

Freie Universität Berlin

Masterarbeit am Institut für Informatik der Freien Universität Berlin
Arbeitsgruppe Software Engineering

mit Unterstützung der adesso SE

Vergleich und Evaluation von Low-Code-Plattformen gegenüber konventioneller Softwareentwicklung

Altan Mehmet Karacan
Matrikelnummer: 4677926
altanmk93@zedat.fu-berlin.de

Erstgutachter: Prof. Dr. Lutz Prechelt
Zweitgutachter: Prof. Dr. Jörn Eichler
Betreuer: Boris Kownatzki

02. Mai 2022

Zusammenfassung

Die Entwicklung von Softwaresystemen von Grund auf ist oft ein kostspieliger und zeitaufwändiger Prozess, für den IT-Dienstleistern oft die notwendigen Entwicklungskapazitäten fehlen. Low-Code-Plattformen wurden daher von verschiedenen Anbietern als Lösung zur Verringerung von Engpässen und zur Steigerung der Entwicklungsproduktivität bei der Softwareentwicklung entwickelt[1]. Ziel dieser Arbeit ist es, einige marktführende Low-Code-Plattformen zu vergleichen, um ihre Gemeinsamkeiten und Unterschiede festzustellen und beurteilen zu können, für welche Art von Anwendungen sie geeignet sind. Ein weiteres Ziel ist es in diesem Zusammenhang auch, die Low-Code-Entwicklung mit der konventionellen Softwareentwicklung zu vergleichen, um einem Softwareentwickler oder Entscheider die Vor- und Nachteile näher zu bringen. Um die Ziele zu erreichen, wurden reale Anwendungen definiert und mit den hier betrachteten Low-Code-Plattformen umgesetzt. Erkenntnisse über die Plattformen, sowie Entwicklungsschritte zur Umsetzung der definierten Anwendungen wurden in Textprotokollen festgehalten, die als Grundlage für die Evaluation der Plattformen dienten. Bei der Implementierung der Anwendungen wurden einige Ähnlichkeiten zwischen den Plattformen, z.B. in Bezug auf die Fähigkeit zur Integration von Diensten Dritter festgestellt. Es zeigten sich jedoch auch Unterschiede, z.B. in den Skalierungsstrategien der Plattformen. Insgesamt wurde festgestellt, dass Low-Code-Plattformen aufgrund verschiedener Einschränkungen weniger flexibel sind als die konventionelle Softwareentwicklung.

Selbstständigkeitserklärung

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Berlin, 2. Mai 2022

Altan Mehmet Karacan

Inhaltsverzeichnis

1	Einführung	1
1.1	Zielsetzung	1
1.2	Auswahl der Low-Code-Plattformen	3
1.3	Aufbau der Arbeit	4
2	Grundlagen	5
2.1	Low-Code-Entwicklung	5
2.2	Low-Code-Entwicklungsprozess	5
2.3	Verwandte Arbeiten	6
3	Methodik	8
3.1	Informationsbeschaffung	8
3.2	Anwendungsbeispiele	8
3.3	Protokollierung	9
4	Low-Code-Plattformen	10
4.1	OutSystems	10
4.1.1	Einführung	10
4.1.2	Architektur	10
4.1.3	Entwicklung	12
4.2	Microsoft Power Apps	17
4.2.1	Einführung	17
4.2.2	Entwicklung	18
4.3	ServiceNow Now Platform	23
4.3.1	Einführung	23
4.3.2	Entwicklung	23
4.4	Mendix Platform	26
4.4.1	Einführung	26
4.4.2	Entwicklung	26
5	Umsetzung und Evaluation	30
5.1	1. Phase: Kfz-Zulassung	31
5.1.1	Ziele	31
5.1.2	Umsetzung	32

5.1.3	Evaluation	45
5.2	2. Phase: Terminanfrage	49
5.2.1	Ziele	49
5.2.2	Umsetzung	50
5.2.3	Evaluation	62
5.3	3. Phase: Bewerbermanagement	66
5.3.1	Ziele	66
5.3.2	Umsetzung	66
5.3.3	Evaluation	79
6	Fazit	81
6.1	Zusammenfassung der Ergebnisse	81
6.2	Diskussion zur Einschätzung der geeignetsten Plattform	83
6.3	Diskussion zum Vergleich mit der klassischen Softwareentwicklung . .	84
6.4	Ausblick	86
A	Anhang	87
A.1	Katalog der Logikwerkzeuge für Aktionen in OutSystems	87
A.2	Katalog der Toolbox-Elemente für Microflows in Mendix	88
A.3	Repositories	89
	Literaturverzeichnis	96

Abbildungsverzeichnis

1.1	Magic Quadrant für Enterprise-Low-Code-Anwendungsplattformen[4]	3
4.1	Architektur der OutSystems Plattform[17]	11
4.2	Service Studio Anwendungsdetails[22]	13
4.3	Arbeitsbereich von Service Studio[22]	14
4.4	Beispiel einer Client-Aktion in OutSystems	15
4.5	Arbeitsbereich von Integration Studio[28]	16
4.6	Layout von Power Apps Studio	19
4.7	Layout des klassischen App-Designers	20
4.8	Layout vom Power Apps-Portalstudio[37]	21
4.9	Ausschnitt der Power Portalverwaltungs-App[38]	22
4.10	App Engine Studio[44]	24
4.11	Layout von ServiceNow Studio IDE [47]	25
4.12	Layout von Mendix Studio[52]	27
4.13	Beispiel eines Microflows in Mendix	28
4.14	Layout von Mendix Studio Pro[54]	29
5.1	Das Datenmodell der Kfz-Zulassungsanwedung in OutSystems	32
5.2	Benutzeroberflächen der Kfz-Zulassungsanwendung mit OutSystems	33
5.3	Client-Aktion zum Anlegen eines Antragstellers in Service Studio	34
5.4	Konfiguration einer REST-API Integration in Service Studio	35
5.5	Aufruf der REST-API in einer Client-Action	36
5.6	Benutzeroberflächen der Kfz-Zulassungsanwendung mit Power Apps	37
5.7	Exemplarische Überwachung der Kfz-Zulassungsanwendung in Power Apps	40
5.8	Benutzeroberflächen der Kfz-Zulassungsanwendung mit Mendix	41
5.9	Microflow zum Anlegen eines Antragstellers	41
5.10	Microflow zum Aufrufen der REST-API	43
5.11	Microflow zum Anlegen eines Systemnutzers in Mendix	51
5.12	Benutzeroberflächen der Terminanfrage-Anwendung mit Mendix	53
5.13	Screen-Aktion zum Anlegen eines Benutzers	54
5.14	UI-Flow für den Rückruf der Anwendung nach der Autorisierung durch Google	55
5.15	Preparation-Aktion zum Aufruf der Google Autorisierungsseite und zum Anlegen eines autorisierten Anwendungsbenutzers	55

5.16	Benutzeroberflächen der Terminanfrageanwendung mit OutSystems .	57
5.17	Ergebnis des BDD-Tests in OutSystems	58
5.18	Given-When-Then-Bildschirmaktionen	59
5.19	Benutzeroberflächen der Portal-Anwendung mit Power Apps	61
5.20	Beispiel eines End-to-End-Tests in Power Apps Test Studio	62
5.21	Zeitvergleich zwischen Low-Code-Plattformen und klassischer Softwareentwicklung in Bezug auf die Implementierung der Funktionalitäten der Terminanfrage-Anwendung	63
5.22	Datenmodell der Bewerber-Management-Anwendung	67
5.23	Ansicht zur Erstellung eines Diagramms in Dataverse	69
5.24	Benutzeroberflächen der Bewerbermanagement-Anwendung mit Power Apps	70
5.25	Aufruf der Erweiterung zur Berechnung von Dashboard-Kennzahlen in einer Datenaktion	72
5.26	Serveraktion zur Berechnung der Kreissektoren eines Kreisdiagramms in OutSystems	73
5.27	Benutzeroberflächen der Bewerbermanagement-Anwendung mit OutSystems	73
5.28	Benutzeroberflächen der Bewerbermanagement-Anwendung mit Mendix	76
5.29	Aufruf der Java-Aktion in einem Microflow zur Berechnung von Dashboard-Kennzahlen	76
5.30	Benutzeroberflächen der Bewerbermanagement-Anwendung mit ServiceNow	78

Tabellenverzeichnis

5.1	Zuordnung von Forschungsfragen zu Phasen in Matrixdarstellung . .	30
5.2	Reihenfolge der Plattformen für die Umsetzung der Anwendungen in den verschiedenen Phasen.	30
A.1	Katalog der Logikwerkzeuge für Aktionen in OutSystems[82]	87
A.2	Katalog der Toolbox Elemente für Microflows in Mendix[53]	88

CI / CD Continuous Integration / Continuous Delivery

DLL Dynamic Link Library

IDE Integrated Development Environment

WYSIWYG What You See Is What You Get

REST Representational State Transfer

API Application Programming Interface

SaaS Software as a Service

PaaS Platform as a Service

SSO Single Sign-on

OAuth Open Authentication

URI Uniform Resource Identifier

BDD Behavior Driven Development

1 Einführung

Der Bedarf an technologischen Lösungen wächst durch die Digitalisierung in den verschiedenen Branchen stetig. Immer mehr Unternehmen und Institutionen nutzen Softwarelösungen, um ihre Arbeitsprozesse zu beschleunigen und zu automatisieren. Die Corona-Pandemie ist ein Beispiel für die Digitalisierung im Gesundheitswesen, bei der viele Prozesse, wie z. B. die Rückverfolgung der Ausbreitung des Corona-Virus, mit Hilfe technischer Softwarelösungen umgesetzt wurden[2]. Die ständige Nachfrage nach technischen Lösungen führt dazu, dass IT-Dienstleister stark unter Druck stehen und ihre Lösungen in kurzer Zeit effizient bereitstellen müssen. Die Software muss dabei dennoch sicher und in hoher Qualität geliefert werden.

Der Einsatz von Low-Code-Plattformen kann hier helfen, die Bereitstellung der Anwendungen deutlich zu beschleunigen. Mit Hilfe von visuellen Schnittstellen lassen sich notwendige Komponenten der Anwendung, wie die Benutzeroberfläche, das Datenmodell und die Arbeitsabläufe der Anwendung ohne tiefergehende Programmierkenntnisse oder technisches Hintergrundwissen umsetzen[3]. Auf dem Markt gibt es eine Vielzahl unterschiedlicher Low-Code-Plattformlösungen, die alle die gleiche Grundintention haben, sich aber insgesamt in Funktionalität und Qualität unterscheiden[4].

1.1 Zielsetzung

Diese Arbeit wurde von der adesso SE, einem IT-Dienstleistungsunternehmen, vorgeschlagen, um herauszufinden, inwieweit Low-Code-Plattformen aus Sicht eines Dienstleisters effizienter sind als die klassische Softwareentwicklung mit den im Unternehmen eingesetzten Frameworks. Ziel dieser Arbeit ist es daher, eine Reihe von Low-Code-Plattformen zu untersuchen und zu vergleichen, um ihre Stärken und Schwächen zu identifizieren. Darauf aufbauend soll eine Einschätzung vorgenommen werden, welche Art von Anwendung mit den untersuchten Plattformen realisiert werden kann. Weiterhin ist das Ziel dieser Arbeit, die Entwicklung mit Low-Code-Plattformen gegenüber der klassischen Softwareentwicklung zu stellen, um deren Vor- und Nachteile näher zu bringen. Als Grundlage für den Vergleich der Plattformen wurden im Vorfeld folgende Forschungsfragen definiert, die in der Arbeit näher untersucht werden.

1. In welchen Prozessen ist beim initialen Entwicklungsaufwand mit Zeitersparnis / Zeitverlust im Vergleich zur klassischen Softwareentwicklung zu rechnen?
2. Wird die Integration / Bereitstellung von Services unterstützt?
3. Ist eine individuelle Anpassung durch selbstgeschriebenen Code möglich?
4. Werden Werkzeuge zur Laufzeitanalyse bereitgestellt?
5. Wie groß ist der Featureumfang für die Umsetzung komplexer Anwendungen?
6. Kann die Anwendung ohne Mehraufwand weiterentwickelt werden?
7. Ist die Testbarkeit der Anwendungen gegeben und wie aufwändig ist die Testdurchführung?
8. Kann Debugging zur Fehlerdiagnose verwendet werden?
9. Wie aufwändig ist eine Automatisierung durch CI / CD?
10. Welche Skalierungsstrategien gibt es und wie hoch ist die Ausfallsicherheit?

1.2 Auswahl der Low-Code-Plattformen

Bei der Auswahl spezifischer Low-Code-Plattformen wurde im Vorfeld eine Einschätzung berücksichtigt, die in Abbildung 1.1 dargestellt ist.



Abbildung 1.1: Magic Quadrant für Enterprise-Low-Code-Anwendungsplattformen[4]

Die Einschätzung stammt von Gartner, einem Marktforschungsunternehmen für Entwicklungen in der IT-Branche, das verschiedene Low-Code-Plattformen basierend auf ihrer Innovationsfähigkeit und Marktreaktion auf ihre Produktentwicklung in Quadranten einteilt. Jeder Quadrant positioniert die bestehenden Low-Code-Plattformen auf dem Markt als Führer, Visionäre, Nischenspieler und Herausforderer.[4] Von den sechs führenden Low-Code-Plattformen sind die drei Plattformen OutSystems, Microsoft Power Apps und ServiceNow Lizenzpartner des Unternehmens adesso, die bereits in einigen Kundenprojekten erfolgreich eingesetzt wurden. Die vierte und letzte Plattform, die für den Vergleich in Betracht gezogen wurde, ist die Mendix Plattform. Ihre Auswahl basiert hauptsächlich auf persönlicher Präferenz. In Gartners Bericht ‚Enterprise Low-Code Application Platforms (LCAP) Reviews and Ratings‘ wurden viele Bewertungen der Mendix-Plattform vorgenommen, die ihre

Auswahl für den Vergleich beeinflussten[5]. Vor allem Features wie die Einfachheit der Plattform und die damit verbundene einfache Bedienbarkeit wurden in den Bewertungen oft genannt, was wichtige Gründe für die Wahl der Plattform waren. Auch die Möglichkeit für schnelles Prototyping ohne Softwareentwicklung und schnelle Anwendungsbereitstellung trugen zur Entscheidung bei.

Für die professionelle Anwendungsentwicklung nutzt die Mendix-Plattform für ihre Erweiterungen die Programmiersprache Java, für die keine nennenswerte Einarbeitungszeit notwendig ist, da bereits Erfahrungen in der Anwendungsentwicklung mit dieser Programmiersprache gesammelt wurden.

1.3 Aufbau der Arbeit

Diese Arbeit beginnt mit Kapitel 2, in dem grundlegende Konzepte und Begriffe im Zusammenhang mit der Low-Code-Entwicklung erläutert werden. Darüber hinaus werden einige verwandte Arbeiten vorgestellt und sie mit der Position dieser Arbeit kontrastiert. Kapitel 3 stellt die methodischen Herangehensweisen vor, die zum Vergleich und zur Evaluation der vorgestellten Low-Code-Plattformen aus Kapitel 4 verwendet werden. Darauf folgend beschreibt Kapitel 5 die Umsetzung der vorgestellten Herangehensweisen in drei Phasen, stellt die Ergebnisse vor und führt nach jeder Phase eine Evaluation durch. Abschließend werden die Ergebnisse in Kapitel 6 zusammengefasst. Darauf aufbauend werden die geeigneten Plattformen für verschiedene Anwendungsfälle diskutiert, Empfehlungen gegeben und die Low-Code-Entwicklung mit der klassischen Softwareentwicklung verglichen. Außerdem wird ein Ausblick auf eine weitere Forschung gegeben.

2 Grundlagen

In diesem Kapitel wird der Begriff Low-Code-Entwicklung definiert und die damit zusammenhängenden Konzepte erläutert, um das Verständnis für diese Arbeit zu verbessern.

2.1 Low-Code-Entwicklung

Der Begriff ‚Low Code‘ wurde erstmals 2014 von dem Marktanalyseunternehmen Forrester verwendet. In diesem Zusammenhang wurden Low-Code-Entwicklungsplattformen auch als Plattformen definiert, die eine schnelle Erstellung und Bereitstellung von Geschäftsanwendungen mit minimalem manuellen Programmieraufwand ermöglichen und möglichst wenig Aufwand für die Installation und Konfiguration von Entwicklungsumgebungen erfordern.[6]

Dabei nutzen die Low-Code-Plattformen Ansätze aus der modellbasierten Softwareentwicklung, der schnellen Anwendungsentwicklung, der automatischen Codegenerierung und der visuellen Programmierung für die Anwendungsentwicklung[3]. Die modellbasierte Softwareentwicklung ist ein allgemeiner Begriff für Techniken, die automatisch einen ausführbaren Code aus formalen Modellen generieren. Bei dieser Technik verwenden die Benutzer eine grafische Darstellungsform, um einen Aspekt der Software auszudrücken. Beispielsweise wird das Datenmodell mit Hilfe der bereitgestellten Werkzeuge in ausführbare Software übersetzt.[7] Die visuelle Programmierung ist eine besondere Form der modellbasierten Softwareentwicklung, bei der auf die Verwendung von textuellem Quellcode verzichtet wird und stattdessen grafische Elemente per Drag-and-Drop zur Spezifikation der Software verwendet werden[8]. Mit der schnellen Anwendungsentwicklung zielen Plattformen darauf ab, kommerzielle Anforderungen zu erfüllen, indem sie funktionierende Geschäftsanwendungen in kürzerer Zeit und mit geringeren Investitionen bereitstellen[9].

2.2 Low-Code-Entwicklungsprozess

Die Anwendungsentwicklung mit Low-Code-Plattformen umfasst im Allgemeinen fünf Schritte. Die **Datenmodellierung** ist in der Regel der erste Schritt, bei dem Modellierungskonstrukte verwendet werden, um das Datenmodell der Anwendung zu

definieren. Hier werden Entitäten, Beziehungen zwischen Entitäten, Einschränkungen und Abhängigkeiten in der Regel per Drag-and-Drop definiert.[10]

Der zweite Schritt ist die **Definition der Benutzeroberflächen** der Anwendung. Mit Hilfe von Drag-and-Drop-Funktionen werden konfigurierbare Formulare und Ansichten für die Erstellung, Bearbeitung und Visualisierung von Daten erstellt.[6]

Der dritte Schritt ist die **Spezifikation der Geschäftslogik** der Anwendung, die durch Kontroll- und Datenflussdiagramm ähnliche Darstellungen implementiert wird. Die Spezifikation von grafischen Arbeitsabläufen und Geschäftsregeln ist ebenfalls Teil dieses Schritts.

Der nächste Schritt ist die **Integration mit externen Diensten** über spezielle APIs von Drittanbietern, wobei Low-Code-Plattformen in der Regel Konnektoren zur Nutzung von Daten als Datengrundlage oder Diensten bereitstellen.[10]

Mit der Unterstützung von Bereitstellungswerkzeugen ist der letzte Schritt die **Bereitstellung** der entwickelten Anwendung in verschiedenen Umgebungen mit nur wenigen Klicks[6].

2.3 Verwandte Arbeiten

Im Jahr 2020 erstellten Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio und Alfonso Pierantonio in ihrer Arbeit ‚Supporting the understanding and comparison of low-code development platforms‘ eine Taxonomie von charakteristischen Merkmalen für den Vergleich von Low-Code-Entwicklungsplattformen, die als Entscheidungshilfe für die Auswahl der richtigen Plattform zur Lösung eines bestimmten Problems dienen soll[10]. Die Merkmale wurden durch die Analyse von acht Plattformen ermittelt und in Kategorien wie Unterstützung der Interoperabilität mit externen Diensten und Datenquellen, Unterstützung der Skalierbarkeit und viele andere gruppiert. Zusätzlich zu den zuvor vorgestellten Taxonomiemerkmalen wurden während der Analyse weitere Aspekte ermittelt, wie z. B. die Art der Anwendung oder der Kosten- und Zeitaufwand für das Erlernen der Plattform. Außerdem wurde eine einfache Benchmark-Anwendung entwickelt, um einen besseren Einblick in die Plattformen zu erhalten. In dieser Arbeit wird keine Taxonomie erstellt, sondern konkrete Forschungsfragen untersucht, die sich jedoch teilweise den Kategorien der Taxonomie zuordnen lassen. Darüber hinaus verfolgt diese Arbeit einen anderen methodischen Ansatz zur Gewinnung von Erkenntnissen über die ausgewählten Plattformen, der

in Kapitel 3 näher erläutert wird. Unter den untersuchten Plattformen befanden sich auch drei Plattformen aus der vorliegenden Arbeit. Ein Mehrwert dieser Arbeit besteht darin, dass die Low-Code-Entwicklung zusätzlich mit der klassischen Softwareentwicklung verglichen wird.

In der 2021 erschienenen Arbeit ‚Characteristics and Challenges of Low-Code Development: The Practitioners’ Perspective‘ von Yajing Luo, Peng Liang, Chong Wang, Mojtaba Shahin und Jing Zhan wurden die Merkmale und Herausforderungen von Low-Code-Plattformen durch die Erhebung von Daten aus einer großen Anzahl von Beiträgen durch Stichwortsuche auf Stack Overflow und Reddit-Subreddits ermittelt[11]. Als Kriterien für die Datenerhebung wurden Forschungsfragen definiert, z. B. welche Art von Anwendungen mit Low-Code entwickelt werden oder welche Programmiersprachen in Low-Code-Plattformen verwendet werden, um die relevanten Beiträge zu filtern. Zur Beantwortung der Forschungsfragen wurden die erhobenen Daten mit Hilfe der deskriptiven Statistik und der Methode der konstanten Vergleichs analysiert. Einige dieser Forschungsfragen werden auch in dieser Arbeit beantwortet. Der Unterschied zur vorliegenden Arbeit besteht darin, dass die Forschungsfragen durch die praktische Umsetzung von realen Anwendungen und der komparativen Analyse der protokollierten Daten beantwortet werden.

