Provider, eingestellt werden. Dafür wurde eine SMTP-Konfiguration für Gmail festgelegt. Anschließend mussten weitere Einstellungen im Google-Konto der eingestellten Gmail-Adresse vorgenommen werden, um den Zugriff durch OutSystems zu erlauben. Diese Einstellungen im Google-Konto haben einen großen Teil der Umsetzungszeit in Anspruch genommen, da nicht offiziell von OutSystems angegeben wurde, dass weitere Einstellungen notwendig sind.

Bei der OutSystems Anwendung ist die E-Mail-Adresse des Absenders ebenfalls fest hinterlegt. Wie bei der Power Apps Umsetzung wäre es möglich weitere Absender-Adressen zuzulassen, die allerdings erst in den Einstellungen des genutzten E-Mail-Kontos festgelegt werden müssen. Dafür ist es bei OutSystems allerdings möglich, dass eine Excel-Datei vom Nutzer hochgeladen werden kann. Diese Datei muss

[55] Vgl. (success.outsystems.com, 2023a)
[56] Vgl. (success.outsystems.com, 2023b)

allerdings einem bestimmten Format entsprechen, nämlich eine Spalte mit der Bezeichnung „E-Mail-Adresse“ und eine zweite Spalte mit der Bezeichnung „Name“. Nur so kann sichergestellt werden, dass die Datei richtig verarbeitet werden kann.

4.1.6. Appian

Von Appian wurde innerhalb des Zeitraums dieser Arbeit keine Testversion zur Verfügung gestellt. Laut Appian wurde aufgrund der steigenden Popularität ein Warteschlangensystem eingeführt, um sicherzustellen, dass jeder Nutzer Zugang zu den notwendigen Ressourcen und zum Support erhalten kann. In diesem System werden Personen, die eine Appian-Zertifizierung anstreben, und kleinere Gruppen, die Appian einführen wollen, bevorzugt behandelt. Aufgrund dessen wird Appian im nachfolgenden Vergleich nicht weiter beachtet.

4.2. Diskussion und Vergleich der Ergebnisse

4.2.1. Vergleichsbeschreibung

Um die Umsetzung des Testszenarios bestmöglich vergleichen zu können, sollen die Ergebnisse in einer Vergleichsmatrix veranschaulicht werden. Aus Gründen der Übersichtlichkeit, wird für jede Vergleichskategorie eine eigene Tabelle genutzt, die in den Punkten 4.2.2 bis 4.2.7 getrennt voneinander dargestellt werden. Die einzelnen Kriterien sind teilweise noch einmal in weitere Kriterien unterteilt, um eine vergleichbare Darstellung in den Tabellen zu ermöglichen.

Für die Kategorien Umsetzungsdauer, Herausforderungen und Dokumentation und Support werden keine weiteren Unterkriterien bewertet. Für die Kategorie Lernkurve und Nutzerfreundlichkeit wurden folgende Unterkategorien werden bewertet:

- Übersichtlichkeit: Die Bewertung der Übersichtlichkeit mit den Attributen schlecht, mittel und gut.
- Lernkurve: Die Bewertung der Lernkurve mit den Attributen niedrig, mittel und hoch.
- Anmerkungen: Welche Besonderheiten sind auffällig in Bezug auf die Nutzerfreundlichkeit.

Für die Kategorie Integration mit anderen Systemen werden folgende Unterkriterien bewertet:

- Anzahl: Die Anzahl der verfügbaren Systeme, auf die zugegriffen werden kann.
- Besonderheiten: Gibt es Besonderheiten bei den verfügbaren integrierbaren Systemen, wie z.B. ein System, was nur von einer der Plattformen angeboten wird?
- Zugang: Kann direkt in der Plattform auf die integrierbaren Systeme zugegriffen werden oder muss erst etwas heruntergeladen werden?

Für die Kategorie Low-Code Umsetzung werden folgende Unterkategorien bewertet:

- Code: Musste Code in einer Programmiersprache manuell geschrieben werden?
- Kenntnisse: Welche Kenntnisse waren für die Umsetzung des Testszenarios notwendig?

Für die Kategorien Umsetzungsdauer, Dokumentation & Support und Herausforderungen werden keine weiteren Unterkategorien bewertet.

4.2.2. Lernkurve & Nutzerfreundlichkeit

Plattform	Bewertung	
Microsoft Power Apps	Übersichtlichkeit:	gut
	Lernkurve:	mittel
	Anmerkungen:	Es können Fehlermeldungen auftreten, deren Grund nicht ersichtlich ist, da die gleiche Funktion an einer anderen Stelle in der App funktioniert.
Microsoft Power Automate	Übersichtlichkeit:	gut
	Lernkurve:	hoch
Oracle APEX	Übersichtlichkeit:	schlecht
	Lernkurve:	niedrig
	Anmerkungen:	Die Plattform wirkt zum Start sehr unübersichtlich und es ist kein direkter Startpunkt für die Entwicklung zu erkennen.
Mendix	Übersichtlichkeit:	mittel
	Lernkurve:	niedrig
	Anmerkungen:	Es ist kein direkter Startpunkt für die Entwicklung zu erkennen. Konnektoren-Module müssen erst heruntergeladen werden, was oftmals Fehlermeldungen mit sich bringt.
OutSystems	Übersichtlichkeit:	mittel
	Lernkurve:	hoch
	Anmerkungen:	Die Drag&Drop - Funktion für die Erstellung der Benutzeroberfläche ist etwas gewöhnungsbedürftig.

Tabelle 3: Bewertung Lernkurve und Nutzerfreundlichkeit

Die Kategorie Lernkurve & Nutzerfreundlichkeit ist mit eine der wichtigsten für die Einschätzung der Eignung einer Low-Code-Plattform für die Nutzung durch einen Citizen Developer. In dieser Kategorie wird bewertet, wie schnell ein neuer Nutzer sich in der jeweiligen Plattform zurechtfinden kann und wie schnell er Fortschritte im Lernprozess erzielen kann. Wichtig sind vor allem eine übersichtliche und intuitiv bedienbare Benutzeroberfläche und eine verständliche Funktionsweise der Plattform. Wie in Tabelle 3 zu erkennen ist konnten die 3 Plattformen Power Apps, Power

Automate und OutSystems Punkten, während Oracle APEX und Mendix eher schlechter abschnitten.

Power Automate sticht vor allem mit seiner sehr kompakten Nutzeroberfläche heraus und ist sowohl im Bereich Lernkurve als auch Im Bereich Nutzerfreundlichkeit die beste Plattform. Es ist ausschließlich dazu ausgelegt „Flows“ zu erstellen und nicht komplette Anwendungen. Die Plattform bietet keine Möglichkeit eine Benutzeroberfläche zu erstellen und ist in ihren Funktionen nur auf das nötigste beschränkt. Das stellt zwar einen Nachteil in Bezug auf den Funktionsumfang und die Flexibilität der Plattform dar, aber ist in Bezug auf die Lernkurve und die Nutzerfreundlichkeit von großem Vorteil. Die Plattform an sich ist sehr übersichtlich gestaltet und der Entwickler muss nicht viel Zeit aufwenden, um an einen Punkt zu kommen, an dem er die Hauptfunktionen beherrscht.

Microsoft Power Apps ist bei der Nutzerfreundlichkeit ebenfalls sehr gut aufgestellt, da die Plattform sehr übersichtlich ist und eine überschaubare Anzahl an Optionen hat, die allerdings ausreichen, um eine vollwertige Anwendung zu erstellen. Bei der Nutzerfreundlichkeit ist nur ein einziger negativer Punkt aufgefallen. Es können Fehlermeldungen bei Funktionen auftreten, bei denen der Grund nicht direkt ersichtbar ist, da die gleiche Funktion an anderen Stellen in der App ohne Fehler funktioniert. Bei der Lernkurve ist Power Apps allerdings nur im Mittelfeld der ausgewählten Plattformen zu finden. So ist es zwar gerade am Anfang möglich mit der Erstellung der Benutzeroberfläche per Drag&Drop schnell ein sichtbares Ergebnis zu erzeugen. Sobald es allerdings notwendig wird Code zu schreiben, kann der Fortschritt schnell ins Stocken geraten. Allerdings wirkt Power Apps dort etwas unterstützend und gibt Standard-Methoden vor, für die angezeigt wird, welche Eingabe als nächste erwartet wird. Die Schwierigkeit ist am Anfang somit eher die passende Methode zu finden und den erwarteten Datentyp richtig einzusetzen.

OutSystems muss sich im Vergleich mit Power Apps bei der Nutzerfreundlichkeit geschlagen geben, kann dafür allerdings mit einer höheren Lernkurve punkten. Die Plattform ist etwas unübersichtlicher, da deutlich mehr verschiedene Funktionen vorhanden sind. Gerade auch die Erstellung der Benutzeroberfläche mit Drag&Drop ist etwas gewöhnungsbedürftig, da die Elemente auf der Oberfläche nicht frei platziert werden können, obwohl die leere Seite dies suggeriert. Die Elemente müssen in Containern angeordnet werden, die allerdings vorher noch nicht vorhanden sind und erst mit dem Element eingefügt werden. Dieses Vorgehen ist von allen anderen Plattformen besser gelöst, sodass z.B. bei Power Apps die Elemente auf einer leeren Oberfläche frei angeordnet werden können, oder bei APEX und Mendix ein festes Layout mit Containern vorgegeben ist. Allerdings bietet die Plattform eine recht

Lernkurve, sodass es schon nach kurzer Zeit möglich ist die Plattform ausreichend zu verstehen. Für die Umsetzung muss zudem kein Code erstellt werden, sondern die Funktionalität kann durch „Server Actions“ oder andere Aktionen abgebildet werden, welche ähnlich wie ein „Flow“ in Microsoft Power Automate funktionieren und einfach zu verstehen sind.

Mendix ist in der Kategorie Lernkurve und Nutzerfreundlichkeit nur auf dem vorletzten Platz zu finden. Die Nutzerfreundlichkeit wird mit mittel bewertet, was vor allem der anfangs sehr unübersichtlichen Plattformen geschuldet ist, aber auch weiteren Dingen, die während der Arbeit mit der Plattform störend sind. So ist es zu Beginn der Arbeit mit der Plattform schwer einen Anfangspunkt zu finden. Während bei anderen Plattformen direkt eine leere Vorlage zur Erstellung der Benutzeroberfläche angezeigt wird, wird bei Mendix bis auf eine Menüleiste am Rand eine weitestgehend leere Plattform angezeigt. Je nach Anforderung müssen zu Beginn außerdem verschiedene Module installiert werden, die oftmals direkt mehrere Fehlermeldungen mitbringen, welche dann zuerst behoben werden müssen. Die Lernkurve wurde zudem mit niedrig bewertet, da es im Vergleich zu anderen Plattformen deutlich länger dauert, bis ein Fortschritt erkennbar ist. Es ist zwar nicht notwendig Code manuell zu schreiben, allerdings ist die Plattform trotzdem recht kompliziert, sodass die wenigsten Dinge ohne fremde Hilfe erarbeitet werden können.

Oracle APEX hat in der Kategorie Lernkurve und Nutzerfreundlichkeit am schlechtesten abgeschnitten. Die Plattform erhielt sowohl bei der Lernkurve als auch bei der Nutzerfreundlichkeit die schlechteste Bewertung. Die Plattform ist ähnlich wie Mendix sehr unübersichtlich und bietet keinen direkten Startpunkt für die Entwicklung. Die Entwicklungsumgebung ist an einzelne Seiten gekoppelt, was bedeutet, dass wenn mehrere Seiten in der Anwendung vorhanden sind und eine andere Seite bearbeitet werden soll, die Entwicklungsumgebung für die aktuelle Seite geschlossen und für die andere Seite erneut geöffnet werden muss. Das ist zwar nicht mit großem Zeitaufwand verbunden, aber ist ein sehr ungewöhnliches Verhalten, da bei allen anderen Plattformen direkt in der Entwicklungsumgebung auf alle Seite zugegriffen werden kann.

4.2.3. Interoperabilität mit anderen Systemen und Datenquellen

Plattform	Bewertung	
Microsoft Power Apps	Anzahl:	> 100 Konnektoren
	Besonderheiten:	Microsoft und Google Dienste sehr umfangreich
	Zugang:	direkt in der Plattform
Microsoft Power Automate	Anzahl:	> 100 Konnektoren
	Besonderheiten:	Microsoft und Google Dienste sehr umfangreich; Vordefinierte Aktionen pro Konnektor
	Zugang:	direkt in der Plattform
Oracle APEX	Anzahl:	nicht genau quantifizierbar
	Besonderheiten:	Konzentration auf Datenbanken und REST-Datenquellen
	Zugang:	Datenquellen über „Data Load“-Seite integrierbar
Mendix	Anzahl:	> 100 Konnektoren
	Besonderheiten:	Direkt integrierte Konnektoren: Datenbanken, Amazon AWS
	Zugang:	Module müssen heruntergeladen werden; Module werden sowohl von Mendix selbst als auch von Dritten angeboten
OutSystems	Anzahl:	> 100 Konnektoren
	Besonderheiten:	SAP, REST, SOAP
	Zugang:	Wenige Konnektoren direkt in der Plattform; Konnektoren von Dritten über „Marktplatz“ zur Verfügung gestellt

Tabelle 4: Bewertung der Interoperabilität mit anderen Systemen und Datenquellen

In Tabelle 4 sind die Bewertungen der Plattform zu den verfügbaren Systemen und Datenquellen dargestellt. Wie zu erkennen ist bieten alle Plattformen, bis auf Oracle APEX mehr als 100 verschiedene Konnektoren für die Anbindung von anderen Systemen und Datenquellen an. Bei APEX werden nicht direkt Konnektoren angeboten, sodass keine Aussage über die Anzahl getroffen werden kann. Bei allen anderen Plattformen wird keine konkrete Zahl der verfügbaren Konnektoren genannt,

allerdings übersteigt die Anzahl bei allen Plattformen die Marke von 100 Stück. Somit kann anhand der Anzahl der Konnektoren kein Sieger in dieser Kategorie festgestellt werden.

Bei den Microsoft Plattformen sticht vor allem die Anbindung anderer Microsoft Systeme, aber auch der Systeme von Google hervor. Außerdem sind alle Konnektoren direkt in den Plattformen verfügbar und müssen nicht erst heruntergeladen und installiert werden. Bei Power Automate ist eine weitere Besonderheit, dass zu den verfügbaren Systemen vordefinierte Aktionen verfügbar sind, sodass direkt ausgewählt werden kann, wie mit dem angebundenen System kommuniziert werden soll. Allerdings ist die Plattform auch auf diese Aktionen beschränkt und es ist nicht möglich anders mit den Systemen zu kommunizieren.

Bei OutSystems ist der Zugriff auf andere Systeme und Datenquellen geteilt. Auf Excel- und JSON-Dateien kann direkt aus einer „Server Action“ zugegriffen werden. Außerdem können direkt in der Plattform Verbindungen zu SAP, SOAP, REST und verschiedenen Datenbanken hergestellt werden. Für weitere Systeme und Datenquellen ist ein Marktplatz verfügbar, auf dem verschiedenste Konnektoren kostenlos heruntergeladen werden können. Diese Konnektoren werden sowohl von OutSystems selbst als auch von Dritten zur Verfügung gestellt.

Mendix nutzt ein ähnliches Vorgehen wie OutSystems. Bei Mendix kann direkt in der Plattform die Verbindung zu verschiedenen Datenbanken und Amazon AWS hergestellt werden. Für alle weiteren Konnektoren müssen Module auf dem Marktplatz heruntergeladen werden, die ebenfalls sowohl von Mendix selbst als auch von Dritten angeboten werden.

Bei Oracle APEX liegt die Konzentration generell eher auf Datenbanken und REST. Excel-Dateien können besipeilsweise über eine „Data Load“-Seite in die Anwendung geladen werden. Allerdings ist dabei nicht ersichtlich, ob dieses Vorgehen auch für weitere Datenquellen verfügbar ist. Auch hier ist das Problem, dass in der Testversion nicht alle Funktionen verfügbar sind, sodass dieser Punkt für APEX nicht weiter beurteilt werden kann.

4.2.4. Umsetzungsdauer

Plattform	Bewertung
Microsoft Power Apps	1 Stunde
Microsoft Power Automate	1 Stunde
Oracle APEX	Nicht fertig. (nach 7 Stunden vorzeitig beendet)
Mendix	Nicht fertig. (nach 16 Stunden beendet)
OutSystems	8,5 Stunden

Tabelle 5: Bewertung der Umsetzungsdauer

Bei der Umsetzungsdauer, die in Tabelle 5 abgebildet ist, ist eine Korrelation zu der Kategorie Lernkurve und Nutzerfreundlichkeit zu erkennen. Umso besser die Bewertung einer Plattform in Bezug auf ihre Lernkurve und ihre Nutzerfreundlichkeit, umso kürzer ist die Umsetzungsdauer des Testszenarios mit dieser Plattform. Somit ist auch in dieser Kategorie Power Automate die beste Plattform mit einer Umsetzungsdauer von 1 Stunde. Mit Power Apps wurde ebenfalls nur 1 Stunde für die Umsetzung benötigt, allerdings ist in diesem Fall zu beachten, dass keine Einarbeitungszeit notwendig war, da bereits Vorkenntnisse mit der Plattform vorhanden waren. Wären für Power Apps keine Vorkenntnisse vorhanden gewesen, wäre die Umsetzungsdauer wahrscheinlich eher im Bereich von 3 bis 4 Stunden gewesen. Somit wäre Power Apps bei gleichen Bedingungen gleichauf mit OutSystems. Bei OutSystems beträgt die Umsetzungsdauer zwar 8,5 Stunden, allerdings wurden ganze 5 Stunden davon für die Behebung eines Fehlers aufgewendet, welcher nicht OutSystems zuzuweisen ist. Der Fehler wurde durch den E-Mail-Versand mit Gmail verursacht und lag auf Seiten von Google, wodurch die eigentliche Entwicklungszeit mit OutSystems nur 3,5 Stunden beträgt.

Mit Mendix und Oracle APEX konnte das Testszenario nicht umgesetzt werden. Bei Mendix wurde die Umsetzung nach überschreiten des festgelegten Zeitlimits von 16 Stunden abgebrochen. Bei Oracle APEX wurde die Umsetzung bereits nach 7 Stunden abgebrochen, da festgestellt wurde, dass das Testszenario mit der kostenlosen Testversion von APEX nicht umgesetzt werden kann.

4.2.5. Dokumentation & Support

In der Kategorie Dokumentation und Support keine großen Unterschiede zwischen den Plattformen. Für jede der Plattformen ist eine Entwicklerdokumentation vorhanden, sowie auch ein offizielles Support-Forum.

Anzumerken sind allerdings einige Punkte die in den Entwicklerdokumentation einzelner Plattformen verbessert werden könnten. Der erste Punkt ist, dass bei OutSystems Informationen in der Entwicklerdokumentation fehlten, die eine längere Fehleranalyse und -behebung zur Folge hatten. Für die Konfiguration des E-Mail-Versands mit Gmail waren weitere Einstellungen direkt im Google-Konto notwendig, um E-Mails versenden zu können. Über diese Einstellungen war nichts in den Entwicklerdokumentationen zu lesen. Das ist zwar kein Thema, was direkt mit der Plattform zusammenhängt, allerdings kann man davon ausgehen, dass die Notwendigkeit der weiteren Einstellungen den Plattformen bekannt ist, da bei Mendix eine ausführliche Anleitung genau dazu zu finden war.