In der Arbeit ‚Developer Experience of a Low-Code Platform: An exploratory study‘ von Daniel Dahlberg, veröffentlicht im Jahr 2020, wurden Interviews mit erfahrenen Low-Code-Entwicklern bei einem auf Low-Code-Lösungen spezialisierten IT-Dienstleister geführt, um ihre Erfahrungen mit der in ihren Lösungen verwendeten Low-Code-Plattform zu verstehen[12]. Für die Durchführung der Interviews wurden Forschungsfragen definiert, die sich z.B. darauf beziehen, wie Entwickler ihre Arbeit auf Low-Code-Plattformen wahrnehmen, aber auch auf Unterschiede in der Erfahrung zwischen klassischer und Low-Code-Entwicklung. Von diesen Erfahrungsunterschieden werden ähnliche Aspekte wie Dokumentation oder Programmiersprache auch in dieser Arbeit angesprochen. Der Unterschied zur vorliegenden Arbeit ist, dass hier auch einige technische Unterschiede und Gemeinsamkeiten zwischen den beiden Entwicklungsansätzen nähergebracht werden. Auch wenn einige Merkmale der Plattformen, die sich aus positiven und negativen Erfahrungen mit den Plattformen ergeben, erwähnt werden, werden insgesamt keine tieferen Erkenntnisse über Low-Code-Plattformen im Vergleich zur vorliegenden Arbeit genannt.

3 Methodik

In diesem Kapitel wird die methodische Herangehensweise beschrieben, die zur Beantwortung der Forschungsfragen verwendet wurde, damit ein Vergleich der Low-Code-Plattformen gezogen werden kann. Außerdem wird ein weiterer Ansatz erläutert, der zur Analyse qualitativer Daten verwendet wird.

3.1 Informationsbeschaffung

Um zunächst einen Überblick über die ausgewählten Plattformen sowie deren Entwicklungsumgebungen und zusätzliche Entwicklungswerkzeuge zu erhalten, wird zunächst eine Literatur- und Internetrecherche durchgeführt. Die Internetrecherche berücksichtigt vor allem die offiziellen Dokumentationsseiten, aber auch hilfreiche Informationen von den Entwicklern der Plattformen in den zugehörigen Datenaustauschforen.

3.2 Anwendungsbeispiele

Zur Beantwortung der Forschungsfragen werden phasenweise reale Anwendungsbeispiele definiert und mit den Low-Code-Plattformen umgesetzt. Eine Phase ist ein sich wiederholender Prozess, in dem ein neu definiertes Anwendungsbeispiel sequenziell mit den Plattformen in einer bestimmten Reihenfolge implementiert wird, um neue Erkenntnisse über die Plattformen zu gewinnen und sie schließlich bewerten zu können. Die Forschungsfragen werden insgesamt über drei Phasen hinweg untersucht. In der ersten Phase wird die Reihenfolge der Plattformen nach dem Zufallsprinzip festgelegt, die in den nachfolgenden Phasen je nach Entwicklungserfahrung und Abweichung von der vorherigen Reihenfolge neu festgelegt wird. Ziel ist es, in den verschiedenen Phasen Anwendungsbeispiele zu definieren, mit denen die beschriebenen Probleme näher untersucht werden können. Beginnend mit einem einfachen Anwendungsbeispiel nimmt die Komplexität der Anwendungsbeispiele entsprechend den Erfahrungen und Erkenntnissen aus den vorangegangenen Phasen zu. Es sollen auch Anwendungsbeispiele definiert werden, die nicht primär mit allen Plattformen umsetzbar sind. Das bedeutet, dass in einigen Phasen einige Plattformen nicht berücksichtigt werden können, weil sie die Anforderungen des Anwendungsbeispiels

nicht erfüllen.

3.3 Protokollierung

Um qualitative Daten über die Plattformen zu erheben, werden in den jeweiligen Phasen ähnlich strukturierte Textprotokolle erstellt, in denen Beobachtungen und Erkenntnisse über die Plattformen während der Umsetzung der Anwendungsbeispiele festgehalten werden. Für jedes Anwendungsbeispiel werden so viele Textprotokolle erstellt, wie es Plattformen der jeweiligen Phase gibt, die für die Umsetzung des Anwendungsbeispiels festgelegt wurden. Die Textprotokolle halten die gesamten Entwicklungsschritte eines Anwendungsbeispiels sowie die entwickelten Lösungen fest, die sowohl aus eigenständigen Lösungsideen als auch aus Verweisen auf hilfreiche Quellen stammen. Darüber hinaus werden auch Lösungsansätze mit Begründungen dokumentiert, die aber nicht zum Ziel der Umsetzung eines Problems führen. Die Protokollierung erfolgt im gleichen Zyklus und während der Entwicklung eines Teilproblems. Sie wird am Ende der Entwicklung vermieden, da die Gefahr besteht, dass persönliche Erfahrungen und Kenntnisse über die Plattformen verloren gehen. Die Analyse der dokumentierten Daten aus den Textprotokollen erfolgt nach jeder Phase mit der komparativen Analyse, um Gemeinsamkeiten und Unterschiede zwischen den Plattformen zu ermitteln. Dabei werden einzelnen Protokolltexten aus verschiedenen Protokollen einer Phase Kategorien zugeordnet, um ein bestimmtes Merkmal der Plattformen auszudrücken. Auf der Grundlage dieser Analyse wird eine Auswertung durchgeführt, die den Vergleich der Plattformen in verschiedenen Kategorien verdeutlicht.

4 Low-Code-Plattformen

In diesem Kapitel werden die für die Evaluation verwendeten Low-Code-Plattformen vorgestellt und ein Überblick über ihre verschiedenen Entwicklungsumgebungen gegeben.

4.1 OutSystems

4.1.1 Einführung

OutSystems ist eine Low-Code-Plattform für Unternehmen zur Entwicklung mobiler und webbasierter Anwendungen mithilfe von visuellen Werkzeugen. Sie verfolgt die Vision, Anwendungen in schnelleren Zyklen bereitzustellen und anzupassen.[13] Das Unternehmen OutSystems wurde 2001 in Lissabon gegründet und hat heute seinen Hauptsitz in Boston[14]. OutSystems bietet ein Preismodell mit zwei kostenpflichtigen Versionen und einer kostenfreien Version an. Letztere verfügt über eine von OutSystems selbst gehostete Cloud-Umgebung[15]. Im Rahmen dieser Arbeit wurde ausschließlich die Testversion untersucht.

Anwendungsentwicklung mit OutSystems basiert in erster Linie auf visueller Full-Stack-Entwicklung. Auch hier kommt die Technik der modellgetriebenen Softwareentwicklung zum Einsatz, bei der formale Modelle in Software übersetzt werden. Nach diesen Prinzipien lassen sich Benutzeroberflächen, Geschäftslogiken und Datenmodelle per Drag-and-Drop erstellen. Eine weitere Besonderheit ist, dass eine Anwendung mit nur einem Klick bereitgestellt werden kann.[16] Mehr zu dieser Funktion ist im Unterabschnitt 4.1.3 beschrieben.

4.1.2 Architektur

Die OutSystems-Plattform besteht aus den Komponenten ‚Platform Server‘, ‚Integration Studio‘, ‚Service Studio‘, ‚LifeTime‘ und ‚Service Center‘, deren Zusammenspiel in Abbildung 4.1 dargestellt ist.

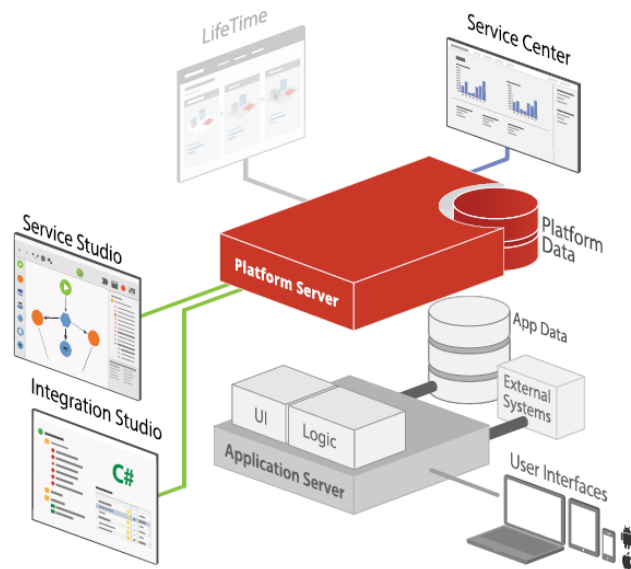


Abbildung 4.1: Architektur der OutSystems Plattform[17]

Der Plattformserver ist die Kernkomponente von OutSystems, die verschiedene Dienste zum Erstellen, Bereitstellen, Verwalten, Ausführen und Überwachen von Anwendungen verwendet. Alle Versionen einer Anwendung werden in der Plattformdatenbank gespeichert. OutSystems verwendet eine Vielzahl von Werkzeugen, um Anwendungen zu entwickeln und zu warten. Das Hauptentwicklungstool ist Service Studio, eine visuelle Entwicklungsumgebung, in der die Kernelemente einer Anwendung erstellt werden. In dieser Umgebung ist es möglich das Datenmodell zu entwerfen, Benutzeroberflächen zu erstellen, externe Dienste zu integrieren und Sicherheitsrichtlinien für Web- und mobile Anwendungen festzulegen.[17][18]

Eine weitere integrierte Entwicklungsumgebung ist Integration Studio. Sie ermöglicht Entwicklern, Komponenten mit benutzerdefiniertem C# Code in Microsoft Visual Studio zu entwickeln, die in jede OutSystems-Anwendung integriert werden können. Wenn eine Komponente veröffentlicht wird, kompiliert die Entwicklungsumgebung ihren Code mit dem .NET-Compiler und sendet dann die generierten DLLs an den Plattformserver. Es ist somit möglich, eine erfolgreich veröffentlichte Komponente als Ressource in Service Studio zu verwenden.[19] Sowohl Service Studio als auch Integration Studio sind nur als Desktopanwendungen verfügbar.

Auf dem Plattformserver läuft ein Code-Generator-Dienst, der alle Anwendungs-

komponenten aus dem in Service Studio entwickelten Anwendungsmodell generiert und einem Bereitstellungsdienst zur Verfügung stellt. Dieser stellt dann die Anwendungskomponenten auf dem Anwendungsserver bereit. Der Plattformservers stellt somit sicher, dass der generierte Code einer Anwendung zur Ausführung auf einem Standard-Webanwendungsserver bereitgestellt wird. Der Anwendungsserver verwendet herkömmliche SQL-, Oracle- und MySQL-Datenbanken sowie externe Systeme, um die Anwendungen auszuführen.[20][21]

Neben den Entwicklungswerkzeugen gibt es auch Verwaltungswerkzeuge für die Bereitstellung und den Lebenszyklus der Anwendungen. Eines dieser Werkzeuge ist das Service Center, eine Verwaltungskonsole des Plattformservers. Sie bietet Umgebungsüberwachung, z. B. die Untersuchung von Protokollnachrichten, die von den Anwendungen generiert werden, und die Konfiguration von Umgebungseinstellungen. LifeTime ist eine weitere Konsole, die entwickelt wurde, um die verschiedenen Umgebungen wie Entwicklung, Qualitätssicherung und Produktion zentral zu verwalten und den gesamten DevOps-Prozess so zu automatisieren, dass Anwendungen von der Entwicklung bis zur Produktion bereitgestellt werden können.[18] Allerdings ist in der Testversion nur ein eingeschränkter Funktionsumfang möglich, sodass nur die Entwicklungsumgebung verwaltet und konfiguriert werden kann.

4.1.3 Entwicklung

Service Studio ist die primäre integrierte Entwicklungsumgebung für die Low-Code-Entwicklung der OutSystem-Plattform. Das Front- und Backend einer Anwendung werden also nur in einer Umgebung entwickelt. Nach erfolgreicher Verbindung mit der persönlichen Umgebung der OutSystems-Cloud werden alle Anwendungen aufgelistet, die sich auf dem Plattformservers befinden. Zu Beginn bietet die Plattform drei Optionen als Startpunkt für die Entwicklung an:

1. Erstellen einer neuen Anwendung
2. Öffnen einer vorhandenen Anwendung
3. Installieren einer Anwendung von ‚Forge‘[22]

Forge ist eine Sammlung von öffentlichen Modulen, Konnektoren, UI-Komponenten und anderen Lösungen, um die erforderlichen Ressourcen in seiner Umgebung nach

Bedarf zu installieren und zu verwenden.[18] Diese Ressourcen können über die Registerkarte Forge durchsucht werden. Bei der ersten Option kann man entweder ganz von Grund auf neu oder mit einer vorgefertigten Anwendung starten. Entscheidet man sich dafür, die Anwendung von Grund auf neu zu entwickeln, kann man zwischen den Typen reaktive bzw. herkömmliche Webanwendung, Tablet-Anwendung, mobile Anwendung oder Service wählen. An dieser Stelle ist es in diesem Zusammenhang notwendig, die Unterschiede zwischen reaktiven und herkömmlichen Webanwendungen zu verstehen.

Reaktive Webanwendungen verwenden im Gegensatz zu herkömmlichen Webanwendungen das clientseitige Entwicklungsparadigma. Bei herkömmlichen Webanwendungen liegt der Fokus bei der serverseitigen Entwicklung.[23] Nach Auswahl und Erstellung des Anwendungstyps wird der in Abbildung 4.2 gezeigte Bildschirm mit den Anwendungsdetails angezeigt.

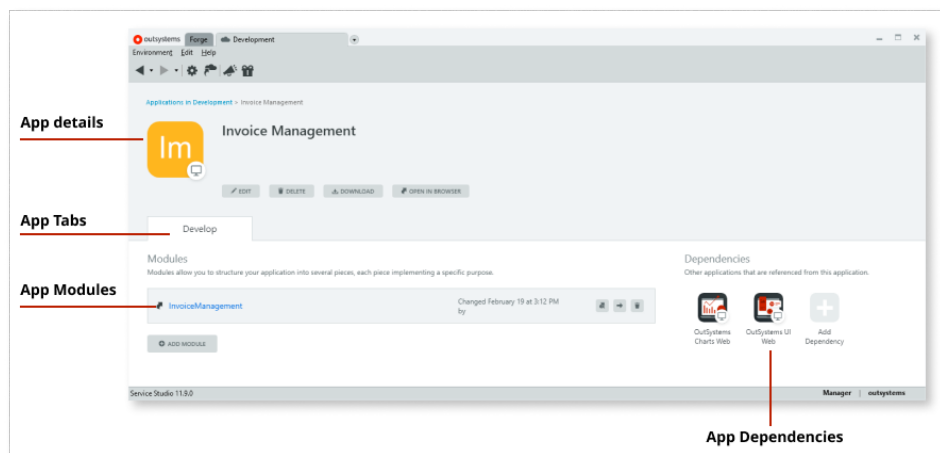


Abbildung 4.2: Service Studio Anwendungsdetails[22]

Hier werden die grundlegenden Informationen über die Anwendung sowie einige Schaltflächen zum Bearbeiten, Löschen, Herunterladen und Öffnen der Anwendung dargestellt. Unterhalb der Details wird die Registerkarte ‚Entwickeln‘ angezeigt, die die Liste der mit der Anwendung verknüpften Module anzeigt. OutSystems verwendet Module, um die Funktionalität einer Anwendung zu trennen, damit die Komplexität der Anwendung besser verwaltet und Abstraktionen gebildet werden können. Dies ermöglicht eine produktive Teamarbeit, bei der unterschiedliche Funktionalitäten einer Anwendung separat entwickelt und entsprechend problemlos zusammengeführt werden können.[24] Bei der Entwicklung einer mobilen Anwendung erscheint

eine weitere Registerkarte ‚Native Plattformen‘, um die IOS- und Android-Pakete zu generieren, die direkt auf mobilen Geräten installiert werden können[25]. Schließlich werden die in dieser Anwendung verwendeten Abhängigkeiten in diesem Bildschirm angezeigt. Nach dem Öffnen eines Moduls erscheint der Arbeitsbereich von Service Studio für die Entwicklung der Anwendung, dessen Aufbau in Abbildung 4.3 dargestellt ist.

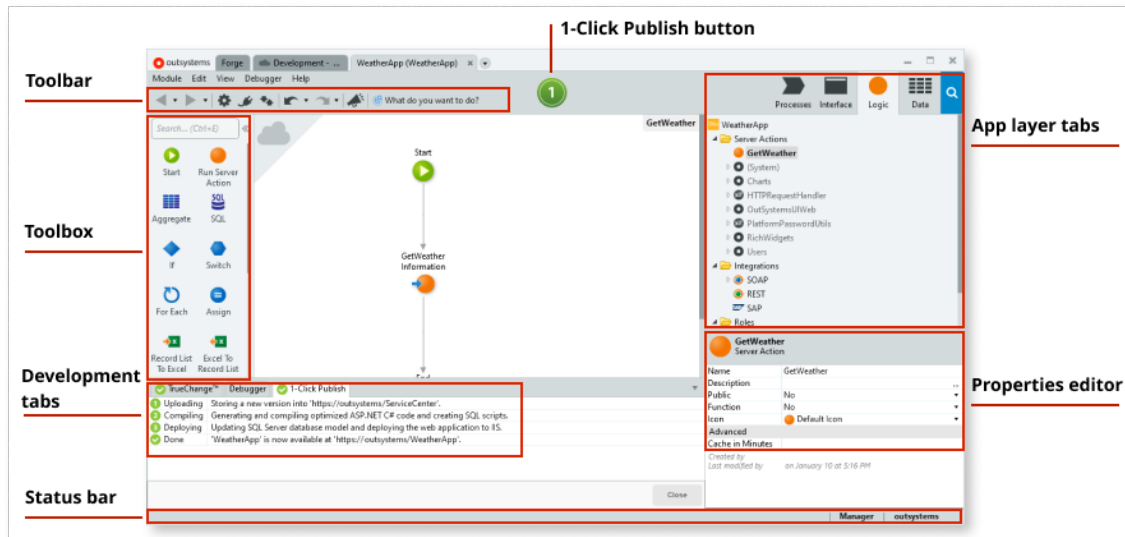


Abbildung 4.3: Arbeitsbereich von Service Studio[22]

Der Arbeitsbereich umfasst das Entwerfen, Debuggen und Bereitstellen der Module einer Anwendung. Er setzt sich aus den folgenden Hauptkomponenten zusammen: Toolbar, 1-Klick-Bereitstellungsschaltfläche, Toolbox, Haupteditor, Entwicklungsregisterkarten, Anwendungsschicht Registerkarten, Eigenschaften Editor, Statusleiste. Für jedes Modul einer Anwendung wird eine separate Registerkarte im oberen Registerkartenbereich erstellt. Es ist möglich, an mehreren Modulen verschiedener Anwendungen gleichzeitig zu arbeiten. Unterhalb der Registerkarten befindet sich eine Menüleiste, die aus einigen Menüs, einer Werkzeugleiste, einer 1-Klick-Bereitstellungsschaltfläche, vier Registerkarten und einer Suche besteht. Die Werkzeuge aus der Werkzeugleiste können verwendet werden, um das Service Center zu erreichen, die Abhängigkeiten zu verwalten und die neueste Version des Moduls vom Server abzurufen und mit der lokalen Version über ein integriertes Versionskontrollsystem zusammenzuführen. Die 1-Klick-Bereitstellungsschaltfläche stellt den lokalen Stand des Moduls in der aktuellen Umgebung bereit und weist den Benutzer darauf hin,

wenn Fehler existieren. Die vier Registerkarten der Anwendungsschicht sind unterteilt in die Schichten Prozesse, Benutzerschnittstelle, Logik und Daten. Die Datenschicht enthält ein Entitätsdiagramm, Entitäten aus verschiedenen Datenbanken, Datenstrukturen in Form von benutzerdefinierten Datentypen und Site Properties als globale Variablen für konstante Konfigurationswerte[26]. Die Implementierung von Server- und Client-Aktionen sowie von Integrationen mit externen Systemen und Diensten, die Definition von Benutzerrollen und die Erstellung von Ausnahmebehandlungen werden in der Logikschicht realisiert.[22] Die folgende Abbildung 4.4 soll zur Veranschaulichung eine Client-Aktion demonstrieren, die bei einem gültigen Formular eine CRUD-Operation aufruft und anschließend den aktuellen Bildschirm auf ein bestimmtes Ziel weiterleitet.

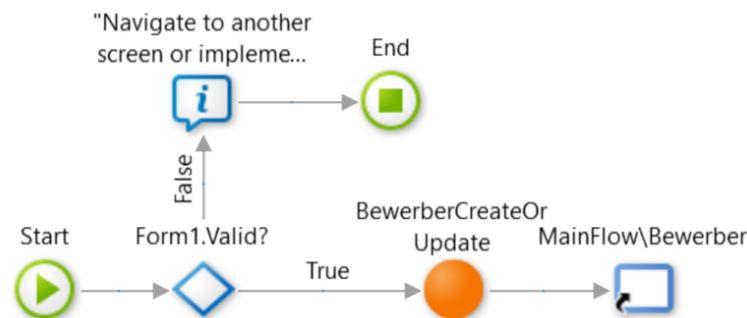


Abbildung 4.4: Beispiel einer Client-Aktion in OutSystems

Aktionen in OutSystems werden in einer Datenflussdiagramm ähnlichen Darstellung implementiert, deren visuelle Symbole im Katalog A.1 ausführlich erläutert werden. In der Benutzerschnittstellenschicht werden die Benutzeroberflächen in sogenannte UI-Flows unterteilt, die die verschiedenen Bildschirme und Blöcke gruppieren. Sie enthalten ebenso die in den Bildschirmen und Blöcken verwendete Logik. Bilder, Themes und JavaScript-Skripte sind ebenfalls hier verfügbar.[27] In der letzten Registerkarte findet man Prozesse zum Integrieren von Geschäftsprozessen mithilfe von verschiedenen Prozessflüssen. Ansonsten können Timer in diesem Bereich erstellt werden, um asynchrone Logik zu einem gewünschten Zeitpunkt auszuführen.