Bei Microsoft Power Apps sind dagegen Code-Beispiele hinterlegt, die in der dargestellten Form von der Plattform gar nicht akzeptiert werden. In der Entwicklerdokumentationen für Power Apps sind verschiedene Eingabeparameter in den Code-Beispielen immer mit einem Komma getrennt. Diese Schreibweise wird von der Plattform allerdings nicht akzeptiert, sondern es wird ein Semikolon gefordert. Der Fehler ist zwar in der Plattform schnell zu bemerken, da dort die Hinweise in der richtigen Form angegeben sind, allerdings sollten auch in der offiziellen Entwicklerdokumentation die angegeben Beispiele die richtige Schreibweise vorweisen.

4.2.6. Herausforderungen

Plattform	Bewertung
Microsoft Power Apps	E-Mail-Inhalt: Strings können in Power Apps nicht mit einem „+“ verknüpft werden, sondern müssen mit einem „&“ verknüpft werden, da Power Apps bei einem „+“ eine Zahl erwartet. ➔ Bewältigung: 10 Minuten; Durchsuchen der Entwicklerdokumentationen
Microsoft Power Automate	Es sind keine besonderen Herausforderungen aufgetreten.
Oracle APEX	Es konnten keine SMTP-Server-Einstellungen für den E-Mail-Versand vorgenommen werden. ➔ Bewältigung: nicht möglich; Umsetzung wurde abgebrochen
Mendix	E-Mail-Versand: Es gab keine vollständige, schrittweise Anleitung, wie die Konfiguration für den E-Mail-Versand vorzunehmen ist. ➔ Bewältigung: 3 Stunden; Lesen in Entwicklerdokumentationen und Support-Foren und selbst ausprobieren Error-Meldungen: Nach der Installation der Konnektoren-Module kam es zu Error-Meldungen, die behoben werden mussten, um die App zu starten. ➔ Bewältigung: 2,5 Stunden; Lesen in Support-Foren „Microflow“: Das Übergeben der Parameter von der Nutzeroberfläche an den „Microflow“ war nicht möglich. ➔ Bewältigung: nicht bewältigt; die Umsetzung wurde nach 16 Stunden abgebrochen
OutSystems	E-Mail-Versand: Für Gmail als E-Mail-Dienst waren weitere Einstellungen im Google-Konto notwendig, was von OutSystems nicht erwähnt wurde. ➔ Bewältigung: 5 Stunden; Lesen in Support-Foren und selbst ausprobieren

Tabelle 6: Bewertung Herausforderungen

Wie in Tabelle 6 dargestellt ist, sind allen Plattformen bis auf Power Automate verschiedene Herausforderungen aufgetreten, die die Entwicklungsarbeiten gestoppt haben und bewältigt werden mussten. Dabei gab es Herausforderungen deren Schwere eher gering war und in wenigen Minuten bewältigt werden konnten, bis hin zu schweren Herausforderungen, die nicht bewältigt werden konnten und einen Abbruch der Umsetzung mit sich führten.

Mit beiden Microsoft Plattformen konnte das Testszenario weitestgehend problemlos umgesetzt werden. Während es bei Power Automate keine Probleme gab, kam bei Power Apps die Herausforderung auf, dass der eingegebene Text für die E-Mail nicht

akzeptiert wurde. Der Text bestand aus einer Zeichenkette und einer Variable, die mit einem „+“ zusammengefügt wurden. Diese Kombination wurde von Power Apps nicht akzeptiert, da die Plattform bei einem „+“ eine Rechenoperation und somit Zahlen auf beiden Seiten des Zeichens erwartet. Die Lösung war die Verwendung des Zeichens „&“ mit dem in der Programmiersprache „Power FX“ Zeichenkette verbunden werden können. Die Bewältigung der Herausforderung gelang mit der Recherche nach einer Lösung in der Entwicklerdokumentation der Plattform und nahm ca. 10 Minuten in Anspruch.

Bei der Umsetzung des Testszenarios mit OutSystems gab es ebenfalls nur eine Herausforderung, welche aber deutlich zeitintensiver in der Bewältigung war. Wie bereits erwähnt waren für den E-Mail-Versand über Gmail weitere Einstellungen in dem verwendeten Google-Konto notwendig, welche allerdings von OutSystems selbst nicht erwähnt wurden. Die Bewältigung dauerte 5 Stunden und bestand größtenteils aus der Suche nach dem eigentlichen Fehler, da nicht ersichtbar war, warum der E-Mail-Versand nicht funktionierte. Mit der Hilfe aus mehreren verschiedenen Foreneinträgen konnte eine Lösung gefunden werden. Dafür musste im Google-Konto ein App-Passwort erstellt werden, welches in der Konfiguration für den E-Mail-Versand in OutSystems hinterlegt wurde. Das war notwendig, da Gmail den Zugang über die SMTP-Konfiguration ohne das App-Passwort aus Sicherheitsgründen nicht zulässt.
Bei Oracle APEX gab es ebenfalls eine Herausforderung, die allerdings gar nicht zu bewältigen war und die Umsetzung dadurch abgebrochen werden musste. Das Problem war auch in diesem Fall die Konfiguration des E-Mail-Versand. Diese konnte allerdings mit der kostenlosen Testversion von APEX gar nicht vorgenommen werden, da dafür eine Datenbank notwendig ist, welche in der Testversion nicht enthalten ist. Mit dieser Erkenntnis musste die Umsetzung des Testszenarios mit Oracle APEX beendet werden.

Deutlich mehr Herausforderungen gab es bei der Umsetzung des Testszenarios mit Mendix. Auch hier war wieder die Konfiguration des E-Mail-Versands ein Problem. In diesem Fall war in der Entwicklerdokumentationen nicht vollständig beschrieben, wie diese vorzunehmen ist. Somit blieb nur die Möglichkeit verschieden Dinge in der Plattform auszuprobieren und zu untersuchen, wie sich die Einstellungen für den E-Mail-Versand konfigurieren lassen. Mit einem Zeitaufwand von ca. 3 Stunden konnte die Herausforderung bewältigt werden. Eine weitere Herausforderung bei Mendix war, dass bei der Installation notwendiger Module mehrere Fehlermeldungen aufgetreten sind, die gelöst werden mussten, um die Anwendung starten zu können. Zur Bewältigung waren vor allem Beiträge in Support-Foren hilfreich, in denen die Beseitigung der meisten Fehlermeldungen beschrieben war. Die Bewältigung nahm 2,5 Stunden in Anspruch. Die dritte Herausforderung bei der Arbeit mit Mendix war,

dass Parameter von der Benutzeroberfläche nicht so einfach ein einen „Microflow“ übergeben werden konnten. Das wäre notwendig gewesen, um beispielsweise den Text der zu versendenden E-Mail vom Nutzer eingeben zu lassen und diesen anschließend an den „Microflow“ zu übergeben und in die E-Mail-Vorlage einzutragen. Diese Herausforderung konnte im vorgegebenen Zeitrahmen nicht mehr bewältigt werden.

4.2.7. Low-Code Umsetzung

Plattform	Bewertung	
Microsoft Power Apps	Code:	Es mussten 4 Zeilen Code in „Power FX“ geschrieben werden.
	Kenntnisse:	Variablen, Datentypen, Schleifen
Microsoft Power Automate	Code:	Es musste kein Code geschrieben werden.
	Kenntnisse:	Variablen, Schleifen
Oracle APEX	Code:	Bei einer weiteren Umsetzung hätten wahrscheinlich mehrere Zeilen Code in „SQL“ geschrieben werden müssen.
	Kenntnisse:	Variablen, Schleifen, SQL
Mendix	Code:	Es musste kein Code geschrieben werden.
	Kenntnisse:	Variablen, Schleifen, Objekte, Datentypen
OutSystems	Code:	Es musste kein Code geschrieben werden.
	Kenntnisse:	Variablen, Schleifen, Datentypen

Tabelle 7: Bewertung Low-Code Umsetzung

In Tabelle 7 zur Bewertung der Low-Code Umsetzung ist zu sehen, dass bei der Umsetzung des Testszenarios nur 2 der 5 Plattformen Code benötigten. Somit können die Plattformen Power Automate, Mendix und OutSystems sogar als No-Code-Plattform angesehen werden, da bei diesen Plattformen kein Code manuell geschrieben werden muss. Trotzdem sind für alle Plattformen einige Kenntnisse aus dem Bereich Programmierung notwendig. Unabhängig von der Plattform waren für das Testszenario Kenntnisse über Variablen und Schleifen notwendig. Das lag daran, dass das Szenario von sich aus so aufgebaut war, dass es notwendig war, mit einer Schleife jeden Datensatz der Excel-Datei einzeln zu verarbeiten, um die darin enthaltenen Daten per Variablen in eine E-Mail-Vorlage einzufügen. Dieser Prozess musste auch mit allen Plattformen in dieser Form umgesetzt werden.
Trotzdem kamen je nach Plattform weitere notwendige Kenntnisse hinzu. Bei Power Apps und OutSystems waren zusätzlich Kenntnisse über Datentypen notwendig. So musste bei OutSystems bei der Erstellung der Variablen ein Datentyp ausgewählt

werden und bei Power Apps kam es aufgrund von verschiedenen Datentypen zu einem Problem, welches bereits in Punkt 4.2.6 erklärt wurde. Bei Mendix kamen neben Datentypen auch Objekte zusätzlich hinzu. Dabei war es notwendig Objekte vom Typ E-Mail zu erstellen, welche anschließend mit Variablen des passenden Datentyps befüllt werden mussten.
Den Fall, dass Code für die Umsetzung manuell geschrieben werden muss, gab es nur bei den beiden Plattformen Power Apps und Oracle APEX. Bei APEX ist allerdings die Besonderheit, dass Code für die Umsetzung notwendig gewesen wäre, dieser Punkt durch einen frühzeitigen Abbruch der Umsetzung aber nicht erreicht wurde. In APEX wird die Abfragesprache SQL verwendet, da die Plattform hauptsächlich für den Gebrauch in Kombination mit einer Datenbank gedacht ist.
Bei der Umsetzung mit Power Apps wurde die Programmiersprache „Power FX“ genutzt. Dabei muss zwar grundsätzlich Code geschrieben werden, allerdings wirkt Power Plattform unterstützend mit, in dem es vorgefertigte Funktionen vorgibt, für die angezeigt wird, welche Eingabe als nächstes erwartet wird. Außerdem werden meist die passenden Eingaben direkt vorgeschlagen, sodass zumindest der Datentyp der Eingabe passend ist. Wenn beispielsweise von einer Funktion eine Sammlung erwartet wird, werden alle vorhandenen Sammlungen für die Eingabe vorgeschlagen. Solange die Anforderungen an die Funktionalität nicht zu komplex werden, müssen für den notwendigen Code in Power Apps eher passende Eingaben ausgewählt, als selbst geschrieben werden.