Die Hauptentwicklung der Module findet im Bearbeitungsbereich statt, der aus dem Haupteditor, der Toolbox, dem Bereich der Elemente aus den Registerkarten der Anwendungsschichten und dem Editor für die Eigenschaften eines ausgewählten Elements besteht. Die Toolbox enthält viele Tools und Widgets, die zum Entwickeln der Bildschirme und Aktionen verwendet werden können. Die Objekte in der Tool-

box werden entsprechend dem ausgewählten Element angepasst. Der Haupteditor befindet sich mittig im Layout der Entwicklungsumgebung, in dem die eigentliche Umsetzung der Anwendungslogik und der Benutzeroberflächen stattfindet. Beim Bearbeiten eines Bildschirms bietet der Haupteditor an, einen Widget-Baum anzuzeigen, der die genaue Struktur und Anordnung der Benutzerschnittstellenelemente zeigt.

Am unteren Bereich des Layouts der Entwicklungsumgebung befinden sich die drei Entwicklungsregisterkarten. Auf der Registerkarte ‚TrueChange‘ werden die Fehler und Warnungen angezeigt. Durch Doppelklick auf den Fehler oder die Warnung gelangt man an die entsprechende Stelle der Fehlerursache. In der Registerkarte ‚Debugger‘ kann man in einer Debugging-Session die einzelnen Inhalte der Variablen genauer untersuchen. Der Deployment-Prozess mit seinem Fortschritt im Detail wird auf der Registerkarte ‚1-Klick-Bereitstellung‘ dargestellt. Das unterste Element des Layouts ist die Statusleiste, die Informationen über den Benutzer, die aktuelle Umgebung und das Datum der letzten Veröffentlichung der Anwendung anzeigt.[22]

OutSystems ermöglicht es Softwareentwicklern, die Funktionalität und das Datenmodell einer Anwendung mit benutzerdefiniertem Code zu erweitern. Das Erstellen und Definieren der Struktur einer Erweiterung wird im Integration Studio implementiert, dessen Arbeitsbereich in Abbildung 4.5 veranschaulicht wird.

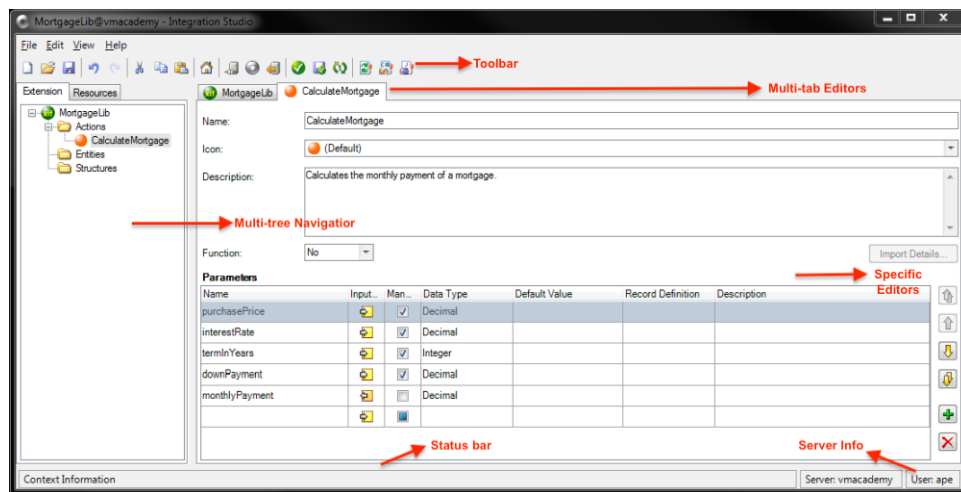


Abbildung 4.5: Arbeitsbereich von Integration Studio[28]

Im Erweiterungs-Tab des Multi-Tree-Navigators werden Aktionen, Entitäten und Strukturen angelegt, die im ‚Multi-Tab-Editor‘ angezeigt und im ‚Specific-Editor‘ angepasst werden können. In der Symbolleiste werden die Erweiterungen auf Fehler

überprüft, veröffentlicht und der zugehörige Code über Visual Studio Code aufgerufen. In der Statusleiste werden Informationen, wie der verbundene Plattformservers und Benutzer sowie der Speicherort der Erweiterung angezeigt.[28]

4.2 Microsoft Power Apps

4.2.1 Einführung

Power Apps ist eine von Microsoft entwickelte Low-Code-Plattform, die Teil der Microsoft Power Platform ist. Microsoft Power Platform ist ein Zusammenspiel verschiedener Softwaredienste, welches die Entwicklung von Anwendungen mithilfe von Workflows, Datenvisualisierung und anderen Tools ermöglicht. Sie besteht aus den vier Diensten Power BI, Power Apps, Power Automate und Power Virtual Agents.

Mit Power BI ist es möglich, Analysen auf Basis von Geschäftsdaten durchzuführen, die in verschiedenen Darstellungsformen wie Reports und Dashboards visualisiert werden können. Durch diese Darstellung der Daten können bessere Erkenntnisse gewonnen und die Entscheidungsfindung unterstützt werden. Power Apps ist die webbasierte Low-Code-Anwendungsentwicklungsumgebung zum Erstellen von Web- und Mobilanwendungen für die gezielte Problemlösung in einem Unternehmen. Die Power Plattform verfügt über einen eigenen Datendienst, das sogenannte Microsoft Dataverse, zur Speicherung und Verwaltung von Daten. Außerdem dient Microsoft Dataverse als gemeinsame Sprache zwischen den verschiedenen Komponenten der Microsoft Power Platform und stellt deren Zusammenarbeit sicher. Konnektoren ermöglichen die Interaktion mit anderen Datenquellen von Drittanbietern. Sie dienen als Brücke zwischen den verschiedenen Datenquellen und den Anwendungen. Die dritte Komponente, Power Automate, dient dazu, sich wiederholende Aufgaben und digitale Prozesse in Unternehmen durch Workflows zu automatisieren. Workflows können mithilfe von Drag-and-Drop-Tools und verschiedenen Konnektoren erstellt werden. Eine weitere Funktion, die den Diensten zur Verfügung gestellt wird, ist der AI Builder, um den Anwendungen und Workflows Funktionen mit künstlicher Intelligenz hinzuzufügen. Mit Power Virtual Agents können Chatbots mithilfe einer grafischen Benutzeroberfläche erstellt werden, ohne dass Code entwickelt werden muss. Auch die Interaktion zwischen den einzelnen Diensten ist gewährleistet, sodass die verschiedenen Funktionalitäten der Dienste in einer Anwendung genutzt werden

können.[29]

Microsoft bietet seinen Benutzern verschiedene Lizenzen mit flexiblen Abonnementplänen an.[30] Im Rahmen dieser Arbeit wurde ein Zugang zur kostenfreien Lizenz bereitgestellt, innerhalb derer alle gängigen Dienste der Microsoft Power Plattform verfügbar sind. Allerdings wurde hauptsächlich der Dienst Power Apps näher betrachtet, da der Fokus dieser Arbeit primär auf der Anwendungsentwicklung liegt.

4.2.2 Entwicklung

Bei der Erstellung einer Anwendung in Power Apps wird grundsätzlich zwischen drei verschiedenen Arten von Anwendungen unterschieden: Canvas-, modellgetriebene und Portal-Anwendungen. Alle drei Arten verfolgen einen unterschiedlichen Ansatz und eignen sich für verschiedene Anwendungsfälle.

Eine Canvas-Anwendung verfolgt den Ansatz, dass die Benutzeroberfläche per Drag-and-Drop-Prinzip frei gestaltet werden kann. Die Anwendung kann entweder auf Basis eines leeren Hintergrunds, einer Vorlage oder Daten aus verschiedenen Datenquellen wie Dataverse, Excel etc. als Entwicklungsausgangspunkt erstellt werden. Sie ist für unterschiedliche Endgeräte wahlweise im Hoch- oder Querformat verfügbar. Logische Operationen an den Daten werden unter Verwendung von Ausdrücken implementiert, die den Funktionen in Microsoft Excel ähneln. Ihr Einsatzgebiet liegt innerhalb einer Organisation, in der nur zugelassene Benutzer auf die Anwendung zugreifen können.[31] Abbildung 4.6 veranschaulicht den Aufbau der Canvas-Anwendungsentwicklungsumgebung, bekannt als Power Apps Studio.

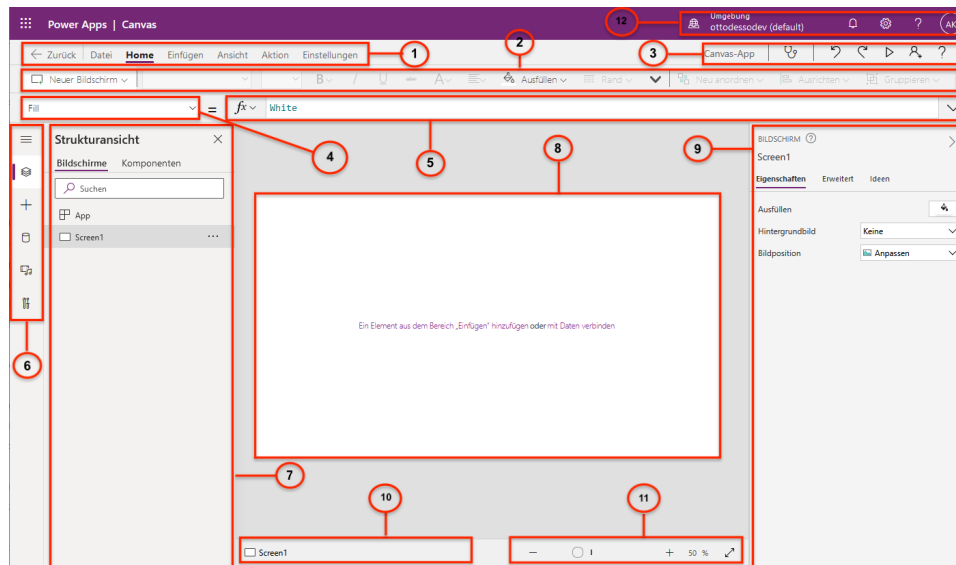


Abbildung 4.6: Layout von Power Apps Studio

Power Apps Studio setzt sich im Wesentlichen aus zwölf Komponenten zusammen. Die erste Komponente in der Abbildung zeigt die Multifunktionsleiste mit mehreren Navigationsregisterkarten zum Hinzufügen von steuer- und benutzerdefinierten Designelementen. Die Elemente der zweiten Komponente passen sich abhängig von der ausgewählten Registerkarte an. In der dritten Komponente befinden sich die Aktionen zum Wiederherstellen von Zuständen, Vorschau, Teilen mit Benutzern und Fehlerprüfung der Anwendung. Eine Liste mit Eigenschaften für ein ausgewähltes Objekt wird in der vierten Komponente angezeigt. Die fünfte Komponente ist für die Erstellung von Formeln mit Funktionen zur Bearbeitung von Daten zuständig. In der sechsten Komponente gibt es einen Auswahlbereich zum Umschalten zwischen Datenquellen, Bildschirmen, Einfügeoptionen, Medien und erweiterte Tools zum Überwachen und Testen der Anwendung. Die siebte Komponente zeigt den Detailbereich mit Optionen, die für das ausgewählte Menüelement zum Erstellen der Anwendung relevant sind. Der aktuelle Bildschirm wird in der achten Komponente angezeigt. Die neunte Komponente ist der sogenannte Eigenschaftsbereich, in dem die Liste der Eigenschaften für das ausgewählte Objekt im Benutzeroberflächenformat dargestellt wird. In der zehnten Komponente kann ein Bildschirm zum Anzeigen ausgewählt werden, dessen Größe in der elften Komponente angepasst werden kann. In der zwölften Komponente wird die Entwicklungsumgebung angezeigt und eingestellt sowie das Benutzerkonto verwaltet.[32]

Bei modellgetriebenen Anwendungen wird vor der eigentlichen Entwicklung Vorarbeit in die Modellierung, der für Anwendung notwendigen Daten investiert. Man verfolgt den Ansatz erst die Daten richtig aufzubereiten, bevor die eigentliche Entwicklung der Anwendung beginnt. Dataverse dient hier als wichtige Grundlage für die Erstellung einer modellgetriebenen Anwendung. Es dient dazu, das Datenmodell zu implementieren, die Formulare und Ansichten für die zugehörigen Datentabellen zu entwerfen und ggf. Diagramme aus den Daten zu erstellen.[33] Nach den Vorarbeiten wird die Anwendung mit vorgefertigten Komponenten entworfen. Im Gegensatz zu Canvas-Anwendungen ist die Gestaltung des Anwendungslayouts wenig flexibel, da es maßgeblich von den Komponenten bestimmt und durch die Konfiguration der Komponenten automatisch generiert wird. Die komponentenorientierte Entwurfsumgebung für die Entwicklung modellgetriebener Anwendungen heißt Power Apps App-Designer, dessen Struktur in Abbildung 4.7 dargestellt ist.

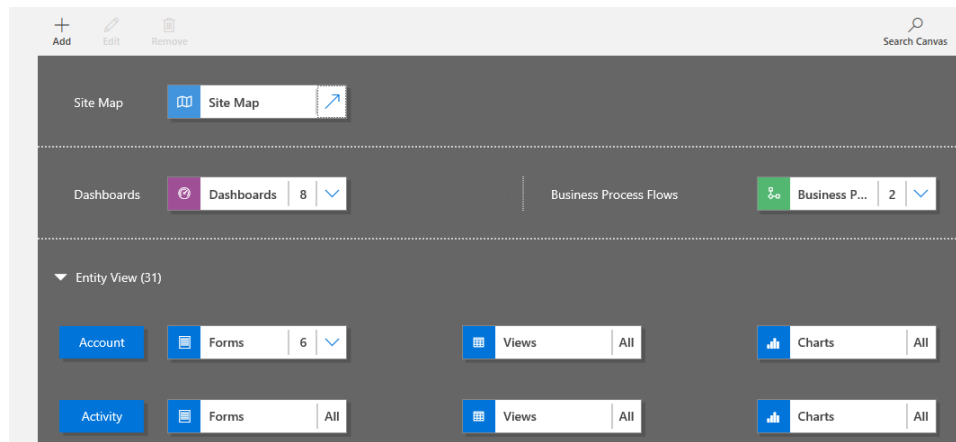


Abbildung 4.7: Layout des klassischen App-Designers

Der App-Designer besteht aus den Strukturen: Siteübersicht, Dashboards, Geschäftsprozessabläufe und Entitätsansichten. In der Siteübersicht wird die Navigation für die Anwendung eingerichtet. Beim Aufruf der Siteübersicht öffnet sich eine neue Ansicht, der sogenannte ‚Sitemap-Designer‘, um die hierarchische Struktur der Seiten zu erstellen. Hier werden Bereiche, Unterbereiche und Gruppen angelegt. Die hierarchische Anordnung der Komponenten ist in der Form dargestellt, dass alle erstellten Seiten der Anwendung als Unterbereiche angelegt werden, die wiederum einer Gruppe zugeordnet sind. Die gruppierten Seiten werden dann einem Bereich zugeordnet, um die Elemente strukturiert darzustellen.[34] Eine Beispielgruppierung wären Formular-

und Ansichtsseiten zum Erstellen oder Anzeigen von Daten einer zugehörigen Entität. Nach Abschluss der Hierarchie für die Seiten werden die Navigationselemente in der linken Seitenleiste der resultierenden Anwendung platziert. Wenn die Anwendung über mehr als einen Bereich verfügt, gibt es in der linken Seitenleiste ein Umschaltsteuerelement zum Wechseln der Bereiche. Unter Dashboards können neue oder fertige Dashboards erstellt werden. Unter anderem können auch Geschäftsprozessabläufe eingebunden werden. Schließlich integriert die Entitätsansicht die verschiedenen Formen, Ansichten, Diagramme und Dashboards einer Entität, die zuvor über Dataverse entworfen wurden.

Um eine Anwendung auch Benutzern außerhalb der Organisation zugänglich zu machen, hat Microsoft die Lösung Power Apps-Portale veröffentlicht. Power Apps-Portale sind öffentliche Websites, die sich die vielfältigen Möglichkeiten von Dataverse zunutze machen.[35] Um ein Portal zu erstellen und anzupassen, gibt es auch eine spezielle Anwendungsentwicklungsumgebung, namens Power Apps Portalstudio mit einem integrierten WYSIWYG-Editor, um das Layout der Webseiten durch einfaches Hinzufügen von Komponenten zu entwerfen[36]. In Abbildung 4.8 wird das Layout von Power Apps Portalstudio dargestellt.

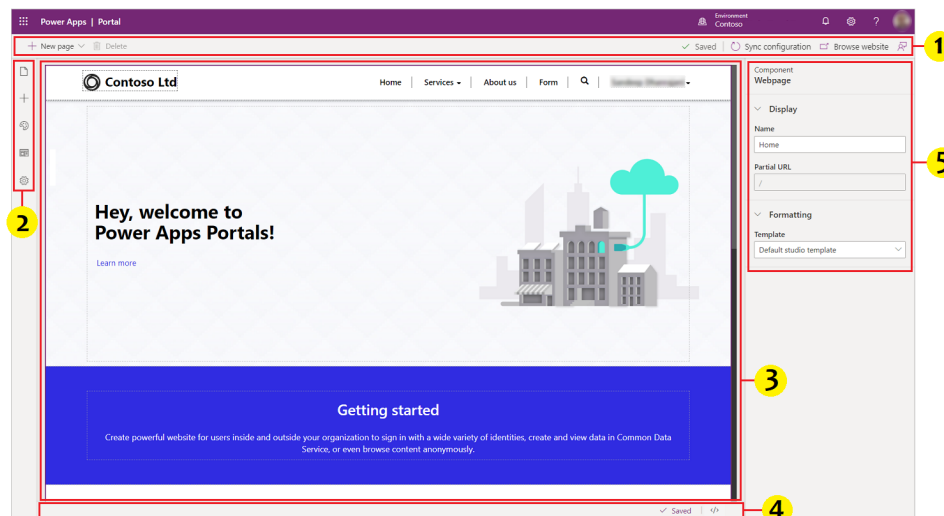


Abbildung 4.8: Layout vom Power Apps-Portalstudio[37]

Power Apps Portalstudio besteht aus einer linken Symbolleiste zum Hinzufügen von Komponenten zu Webseiten, ein Eigenschaftsbereich auf der rechten Seite des Layouts zum Bearbeiten der ausgewählten Komponente, einem mittleren Canvas zum Anzeigen der Webseiten, einer Fußzeile zum Öffnen des Quellcode-Editors und einer

Kopfzeile zum Erstellen neuer Webseiten und zum Löschen von Komponenten[37].

Wie klassische Websites können auch Power Apps-Portale verwaltet und konfiguriert werden. Dafür gibt es eine Verwaltungsanwendung namens Portal Management, deren Struktur in Abbildung 4.9 dargestellt ist.

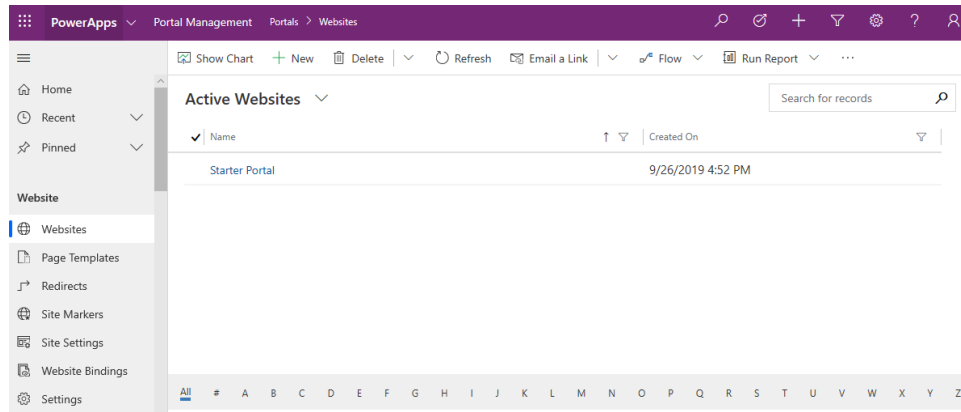


Abbildung 4.9: Ausschnitt der Power Portalverwaltungs-App[38]

Der Einsatz der Portal Management App ist dann notwendig, um das Portal zu verfeinern, beispielsweise durch das Hinzufügen von Formularen und das Erstellen von Website-Templates. Die wichtigste Komponente der Portalverwaltungs-App für die Konfiguration einer Portalanwendung ist die linke Seitenleiste, die aus den sechs Abschnitten Website, Inhalt, Sicherheit, Werbeanzeigen, Umfragen und Administration besteht.[38]

Im Abschnitt Website werden die erweiterten Konfigurationen für eine bestimmte Website durchgeführt. Hier ist es möglich, Seitenvorlagen zu erstellen, Weiterleitungen auf externe URLs festzulegen, Liquid-Objekte zu verwenden, um dynamische Inhalte nicht über fest codierte Links anzuzeigen, die Authentifizierung zu konfigurieren und portalspezifische Einstellungen wie die eindeutige URL festzulegen. Im Inhaltsbereich werden alle Inhalte in Form von Formularen, Listen, Sprachen, textuellen Inhaltsschnipseln und anderen Arten von Inhalten erstellt. Der Abschnitt Sicherheit listet alle authentifizierten Benutzer auf, legt Zugriffsberechtigungen über Rollen für bestimmte Informationen fest und steuert die Kontrolle und den Zugriff auf die Websites. Im letzten Bereich Administration können Analysen für bestimmte Websites aktiviert werden.[39]

4.3 ServiceNow Now Platform

4.3.1 Einführung

Service Now ist ein Softwareunternehmen, das seinen Nutzern eine cloudbasierte Plattform bietet, um manuelle Arbeitsabläufe in Unternehmen durch digitale Prozesse zu ersetzen. Hinter dieser cloudbasierten Lösung steht die Now Platform, die ihren Nutzern sowohl SaaS- als auch PaaS-Angebote zur Verfügung stellt.[40] Unter den zahlreichen Angeboten an Diensten bietet die Now Platform die Entwicklung von Mobil- und Webanwendungen auf der Grundlage von No- und Low-Code-Entwicklung mit verschiedenen Tools.