4.2.8. Auswertung

Im Folgenden sollen die in den Punkten 4.2.2 bis 4.2.7 veranschaulichten und beschrieben Ergebnisse abschließend verglichen werden. Zuerst einmal lässt sich festhalten, dass das gewählte Testszenario nur mit den 3 Plattformen Power Apps, Power Automate und OutSystems in der vorgegebenen Zeit von 16 Stunden umgesetzt werden konnte. Bei der Plattform Oracle APEX wurde die Umsetzung vorzeitig beendet, da mit der kostenlosen Testversion ohne Datenbankzugriff keine SMTP-Server-Konfiguration für den E-Mail-Versand möglich ist. konnte das Testszenario mit der Plattform Mendix nicht in der vorgegebenen Zeit umgesetzt werden, sodass die Arbeit mit der Plattform nach Ablauf der 16 Stunden beendet wurde. Da der Zugang zu der Plattform Appian während der Bearbeitungszeit dieser Arbeit nicht möglich war, wird diese, wie bereits in Punkt 4.1.6 angemerkt, nicht weiter betrachtet.

Von der minimalen Funktionalität der entwickelten Anwendung ausgehend ist Microsoft Power Automate der klare Sieger des Vergleichs. Power Automate hatte sowohl die kürzeste Umsetzungsdauer als auch die beste Nutzerfreundlichkeit und die höchste

Lernkurve. Außerdem sind keine weiteren Herausforderungen bei der Umsetzung aufgetreten und es waren im Vergleich zu den anderen Plattformen die wenigsten Kenntnisse im Bereich Programmierung notwendig. In den Kategorien Dokumentation & Support und Interoperabilität mit anderen Systemen und Datenquellen konnten zudem keine Nachteile gegenüber den anderen Plattformen festgestellt werden. Der einzige Nachteil von Power Automate ist, dass keine komplette Anwendung erstellt werden kann und somit keine Eingaben über eine Benutzeroberfläche entgegengenommen werden können. Somit ist es nicht möglich den Betreff oder den Inhalt der E-Mail zu individualisieren. Ein mögliches Szenario für die in Power Automate umgesetzte Lösung könnte eine wiederkehrende gleichbleibende Benachrichtigung sein, die an eine individuelle Auswahl an Personen gesendet werden soll. Die Auswahl an Empfängern kann direkt über eine Änderung in der Excel-Datei angepasst werden.

In etwa gleichauf auf Platz 2 sind Power Apps und OutSystems. Der einzige größere Unterschied zwischen den Plattformen ist die Dauer der Umsetzung, allerdings wurde in Punkt 4.2.4 bereits erklärt, dass die Umsetzungsdauer bei gleichen Bedingungen bei den beiden Plattformen ungefähr gleich gewesen wäre. Ebenso gibt es keine großen Unterschiede in der Kategorie Dokumentation & Support. In der Kategorie Interoperabilität sind Unterschiede vorhanden, bei denen allerdings nicht grundsätzlich eine Plattform als besser bewertet werden kann, sondern je nach Anforderung entschieden werden muss. Für das umgesetzte Testszenario waren die zur Verfügung stehenden Systeme und Datenquellen bei beiden Plattformen ausreichend. Bei beiden Plattformen gab es keine größeren Herausforderungen, zumindest keine, die direkt von der Plattform selbst verschuldet gewesen wären. In der Kategorie Lernkurve & Nutzerfreundlichkeit gleichen sich die Vor- und Nachteiler beider Plattformen aus und auch im Bereich Low-Code Umsetzung sind beide Plattformen ähnlich aufgestellt, bis auf den Punkt, dass bei Power Apps manueller Code geschrieben werden muss, was allerdings bei der geringen Komplexität des Testszenarios nicht groß ins Gewicht fällt. Power App überzeugt vor allem mit seiner übersichtlichen und aufgeräumten Plattform und der guten Interoperabilität mit anderen Systemen von Microsoft und der direkten Einbindung dieser in die Anwendung. Dagegen bietet OutSystems eine höhere Lernkurve und eine Erstellung der Funktionalität der Anwendung über „Server Actions“, wodurch es nicht notwendig ist Code manuell zu schreiben. Außerdem ist OutSystems durch die Unterstützung von REST-APIs sehr gut für die Entwicklung von Web-Anwendungen geeignet und kann dabei sowohl für den Frontend- als auch für den Backend-Part verwendet werden.

Für die Plattformen Power Automate, Power Apps und OutSystems kann festgehalten werden, dass diese für das umgesetzte Testszenario für einen Citizen Developer

geeignet sind. Die Plattformen Mendix und Oracle APEX sind für die Umsetzug des Testszenarios durch einen Citizen Developer nicht geeignet. Mit Mendix wäre das Testszenario zwar höchstwahrscheinlich umsetzbar gewesen, allerdings mit einem deutlich höheren Zeitaufwand im Vergleich zu den anderen Plattformen. Im Vergleich ist Mendix weniger geeignet für den Citizen Developer, da die Plattform eine schlechtere Lernkurve und Nutzerfreundlichkeit bietet und deutlich mehr Herausforderungen mit sich brachte. Allerdings schneidet die Plattform in den Kategorien Interoperabilität mit anderen Systemen und Datenquellen, Dokumentation & Support und vor allem auch in der Low-Code Umsetzung nicht schlechter ab als die anderen Plattformen.

Mit Oracle APEX konnte das Testszenario aufgrund des geringeren Funktionsumfangs der Testversion nicht umgesetzt werden. Die Plattform ist generell ungeeignet für einen Citizen Developer, da zum einen die Nutzerfreundlichkeit und die Lernkurve im Vergleich am schlechtesten bewertet wurden und zum anderen aufgrund der bevorzugten Nutzung in Verbindung mit einer Oracle Datenbank SQL-Kenntnisse notwendig sind.

Die beste Lösung für die Umsetzung des Testszenarios, auch über die Mindestanforderungen hinaus, wäre vermutlich die Kombination den Plattformen Power Apps und Power Automate gewesen. Die Möglichkeit der Vereinigung der beiden Plattformen wird von Microsoft selbst angeboten, sodass in einer Power App ein „Flow“ integriert werden kann. Somit wäre es möglich gewesen eine Anwendung zu erstellen, die eine Benutzeroberfläche besitzt, aber für die kein Code für die Erstellung notwendig gewesen wäre. Somit hätte die Anwendung mehr Funktionen, als die Lösung mit Power Automate umgesetzt wurde und wäre einfacher zu erstellen gewesen als die Anwendung, die mit Power Apps entwickelt wurde. Ein ähnliches Vorgehen wird zwar ebenfalls von OutSystems in einer einzelnen Plattform angeboten, aber aufgrund der guten Interoperabilität der Microsoft Plattformen zu anderen Microsoft Systemen wie Excel oder Outlook, sind diese Plattformen für das gewählte Szenario im Vorteil.

5. Ergebnisse

5.1. Fazit

Aufgrund vorangegangener Erfahrungen mit Microsoft Power Apps konnte die Erkenntnis erlangt werden, dass ein Citizen Developer bei der Arbeit mit der Plattform recht schnell an seine Grenzen gelangen kann. Es sollte untersucht werden, ob sich diese Beobachtungen auf andere Low-Code-Plattformen übertragen lassen. Um die Eignung verschiedener Low-Code-Plattformen für die Zielgruppe des Citizen Developers zu untersuchen, wurde ein einheitliches, realitätsnahes Testszenario entworfen, welches mit einer Auswahl an Plattformen umgesetzt wurde. Die Umsetzung mit den einzelnen Plattformen wurde anhand von verschiedenen qualitativen und quantitativen Kriterien bewertetet, woraus eine Eignung der jeweiligen Plattform für den Citizen Developer abgeleitet wurde.
Zudem wurde eine Umfrage unter Mitarbeitern der BE-terna GmbH durchgeführt, um die Anzahl der potenziellen Citizen Developer in einem IT-Unternehmen aufzuzeigen und diese Anzahl ins Verhältnis zu den professionellen Softwareentwicklern zu setzen. Außerdem sollte untersucht werden, inwiefern die Personen, die, aufgrund ihrer technischen Kenntnisse, als potenzielle Citizen Developer eingestuft werden können, sich mit dem Begriff identifizieren können.

Mit Blick auf die Ergebnisse des Plattformenvergleichs kann festgehalten werden, dass es tatsächlich Plattformen gibt, die es einem Citizen Developer, also einer Person ohne umfassende Kenntnisse im Bereich der Softwareentwicklung, ermöglichen eigene Anwendungen zu erstellen. Im Besonderen erscheinen die Plattformen Microsoft Power Automate, Microsoft Power Apps und OutSystems geeignet, um die Entwicklung einer einfachen Anwendung durch einen Citizen Developer zu ermöglichen. Es konnte allerdings auch herausgefunden werden, dass die Nutzung durch einen Citizen Developer nicht bei jeder Plattform gleichermaßen empfehlenswert ist.
In der Vergleichsbewertung ist zu erkennen, dass eine gute Nutzerfreundlichkeit und eine hohe Lernkurve bis zu einem bestimmten Punkt deutlich wichtiger sind, als die Möglichkeit Anwendungen vollständig ohne Code entwickeln zu können. So schneiden Plattformen, bei denen Code manuell geschrieben werden muss, im Vergleich teilweise sogar besser ab als Plattformen, die eine Entwicklung ohne Code anbieten. Das ist vor allem durch eine leicht verständliche und intuitive Plattform und einer gezielten Hilfestellung bei der Erstellung des Codes möglich. Es lässt sich sagen, dass vor allem eine übersichtliche und intuitiv bedienbare Plattform, deren Hauptfunktionen schnell zu erlernen sind, notwendig ist, um von einem Citizen Developer genutzt werden zu können.

Allerdings spielen ebenso die Anforderungen an die Anwendung, die erstellt werden soll, eine große Rolle. Umso komplexer diese sind, umso wahrscheinlicher ist es auch, dass eine Umsetzung durch einen Citizen Developer nicht möglich ist. Ab einem bestimmten Komplexitätsgrad kann es notwendig werden eine größere Menge an Code zu schreiben, was durch einen Citizen Developer nicht so einfach umzusetzen ist. Außerdem ist es ebenfalls möglich, dass die Plattform selbst, durch die höhere Komplexität, an ihre Grenzen kommt und die Anwendung gar nicht umsetzbar ist.
Außerdem sollte bei der Auswahl der Plattformen darauf geachtet werden, dass diese zu den Anforderungen der zu erstellenden Anwendung passt. Umso besser eine Plattform für die Umsetzung eines bestimmten Szenarios geeignet ist, umso einfacher lässt sich dieses Szenario mit der Plattform umsetzen. So ist es logisch, dass beide Microsoft Plattformen im Vergleich sehr gute Ergebnisse erzielen konnte, obwohl Power Apps in Punkt 2.3 sogar als Negativbeispiel aufgeführt wurde. Das liegt daran, dass die Anforderungen des Testszenarios sehr gut zu den Stärken der Plattformen passen und die Umsetzung somit gut bewältigt werden konnte. Gerade bei den Plattformen von Microsoft liegen die Vorteile in der guten Interoperabilität mit anderen Systemen von Microsoft. Davon konnte vor allem beim Import der Excel-Datei profitiert werden, da die Anbindung von Excel an die Plattformen sehr einfach umzusetzen war.

Anhand der Ergebnisse der Vergleichsbewertung lässt sich schlussfolgern, dass der Umsetzung von kleineren, weniger komplexen Anwendungen durch einen Citizen Developer keine Hindernisse im Weg stehen würden. Somit könnten bis zu einem bestimmten Komplexitätsgrad und mit der Wahl der richtigen Plattform viele Anwendungen durch einen Citizen Developer umgesetzt und die professionellen Softwareentwickler damit entlastet werden. Der einzige limitierende Faktor für die Anwendungsentwicklung durch einen Citizen Developer scheint also der Komplexitätsgrad der gestellten Anforderungen zu sein.
Ein anderes Bild liefern dagegen die Ergebnisse der durchgeführten Umfrage. Dort ist klar erkennbar, dass der potenzielle Citizen Developer ebenfalls einen limitierenden Faktor darstellt, indem er zum einen gar nicht gewillt ist, als solcher zu agieren, und zum anderen sich nicht bewusst ist, dass er die notwendigen Fähigkeiten besitzt, um als Citizen Developer aktiv zu werden.
Aus den Umfrageergebnissen ist zu entnehmen, dass eine Vielzahl an Personen vorhanden ist, die anhand ihrer Fähigkeiten als potenzielle Citizen Developer einzuschätzen sind. So wurde herausgefunden, dass die Gruppe der potenziellen Citizen Developer im Vergleich zu den professionellen Softwareentwicklern und den Nicht-Entwicklern mit 42% den größten Anteil bei BE-terna besitzt. Vor allem in den technischen Tätigkeitsfeldern sind viele Personen anzutreffen, die per Definition dem Citizen Developer zugeordnet werden können. Im Gegenzug dazu wurde allerdings ermittelt, dass sich nur ein kleiner Teil der Personen, die als potenzielle Citizen

Developer anzusehen sind, auch als solche identifizieren können. So gaben die meisten Personen, unabhängig von ihren technischen Kenntnissen, an, sich wenig bis gar nicht mit dem Begriff „Citizen Developer“ identifizieren zu können.

Aus den Umfrageergebnissen kann also geschlussfolgert werden, dass die Umsetzung der Anwendungsentwicklung durch Citizen Developer nicht nur von Seiten der Low-Code-Plattformen limitiert wird, sondern auch durch die Citizen Developer selbst. Um den Einsatz von Low-Code-Plattformen voranzubringen, müssten die potenziellen Citizen Developer vermehrt an diese herangeführt werden. Somit ist es nicht ausreichend, dass Low-Code-Plattformen von der IT-Abteilung zur Verfügung gestellt werden, sondern die Citizen Developer müssten im Umgang mit den Plattformen geschult und betreut werden. Die Schulungen sollte den Citizen Developern allerdings nicht nur die Funktionen und die Bedienung der Plattformen nahebringen, sondern vor allem auch zeigen mit welchen Erwartungen sie an die Plattformen herantreten können. So wäre es beispielsweise wichtig für den Citizen Developer zu wissen, welche Anforderungen umgesetzt werden können und auch welche Anforderung, mit Blick auf die Datensicherheit, umgesetzt werden dürfen.
Die IT-Abteilung müsste somit eine deutlich größere Rolle einnehmen als bisher angenommen. Sie müsste also nicht nur die Plattformen bereitstellen, sondern wäre darüber hinaus für die Schulung und Betreuung im Umgang mit diesen verantwortlich. Zudem müsste die Richtlinien für den Umgang mit den Plattformen entwickeln und bereitstellen. Ebenfalls wäre es denkbar, dass Key-User für die Plattformen zur Verfügung gestellt werden, die die Citizen Developer bei der Umsetzung von Anwendungen unterstützen können.

5.2. Ausblick

Aufgrund der validen Ergebnisse der Arbeit durch die angewandte Methodik ist eine Weiterverfolgung des Ansatzes denkbar. So ist es möglich die Untersuchungen der Arbeit zu vertiefen und zum anderen die Perspektive der Arbeit zu erweitern.
Für die Vertiefung der Forschungsarbeit wäre es notwendig für die Plattformen mehrere differenzierte Szenarien umzusetzen und zu testen. Somit wäre es möglich ein umfassenderes Bild zur Eignung der Plattformen für den Citizen Developer zu erhalten. Ein mögliches Vorgehen wäre, auf den bereits evaluierten Plattformen weitere Testszenarien umzusetzen. Dabei sollten die verschiedenen Szenarien stufenweise in der Komplexität gesteigert werden, um herausfinden zu können, bis zu welchem Komplexitätsgrad die Nutzung der Plattform möglich ist. Bei einer ausreichend differenzierten Auswahl der Testszenarien wäre es somit auch möglich die Plattformen hinsichtlich ihrer Stärken und Schwächen tiefer zu bewerten und ausführlicher auf die Zielszenarien der einzelnen Plattformen eingehen zu können.

Für eine weitere Perspektive der Arbeit wäre es möglich Low-Code-Plattformen unabhängig vom Faktor Citizen Developer auf ihre Grenzen zu testen. Dabei könnte untersucht werden, ab welchem Komplexitätsgrad die Plattformen mit einem erfahren Softwareentwickler an ihre Grenzen geraten und ab welchem Punkt es ratsam ist auf die traditionelle Softwareentwicklung zu setzen. Dabei könnte mit einem ähnlichen Konzept vorgegangen werden, dass differenzierte Szenarien mit steigendem Komplexitätsgrad umgesetzt werden. Diese Umsetzung würde sowohl mit einer Auswahl an Low-Code-Plattformen als auch mit einer traditionellen Programmiersprache erfolgen. Dabei wäre es sogar denkbar, dass, aufgrund des deutlich umfassenderen Wissensstands bezüglich Programmiersprachen und ihrem bevorzugten Anwendungsbereich, für jeden Testszenario die jeweils am besten geeignete Programmiersprache ausgewählt wird.