Darüber hinaus verfügt ServiceNow über ein zentrales IT-Service-Management-Tool, das sogenannte Service Management, zur Bereitstellung wiederverwendbarer Softwaredienste, sodass die Gesamtproduktivität gesteigert werden kann. Die Tools der Now Platform sind auch im Service Management verfügbar.[41] ServiceNow vertreibt ebenso seine Plattform als ein kommerzielles Produkt mit individueller Preisgestaltung[42]. Im Rahmen dieser Arbeit wurde ausschließlich die Testversion betrachtet.

4.3.2 Entwicklung

Für die Erstellung benutzerdefinierter Anwendungen bietet die Now Platform zwei vereinfachte Tools, mit denen es möglich ist, in wenigen Schritten eine Anwendung mit ihren wesentlichen Bestandteilen zu erstellen. Zu den Bestandteilen einer Anwendung gehören das Definieren eines Datenmodells, das Erstellen von Benutzeroberflächen, das Hinzufügen von der Geschäftslogik und Automatisierung, sowie das Definieren von Rollen für die Anwendungssicherheit. Eines der beiden Tools ist der ‚Guided Application Creator‘, der in den Systemanwendungen des Service Managements als integriertes Modalfenster zu finden ist[43]. Wie der Name schon verrät, führt es den Benutzer in verschiedenen Schritten durch die Erstellung der Anwendung. Das zweite Tool ist ‚App Engine Studio‘, eine eigenständige Webanwendung zum Erstellen von Low-Code-Anwendungen, deren Startbildschirm in Abbildung 4.10 dargestellt ist.

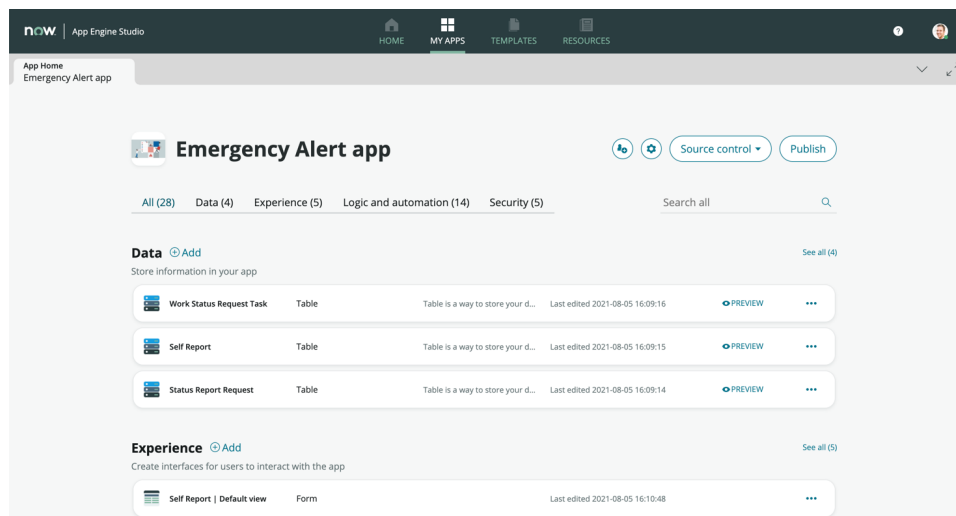


Abbildung 4.10: App Engine Studio[44]

Es besteht aus vier Kernkomponenten: Daten, Experience, Logik und Automatisierung, Sicherheit. Das Datenmodell der Anwendung wird in der Datenkomponente implementiert. Hier können Datentabellen von Grund auf aus einer vorhandenen Tabelle oder durch Hochladen einer Tabellenkalkulation erstellt werden. Die Struktur und der Inhalt der Datentabellen können hier ebenso eingesehen werden. In der Experience-Komponente werden die Benutzerschnittstellen für die Anwendung mit Hilfe eines Widget-basierten UI-Frameworks namens ‚UI-Builder‘ erstellt[45]. Eine Experience ist eine Sammlung an Seiten, über die Benutzer mit einer Anwendung interagieren können. Hier können verschiedene Experience-Typen wie ein Portal oder ein Arbeitsbereich verwendet werden. In einem Portal hat jede Seite eine gemeinsame Kopf- und Fußzeile, während in einem Arbeitsbereich jede Seite eine gemeinsame Kopf- und Seitenleiste hat.[46] Der nächste Schritt besteht darin, die Anwendungslogik mithilfe automatisierter Workflows hinzuzufügen. Für die Erstellung von Workflows existiert die Anwendung Flow Designer, mit der sich Geschäftslogik, Prozessautomatisierung und Integration mit einfacher Sprache und visuellen Elementen umsetzen lassen. Im letzten Schritt wird die Zugriffskontrolle der Anwendung durch das Definieren von Benutzerrollen festgelegt.[44]

Für die Weiterentwicklung einer erstellten Anwendung an einer zentralen Stelle bietet die ServiceNow Plattform ‚ServiceNow Studio IDE‘ als Schnittstelle an, die viele ähnliche Funktionen wie eine klassische, integrierte Entwicklungsumgebung hat. Das Layout von ServiceNow Studio wird in der Abbildung 4.11 demonstriert.

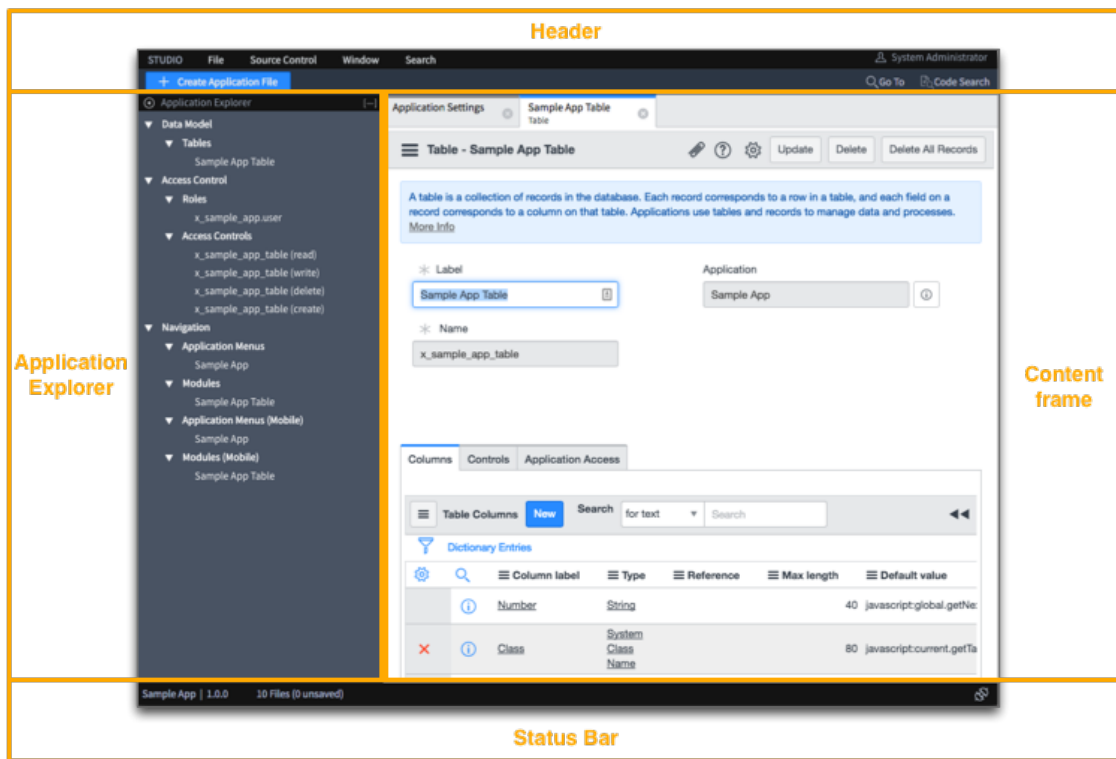


Abbildung 4.11: Layout von ServiceNow Studio IDE [47]

Die ServiceNow Studio IDE besteht im Wesentlichen aus einer Kopfzeile, einem Anwendungsexplorer, einem Inhaltsrahmen und einer Statusleiste. Die Menüs und Bedienelemente werden in der Kopfzeile angezeigt. Das Dateimenü kann verwendet werden, um eine Anwendungsdatei zu erstellen, den Guided Application Creator zu starten, um eine Anwendung zu erstellen, die aktuelle Anwendung in verschiedenen Umgebungen zu veröffentlichen, eine Anwendung aus einem Versionskontrollsystem zu importieren, Entwickler zu verwalten und Anwendungseinstellungen anzupassen. Viele wichtige Funktionen eines Versionskontrollsystems werden im Quellcodeverwaltungs-menü bereitgestellt, z. B. das Verwalten der Repository-Einstellungen, das Erstellen von Commits, Stashes und Branches sowie das Anzeigen der Historie. Der Anwendungsexplorer listet alle Dateien der Anwendung kategorisiert auf. Dabei sind beispielsweise Datentabellen, Navigationselemente, Benutzeroberflächen, Formulare, Skripte etc. modular gehalten. Ein Detailformular für jeden Datensatz wird in eigenen Registerkarten im Inhaltsrahmen angezeigt. In der Statusleiste werden der Anwendungsname, die Anwendungsversion, die Gesamtzahl der nicht gespeicherten Dateien und der aktuelle Branch angezeigt.[47]

4.4 Mendix Plattform

4.4.1 Einführung

Mendix ist eine Low-Code-Plattform zur Entwicklung mobiler und webbasierter Anwendungen. Sie ermöglicht die Entwicklung von Anwendungen mit wenig bis keiner Programmiererfahrung. Die Mendix Plattform bietet eine webbasierte Anwendungsentwicklungsumgebung mit visuellen Modellierungswerkzeugen für die reine Entwicklung einer Anwendung ohne Programmierung. Darüber hinaus gibt es auch eine Anwendungsentwicklungsumgebung als Desktop-Anwendung mit visuellen Modellierungswerkzeugen für die professionelle Entwicklung, die sich auch in klassische Entwicklungsumgebungen wie Eclipse integrieren lässt.[48]

Das Unternehmen hinter der Plattform wurde 2005 mit der Philosophie gegründet, durch modellgetriebene Anwendungsentwicklung eine gemeinsame Sprache zwischen Business und IT zu schaffen, damit kein Softwareprojekt an unterschiedlichen Denkweisen der Beteiligten scheitert[49]. Mendix bietet seinen Benutzern auch ein Preismodell an[50]. Für die Evaluation wurde hier nur die freie Version berücksichtigt. Die Mendix-Plattform besteht aus den drei Kernkomponenten Entwicklerportal, Mendix Studio und Mendix Studio Pro, die im Unterabschnitt 4.4.2 näher erläutert werden.

4.4.2 Entwicklung

Das Entwicklerportal, ein Online-System, stellt verschiedene Funktionalitäten wie unter anderem das Erstellen und Bereitstellen von Anwendungen, das Verwalten von Anwendungsbenutzern und das Einladen anderer Benutzer in das Anwendungsteam bereit[51]. Eine erstellte Anwendung kann entweder mit Mendix Studio oder Mendix Studio Pro bearbeitet werden. Es bietet auch Zugriff auf das Kontrollzentrum, das Unternehmensaktivitäten wie das Anzeigen externer Mitglieder mit Diagrammen oder ähnlichem darstellt.

Mendix Studio ist die webbasierte No-Code-Anwendungsentwicklungsumgebung der Mendix-Plattform, die darauf abzielt, die Benutzeroberfläche und die Funktionalität einer Anwendung zu entwickeln, ohne eine Programmiersprache zu verwenden. Sie ermöglicht die Umsetzung komplexer Logik in einer vereinfachten, visuellen Sprache.[52] Die Abbildung 4.12 zeigt den Aufbau von Mendix Studio.

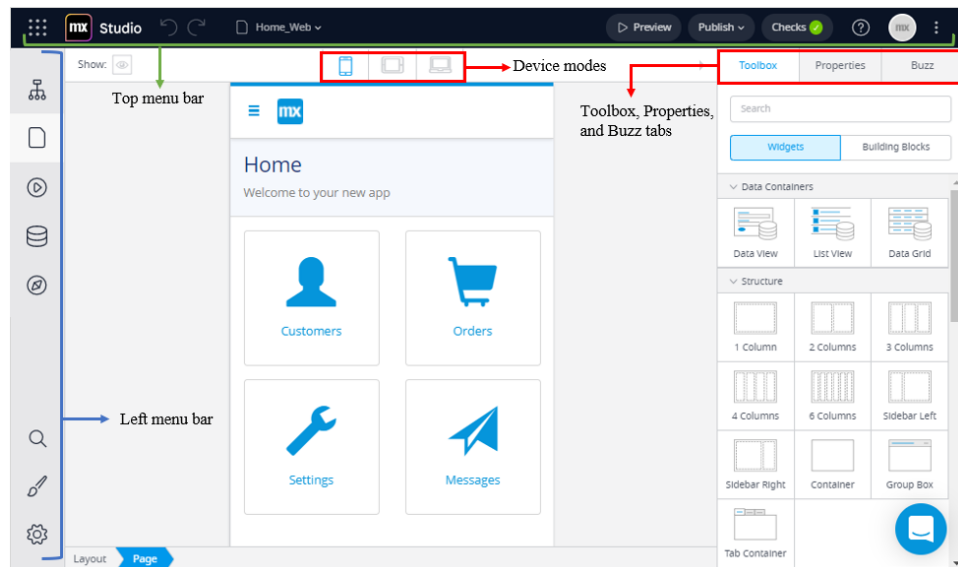


Abbildung 4.12: Layout von Mendix Studio[52]

Mendix Studio besteht aus einer oberen und linken Menüleiste, einem oberen rechten Menü, das aus den Registerkarten Toolbox, Properties und Buzz besteht, und drei oberen mittleren Schaltflächen zum Umschalten der Ansichten. In der linken Menüleiste sind Elemente wie das Datenmodell, auch Domänenmodell genannt, Seiten, Mikro- und Workflows und das Navigationsdokument erreichbar. Auch eine Suche innerhalb der Anwendung, eine Designanpassung und der Aufruf der Anwendungseinstellungen stehen hier zur Verfügung. Die Toolbox-Registerkarte zeigt alle Werkzeuge, die für den aktuellen Editor benötigt werden. Mit der Properties-Registerkarte können die Eigenschaften des ausgewählten Elements angepasst werden. Die Buzz-Registerkarte dient hauptsächlich der Kommunikation zwischen App-Entwicklungsteams über Kommentare. Die obere Menüleiste kann verwendet werden, um vorherige Zustände wiederherzustellen, zwischen kürzlich besuchten Seiten zu navigieren, den aktuellen Zustand der Anwendung in der Vorschau oder zu veröffentlichen und eine Fehlerprüfung anzuzeigen.[52] Mendix Studio bietet bereits viele Out-of-the-Box-Funktionalitäten, z.B. eingebaute Logik in Schaltflächen. Die Implementierung zusätzlich notwendiger Logik kann jedoch mit Hilfe sogenannter Microflows realisiert werden.

Ein Microflow ist eine visuelle Programmiertechnik, um textuellen Programmcode mit visuellen Werkzeugen auszudrücken[53]. Die klassischen Kernelemente

der Softwareentwicklung wie Schleifen, Variablen etc. können in einem Microflow verwendet werden. Der folgende Microflow aus der Abbildung 4.13 demonstriert die Ausführung einer Datenbankabfrage, deren Ergebnis als Rückgabewert zurückgegeben wird. Die ausführlichen Erläuterungen zu den visuellen Symbolen aus dem Microflow können dem Katalog A.2 entnommen werden.

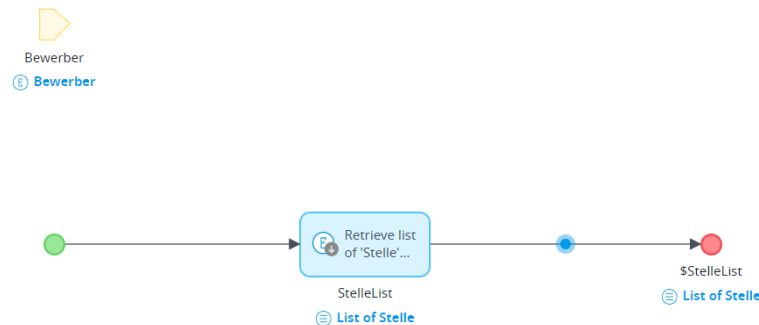


Abbildung 4.13: Beispiel eines Microflows in Mendix

Für die professionelle Anwendungsentwicklung bietet die Mendix-Plattform eine Desktop-basierte Low-Code-Anwendungsumgebung namens Mendix Studio Pro. Je nach Anwendungsfall kann die Erstellung einer Anwendung vorgefertigte Vorlagen als Start verwenden oder mit einer leeren Seite beginnen. Zusätzlich zu den Funktionalitäten der webbasierten Lösung bietet Mendix Studio Pro auch Erweiterbarkeit in Bezug auf externe Module oder benutzerdefinierten Code, die Integration von API-Schnittstellen und die Konfiguration der Anwendungssicherheit. Darüber hinaus werden essenzielle Werkzeuge wie ein Debugger zur Fehlerdiagnose oder die Datenbeschaffung mittels spezieller Abfragesprachen angeboten, die für eine professionelle Anwendungsentwicklung unverzichtbar sind. Ein Versionskontrollsystem wurde ebenfalls integriert, um die kollaborative Entwicklung zu unterstützen.[54] In Abbildung 4.14 wird das Layout von Mendix Studio Pro präsentiert.

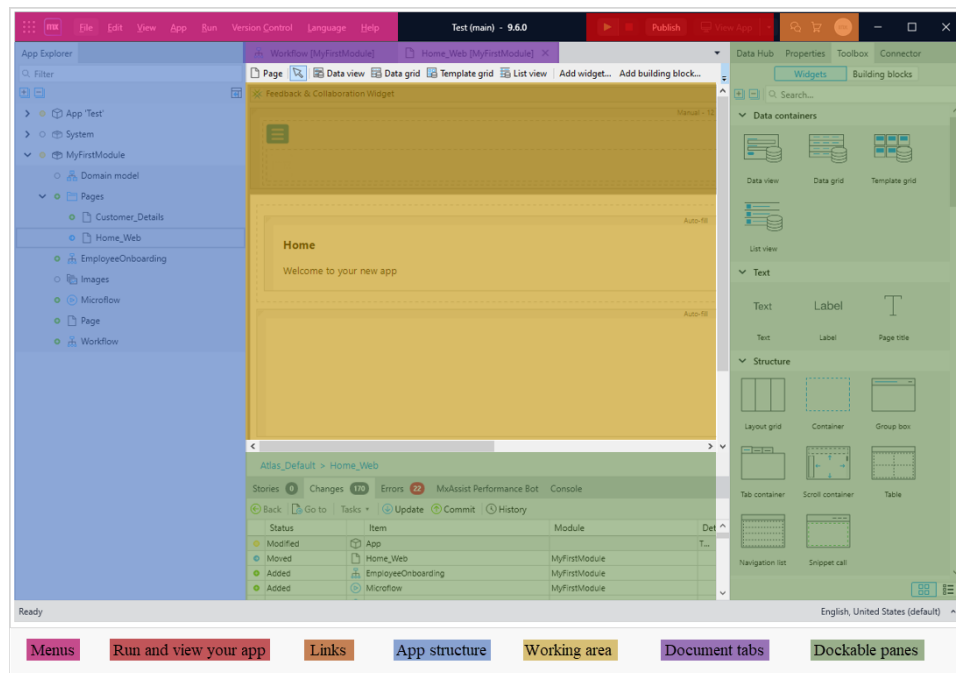


Abbildung 4.14: Layout von Mendix Studio Pro[54]

Mendix Studio Pro ist Mendix Studio strukturell sehr ähnlich. Die gesamte Anwendungsstruktur inklusive System- und Sicherheitseinstellungen, Navigation, Fehlermeldungen vom Server oder Client und alle Dokumente von externen Modulen aus dem Marketplace findet man in der linken Menüleiste. Ein bemerkenswerter Unterschied zu Service Studio ist der andockbare Bereich, der Zugriff auf Anpassungen, Fehler, eine Konsole, Variablen und einen Debugger bietet. Ein weiterer Unterschied besteht darin, dass mehrere Dokumente im Arbeitsbereich geöffnet und bearbeitet werden können. In den oberen Menüs können die Sprachen eingestellt und das Versionskontrollsystem genutzt werden. Es ist auch möglich, die Anwendung lokal zu starten und in der Mendix-Cloud zu veröffentlichen. Die Links erreichen das Entwicklerportal und den Marketplace.[54]

5 Umsetzung und Evaluation

In diesem Kapitel werden die verschiedenen Ziele und die Umsetzung der Anwendungsbeispiele beschrieben. Darüber hinaus werden die mit den verschiedenen Plattformen erzielten Ergebnisse zusammengefasst, die zuvor in den Protokollen dokumentiert wurden. Abschließend werden die Plattformen auf der Grundlage der Ergebnisse evaluiert. Die Evaluation erfolgte in drei Phasen, in denen jeweils unterschiedliche Forschungsfragen genauer untersucht wurden. Die Zuordnung einer Forschungsfrage zu einer Phase kann der folgenden Tabelle 5.1 in Matrixdarstellung entnommen werden.

	1.Phase	2.Phase	3.Phase
Zeitersparnis / Zeitverlust		×	
Integration / Bereitstellung von Services	×		
Anpassung durch selbstgeschriebenen Code			×
Werkzeuge zur Laufzeitanalyse		×	
Featureumfang für komplexe Anwendungen			×
Weiterentwicklung		×	
Testbarkeit der Anwendungen		×	
Debugging	×		
CI / CD			×
Skalierbarkeit und Ausfallsicherheit			×

Tabelle 5.1: Zuordnung von Forschungsfragen zu Phasen in Matrixdarstellung

Die Reihenfolge, in der die Plattformen für die Implementierung der Anwendungsbeispiele in Phasen genutzt wurden, ist der folgenden Tabelle 5.2 zu entnehmen.