Erweiternd zum Thema Low-Code-Plattformen wäre es interessant den Einfluss von künstlicher Intelligenz auf das berufliche Umfeld und vor allem auf den Bereich der Anwendungsentwicklung, auch ohne Programmierkenntnisse, zu untersuchen. Besonders die zukünftige Entwicklung von Low-Code-Plattformen könnte stark von der Integration fortschrittlicher KI-Technologien geprägt sein. Diese könnten die Hürden für den Citizen Developer weiter senken, indem sie intuitive Schnittstellen bieten, die auf natürlicher Sprache oder visuellen Gesten basieren. Eine Möglichkeit wäre die Implementierung einer Conversational AI, die es Nutzern ermöglicht, mit der Plattform in natürlicher Sprache zu agieren und so Anforderungen und Funktionen für ihre Anwendung zu definieren. Dadurch würde die Notwendigkeit, technische Spezifikationen in einer Programmiersprache zu formulieren, weiter reduziert.
Darüber hinaus könnte KI-gestützte Code-Generierung eine Schlüsselrolle bei der Vereinfachung der Anwendungsentwicklung für Citizen Developer spielen. Plattformen könnten in der Lage sein, die Absicht des Nutzers zu verstehen und automatisch den erforderlichen Code zu generieren, um die gewünschten Funktionen umzusetzen. Das würde die Abhängigkeit von vorgefertigten Bausteinen und Vorlagen nochmals reduzieren und eine größere Flexibilität bei der Anwendungsentwicklung bieten.
Ein vielversprechender Bereich ist ebenfalls die automatisierte Fehlererkennung und -behebung mithilfe von KI. Dies könnte Citizen Developer dabei unterstützen Fehler und Inkonsistenzen in ihren Anwendungen zu identifizieren und automatische Lösungsvorschläge zu erhalten, ohne tief in den Programmierdetails stecken zu müssen.
Es ist allerdings auch denkbar, dass Low-Code-Plattformen aufgrund der fortschreitenden Entwicklung von KI-Technologien, zunehmend an Bedeutung verlieren könnten. Wenn eine KI in der Lage sein sollte, den gesamten Entwicklungsprozess, einschließlich der Architektur, des Designs und der Implementierung, zu automatisieren, könnte das einen Paradigmenwechseln in der

Softwareentwicklung bedeuten. In einem solchen Szenario könnten Low-Code-Plattformen ihre Daseinsberechtigung verlieren, da die traditionelle Softwareentwicklung mit KI-gestützter Code-Generierung möglicherweise eine größere Flexibilität und Anpassungsfähigkeit bieten würde, da nicht auf vorgefertigte Elemente zurückgegriffen werden muss.
Es sollte allerdings beachtet werden, dass die automatische Generierung einer vollständigen Anwendung nicht nur das Verständnis der funktionalen Anforderungen erfordert, sondern auch das Verständnis der zugrundeliegenden Geschäftslogik und der Benutzererwartungen. In komplexen Anwendungen kann es schwierig sein, alle diese Aspekte zu erfassen und von einer KI-Instanz umsetzen zu lassen. Entwickler könnten somit trotzdem eine Rolle bei der Feinabstimmung und der Sicherstellung der Qualität und Sicherheit spielen.
Insgesamt wird die Integration von KI-Technologien die Softwareentwicklung, vor allem auch ohne Programmierkenntnisse, verändern. Die Möglichkeiten reichen von der Erweiterung der kreativen Möglichkeiten von Nicht-Programmierern bis hin zur Automatisierung technischer Aspekte des Entwicklungsprozesses. Die Zukunft könnte eine Ära sein, in der KI und Mensch Hand in Hand arbeiten, um innovative und maßgeschneiderte Anwendungen zu schaffen.

Literaturverzeichnis

ADRIAN, B., HINRICHSEN, S., SCHULZ, A., & VOß, E. (2020). Low-Code-Programmierung als Ansatz zur Gestaltung bedarfsgerechter informatorischer Assistenzsysteme - eine Fallstudie. In B. ADRIAN, S. HINRICHSEN, A. SCHULZ, & E. VOß, *Informatorische Assistenzsysteme in der variantenreichen Montage : Theorie und Praxis.* Springer Berlin Heidelberg. Abgerufen am 26. Juni 2023

AMBATI, S. M. (06. August 2018). *triology.de*. Abgerufen am 04. Juli 2023 von Mehrsprachige Datenbankanwendungen mit Oracle APEX: https://www.triology.de/blog/datenbankanwendungen-oracle-apex

apex.oracle.com. (kein Datum). Abgerufen am 03. August 2023 von https://apex.oracle.com/de/

appian.com. (kein Datum). Abgerufen am 03. August 2023 von Appian Low-Code Platform Pricing: https://appian.com/products/platform/pricing.html

AUGSTEN, S. (31. Mai 2019). *dev-insider.de*. Abgerufen am 04. Juli 2023 von Was ist ein Citizen Developer?: https://www.dev-insider.de/was-ist-ein-citizen-developer-a-827524/

BREHME, S. (30. Januar 2023). *Computerwoche.de*. Abgerufen am 28. Juni 2023 von Was Sie über No-Code-Plattformen wissen müssen: https://www.computerwoche.de/a/was-sie-ueber-no-code-plattformen-wissen-muessen,3550759

ComputerWeekly.de. (Mai 2020). *computerweekly.com*. Abgerufen am 04. Juli 2023 von Microsoft Power Automate (ehemals Flow): https://www.computerweekly.com/de/definition/Microsoft-Flow

docs.mendix.com. (20. April 2023). Abgerufen am 1. August 2023 von Microflows and Nanoflows: https://docs.mendix.com/refguide/microflows-and-nanoflows/

ETEMADIAN, R. (07. Mai 2023). *Computerwoche.de*. Abgerufen am 28. Juni 2023 von Was ist ein Citizen Developer?: https://www.computerwoche.de/a/was-ist-ein-citizen-developer,3550357

FELDHERR, T. (12. Mai 2021a). *mission-mobile.de*. Abgerufen am 05. Juli 2023 von Mendix: https://mission-mobile.de/knowhow/mendix/#was-ist-mendix

FELDHERR, T. (09. Juni 2021b). *mission-mobile.de*. Abgerufen am 05. Juli 2023 von OutSystems: https://mission-mobile.de/knowhow/outsystems/

FELDHERR, T. (02. März 2023a). *mission-mobile.de*. Abgerufen am 04. Juli 2023 von Die Top 20 Low-Code Plattformen im Vergleich: https://mission-mobile.de/knowhow/low-code-plattformen-im-vergleich/

FELDHERR, T. (23. Mai 2023b). *mission-mobile.de*. Abgerufen am 30. Juni 2023 von Microsoft Power Apps: https://mission-mobile.de/knowhow/microsoft-power-apps/

go-make-it.de. (31. August 2022). Abgerufen am 04. Juli 2023 von Was ist eigentlich - Ein Citizen Developer: https://go-make-it.de/was-ist-eigentlich-ein-citizen-developer/

HAUSCHILDT, P.-S. (kein Datum). *nativdigital.com*. Abgerufen am 03. Juli 2023 von Was ist ein Citizen Developer?: https://nativdigital.com/citizen-developer/

HELLER, M. (28. 05 2023). *Computerwoche.de*. Abgerufen am 28. Juni 2023 von Was ist Low-Code?: https://www.computerwoche.de/a/was-ist-low-code,3551643

Ionos.de. (28. April 2020). Abgerufen am 28. Juni 2023 von Low Code: Definition, Besonderheiten und Einsatzgebiete: https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-low-code/

KAY, C. (19. Juli 2023). *employer-it-talents.de*. Abgerufen am 08. August 2023 von it-fachkräftemangel: https://employer.it-talents.de/blog/it-fachkraeftemangel/

learn.microsoft.com. (05. Juni 2023). Abgerufen am 09. August 2023 von Übersicht über die verschiedenen Arten von Flows: https://learn.microsoft.com/de-de/power-automate/flow-types

LINKE, P. (14. Januar 2021). *promx.net*. Abgerufen am 03. Juli 2023 von Canvas vs. Model-driven Power Apps: Wie unterscheiden sie sich?: https://promx.net/2020/10/canvas-vs-model-driven-power-apps/

mendix.com. (kein Datum). Abgerufen am 03. August 2023 von preisgestaltung: https://www.mendix.com/de/preisgestaltung/

mendix.com. (03. September 2020). Abgerufen am 03. August 2023 von Low-Code-Plattform von Mendix setzt neue Maßstäbe mit neuen Gold-Standard-Zertifizierungen für die Verarbeitung von Gesundheits- und Zahlungsdaten: https://www.mendix.com/de/presse/low-code-plattform-von-mendix-setzt-neue-massstaebe-mit-neuen-gold-standard-zertifizierungen-fu%CC%88r-die-verarbeitung-von-gesundheits-und-zahlungsdaten/

NOACK, K. (02. Juli 2018). *t2informatik.de*. Abgerufen am 28. Juni 2023 von Was sind Low-Code-Plattformen?: https://t2informatik.de/blog/was-sind-low-code-plattformen/

oracle.com. (kein Datum). Abgerufen am 03. August 2023 von https://www.oracle.com/de/cloud/costestimator.html

outsystems.com. (kein Datum). Abgerufen am 03. August 2023 von OutSystems pricing: https://www.outsystems.com/de-de/pricing-and-editions/

powerapps.microsoft.com. (kein Datum). Abgerufen am 03. August 2023 von Power Apps - Preise: https://powerapps.microsoft.com/de-de/pricing/

powerautomate.mircosoft.com. (kein Datum). Abgerufen am 03. August 2023 von Power Automate - Preisübersicht: https://powerautomate.microsoft.com/de-de/pricing/

SAFAR, M. (kein Datum). *Weissenberg-Group.de*. Abgerufen am 28. Juni 2023 von Citizen Developer - Entwicklungsressource von Morgen: https://weissenberg-group.de/citizen-developer-entwicklungsressource-von-morgen/

sharepoint360.de. (07. Februar 2019). *sharepoint360.de*. Abgerufen am 03. Juli 2023 von Vor- und Nachteile von Power Automate (Flow): Für welche Szenarien es sich optimal eignet, und wo die Grenzen liegen: https://sharepoint360.de/vor-und-nachteile-von-microsoft-flow-fuer-welche-szenarien-es-sich-optimal-eignet-und-wo-die-grenzen-liegen/