	1.Phase	2.Phase	3.Phase
OutSystems	1	2	3
Power Apps	2	3	1
Mendix	3	1	2
ServiceNow	4	×	4

Tabelle 5.2: Reihenfolge der Plattformen für die Umsetzung der Anwendungen in den verschiedenen Phasen.

Die Umsetzung und die damit erzielten Ergebnisse der Anwendungsbeispiele mit

den Plattformen werden in der aufgeführten Reihenfolge vorgestellt.

5.1 1. Phase: Kfz-Zulassung

5.1.1 Ziele

Die erste Phase diente in erster Linie dem Einstieg in die Low-Code-Entwicklung. Daher war es von großer Bedeutung das Anwendungsbeispiel so zu formulieren, dass ein erster Einblick in die Plattformen gewonnen werden konnte. Da keine Vorerfahrungen im Bereich der Low-Code-Entwicklung vorhanden waren, wurde die Komplexität der Anwendung zunächst gering gehalten. Das erste Anwendungsbeispiel stellte einen klassischer Antragsprozess dar, bei dem die Interaktion des Benutzers mit der Benutzeroberfläche der Anwendung im Vordergrund stand. Der Benutzer sollte in mehreren Schritten verschiedene Formulare ausfüllen, um alle relevanten Informationen für den Antragsprozess zu erfassen.

Als Anwendungsfall wurde ein Antragsprozess aus dem Bereich der öffentlichen Verwaltung für Bürgerangelegenheiten formuliert, der per elektronischer Abwicklung umgesetzt werden sollte. Da dieser Bereich sehr umfangreich und komplex war, wurden einfache Dienstleistungen der Kfz-Zulassungsbehörde implementiert. Als erstes Anwendungsbeispiel für die erste Phase wurde das Antragsverfahren zur Umschreibung eines zugelassenen Kraftfahrzeugs auf einen anderen Halter mit Kennzeichenwechsel ausgewählt.

Dieser bestand aus vier Schritten, in denen der Nutzer die Formulare Schritt für Schritt ausfüllen sollte. Im ersten Schritt sollte der Nutzer seine persönlichen Daten, wie Vor- und Nachname, Geschlecht, Adresse etc. über ein Formular eingeben. Daran folgte das Fahrzeugdatenformular, in dem der bisherige Halter und das aktuelle Kennzeichen des Fahrzeugs anzugeben waren. Außerdem sollte hier optional auch die Beantragung eines Wunschkennzeichens über ein Eingabefeld zur Verfügung stehen. Im nächsten Schritt sollten die eingegebenen Daten aus den vorangegangenen Schritten zusammengefasst werden, so dass der Nutzer entweder die Richtigkeit der Daten bestätigen oder Korrekturen vornehmen konnte. Schließlich sollte es möglich sein, den Antragsprozess über eine abschließende Komponente zu beenden.

Das beschriebene Anwendungsbeispiel sollte als reine Webanwendung realisiert werden. Dabei sollte vor allem untersucht werden, ob und wie essenzielle Kom-

ponenten einer Webanwendung, wie das Datenmodell, die Geschäftslogik und die grafische Benutzeroberfläche mit Low-Code-Plattformen umgesetzt werden können. Ziel war es, neben der Untersuchung der genannten Komponenten auch die Integration externer Services, wie z.B. die Verfügbarkeit eines Wunschkennzeichens über eine REST-Schnittstelle zu prüfen. Ein weiteres Ziel war es, sich mit den Debugging-Tools der Plattformen vertraut zu machen. Die Anzahl der zu untersuchenden Forschungsfragen wurde für das erste Anwendungsbeispiel bewusst klein gehalten, um zunächst erste Erfahrungen mit den verschiedenen Plattformen zu sammeln.

5.1.2 Umsetzung

OutSystems

Als Anwendungstyp wurde eine reaktive Webanwendung gewählt, die bei hoher Benutzerinteraktion eine hohe Performance aufweist. Im ersten Schritt wurde das Datenmodell in der Datenschicht implementiert, das in Abbildung 5.1 zu sehen ist.

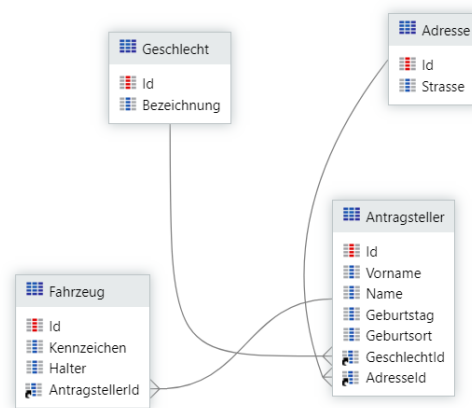


Abbildung 5.1: Das Datenmodell der Kfz-Zulassungsanwendung in OutSystems

Das Datenmodell wurde in der Datenschicht von Service Studio umgesetzt. Neben der manuellen Erstellung von Entitäten bietet OutSystems auch die Möglichkeit, Entitäten aus einer Excel-Datei zu erstellen. Als Datengrundlage wurden zufällig ausgewählte Straßen aus dem Umkreis Berlin-Steglitz in einer Excel-Datei gespeichert und in die Datentabelle ‚Adresse‘ importiert. Die Excel-Datei darf für den Import nur aus einer Tabelle bestehen, deren Spaltennamen in der ersten Zeile der Tabelle stehen

müssen. Beziehungen zwischen Tabellen wurden im Entitätsdiagramm über Verbindungslinien hergestellt. Ausgehend von der Entität, bei der die Verbindungslinie beginnt, wird zwischen zwei Entitäten immer eine 1:n-Beziehung erstellt. Das Referenzattribut wurde ebenfalls automatisch in der entsprechenden Tabelle generiert. Durch Drag-and-Drop einer Datentabelle aus der Datenschicht auf den MainFlow in der Schnittstellenschicht wurde automatisch sowohl eine Detailseite mit einem Formular zum Hinzufügen und Bearbeiten von Daten als auch eine Listenansicht zum Anzeigen aller vorhandenen Daten erstellt. Nach dieser Vorgehensweise wurden die in der Abbildung 5.2 gezeigten Detailseiten für die Entitäten Antragsteller und Fahrzeug erstellt, die nahezu unverändert für den Antragsprozess genutzt wurden.

The image displays three screenshots of the Kfz-Zulassungsanwendung user interface, showing the process of entering and summarizing application data.

Screenshot 1: Antragsdaten

Form fields for personal information:

- Vorname *
- Name *
- Geburtsort *
- Geburtsort *
- Geschlecht *
- Adresse *

Buttons: Zurück, Weiter

Screenshot 2: Fahrzeugdaten

Form fields for vehicle information:

- Halter *
- Kennzeichen *
- Antragsteller *

Buttons: Zurück, Weiter

Screenshot 3: Zusammenfassung

Summary of entered data:

Antragsdaten	Fahrzeugdaten
Vorname	Max
Name	Muster
Geburtsort	8 Feb 2022
Geburtsort	Musterstadt
Geschlecht	männlich
Strasse	Adolfstr, 12167, Berlin-Steglitz
Halter	Max Muster Neu
Kennzeichen	ABCD12

Buttons: Zurück, Bestätigen

Screenshot 4: Abschluss

Das Kraftfahrzeug wurde erfolgreich umgeschrieben!

Button: Fahrzeug Umschreibung beenden

Abbildung 5.2: Benutzeroberflächen der Kfz-Zulassungsanwendung mit OutSystems

Die Herausforderung bestand darin, den eindeutigen Identifikator des erstellten Antragstellers an die Fahrzeugdatendetailseite weiterzugeben, damit das referenzierte Datenfeld des Fahrzeugobjekts gesetzt werden konnte. Dieses Problem wurde durch Eingabeparameter gelöst, die über die URL übergeben wurden. In Service Studio ist es möglich, innerhalb des Sichtbarkeitsbereichs eines Bildschirms Eingabeparameter, lokale Variablen, Aggregate, Client-Aktionen usw. zu definieren, die ausschließlich in-

nerhalb dieses Bildschirms verfügbar sind, um beispielsweise Daten anzuzeigen oder Logik auszuführen. Mit diesem Wissen wurde im Sichtbarkeitsbereich der Antragstellerdetailseite eine lokale Variable angelegt, die den eindeutigen Identifikator des aktuellen Antragstellers speichert. Durch Klicken des Benutzers auf die Schaltfläche ‚Weiter‘ wurde eine Client-Aktion ausgelöst, deren Implementierung in Abbildung 5.3 detailliert dargestellt ist.

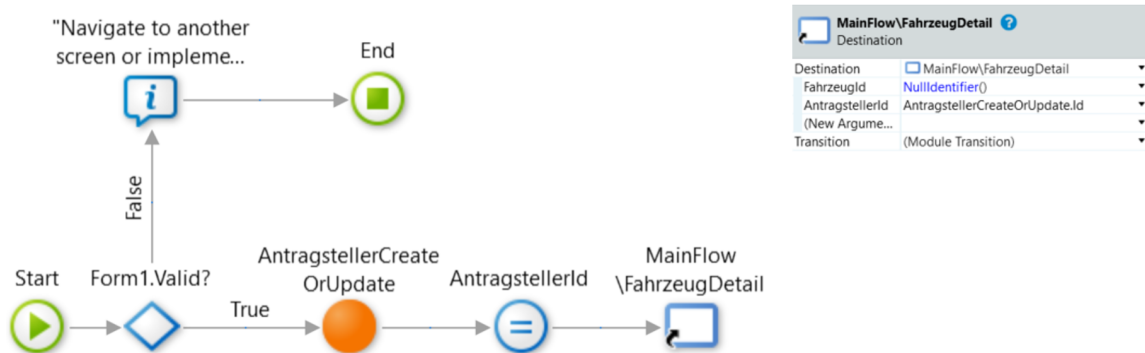


Abbildung 5.3: Client-Aktion zum Anlegen eines Antragstellers in Service Studio

Wenn das Formular gültig war, wurde der Antragsteller zuerst über die Server-Aktion ‚AntragstellerCreateOrUpdate‘ erstellt und dann sein eindeutiger Identifikator der lokalen Variablen ‚AntragstellerId‘ zugewiesen. Bei der Weiterleitung auf die Fahrzeugdetailseite wurde die lokale Variable als Argument übergeben, welche dann als Eingabeparameter im Sichtbarkeitsbereich der Fahrzeugdetailseite zur Verfügung stand. Da die Entität ‚Fahrzeug‘ in Beziehung mit der Entität ‚Antragsteller‘ stand, erforderte der Sichtbarkeitsbereich der Fahrzeugdetailseite die Kenntnis über den eindeutigen Identifikator des Antragstellers, um den richtigen Antragsteller in der zugehörigen Dropdown-Liste des Formulars anzuzeigen. Unter Verwendung des eindeutigen Identifikators des Antragstellers wurde ein Aggregat geschrieben, um die zugehörigen Daten aus der Datenbank zu extrahieren. Die gleiche Vorgehensweise wurde für die Übersichtsseite angewandt, mit dem Unterschied, dass sowohl der eindeutige Identifikator des Fahrzeugs als auch des Antragstellers übergeben wurden. Die entsprechenden Identifikatoren wurden auch als Eingabeparameter über den Zurück-Button an die Detailseiten übergeben, so dass Anpassungen am richtigen Datensatz vorgenommen werden konnten.

Der nächste Schritt bestand darin, die REST-API zur Verfügbarkeitsprüfung eines gewünschten Kennzeichens in die Anwendung zu integrieren. Da aktuell keine öffent-

liche REST-API existiert, die dieses Problem löst, war es notwendig, eine eigenständige Programmierschnittstelle als Mockup zu entwickeln. Die erforderliche REST-API wurde mit Spring Boot entwickelt und auf Heroku, einer kostenfreien Cloud Plattform as a Service, bereitgestellt. Die Bereitstellung war notwendig, da veröffentlichte Anwendungen auf einer Cloud nicht mit lokalen Umgebungen kommunizieren können. Die REST-API diente nur als Beispieldienst im Rahmen der Untersuchung der Forschungsfrage, so dass die Verfügbarkeit eines Kennzeichens zufällig ermittelt wurde. Ihre Integration wurde in der Datenschicht über eine vorgefertigte Lösung realisiert, die in Abbildung 5.4 dargestellt ist. Es können entweder einzelne oder mehrere Methoden einer REST-API integriert werden. Für die Nutzung mehrerer Methoden einer REST-API war der Import einer OpenAPI-Spezifikation mit Version 3.0 notwendig.

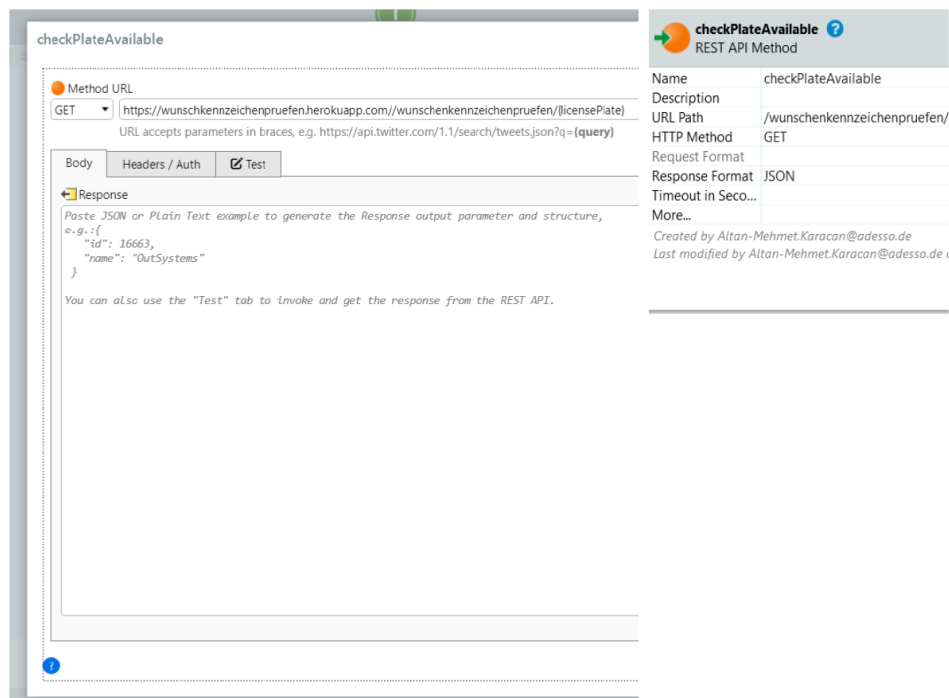


Abbildung 5.4: Konfiguration einer REST-API Integration in Service Studio

Diese Lösung ermöglichte es, einzelne Methoden zu testen, das Ausgabeformat zu definieren und die Ausgabe anzuzeigen sowie Request-Header zu setzen. Sowohl hier als auch im Eigenschaftseditor wurden die HTTP-Anfragemethode und das Ausgabeformat definiert. Der Aufruf der REST-API wird als Server-Aktion in die folgende Client-Aktion aus Abbildung 5.5 aufgenommen, die über das Ereignis ‚onChange‘ aufgerufen wurde, wenn sich das Kennzeichen-Eingabefeld auf der Fahrzeugdetail-

seite änderte.

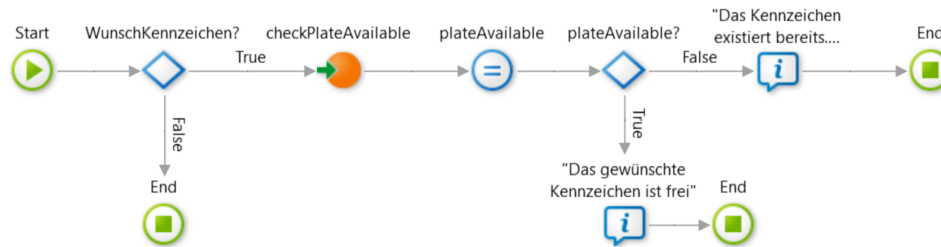


Abbildung 5.5: Aufruf der REST-API in einer Client-Action

Der REST-API-Aufruf erfolgte, wenn das Kontrollkästchen des Formulars auf der Fahrzeugdetailseite aktiv war. Das Ergebnis wurde in der Variablen ‚plateAvailable‘ gespeichert. Wenn bereits ein Kennzeichen vergeben war, wurde die Schaltfläche ‚Weiter‘ deaktiviert, bis ein freies Kennzeichen eingegeben wurde.

Insbesondere wurde das Debugging-Tool von Service Studio bei der Entwicklung der Client-Aktion aus Abbildung 5.5 verwendet, um die REST-API-Ausgabe genauer zu untersuchen. Ein Breakpoint konnte auf jedem Element der Client-Aktion platziert werden. Auf diese Weise konnten die Werte aller Variablen, die sich während des gesamten Ablauf der Client-Aktion geändert haben, untersucht werden, um mögliche Fehler zu diagnostizieren.

Power Apps

Das Datenmodell der Anwendung wurde in Dataverse umgesetzt. Alle Datenspalten wurden als Pflichtfelder definiert, um zu verhindern, dass die entsprechenden Felder in den Formularen unausgefüllt bleiben. Es bestand auch die Möglichkeit, die Datentabellen aus einer SharePoint- oder Excel-Liste zu erstellen. Der Dataverse-Ansatz wurde hier bevorzugt, weil er bereits eine intuitive Lösung für die Erstellung von Tabellenbeziehungen bietet. Für jede Tabelle gibt es eine Registerkarte, auf der die Kardinalität und die verknüpfte Tabelle ausgewählt werden können. Mögliche Kardinalitäten in Dataverse waren n:1, 1:n und n:m. Nachdem die Datentabellen erstellt wurden, wurde eine Canvas-App erstellt. Sie wurde gegenüber den anderen Anwendungstypen aufgrund ihrer Einfachheit und Flexibilität bei der Erstellung benutzerdefinierter Bildschirme sowie der Integration einer REST-API über einen Konnektor bevorzugt. Der nächste Schritt bestand darin, die erstellten Tabellen in die Anwendung zu integrieren. Über den Auswahlbereich der linken Seitenleiste in

Power Apps Studio wurden die Tabellen aus Dataverse in den Sichtbarkeitsbereich der Anwendung eingefügt.

Für die Erstellung der Antragsteller- und Fahrzeugdetailseite war es notwendig, für jede Entität zwei Bildschirme zum Anlegen und Bearbeiten eines Datensatzes zu erstellen, da ein Formular nur einen der Zustände ‚Neu‘, ‚Bearbeiten‘ und ‚Anzeigen‘ als Standardmodus annehmen kann. Die verschiedenen Formulare wurden über ihren Eigenschaftsbereich mit der entsprechenden Datentabelle verknüpft und mit frei wählbaren Attributen der Datentabelle zusammengestellt. Ein Unterschied beim Einrichten von Formularen im Bearbeitungsmodus bestand darin, dass zusätzlich der zu bearbeitende Datensatz angegeben werden musste. Dieser Datensatz wurde mit einer vordefinierten Power Apps-Funktion bestimmt, die direkt auf den zuletzt eingegebenen Datensatz eines bestimmten Formulars zugreift. Auf diese Weise wurde auf der Fahrzeugdetailseite im Dropdown-Menü der richtige Antragsteller gesetzt. Für die Übersichtsseite wurden zwei Anzeigeformulare verwendet, um Daten aus zwei verschiedenen Datenquellen auf einem Bildschirm anzuzeigen. Abbildung 5.6 zeigt die Benutzeroberflächen der mit Power Apps entwickelten Kfz-Zulassungsanwendung.

The image displays four screens of a Power Apps application for vehicle registration:

- Antragsdaten:** A form with fields for Vorname, Name, Geschlecht, Geburtstag (date and time pickers), Geburtsort, Straße, Hausnummer, Plz, and Ort. Navigation buttons 'Zurück' and 'Weiter' are at the bottom.
- Fahrzeugdaten:** A form with fields for Halter, a checkbox for 'Wünschen Sie sich ein neues Kennzeichen?', Kennzeichen, and a dropdown for 'Antragsteller' (set to 'Muster'). Navigation buttons 'Zurück' and 'Weiter' are at the bottom.
- Zusammenfassung:** A summary screen showing the data entered in the previous screens in two columns: Antragsdaten and Fahrzeugdaten. Navigation buttons 'Zurück' and 'Bestätigen' are at the bottom.
- Abschluss:** A completion screen with a checkmark icon, the message 'Die Kraftfahrzeug Umschreibung war erfolgreich!', and a button 'Fahrzeug Umschreibung beenden'.

Abbildung 5.6: Benutzeroberflächen der Kfz-Zulassungsanwendung mit Power Apps

Um das Speichern von doppelten Datensätzen zu vermeiden, wurden Validierungsformeln geschrieben, die über das ‚onSelect‘-Ereignis der ‚Weiter‘-Schaltflächen aufgerufen wurden. Mithilfe einer Lookup-Funktion wurde ermittelt, ob der zu speichernde Datensatz bereits in der entsprechenden Datentabelle existierte. Auf diese Weise wurde die Anwendungslogik in Form von ähnlichen Formeln in Excel ausschließlich über klassische HTML-Ereignisse wie ‚onChange‘, ‚onSelect‘, etc. aufgerufen. Im Folgenden wird die Validierungsformel zur Erkennung eines vorhandenen Antragstellers dargestellt.

```
If (
  IsBlank (
    Lookup( Antragsteller ;
            Name=DataCardValue2.Text And
            Vorname=DataCardValue1.Text
    )
  );
  SubmitForm(Form1) ;;
  Set(antragstellerFree; true) ;;
  Set(AntragstellerEditScreen; false);
  Notify("Dieser Antragsteller existiert bereits" );;
  Set(antragstellerFree; false)
)
```

Listing 1: Validierungsformel zur Erkennung vorhandener Antragssteller in Power Apps

Mit der Lookup-Funktion wurde die Tabelle aller Antragsteller nach einem Antragsteller durchsucht, der die angegebenen Kriterien erfüllt. Wenn kein Antragsteller gefunden wurde, wurde mit der Funktion ‚SubmitForm‘ ein neuer Antragsteller aus den Formularfeldern erstellt. In der Variable ‚AntragstellerEditScreen‘ wurde der letzte Zustand des entsprechenden Formulars gespeichert, um auf dessen Datensatz zuzugreifen.

Um eine REST-API in die Anwendung zu integrieren, musste zunächst ein benutzerdefinierter Konnektor erstellt werden. Es gab mehrere Optionen, die als Grundlage für die Erstellung eines Konnektors verwendet werden konnten, z. B. ein GitHub-

Repository, ein Azure-Dienst, eine OpenAPI-Datei oder -Link, eine Postman-Collection-Datei oder von Grund auf ohne Vorlage. Der Import einer OpenAPI-Datei wurde in diesem Fall bevorzugt, da eine solche Spezifikationsdatei bereits vorhanden war. Da Power Apps jedoch nur die OpenAPI-Spezifikation 2.0 unterstützt, wurde sie unter Verwendung der SpringFox-Abhängigkeit generiert. Unter Verwendung eines Dienstes von Power Apps wurde der Konnektor in den folgenden fünf Schritten erstellt.

1. Allgemein: Hier wurden allgemeine Informationen wie der Host und eine Beschreibung der REST-API angegeben. Host und die Basis-URL wurden automatisch aus der OpenAPI-Datei generiert.
2. Sicherheit: Hier wurde der Authentifizierungstyp aus den möglichen Authentifizierungstypen wie Standardauthentifizierung mit API-Schlüssel, OAuth2.0 oder keine Authentifizierung festgelegt. Letzteres wurde als Authentifizierungstyp ausgewählt.
3. Definition: Hier wurden alle Aktionen aus der Spezifikation aufgelistet und die Anforderungen wie URL, Pfadvariablen und die Antwort der Aktion definiert. Durch die Auswahl einer vorhandenen Aktion wurden die Anforderungen automatisch ermittelt. Eine Beispielausgabe des Endpunkts wurde verwendet, um die Antwortstruktur zu erstellen.
4. Code: Dieser Schritt war optional. Hier konnte benutzerdefinierter Code in C# verwendet werden, um Nutzdaten einer API-Antwort zu manipulieren. Es wurden jedoch keine Anpassungen vorgenommen.
5. Testen: Schließlich konnte der Konnektor getestet werden, nachdem er erstellt wurde. Zu diesem Zweck wurde dieser Schritt erneut aufgerufen, um den Konnektor mit Beispieldaten zu testen.[55]

Der Konnektor wurde dann über den Datenauswahlbereich in die Anwendung integriert. Sein Aufruf erfolgte über das Ereignis ‚onChange‘, wenn der Wert des Eingabefelds für das Kennzeichen geändert wurde.

Zur Fehlererkennung wurde unter den erweiterten Tools das Monitoring-Tool für die Canvas-App verwendet. Hier wurde jede Interaktion mit der Benutzeroberfläche und die damit aufgerufenen Ereignisse mit ihren Metriken wie Zeit, Kategorie,

5 Umsetzung und Evaluation

Vorgang, Ergebnis, Dauer usw. aufgezeichnet. In Abbildung 5.7 wird eine detaillierte Auflistung der Ereignisse abgebildet.

ID	Uhrzeit	Kategorie	Vorgang	Ergebnis	Ergeb...	Status	Dauer (ms)	Datenquelle	Steuerelement	Eigenschaft	Antwortgröße
43	22:11:11.079	UserAction	Select	Erfolgreich					DataCardValu...	Unselect	
44	22:11:11.380	UserAction	SetProperty	Erfolgreich					DataCardValu...	Text	
45	22:11:11.454	UserAction	Select	Erfolgreich					DataCardValu...	OnSelect	
46	22:11:11.700	UserAction	SetProperty	Erfolgreich					DataCardValu...	Text	
47	22:11:11.802	UserAction	Select	Erfolgreich					DataCardValu...	OnSelect	
48	22:11:12.100	UserAction	SetProperty	Erfolgreich					DataCardValu...	Text	
49	22:11:12.199	UserAction	Select	Erfolgreich					ErrorMessage7	OnSelect	
50	22:11:12.606	UserAction	Select	Erfolgreich					DataCardValu...	OnSelect	
51	22:11:12.931	UserAction	SetProperty	Erfolgreich					DataCardValu...	Text	
52	22:11:13.006	UserAction	Select	Erfolgreich					DataCardValu...	OnSelect	
53	22:11:15.273	UserAction	SetProperty	Erfolgreich					DataCardValu...	Text	
54	22:11:16.752	UserAction	Select	Erfolgreich					Button2_1	OnSelect	
55	22:11:16.940	Network	getRows	Erfolgreich		200	166	Antragsteller	Button2_1	OnSelect	268
56	22:11:17.206	Network	createRow	Erfolgreich	Creat...	201	249	Antragsteller	Button2_1	OnSelect	2.371
57	22:11:17.799	Network	getRows	Erfolgreich		200	193	Antragsteller	DataCardValu...	SearchItems	10.729
58	22:14:05.655	UserAction	Select	Erfolgreich					DataCardValu...	OnSelect	
59	22:14:08.960	UserAction	SetProperty	Erfolgreich					DataCardValu...	Text	
60	22:14:09.056	UserAction	Select	Erfolgreich					Checkbox2	OnSelect	
61	22:14:09.056	UserAction	SetProperty	Erfolgreich					Checkbox2	Value	
62	22:14:12.798	UserAction	Select	Erfolgreich					DataCardValu...	OnSelect	
63	22:14:13.978	UserAction	SetProperty	Erfolgreich					DataCardValu...	Text	
64	22:14:38.140	Network	CheckPlateAv...	Erfolgreich		200	23.278	ApiDocument...	DataCardValu...	OnChange	63

CheckPlateAvailable

Details **Formel** Anforderung Antwort

DataCardValue32.OnChange

```
1  If(Checkbox2.Value = true; If(ApiDocumentation.CheckPlateAvailable(DataCardValue32).
    isDesiredLicensePlateFree = true; Set(isFree; true);; Notify("Das gewünschte Kennzeichen
    ist frei!"); Set(isFree; false);;Notify("Das gewünschte Kennzeichen ist bereits vergeben.
    Versuchen Sie ein anderes Kennzeichen")))
```

Abbildung 5.7: Exemplarische Überwachung der Kfz-Zulassungsanwendung in Power Apps

Ein Vorgang entsprach beispielsweise dem Aufruf einer Funktion aus einer Formel, dessen Parameter wie die Zieladresse und die Antwort des API-Endpunkts erfasst wurden. Als Vorgang wurde in diesem Fall der API-Aufruf für die Wunschkennzeichenprüfung aus der Formel in Abbildung 5.7 hervorgehoben.

Mendix

Die Umsetzung des Anwendungsbeispiels wurde zunächst mit Mendix Studio realisiert und im Zuge der Entwicklung mit Mendix Studio Pro finalisiert. Der Grund für den Umstieg wird im späteren Verlauf dieses Unterabschnittes näher erläutert.

Als Grundlage der Anwendung wurde keine vorhandene Vorlage verwendet. Das Datenmodell der Anwendung wurde im Domänenmodell implementiert. Entitäten

5 Umsetzung und Evaluation

mit ihren Attributen des Datenmodells wurden über ein visuelles Entitätselement aus der Toolbox per Drag-and-Drop erstellt. Assoziationen zwischen Tabellen wurden über Verbindungslinien hergestellt. Für jede Verbindungslinie konnte zudem über die Registerkarte ‚Eigenschaften‘ die Kardinalität (1:1, 1: n und n: m) festgelegt werden.

Als Grundlage für die Erstellung der Antragsteller- und Fahrzeugdetailseite wurden vorgefertigte Vorlagen verwendet, die nur geringfügige Anpassungen, wie die Konfiguration der Datenquelle für die Formulare erforderten. Abbildung 5.8 zeigt die Benutzeroberflächen der mit Mendix entwickelten Kfz-Zulassungsanwendung.

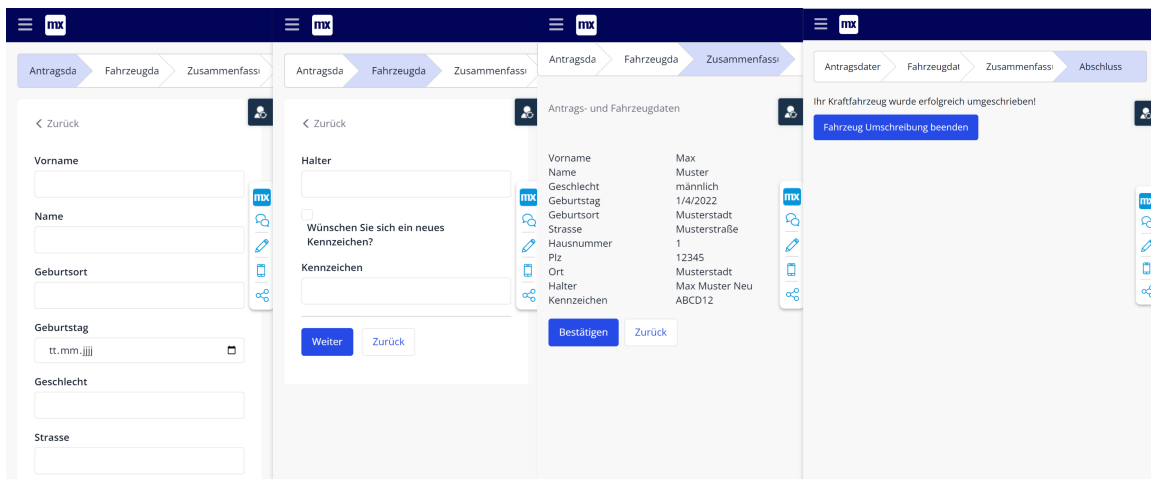


Abbildung 5.8: Benutzeroberflächen der Kfz-Zulassungsanwendung mit Mendix

Das Anlegen eines Antragsteller- und Fahrzeugobjektes, sowie die Weiterleitung zu den entsprechenden Seiten wurden mithilfe von Microflows realisiert. Diese wurden entsprechend aufgerufen, wenn der Benutzer auf die ‚Weiter‘- Schaltflächen klickte. Abbildung 5.9 zeigt die Umsetzung des Microflows zum Anlegen eines Antragstellers.

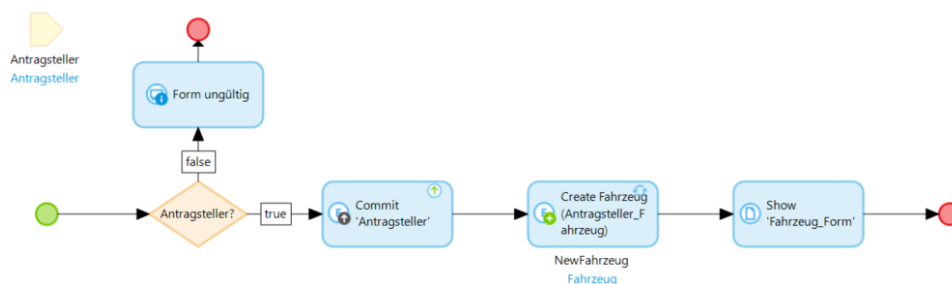


Abbildung 5.9: Microflow zum Anlegen eines Antragstellers

Der Microflow prüfte, ob das über die Formularfelder übergebene Antragsteller-

Objekt existierte, speicherte es in der Datenbank, erstellte ein Fahrzeugobjekt und setzte den assoziierten Antragsteller. Die Zuweisung des Antragsteller auf das Objekt ‚Fahrzeug‘ wurde mit dem Ausdruck *\$Antragsteller* in den Eigenschaften der Create-Object-Objektaktivität realisiert. Im letzten Schritt wurde das Fahrzeugobjekt an die Fahrzeugdetailseite übergeben. Die Objektübergabe war an dieser Stelle notwendig, da die Fahrzeugdetailseite für das Formular ein Fahrzeugobjekt erwartete. Für die Übersichtsseite wurde das Datenansicht-Widget verwendet, dem das Fahrzeugobjekt als Datenquelle zugewiesen wurde. Ausdrücke wie *\$Fahrzeug/Antragsteller/Name* ermöglichten den Zugriff auf die Attribute, der in Beziehung zur Fahrzeugentität stehende Antragstellerentität, um sie in der Datenansicht anzuzeigen.

Die Integration einer REST-API wurde in Mendix Studio nicht unterstützt, da dies durch den Umfang der No-Code-Entwicklung nicht abdeckt wurde. Daher erfolgte der Umstieg auf Mendix Studio Pro. Bevor die REST-API in die Anwendung integriert werden konnte, war es zunächst notwendig, eine Anpassung am Datenmodell vorzunehmen. Der Zustand, ob der Benutzer ein Wunschkennzeichen angefordert hat, der mit einem Kontrollkästchen auf der Fahrzeugdetailseite abgebildet wurde, konnte nicht in einer lokalen Variablen gespeichert werden. Da Formularfelder auf Attribute einer Entität abgebildet wurden, war es nicht möglich, ein Formular um zusätzliche Felder zu erweitern, die nicht in der Entität abgebildet waren. Dies erzwang die Erweiterung der Fahrzeugentität um das Attribut des beschriebenen Zustands.

Die Integration der REST-API wurde im Wesentlichen in drei Schritten realisiert. Der erste Schritt bestand darin, die JSON-Struktur der REST-API-Antwort zu definieren. Zu diesem Zweck wurde ein vorgefertigtes Tool verwendet, welches die gewünschte JSON-Struktur aus einem Ausschnitt einer beispielhaften JSON-Antwort der REST-API extrahierte. Im nächsten Schritt wurde ein sogenanntes Import-Mapping erstellt, welches die JSON-Antwort der REST-API auf die definierte JSON-Struktur abbildete. Da in Mendix Daten nur durch Entitäten repräsentiert werden, war es notwendig, eine nicht-persistente Entität im Domänenmodell anzulegen, damit die JSON-Struktur auf diese abgebildet werden konnte. Der letzte Schritt bestand darin, den Microflow aus Abbildung 5.10 zu erstellen, um die REST-API aufzurufen.

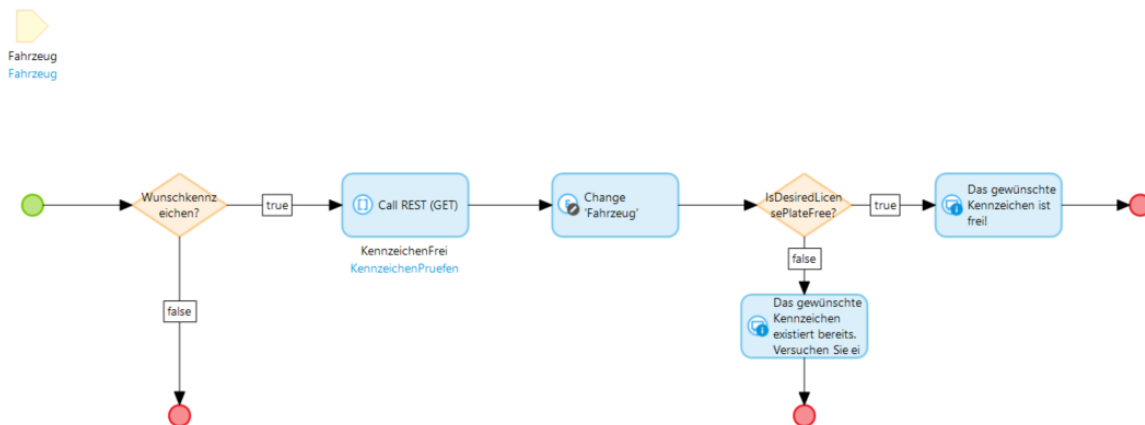


Abbildung 5.10: Microflow zum Aufrufen der REST-API

Der REST-API-Aufruf erfolgte über die Integrationsaktivität ‚Call REST service‘, zu der in den Eigenschaften die URL und die HTTP-Methode der REST-API sowie das Import-Mapping festgelegt wurden.

Das integrierte Debugging-Tool von Mendix Studio Pro wurde eingesetzt, um potenzielle Anwendungsfehler zu finden. Dazu wurde auf einem Element des Microflows ein Breakpoint gesetzt, um jede Aktion des Microflows Schritt für Schritt zu verfolgen. Sobald der Microflow ausgelöst wurde, wurden die Variablen mit ihren entsprechenden Typen und Werten bei jedem Aktionsschritt angezeigt.

ServiceNow

Für die Entwicklung der Anwendung wurde der webbasierte Dienst App-Engine Studio verwendet. Nach der Erstellung der Anwendung wurde zunächst das Datenmodell von Grund auf in der Datenkomponente implementiert, indem die Tabellen und ihre Attribute mit den zugehörigen Datentypen definiert wurden. Eine Beziehung zwischen zwei Tabellen konnte durch die Definition des Referenzattributs in der verknüpften Tabelle hergestellt werden. Hier war nicht eindeutig erkennbar, welche möglichen Kardinalitäten für Beziehungen verwendet werden können.

Der nächste Schritt war die Implementierung der Benutzerschnittstellen. In der Experience-Komponente war es erforderlich, eine Experience auf der Grundlage einer bestehenden Vorlage zu verwenden. Da nicht ersichtlich war, welche Vorlage die Anforderungen der Anwendung abdecken würde, wurden einige Vorlagen teilweise verwendet, um herauszufinden, wie sie aufgebaut waren und ob Anpassungen möglich waren. Die meisten Vorlagen entsprachen nicht den Anforderungen des

Anwendungsbeispiels, so dass im Servicemanagement eine neue Experience mit den entsprechenden Konfigurationen erstellt wurde, welche keine Vorlage verwendete. Zum Erstellen einer Experience war es notwendig den Experience-Typ, den URL-Pfad und die Landingpage der Anwendung zu konfigurieren. Als Experience-Typ wurde ein leerer Arbeitsbereich erstellt, der keine Vorlagen enthielt. Für die Gestaltung der Benutzeroberflächen der Detailseiten wurde das Formular-Widget im UI-Builder verwendet. Auch hier konnte die Datenquelle angegeben werden, allerdings gab es das Problem, dass die Formularfelder nicht editierbar waren. Teilweise wurden in der Vorschauansicht sogar keine Formularfelder angezeigt. Daher wurde ein eigenständiges Formular erstellt, das aus Eingabefeldern und Labels bestand, die die Attribute einer Entität abbildeten.

Um die Werte aus den Formularfeldern zu extrahieren und daraus einen Datensatz für die entsprechende Datentabelle zu erstellen, war es notwendig, ein Client-Skript mit JavaScript zu entwickeln. Das Skript sollte einen Eintrag in der entsprechenden Datentabelle vornehmen und den eindeutigen Identifikator als Parameter an die Folgeseite übergeben, damit der zugehörige Datensatz auf dieser Seite angezeigt werden konnte. Es ermöglichte den Zugriff auf die Werte der Formularfelder und die Erstellung eines Datensatzes in der entsprechenden Tabelle über einen API-Aufruf. Allerdings war es mit dem Skript nicht möglich, den eindeutigen Identifikator des erstellten Datensatzes vom Server abzufragen.

Ein neuer Lösungsansatz für das Problem bestand darin, eine Geschäftsregel in ServiceNow zu definieren. Eine Geschäftsregel ist ein serverseitiges Skript, das ausgeführt wird, wenn eine CRUD-Operation an einem Datensatz durchgeführt oder eine Tabelle abgefragt wird[56]. Für die Geschäftsregel konnte angegeben werden, wann und für welche Datenbankoperation auf einer bestimmten Tabelle sie ausgeführt werden soll. Das serverseitige Skript, das beim Eintreffen dieser Konfiguration ausgelöst werden sollte, musste an dieser Stelle eigenständig mit JavaScript entwickelt werden. Eine vordefinierte Funktion wurde verwendet, um den eindeutigen Identifikator des erstellten Datensatzes zu ermitteln. Darüber hinaus gab es eine weitere Funktion zur Weiterleitung zu einer bestimmten URL. Als die Geschäftsregel ausgelöst wurde, wurde auf dem Client keine Umleitung durchgeführt. Um das Problem zu beheben, wurde der Skript-Debugger von ServiceNow verwendet, um das zugehörige Geschäftsregel-Skript genauer zu untersuchen, was jedoch die Ursache des Problems nicht identifizieren konnte. Da wesentliche Teile der Anwendung von der Lösung dieses Problems abhingen, schränkte dies die Entwicklung der Anwendung ein, so

dass die Forschungsfragen nur in begrenztem Umfang untersucht werden konnten.

5.1.3 Evaluation

Mit Ausnahme von ServiceNow wurden die in dieser Phase definierten Ziele mit allen Plattformen weitgehend erreicht. Die Ergebnisse zeigten jedoch sowohl Gemeinsamkeiten als auch Unterschiede zwischen den Plattformen bei der Implementierung der Anwendung, die in den folgenden Kategorien dargestellt werden.

Datenmodellierung

Mendix und OutSystems verwenden Entitätsdiagramme mit visuellen Objekten zur Erstellung von Datentabellen. Ein Unterschied zwischen den beiden Plattformen besteht darin, dass bei Verwendung einer Verbindungslinie zwischen zwei Entitäten im Entitätsdiagramm eine Beziehung in OutSystems standardmäßig immer mit einer 1:n-Kardinalität abgebildet wird. An dieser Stelle war es nicht selbsterklärend, wie eine 1:1-Kardinalität zwischen zwei Entitäten abgebildet werden konnte. Mendix hingegen stellt dem Benutzer beim Einrichten einer Beziehung eine Liste aller möglichen Kardinalitäten zur Verfügung. Power Apps verfolgt jedoch einen anderen Ansatz und verwendet keine visuellen Objekte für die Datenmodellierung. Die Daten werden an einer zentralen Stelle im Dataverse modelliert. Auch bei der Erstellung einer Beziehung zwischen zwei Tabellen gibt es keine 1:1-Kardinalität in der Liste der vorgeschlagenen Kardinalitäten. ServiceNow verwendet ebenfalls kein Entitätsdiagramm zur Modellierung der Daten. Die Struktur einer Tabelle einschließlich der Referenzattribute muss von Grund auf neu modelliert werden. Dies erfordert eine umfassende, manuelle Modellierung der Daten, da es keine Automatismen zur Erstellung von Beziehungen gibt.

Zur Anzeige der Tabelleninhalte verfügt Service Studio über eine Ansicht aller Datensätze für eine Tabelle, in der direkte Anpassungen möglich sind. Insbesondere können auf diese Weise auch Ausgangsdaten importiert werden. Mendix bietet in beiden Entwicklungsumgebungen keine Anzeige für Tabelleninhalte an. Die einzige Möglichkeit, die Daten anzuzeigen, wäre über ein Datenansichts-Widget auf einem Bildschirm. Die Einspeisung von Daten in eine bestehende Anwendung würde die Erstellung eines Microflows erfordern. In ServiceNow können Tabellen direkt über das Service Management durchsucht werden. Das Suchergebnis bietet eine Tabellenansicht mit allen Daten sowie die Möglichkeit, neue Datensätze zu erstellen. Für

Power Apps kann Dataverse verwendet werden, um Tabelleninhalte anzuzeigen. Um jedoch einen Datensatz zu einer Tabelle hinzuzufügen, muss zunächst ein Hauptformular für die Tabelle im Dataverse erstellt werden.

Variablen

Die Plattformen unterscheiden sich auch darin, wie Variablen definiert werden. In Mendix werden Variablen innerhalb eines Microflows über eine Aktivität erstellt. Sie sind nur innerhalb des Microflows sichtbar. In OutSystems werden Variablen im Sichtbarkeitsbereich eines Bildschirms außerhalb einer Aktion definiert. Der Variablenwert wird jedoch in einer Aktion zugewiesen. In den Canvas-Apps von Power Apps werden Variablen in einer Formel definiert, sodass sie in der gesamten Anwendung sichtbar sind. Zu ServiceNow kann an dieser Stelle keine Aussage getroffen werden, da bei der Anwendungsentwicklung kein Umgang mit Variablen zustande kam.

Anwendungslogik

Für die Umsetzung von Logik verwenden OutSystems und Mendix visuelle Programmiertechniken mit einer sehr geringen Codebasis. Power Apps hingegen verwendet Formeln mit Funktionen, die der Programmierung nahekommen. Im Vergleich zu anderen Plattformen verwendet ServiceNow Programmierwerkzeuge, wie echten JavaScript-Code, der in Client-Skripten und Geschäftsregeln eingesetzt wird.

OutSystems bietet standardmäßig viele Out-of-the-Box-Funktionen, wie z.B. CRUD-Operationen, die automatisch über Serveraktionen bereitgestellt werden, wenn eine Entität erstellt wird. In Mendix hingegen müssen diese Operationen manuell über Objektaktivitäten in einem Microflow implementiert werden. Power Apps bietet ähnliche Features wie OutSystems, nur mit dem Unterschied, dass diese Operationen über Formularfunktionen bereitgestellt werden.

Um Daten aus der Datenbank abzurufen, verwendet OutSystems Aggregate mit Filter- und Sortiermöglichkeiten. Die ausgeführte SQL-Abfrage eines Aggregats kann in Service Studio angezeigt, aber nicht manipuliert werden. Mendix hingegen bietet eine eigene Abfragesprache namens X-Path zum Abrufen von Daten. Sie kann als Constraints in Objektaktivitäten von Microflows verwendet werden. Darüber hinaus können innerhalb einer X-Path-Abfrage Aggregatfunktionen, wie die Berechnung des Minimums definiert werden. Eine Sortierung der zu extrahierenden Daten innerhalb einer X-Path-Abfrage ist jedoch nicht möglich. Power Apps verfügt über verschiedene

Suchfunktionen wie Filter, Lookup und Search, die in Formeln verwendet werden können. Einige Funktionen verwenden jedoch keine Delegation, sodass die Anwendung aufgrund größerer Datenmengen längere Ladezeiten haben kann[57].

REST-API-Integration

Bei der Integration einer REST-API wurde festgestellt, dass keine OpenAPI Spezifikation in Mendix importiert werden kann, um mehrere Endpunkte einer REST-Schnittstelle zu verwenden. Es ist nur möglich, einzelne Endpunkte zu konsumieren. Mit OutSystems können jedoch sowohl einzelne, als auch mehrere Endpunkte konsumiert werden. Um mehrere Endpunkte zu verwenden, ist es notwendig, eine OpenAPI-Spezifikation mit Version 3.0 zu importieren. OutSystems generiert automatisch eine Serveraktion für jeden Endpunkt aus dieser Spezifikation. In Power Apps können ähnlich wie in OutSystems mehrere Endpunkte einer REST-Schnittstelle konsumiert werden. Außerdem bietet Power Apps die Möglichkeit, einen Konnektor für den Aufruf einer REST-API aus verschiedenen Quellen zu erstellen. Für den Import einer OpenAPI-Spezifikation verwendet Power Apps jedoch eine veraltete Version, die weit hinter der aktuellen Version 3.0.1 liegt. Für ServiceNow kann keine Aussage über die Nutzung einer REST-API getroffen werden, da dies aufgrund der beschriebenen Probleme nicht untersucht werden konnte.

Drag-and-Drop

Im Großen und Ganzen ist die Drag-and-Drop-Funktionalität in alle Plattformen integriert, insbesondere bei der Gestaltung der Benutzeroberfläche. OutSystems hebt sich jedoch in diesem Punkt von den anderen Plattformen ab, indem es automatisch Bildschirme aus Entitäten generiert.

IDE-Werkzeuge

Das Debugging-Tool von Mendix und OutSystems ist dem einer klassischen IDE sehr ähnlich, bei dem alle Zustände einer Variablen oder eines Objekts aufgezeichnet werden. Power Apps hingegen überwacht vielmehr das Verhalten des Benutzers, indem es jede Aktion auf der Benutzeroberfläche und die damit verbundenen Funktionsaufrufe aufzeichnet. Dabei werden allerdings nicht alle Anweisungen in einer Formel, wie z.B. die Zuweisung eines Variablenwertes oder der Aufruf einer Benachrichtigung an einen Benutzer, aufgezeichnet. Der Skript-Debugger von ServiceNow kann mit serverseitigem JavaScript wie bei einer Geschäftsregel verwendet werden. Wie bei

klassischen IDEs wird an der entsprechenden Zeile im Code ein Breakpoint gesetzt, so dass der Debugger jede Codezeile ab dieser Zeile durchläuft und die Werte aller Variablen anzeigt.

Entwicklungserfahrung

In dieser Kategorie wird über persönliche Entwicklungserfahrungen im Umgang mit den Plattformen berichtet. Die Entwicklungserfahrung wird in diesem Zusammenhang mit den Merkmalen, wie Erlernbarkeit, Verständlichkeit, Benutzerfreundlichkeit und Einfachheit der Low-Code-Plattform definiert. Da zu Beginn keine Erfahrung mit Low-Code-Entwicklung bestand, war der Einstieg unabhängig von der Plattform als schwierig einzustufen. Mit der intensiven Beschäftigung mit verschiedenen Artikeln auf der Dokumentationsseite von OutSystems, wie z. B. der Entwicklung der ersten Anwendung oder dem Erlernen der Grundlagen von Service Studio, stieg die zu Beginn sehr flache Lernkurve deutlich an. Insbesondere bestand die größte Herausforderung darin, die Verwendung von Eingabeparametern zu verstehen, um die Verknüpfung zwischen zwei in Beziehungen stehenden Objekten herzustellen. Die erkennbare Trennung der Schichten spricht insgesamt für einen modularen Aufbau, der die Bedienbarkeit und Benutzerfreundlichkeit der Plattform erhöht. Allerdings fehlt an dieser Stelle ein klarer Einstiegspunkt, der dem Entwickler zeigt, wo er mit der Entwicklung beginnen soll. Auch die Entwicklung der ersten Client-Aktion mit visuellen Elementen war anfangs gewöhnungsbedürftig. Erst durch intensive Lernprogramme in Form von Tutorials oder durch das Lesen von Dokumentationsartikeln verbesserte sich das Verständnis. Insgesamt ist festzuhalten, dass die kompakte Unterbringung aller für die Anwendungsentwicklung erforderlichen Komponenten in einer Entwicklungsumgebung für die Einfachheit und gute Benutzbarkeit der Plattform spricht.

Mit Power Apps wurden insgesamt gute Entwicklungserfahrungen gesammelt, trotz einiger Unterschiede zu OutSystems. Das lag vor allem daran, dass zum einen die verwendeten Formeln für den Einsatz der Anwendungslogik Ähnlichkeiten mit klassischen Funktionen aus der Programmierung haben und zum anderen die Anwendungsentwicklung eine gut erkennbare Struktur und Wiedererkennung von Microsoft Office Produkten aufweist. Dennoch erforderte der Wechsel von visuellen Programmieretechniken zu formelbasierten Techniken ein Umdenken bei der Implementierung der Anwendungslogik. Insbesondere bei der Syntax von Formeln, z.B. beim Einfügen mehrerer Anweisungen in eine Bedingung, traten anfangs Probleme auf, die aber

durch hilfreiche Dokumentationsartikel und eine gute Online-Community gelöst werden konnten. Durch ausführliches Testen der Anwendung wurden jedoch neue Fehler entdeckt, da die Elemente der grafischen Benutzeroberfläche eine genaue Konfiguration erforderten. Insgesamt war die Einarbeitung unkompliziert, da die Werkzeuge für die Anwendungsentwicklung sehr verständlich waren.

Aufgrund der bisherigen Erfahrungen und der Tatsache, dass Mendix analog zu OutSystems bei der Implementierung der Anwendungslogik visuelle Programmier-techniken verwendet, war der Einstieg in Mendix Studio wenig problematisch. Dennoch war es notwendig, bei der Umsetzung der Microflows objektorientiert zu denken, da Daten hier üblicherweise als Objekte modelliert werden. Im Zuge der Integration einer REST-API musste auf Mendix Studio Pro umgestiegen werden. Dies erforderte trotz einiger Ähnlichkeiten zu Mendix Studio eine erneute Einarbeitung in eine neue Entwicklungsumgebung.

Mit ServiceNow wurde insgesamt keine gute Entwicklungserfahrung gesammelt. App Engine Studio verfolgt den Ansatz, die Anwendungsentwicklung schrittweise durchzuführen, was prinzipiell für eine gute Struktur spricht. Eine gute Nutzbarkeit der einzelnen Komponenten innerhalb dieser Schritte konnte jedoch nicht beobachtet werden. Das lag zum einen daran, dass einfache Dinge, wie das Erstellen einer Benutzeroberfläche, komplexe Konfigurationen im Service Management erforderten. Zum anderen ging durch die Verteilung dieser Komponenten auf eine große Anzahl von Diensten der Überblick über die Plattform verloren.

5.2 2. Phase: Terminanfrage

ServiceNow wurde für diese Phase aufgrund der in der ersten Phase beschriebenen Probleme und der negativen Entwicklungserfahrungen nicht berücksichtigt.

5.2.1 Ziele

Mit der ersten Phase war der Erfahrungsschatz für die Entwicklung von Low-Code-Anwendungen gewachsen, weshalb in der zweiten Phase die Komplexität des Anwendungsbeispiels erhöht wurde. In der zweiten Phase lag der Fokus vor allem auf der Benutzerauthentifizierung. Unter anderem sollte auch geprüft werden, welche Authentifizierungsmethoden in den verschiedenen Plattformen unterstützt werden.

Da viele moderne Anwendungen eine Benutzeranmeldung erfordern, war es auch wichtig herauszufinden, inwieweit die Benutzerauthentifizierung in einer Low-Code-Anwendung standardmäßig verfügbar ist. Insbesondere musste abgeschätzt werden, wie hoch der Aufwand wäre, eine alternative Authentifizierungsmethode einzusetzen. Dabei stellte sich auch die Frage, ob es eine Benutzerverwaltung gibt und wie diese funktioniert. Neben der Authentifizierung von Benutzern stellte sich in diesem Zusammenhang auch die Frage, wie Nutzerrollen und der Zugriff auf bestimmte Ressourcen gehandhabt werden.

In der Anwendung der zweiten Phase sollte eine Anwendung mit klassischem Login entwickelt werden, mit dem sich Nutzer anmelden und registrieren können. Da die meisten Anwendungen eine Anmeldung für ihre Nutzer mit unterschiedlichen Accounts anbieten, sollte auch geprüft werden, ob die Benutzerauthentifizierung in Low-Code-Anwendungen per SSO eines Identitätsanbieters wie Google integriert werden kann. Die Benutzerregistrierung und -anmeldung sowie die Einbindung des externen Identitätsanbieters Google sollten zudem auf traditionelle Weise mit Spring Boot programmiert werden, um einschätzen zu können, welche Prozesse mit einem Low-Code-Ansatz zeitsparender sind als der klassische Softwareentwicklungsprozess.

Im Rahmen der Untersuchung der Forschungsfragen sollte die Anwendung um die Funktionalität erweitert werden, dass registrierte Nutzer anderen Nutzern der Anwendung eine Terminanfrage senden können. Dabei war das Ziel herauszufinden, ob es Werkzeuge gibt, die das Laufzeitverhalten von Programmen analysieren. Abschließend sollte eine Reihe verschiedener Testmöglichkeiten untersucht und gegebenenfalls Tests entwickelt werden, um die Korrektheit der Funktionalitäten sicherzustellen.

5.2.2 Umsetzung

Mendix

Standardmäßig bietet Mendix eine Login-Funktion über das ‚Mendix SSO‘-Modul an, jedoch können sich nur Benutzer mit einem Mendix-Konto anmelden. Dies setzte voraus, dass die Anwendung in der Mendix Cloud veröffentlicht werden musste. Um auch Benutzern ohne Mendix-Account die Möglichkeit zu geben, sich anzumelden, wurde eine Template-basierte Login-Seite mit der Möglichkeit zur Registrierung eines Benutzers erstellt. Die Benutzeranmeldung war bereits in der Vorlage enthalten,

jedoch musste die Registrierung an dieser Stelle eigenständig implementiert werden. Zu diesem Zweck wurde zunächst eine nicht-persistente Entität erstellt, die die Felder eines Registrierungsformulars auf einer Registrierungsseite darstellt. An dieser Stelle war es notwendig, das Sicherheitskonzept von Mendix und die Funktionalität zum Anlegen eines Systembenutzers zu verstehen. Die Sicherheit in Mendix wird auf System-, Modul- und Anwendungsebene definiert, jedoch können Anpassungen auf Systemebene nicht vorgenommen werden. Die Anwendungssicherheit wird auf Modulebene definiert. Hier werden Nutzerrollen für das Modul definiert und der Zugriff auf Formulare, Entitäten, Microflows und Datensätze konfiguriert. In der Anwendungsebene wird unter anderem die Sicherheitsstufe festgelegt. Insgesamt gibt es drei Sicherheitsstufen. Auf der Sicherheitsstufe ‚Prototyp/Demo‘ wird die Sicherheit auf Seiten- und Microflow-Zugriff angewendet. Auf der Produktionssicherheitsstufe wird zusätzlich dazu die Sicherheit auf Entitätszugriff und Datensatzzugriff angewendet. Es besteht auch die Möglichkeit, keine Sicherheit als Sicherheitsstufe einzurichten. Weiterhin wird hier der Administrator definiert, ein anonymer und Demob Benutzer angelegt und die Passworrichtlinie festgelegt.[58]

Mit dem Erwerb dieses Wissens wurde auf der Suche nach einer Lösung zum Anlegen und Verwalten von Systembenutzern das von Mendix entwickelte Modul ‚Administration‘ gefunden, welches als Abhängigkeit der Anwendung installiert wurde. Das Domänenmodell dieses Moduls bestand aus einer einzigen Entität ‚Account‘, die eine Spezialisierungsentität der Generalisierungsentität ‚System.User‘ war. Zusätzlich stand sie mit einer nicht-persistenten Entität ‚AccountPasswordData‘ zur Erstellung von Passwortdaten in Beziehung. Auf diesem Weg war es möglich, einen Systembenutzer durch ein Account-Objekt anzulegen. Basierend auf dieser Erkenntnis wurde der Microflow aus Abbildung 5.11 zum Registrieren eines Benutzers entwickelt.

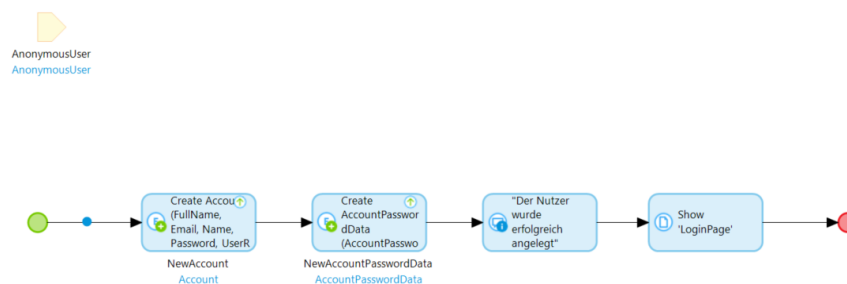


Abbildung 5.11: Microflow zum Anlegen eines Systemnutzers in Mendix

Die erste Objektaktivität erstellte ein Account-Objekt aus den Daten der nicht-persistenten Entität ‚AnonymousUser‘, die über das Registrierungsformular übergeben wurde. Das Account-Objekt wurde dann dem verknüpften ‚AccountPasswordData‘-Objekt zugewiesen, damit die Passwortdaten in der Datenbank gespeichert wurden. Um die Registrierungsseite für anonyme Benutzer zugänglich zu machen, wurde in den Sicherheitseinstellungen des Moduls eine Benutzerrolle ‚Anonymous‘ erstellt und ihr der Zugriff auf die Registrierungsseite gewährt.

Mit der Implementierung der Anmelde- und Registrierungsseite war der nächste Schritt die Integration des externen Identitätsanbieters Google. Vor der eigentlichen Implementierung war es erforderlich, die Anwendung bei den Google-API-Diensten zu registrieren. Zu diesem Zweck wurde zunächst ein neues Projekt in der Google Cloud Console erstellt und ein OAuth 2.0-Client mit einer Weiterleitungs-URI im Projekt angelegt. Die Weiterleitungs-URL sollte den Rückruf der Anwendung nach erfolgreicher Autorisierung beim Identitätsanbieter auslösen. Sie setzt sich aus der Anwendungs-URL und ‚/oauth/v2/callback‘ zusammen, so dass sie die Form ‚https://url-of-app/oauth/v2/callback‘ hat. Nachdem die Anwendung bei den Google-API-Diensten registriert wurde, wurden eine Client-ID und ein Client-Schlüssel zum OAuth 2.0-Client generiert, mit denen nur eine Authentifizierung möglich war. Schließlich kam auf Empfehlung einiger Mendix-Entwickler das ‚OIDC-Modul‘ zum Einsatz, das die Konfiguration von Google gewährleistete. Im Wesentlichen stellte das Modul eine Konfigurationsseite für einen Identitätsanbieter bereit, auf der die Client-ID und der Client-Schlüssel der erstellten Google-App sowie die Endpunkte für die OpenID-Konfiguration angegeben wurden, die von der folgenden URL importiert wurden: <https://accounts.google.com/.well-known/openid-configuration>.

Mit der Integration von Google als weiteren Identitätsanbieter wurde die Anwendung um die Funktionalität erweitert, dass ein registrierter Benutzer über ein Auswahlfeld und eine Datumsauswahl eine Terminanfrage an andere Benutzer der Anwendung sendet. Um eine Terminanfrage ausschließlich an Benutzer der Anwendung und keinen Systembenutzern senden zu können, wurde eine neue Benutzerrolle innerhalb des Moduls definiert. Des Weiteren wurde der Microflow zur Registrierung eines Benutzers um die Funktionalität erweitert, dass die neue Benutzerrolle zusätzlich dem Objekt ‚Account‘ zugeordnet wurde. Allerdings gab es eine Schwierigkeit, diese Benutzerrolle Benutzern zuzuweisen, die sich über Google SSO registriert haben. Dies lag vor allem daran, dass die Dokumentation des Moduls keine Hinweise

auf mögliche Anpassungen preisgab. Um eine Terminanfrage an mindestens einen Benutzer senden zu können, wurde das ‚Bootstrap Multi Select‘ Widget aus dem Marketplace verwendet, mit dem mehrere Benutzer in einer Dropdown-Liste ausgewählt werden können. Die Benutzeroberflächen der Terminanfrage-Anwendung werden in Abbildung 5.12 dargestellt.

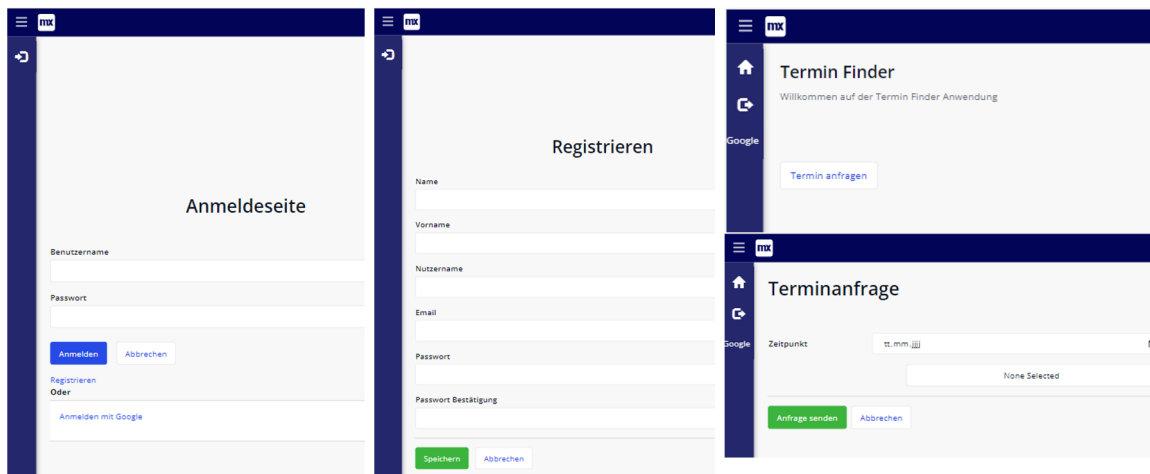


Abbildung 5.12: Benutzeroberflächen der Terminanfrage-Anwendung mit Mendix

Zur Überwachung der Performance der Anwendung bietet Mendix die Lösung ‚Application Performance Monitor‘, mit der das Laufzeitverhalten analysiert werden kann[59]. Diese Lösung setzt sich aus mehreren Werkzeugen zusammen, wobei das ‚Performance Tool‘ die Ausführungszeit einzelner Microflows misst[60]. Dabei werden alle Aktionen und SQL-Anweisungen mit ihrer Dauer aufgezeichnet, wenn die Anwendung ausgeführt wird. Diese Lösung ist standardmäßig in keiner Mendix-Anwendung enthalten, kann aber über den Marketplace installiert werden.