STAAR, L. (04. August 2022). *flixcheck.de*. Abgerufen am 05. Juli 2023 von 10 Low-Code-Plattformen im Überblick: https://www.flixcheck.de/low-code-plattformen/

success.outsystems.com. (23. März 2023a). Abgerufen am 01. August 2023 von Client Action: https://success.outsystems.com/documentation/11/reference/outsystems_language/logic/implementing_logic/logic_elements/client_action/

success.outsystems.com. (23. März 2023b). Abgerufen am 01. August 2023 von Server Action: https://success.outsystems.com/documentation/11/reference/outsystems_language/logic/implementing_logic/logic_elements/server_action/

t2informatik. (kein Datum). Abgerufen am 27. Juni 2023 von Low-Code-Entwicklung: https://t2informatik.de/wissen-kompakt/low-code-entwicklung/

VASKE, H. (11. Februar 2022). *computerwoche.de*. Abgerufen am 04. Juli 2023 von Low-Code-Plattformen auf einen Blick: https://www.computerwoche.de/a/low-code-plattformen-auf-einen-blick,3544905

VINCENT, P., & a., u. (31. Dezember 2022). *gartner.com*. Abgerufen am 02. August 2023 von Magic Quadrant for Enterprise Low-Code Application Platforms: https://www.gartner.com/doc/reprints?id=1-2C8VSOAH&ct=230113&st=sb

WELSCH, L. (01. Februar 2023). *dynamics.konicaminolta.de*. Abgerufen am 03. Juli 2023 von Die Microsoft Power Platform erklärt: https://dynamics.konicaminolta.de/blog/power-platform/

Anhangsverzeichnis

Umfrage – Citizen Developer bei BE-terna

Hallo zusammen,
mein Name ist Ruben, ich studiere Wirtschaftsinformatik an der BA Glauchau und ich schreibe aktuell meine Bachelorarbeit zum Thema "Eignung von Low-Code-Plattformen für die Zielgruppe des Citizen Developers".
Hierzu habe ich eine Umfrage erstellt, um herauszufinden wie viele Mitarbeiter bei BE-terna potenzielle Citizen Developer sind und wie diese Anzahl im Verhältnis zu den hauptberuflichen Softwareentwicklern steht.
Ich würde mich sehr freuen, wenn ihr 2-3 Minuten Zeit hättet, um an der Befragung teilzunehmen.
Die Daten werden anonymisiert und vertraulich behandelt.
Bei Fragen könnt ihr mich gerne kontaktieren.
Vielen Dank
Ruben

Frage 1: In welchem Bereich bist du bei BE-terna tätig?

- Consulting
- Development
- Architect
- Management
- HR/Recruiting
- Sales/Marketing
- Finance/Accounting
- Administration/Controlling
- Sonstiges

Frage 2: Besitzt du Programmierkenntnisse?

- Ja
- Nein

Frage 3: Ist das Programmieren deine Haupttätigkeit?

- Ja
- Nein

Frage 4: Mit was programmierst du?

- Abfragesprachen (z.B. SQL)
- Skriptsprachen
- Objektorientierte Programmiersprachen
- Low-Code (z.B. Power Apps)

Frage 5: Hast du schon einmal mit einer dieser Low-Code-Plattformen gearbeitet?

- Microsoft Power Apps
- Microsoft Power Automate
- Oracle APEX
- Mendix
- OutSystems
- Appian
- Sonstige

Frage 6: Der Begriff Citizen Developer beschreibt Mitarbeiter, die keine formale Ausbildung in der Softwareentwicklung, aber ein gewisses technisches Verständnis haben, wodurch sie in der Lage sind Anwendungen für ihren jeweiligen Fachbereich zu erstellen. Citizen Developer schaffen eigene Lösungen für Anforderungen in ihrem Arbeitsalltag, da sie als Endnutzer und Experten ihres Fachbereichs die Prozesse innerhalb ihrer Abteilung sehr gut kennen. Dafür nutzen sie meist Low-Code-Plattformen, in denen mit Drag&Drop und einfachen Skripten beispielsweise Prozesse automatisiert, Tätigkeiten zeitgesteuert oder Verbindungen zwischen bestehenden Programmen geschaffen werden.
Wie sehr kannst du dich mit dem Begriff „Citizen Developer“ identifizieren? (1 – nicht; 5 - stark)

- 1
- 2
- 3
- 4
- 5

ID	Zeitstempel	Frage 1	Frage 2	Frage 3	Frage 4	Frage 5	Frage 6
1	7.25.23 7:55:20	Development	Ja	Ja	objektorientierte Programmiersprachen;Low-Code (z.B. Power Apps);Abfragesprachen (z.B. SQL);		
2	7.25.23 8:06:57	Sales/Marketing	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	4
3	7.25.23 8:07:25	Development	Ja	Ja	Skriptsprachen;objektorientierte Programmiersprachen;Abfragesprachen (z.B. SQL);		
4	7.25.23 8:07:29	Development	Ja	Ja	objektorientierte Programmiersprachen;Abfragesprachen (z.B. SQL);		
5	7.25.23 8:07:47	Consulting	Ja	Ja	objektorientierte Programmiersprachen;C/AL, AL;		
6	7.25.23 8:07:48	Development	Ja	Ja	objektorientierte Programmiersprachen;Skriptsprachen;Abfragesprachen (z.B. SQL);Low-Code (z.B. Power Apps);		
7	7.25.23 8:07:51	Administration/Controlling	Nein			Microsoft Power Apps;	1
8	7.25.23 8:07:55	Development	Ja	Ja	Abfragesprachen (z.B. SQL);objektorientierte Programmiersprachen;		
9	7.25.23 8:08:06	Development	Ja	Nein			1
10	7.25.23 8:08:09	Consulting	Nein			Microsoft Power Apps;	1
11	7.25.23 8:08:11	Development	Ja	Ja	objektorientierte Programmiersprachen;		

12	7.25.23 8:08:25	Consulting	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	3
13	7.25.23 8:08:30	Development	Ja	Ja	objektorientierte Programmiersprachen;		
14	7.25.23 8:08:38	Service (Consulting+ Development)	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;		
15	7.25.23 8:09:00	Development	Ja	Ja	objektorientierte Programmiersprachen;		
16	7.25.23 8:09:29	Consulting	Nein			Turbo Pascal;	1
17	7.25.23 8:09:50	Consulting	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	2
18	7.25.23 8:09:53	Management	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	4
19	7.25.23 8:10:01	HR/Recruiting	Nein				1
20	7.25.23 8:10:14	Consulting	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	2
21	7.25.23 8:10:32	Development	Ja	Ja	objektorientierte Programmiersprachen;Low-Code (z.B. Power Apps);Abfragesprachen (z.B. SQL);		
22	7.25.23 8:11:17	Consulting	Ja	Nein			2
23	7.25.23 8:14:46	Development	Ja	Ja	objektorientierte Programmiersprachen;Abfragesprachen (z.B. SQL);		

24	7.25.23 8:14:52	Consulting	Nein			Microsoft Power Apps;Microsoft Power Automate;	5
25	7.25.23 8:15:11	Development	Ja	Ja	Skriptsprachen;objektorientierte Programmiersprachen;Low-Code (z.B. Power Apps);		
26	7.25.23 8:16:29	Development	Ja	Ja	objektorientierte Programmiersprachen;Abfragesprachen (z.B. SQL);		
27	7.25.23 8:16:48	Consulting	Ja	Nein			4
28	7.25.23 8:17:31	Development	Ja	Ja	Skriptsprachen;objektorientierte Programmiersprachen;Abfragesprachen (z.B. SQL);		
29	7.25.23 8:18:27	Sales/Marketing	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	5
30	7.25.23 8:18:56	Administration/Controlling	Nein			Microsoft Power Apps;	2
31	7.25.23 8:21:26	Consulting	Nein			Microsoft Power Apps;Microsoft Power Automate;	2
32	7.25.23 8:21:30	Development	Ja	Ja	Abfragesprachen (z.B. SQL);C/AL (ist nicht objektorientiert);		
33	7.25.23 8:21:33	Support Center (für Kunden NAV/BC-Systeme)	Ja	Ja	Abfragesprachen (z.B. SQL);C/AL (demnächst AL);		
34	7.25.23 8:22:34	Consulting	Ja	Nein			1

35	7.25.23 8:22:48	Architect	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;objektorientierte Programmiersprachen;Low-Code (z.B. Power Apps);		
36	7.25.23 8:24:31	Development	Ja	Ja	Abfragesprachen (z.B. SQL);objektorientierte Programmiersprachen;Skriptsprachen;		
37	7.25.23 8:27:54	Management	Ja	Nein			5
38	7.25.23 8:30:00	Solution Engineer	Ja	Ja	Abfragesprachen (z.B. SQL);		
39	7.25.23 8:32:34	Development	Ja	Nein			1
40	7.25.23 8:33:51	Consulting & Development	Ja	Nein			2
41	7.25.23 8:34:58	Consulting/Development	Ja	Ja	Abfragesprachen (z.B. SQL);objektorientierte Programmiersprachen;		
42	7.25.23 8:35:16	Consulting	Ja	Nein		Microsoft Power Automate;	2
43	7.25.23 8:37:07	Consulting	Ja	Nein		Microsoft Power Apps;	1
44	7.25.23 8:39:28	Development	Ja	Ja	C\AL, AL;		
45	7.25.23 8:39:57	Finance/Accounting	Nein			Nein, bisher nicht.;	2
46	7.25.23 8:40:14	Consulting	Nein			Microsoft Power Apps;	2
47	7.25.23 8:43:26	Consulting	Ja	Ja	objektorientierte Programmiersprachen;		