Mendix bietet im Allgemeinen die drei Testvarianten Komponententest, Integrationstest und Akzeptanztest über verschiedene Werkzeuge an. Für den Test von Microflows gibt es das Unit-Testing-Modul von Mendix. Für einen Integrationstest empfiehlt Mendix die Verwendung von SoapUI für Webservices. Für die Erstellung von Akzeptanztests wird die Selenium IDE empfohlen.[61]

OutSystems

In jeder OutSystems-Anwendung ist standardmäßig eine Login-Seite integriert. Eine Registrierung eines Benutzers musste an dieser Stelle jedoch eigenständig implementiert werden. Ein Benutzer wird in OutSystems durch die Entität ‚User‘ aus der

Abhängigkeit ‚System‘ repräsentiert. Um einen Benutzer anlegen zu können, wurde diese Abhängigkeit in der Anwendung installiert. Durch den Import dieser Entität wurden zusätzliche Serveraktionen zum Anlegen, Aktualisieren und Löschen eines Benutzers bereitgestellt. Für die Registrierungsseite wurde diese Entität verwendet, indem ein Formular per Drag-and-Drop daraus erstellt wurde. Das Kennwortfeld musste jedoch manuell in das Formular eingefügt werden. Um das Problem der Speicherung von Klartextkennwörtern in der Datenbank zu vermeiden, wurde die Serveraktion ‚EncryptPassword‘ aus der Abhängigkeit ‚usersLibrary‘ installiert, die Kennwörter verschlüsselt. Die daraus resultierende Aktion kann aus der folgenden Abbildung 5.13 entnommen werden.



Abbildung 5.13: Screen-Aktion zum Anlegen eines Benutzers

Als ersten Schritt wird das Erstellungsdatum zugewiesen. Darauf folgt die Verschlüsselung des Passworts, das dann dem Benutzerobjekt zugewiesen wird. Im letzten Schritt wird der Benutzer über eine bereits existierende Serveraktion angelegt.

Für die Integration von Google als Identitätsanbieter über SSO wurden einige externe Module im Forge durchsucht. Die Suche ergab die Module ‚IdP‘ und ‚Google Services OAuth 2.0‘. Service Studio analysiert vor jeder Installation von externen Modulen aus Forge, ob das Modul Fehler enthält. Beim IdP-Modul wurden diverse Fehler gefunden. Bei Google Services OAuth 2.0 konnte nur ein Bruchteil der Komponenten aus dem Modul installiert werden. Eine weitere Suche nach einer alternativen Lösung ergab eine praktische Übung von OutSystems mit einer Anleitung zur Integration von OAuth 2.0 für die Autorisierung mit Google. Dieser Ansatz wurde letztendlich bevorzugt. Vor der Implementierung war es notwendig, auch diese Anwendung in den Google API-Diensten zu registrieren. Die Weiterleitungs-URI musste jedoch in der Anwendung manuell erstellt werden. Abbildung 5.14 veranschaulicht die Modellierung des UI-Flows für den Rückruf der Anwendung nach der Autorisierung durch Google.



Abbildung 5.14: UI-Flow für den Rückruf der Anwendung nach der Autorisierung durch Google

Um den Weiterleitungs-URI in der Anwendung zu implementieren, wurde zunächst ein neuer UI-Flow, bestehend aus dem Bildschirm ‚GoogleCallback‘, erstellt. Beim Aufruf dieses Bildschirms soll die Autorisierungsseite von Google aufgerufen werden, die den Benutzer nach erfolgreicher Autorisierung wieder auf diesen Bildschirm zurückleitet. Der Benutzer soll dann in der Anwendung registriert und auf den Startbildschirm weitergeleitet werden. Da der beschriebene Weiterleitungs-URI in reaktiven Webanwendungen nicht implementierbar war, musste die ursprünglich als reaktive Webanwendung erstellte Anwendung durch eine traditionelle Webanwendung ersetzt werden. Dies lag vor allem daran, dass bei reaktiven Webanwendungen die clientseitigen Aktionen erst aufgerufen wurden, nachdem die Seite bereits gerendert worden ist. Jedoch war an dieser Stelle der Aufruf einer Aktion beim initialen Laden der Seite erforderlich. Dies wird in herkömmlichen Webanwendungen mit sogenannten ‚Preparation‘-Aktionen gelöst, die beim initialen Aufruf einer Seite aufgerufen werden. Um die beschriebene Weiterleitungslogik zu implementieren, wurde im Sichtbarkeitsbereich des GoogleCallback-Bildschirms eine Preparation-Aktion erstellt, deren Implementierung in Abbildung 5.15 dargestellt wird.

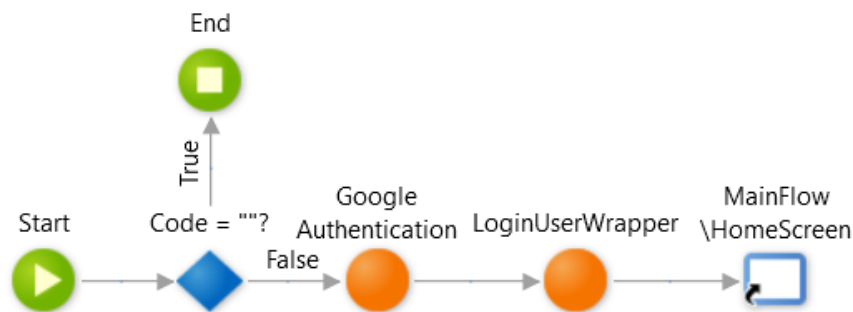


Abbildung 5.15: Preparation-Aktion zum Aufruf der Google Autorisierungsseite und zum Anlegen eines autorisierten Anwendungsbenutzers

Standardmäßig wurde nach erfolgreicher Autorisierung bei Google ein Autorisie-

rungscode vom Autorisierungsserver als Eingabeparameter an den Weiterleitungs-URI übergeben. Wenn dieser nicht vorhanden war, wurde eine Serveraktion aufgerufen, um eine Umleitung auf die Autorisierungsseite von Google unter Verwendung der Client-ID und des Client-Schlüssels auszulösen. Bei erfolgreicher Autorisierung wurde eine weitere Serveraktion aufgerufen, die den autorisierten Benutzer als Benutzer der Anwendung speicherte. Schließlich wurde eine Weiterleitung zum Startbildschirm durchgeführt.

Für Zugriffsbeschränkungen verwendet OutSystems eine rollenbasierte Zugriffskontrolle für die Anwendung, die standardmäßig durch die Benutzerrollen ‚Anonym‘ und ‚Registriert‘ geregelt wird. Mit der Benutzerrolle ‚Registriert‘ kann ein Benutzer auf die Anwendung zugreifen, wenn er sich bereits bei einer anderen Anwendung auf demselben Plattformservers angemeldet hat[62]. Um eine Terminanfrage ausschließlich an Benutzer dieser Anwendung versenden zu können, wurde eine neue Benutzerrolle definiert, mit der gleichzeitig eine Serveraktion zur Zuweisung dieser Benutzerrolle an einen Benutzer generiert wurde. Diese Serveraktion wurde sowohl in die Bildschirmaktion zum Anlegen eines Benutzers als auch in der Preparation-Aktion zum Anlegen eines bei Google autorisierten Benutzers integriert.

Um Terminanfragen an Benutzer in der Datenbank zu speichern, musste eine neue Terminanfrage-Entität erstellt und mit der Systembenutzertabelle verknüpft werden. Allerdings können Systemtabellen in OutSystems nicht angepasst werden. Daher wurde eine neue Entität erstellt, die einen Benutzer der Terminanfrageanwendung darstellt. Das Anlegen dieses Benutzers wurde ebenfalls in die genannten Aktionen integriert. Der nächste Schritt bestand darin, diese Benutzer in einer Dropdown-Liste anzuzeigen, bei der mehrere Elemente ausgewählt werden können. Eine derartige Liste zur Auswahl mehrerer Benutzer war in OutSystems standardmäßig nicht verfügbar. Es war jedoch möglich, die Daten einer Entität in Form einer Listenansicht anzuzeigen und diese durch Kontrollkästchen in jeder Tabellenzeile zu ergänzen, so dass mehrere Einträge aus der Liste ausgewählt werden konnten. In Abbildung 5.16 werden die Benutzeroberflächen der Terminanfrageanwendung dargestellt, die unter anderem die erwähnte Listenansicht veranschaulichen soll.

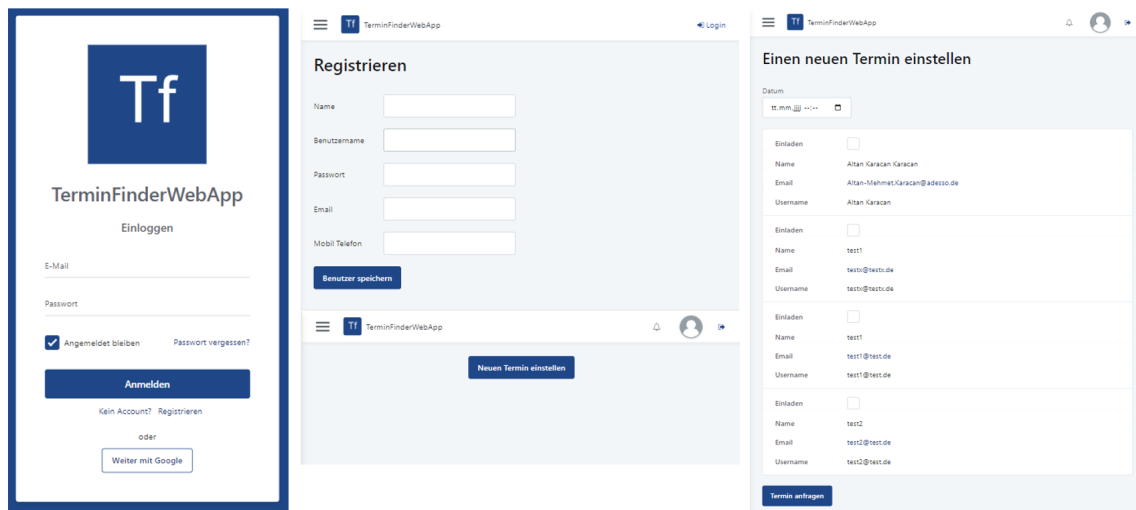


Abbildung 5.16: Benutzeroberflächen der Terminanfrageanwendung mit OutSystems

Die Liste wurde aus der Benutzerentität per Drag-and-Drop erstellt. Der Status, ob ein Benutzer eine Terminanfrage erhalten hat, wurde mit einer Struktur gelöst, die mit einem booleschen Wert abgebildet wird. Um den Status einem Benutzer zuzuordnen, wurde eine weitere Struktur angelegt, die sich aus der Statusstruktur und der Benutzerentität zusammensetzt. Der Grund für diese Lösung war, dass der Status für jede Terminanfrage an einen Benutzer variabel ist und somit nicht ein fester Bestandteil des Benutzers ist.

Für die Verwaltung der Benutzer gibt es in LifeTime bereits eine Lösung, in der alle aktiven Benutzer und die verschiedenen Benutzerrollen aus den Anwendungen des Plattformservers aufgelistet sind. Allerdings ist diese Lösung nur für Premium-Mitglieder zugänglich. Die Tabelleninhalte der Systembenutzer können jedoch im Service Studio eingesehen werden, ohne dass Anpassungen vorgenommen werden können.

In OutSystems sind verschiedene Überwachungs- und Analysewerkzeuge bereits im Service Center und LifeTime für alle Anwendungen integriert. Im Service Center erfolgt die Überwachung in Form von Überwachungsprotokollen, die häufige Fehler, langsame Abfragen, Bildschirmladezeiten für herkömmliche Webanwendungen, Anforderungszeiten für Bildschirmanforderungen, Ausführungszeiten für REST-API-Aufrufe und vieles mehr erfassen. Da die Überwachungsprotokolle für alle Module von allen Anwendungen auf dem Plattformserver erfasst werden, ist es möglich, die Protokolle nach verschiedenen Kriterien zu filtern.[63] LifeTime bietet Analysetools

in Form von Diagrammen, die über den Menüpunkt Analytics auf einem Dashboard angezeigt werden können. Insgesamt zeigt das Dashboard Leistungsprobleme pro Webseite oder Bildschirmaktion, Leistungsprobleme bei bestimmten Netzwerkbedingungen und Server-Antwortzeiten bei langsamen Abfragen. Auf der Grundlage dieser Metriken wurde ein abschließendes Diagramm zur Nutzerzufriedenheit erstellt, das anhand eines Punktesystems den Verbesserungsbedarf der einzelnen Metriken aufzeigt.[64] Das Analysetool im LifeTime ist ebenfalls nur für Premium-Mitglieder verfügbar.

Wie bei klassischen Webanwendungen können auch für die mit OutSystems entwickelten Webanwendungen Komponenten-, Integrations- und API-Tests durchgeführt werden. In der Forge gibt es verschiedene Anwendungen wie ‚TestFramework‘ für API-Tests sowie ‚Unit Testing Framework‘ für Komponententests. Für Komponententests empfiehlt OutSystems jedoch das ‚BDD-Framework‘, das im Rahmen der Forschungsfrage näher untersucht wurde[64]. Bei der verhaltensgesteuerten Entwicklung werden die Anforderungen und Ergebnisse für eine Software in einem verständlichen Text niedergeschrieben und später mit Unit-Tests validiert. Der Test sollte die Korrektheit des ‚LoginUserWrapper‘ aus der Preparation-Aktion von Abbildung 5.15 sicherstellen. Abbildung 5.17 zeigt das Ergebnis des Komponententests mit dem BDD-Framework.

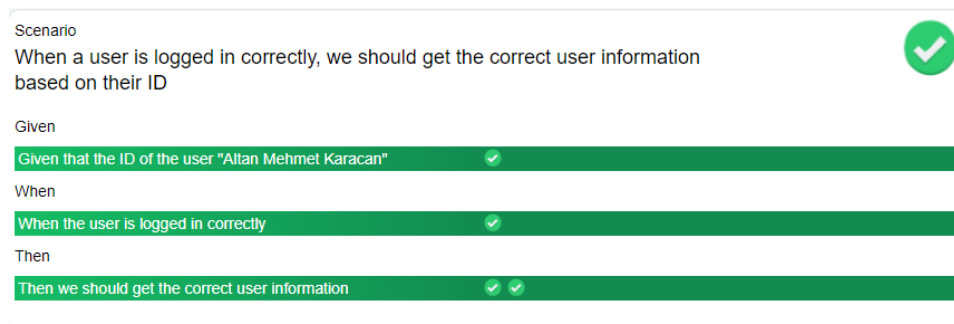


Abbildung 5.17: Ergebnis des BDD-Tests in OutSystems

Der Anforderungstext wurde in diesem Framework mit Szenarien definiert. Mit dem definierten Szenario aus Abbildung 5.17 sollte sichergestellt werden, dass die Registrierung eines Google-Benutzers mit der Serveraktion LoginUserWrapper aus der Preparation-Aktion in Abbildung 5.15 keinen neuen Benutzer in der Anwendung erstellt, wenn dieser Benutzer bereits existiert. Zu diesem Zweck wurden der Identifikator des vorhandenen Benutzers und der von der Anmeldefunktion zurückgegebene

Identifikator verwendet, um schließlich die Gleichheit der Benutzerinformationen zu überprüfen. Wie bei klassischen Modultests wurde auch hier die Given-When-Then-Methode verwendet. Für jeden dieser drei Zustände wurden sowohl die textliche Beschreibung als auch eine Bildschirmaktion erstellt, die diesen Zustand implementiert. Abbildung 5.18 zeigt die zugehörigen Bildschirmaktionen der jeweiligen Zustände.

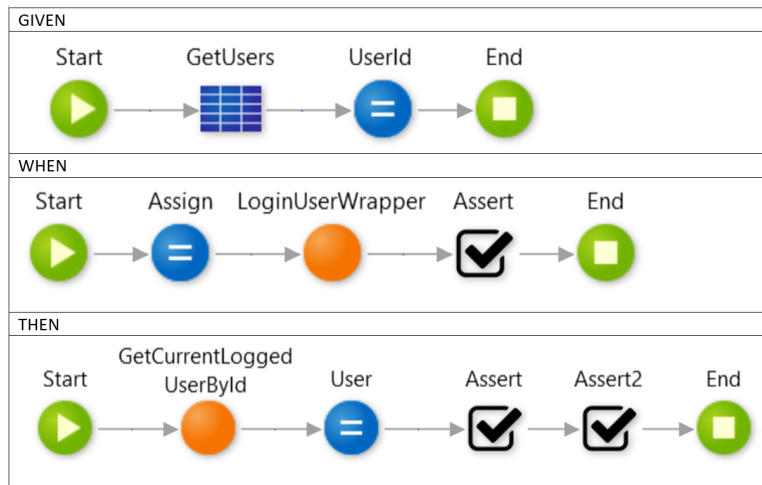


Abbildung 5.18: Given-When-Then-Bildschirmaktionen

Für den Zustand ‚Given‘ wurde ein beliebiger Benutzer aus der Datenbank extrahiert und sein eindeutiger Identifikator einer lokalen Variablen zugewiesen. Die Serveraktion LoginUserWrapper aus dem ‚When‘-Zustand erwartete Token-Informationen wie das Zugriffstoken und das Refresh-Token als Parameter, die vom Autorisierungsserver nach erfolgreicher Autorisierung zurückgegeben werden, um die Google-Profilinformationen eines Benutzers zu extrahieren. Die Token-Informationen des beliebigen Benutzers wurden auf klassischem Wege ermittelt und als Mock-Objekt für den Aufruf der Server-Aktion LoginUserWrapper verwendet, die als Rückgabewert den Identifikator des mit den Token-Informationen zugehörigen Benutzers zurückgibt. Ein Assert wurde verwendet, um den zurückgegebenen Identifikator auf Gleichheit mit dem Identifikator der lokalen Variablen zu vergleichen. Im abschließenden ‚Then‘-Zustand wurden die Benutzerinformationen wie der Name und die E-Mail-Adresse des Benutzers, die durch den zurückgegebenen Bezeichner der Serveraktion LoginUserWrapper bestimmt werden, mit denselben Benutzerinformationen des beliebig ausgewählten Benutzers verglichen.

Power Apps

Mit der Perspektive, eine Anwendung nach außen sichtbar zu machen und einen externen Identitätsanbieter wie Google zu integrieren, waren Apps-Portale die einzige in Frage kommende Lösung, da sowohl Canvas-Apps als auch modellgesteuerte Apps nur innerhalb einer Organisation sichtbar sind. Ein Portal wird immer mit einer grundlegenden Website-Struktur erstellt. Die Grundstruktur umfasst unter anderem eine Anmelde- und Registrierungsseite, deren Funktionalität ebenfalls implementiert ist. Auch die Integration eines Identitätsanbieters war bereits standardmäßig vollständig im Portal implementiert und musste nur noch konfiguriert werden. In den Einstellungen der Portal-App unter den Authentifizierungseinstellungen wurde eine Liste mit einigen Identitätsanbietern, einschließlich Google, aufgeführt. Bei der Auswahl eines Identitätsanbieters mussten nur die Client-ID und der Client-Schlüssel des OAuth 2.0-Clients eingegeben werden. Zu diesem Zweck wurde das Portal auch bei den Google-API-Diensten registriert, indem wiederum ein OAuth 2.0-Client erstellt wurde, dessen Weiterleitungs-URI aus der URL der Portalanwendung gefolgt von `./signin-google` bestand. Bei erfolgreicher Konfiguration des Identitätsanbieters wurde automatisch eine Schaltfläche zur Weiterleitung auf die Google-Autorisierungsseite in der Anmeldeseite eingefügt.

Die Sicherheit einer Portalanwendung wird in der Portal Management App verwaltet. Im Abschnitt `Security` werden alle authentifizierten Benutzer unter `Contacts` aufgeführt. Es ist möglich, direkte Anpassungen an einem Benutzer vorzunehmen. Der allgemeine Zugriff auf eine Seite wird mit Webrollen geregelt. Standardmäßig existieren die Webrollen `Administratoren`, `Anonyme Benutzer` und `Authentifizierte Benutzer`, die jedoch um neue Webrollen erweitert werden können. Eine neu erstellte Seite des Portals ist anfangs für alle zugänglich. Um sie jedoch einzuschränken, ist es notwendig eine neue Zugriffskontrollregel für diese Seite zu erstellen. Die Zugriffskontrollregel konfiguriert die Zugriffskontrolle für eine bestimmte Webseite der Portalanwendung und definiert die Webrollen, die auf diese Seite zugreifen können.

Der nächste Schritt war die Implementierung der Terminanfragefunktion. Power Apps Portal Studio wurde für die Gestaltung der Portalseiten verwendet, um eine neue Terminanfrage-Seite zu erstellen. Für diese Seite wurde auch eine Zugriffskontrollregel erstellt, die authentifizierten Benutzern den Zugriff gewährt. Außerdem wurde eine neue Tabelle zur Speicherung von Terminen in Dataverse erstellt, die mit der Tabelle `Contacts` verknüpft wurde, damit authentifizierte Benutzer eine

Terminanfrage erhalten können. An dieser Stelle war es notwendig, das Formular zum Anlegen eines Termins in Dataverse zu modellieren, das auf der Portalseite angezeigt werden sollte. Allerdings gab es das Problem, dass trotz der Modellierung kein Formularfeld für die Auswahl eines Benutzers im Formular auf der Seite angezeigt wurde. Abbildung 5.19 zeigt die Benutzeroberflächen der Portalanwendung.