48	7.25.23 8:44:11	Consulting	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;Low-Code (z.B. Power Apps);		
49	7.25.23 8:44:52	Sales/Marketing	Nein			Microsoft Power Apps;	3
50	7.25.23 8:45:50	Development	Ja	Ja	Abfragesprachen (z.B. SQL);C/AL;Skriptsprachen;objektorientierte Programmiersprachen;		
51	7.25.23 8:46:25	Sales/Marketing	Nein			Microsoft Power Apps;	1
52	7.25.23 8:47:38	Development	Ja	Ja	objektorientierte Programmiersprachen;Low-Code (z.B. Power Apps);Skriptsprachen;Abfragesprachen (z.B. SQL);		
53	7.25.23 8:49:16	Consulting	Nein			Microsoft Power Apps;Microsoft Power Automate;	3
54	7.25.23 8:49:20	IT	Ja	Nein		Microsoft Power Automate;	2
55	7.25.23 8:49:24	Consulting	Ja	Nein		Nein;	2
56	7.25.23 8:49:25	Sales/Marketing	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	3
57	7.25.23 8:54:11	Consulting	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	5
58	7.25.23 8:56:11	Consulting	Ja	Nein			3
59	7.25.23 8:56:43	Development	Ja	Ja	Abfragesprachen (z.B. SQL);AL;		

60	7.25.23 9:02:42	Consulting	Ja	Nein		Microsoft Power Automate;	2
61	7.25.23 9:02:53	Development	Ja	Ja	Low-Code (z.B. Power Apps);objektorientierte Programmiersprachen;Skriptsprachen;Abfragesprachen (z.B. SQL);		
62	7.25.23 9:07:01	Management	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	1
63	7.25.23 9:07:43	Consulting	Nein				2
64	7.25.23 9:09:15	Development	Ja	Ja	objektorientierte Programmiersprachen;Skriptsprachen;		
65	7.25.23 9:10:02	Development	Ja	Ja	C/AL;		
66	7.25.23 9:15:12	Development	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;objektorientierte Programmiersprachen;Low-Code (z.B. Power Apps);		
67	7.25.23 9:19:14	Sales/Marketing	Nein			nein;	1
68	7.25.23 9:19:15	HR/Recruiting	Nein				1
69	7.25.23 9:19:49	Sales/Marketing	Nein			nein;	1
70	7.25.23 9:21:57	Development	Ja	Ja	objektorientierte Programmiersprachen;		
71	7.25.23 9:27:26	Consulting/Development	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;Low-Code (z.B. Power Apps);C/AL und AL;		

72	7.25.23 9:38:07	Development	Ja	Ja	Abfragesprachen (z.B. SQL);objektorientierte Programmiersprachen;		
73	7.25.23 9:41:47	teamlead	Ja	Nein		Microsoft Power Automate;Microsoft Power Apps;	4
74	7.25.23 9:52:46	Development	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;objektorientierte Programmiersprachen;prozedurale Programmierung wie jeder be-terna BC/NAV Entwickler auch;		
75	7.25.23 9:53:07	Consulting	Nein			keiner;	1
76	7.25.23 9:56:14	Development	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;objektorientierte Programmiersprachen;Low-Code (z.B. Power Apps);		
77	7.25.23 9:59:18	Architect	Nein			Microsoft Power Apps;Microsoft Power Automate;	2
78	7.25.23 10:03:11	Projektleitung (incl. Development)	Ja	Ja	C/AL, AL;Abfragesprachen (z.B. SQL);		
79	7.25.23 10:05:29	Development	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;objektorientierte Programmiersprachen;		
80	7.25.23 10:06:37	Consulting	Nein			Microsoft Power Apps;Microsoft Power Automate;	4
81	7.25.23 10:07:01	Consulting	Nein				2

82	7.25.23 10:24:48	BE-Analytics / BI	Ja	Ja	Abfragesprachen (z.B. SQL);		
83	7.25.23 10:29:42	Development	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	2
84	7.25.23 10:31:53	Consulting	Ja	Ja	Abfragesprachen (z.B. SQL);C/AL;		
85	7.25.23 10:39:30	Consulting	Nein			Microsoft Power Automate;	4
86	7.25.23 10:40:40	Consulting	Ja	Nein		Microsoft Power Apps;	4
87	7.25.23 10:43:23	Development	Ja	Ja	objektorientierte Programmiersprachen;Skriptsprachen;		
88	7.25.23 10:45:05	Consulting	Nein			Nein;	1
89	7.25.23 11:30:31	Consulting	Nein			Microsoft Power Apps;Microsoft Power Automate;	4
90	7.25.23 11:43:32	Development	Ja	Ja	objektorientierte Programmiersprachen;Abfragesprachen (z.B. SQL);		
91	7.25.23 12:26:58	Consulting	Nein				1
92	7.25.23 12:55:13	Consulting	Ja	Nein		Microsoft Power Apps;	1
93	7.25.23 12:55:30	Administration/Controlling	Nein				1
94	7.25.23 13:08:04	Development	Ja	Ja	AL;		

95	7.25.23 13:39:40	Consulting	Nein			Microsoft Power Automate;Microsoft Power Apps;Power BI;	1
96	7.25.23 13:50:38	Teamlead Consulting/Development	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;Zapier;	5
97	7.25.23 14:19:32	Development	Ja	Ja	Abfragesprachen (z.B. SQL);objektorientierte Programmiersprachen;Skriptsprachen;		
98	7.25.23 14:21:42	Architect	Ja	Ja	Skriptsprachen;objektorientierte Programmiersprachen;		
99	7.25.23 14:21:47	Management	Nein				1
100	7.25.23 15:26:35	Consulting	Ja	Nein			1
101	7.25.23 15:37:58	Sales/Marketing	Nein			Microsoft Power Apps;	2
102	7.25.23 16:44:02	Consulting	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	3
103	7.25.23 18:55:42	Consulting	Nein			Microsoft Power Apps;	2
104	7.26.23 8:17:58	Consulting	Ja	Nein			2
105	7.26.23 8:19:17	Development	Ja	Ja	CAL/AL;		
106	7.26.23 9:12:03	Consulting	Ja	Nein			1

107	7.26.23 9:17:00	Consulting	Ja	Nein			2
108	7.26.23 10:05:13	Development	Ja	Ja	Abfragesprachen (z.B. SQL);Skriptsprachen;objektorientierte Programmiersprachen;		
109	7.26.23 10:18:58	Consulting	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	5
110	7.26.23 10:53:31	Management	Ja	Nein		Microsoft Power Automate;Microsoft Power Apps;	1
111	7.26.23 13:26:11	Management	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	1
112	7.26.23 14:43:46	Architect	Ja	Nein		Microsoft Power Automate;	2
113	7.26.23 14:49:54	Management	Ja	Nein		Microsoft Power Apps;	1
114	7.26.23 16:50:33	Management	Ja	Nein		Microsoft Power Apps;UiPath;Microsoft Power Automate;	3
115	7.27.23 8:10:50	Consulting	Nein				2
116	7.27.23 8:15:15	Development	Ja	Ja	objektorientierte Programmiersprachen;Skriptsprachen;Abfragesprachen (z.B. SQL);Low-Code (z.B. Power Apps);		
117	7.27.23 13:51:55	Management	Ja	Nein		Microsoft Power Automate;Microsoft Power Apps;	4

118	7.28.23 7:58:12	Consulting	Nein				4
119	7.28.23 13:38:10	Consulting	Ja	Nein		Microsoft Power Apps;Microsoft Power Automate;	3
120	7.31.23 9:11:15	Consulting	Ja	Nein		Microsoft Power Automate;	2
121	7.31.23 9:49:24	Consulting	Ja	Ja	objektorientierte Programmiersprachen;Abfragesprachen (z.B. SQL);		
122	7.31.23 10:04:43	Consulting	Ja	Nein		Microsoft Power Automate;	4
123	7.31.23 10:11:27	Consulting	Ja	Nein		Microsoft Power Automate;Microsoft Power Apps;	2
124	7.31.23 10:47:26	Consulting	Ja	Nein			4
125	7.31.23 12:47:36	Architect	Ja	Nein		Microsoft Power Automate;	1

Eidesstattliche Erklärung

Ich erkläre an Eides statt,

dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht.

Die Zustimmung des/der beteiligten Unternehmen/s zur Verwendung betrieblicher Unterlagen habe ich eingeholt.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder veröffentlicht noch einer anderen Prüfungsbehörde/-stelle vorgelegt.

Lohse, Ruben Gottfried

Name, Vorname Verfassender

Altenburg, 11.08.2023

Ort, Datum Abgabetermin

Unterschrift Verfassender

BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
GLAUCHAU
UNIVERSITY OF COOPERATIVE EDUCATION

Erklärung zur Prüfung wissenschaftlicher Arbeiten

Die Bewertung wissenschaftlicher Arbeiten erfordert die Prüfung auf Plagiate. Die hierzu von der Staatlichen Studienakademie Glauchau eingesetzte Prüfungskommission nutzt sowohl eigene Software als auch diesbezügliche Leistungen von Drittanbietern. Dies erfolgt gemäß § 7 des Gesetzes zum Schutz der informationellen Selbstbestimmung im Freistaat Sachsen (Sächsisches Datenschutzgesetz – SächsDSG) vom 25. August 2003 (Rechtsbereinigt mit Stand vom 31. Juli 2011) im Sinne einer Datenverarbeitung im Auftrag.
Der Studierende bevollmächtigt die Mitglieder der Prüfungskommission hiermit zur Inanspruchnahme o.g. Dienste. In begründeten Ausnahmefällen kann der Datenschutzbeauftragte der Staatlichen Studienakademie Glauchau sowohl vom Verfasser der wissenschaftlichen Arbeit als auch von der Prüfungskommission in den Entscheidungsprozess einbezogen werden.

Name:	Lohse
Vorname:	Ruben Gottfried
Matrikelnummer:	4004054
Studiengang:	Wirtschaftsinformatik
Titel der Arbeit:	Analyse und Bewertung von verschiedenen Low-Code-Plattformen für die Zielgruppe des Citizen Developers
Datum:	14.08.2023
Unterschrift: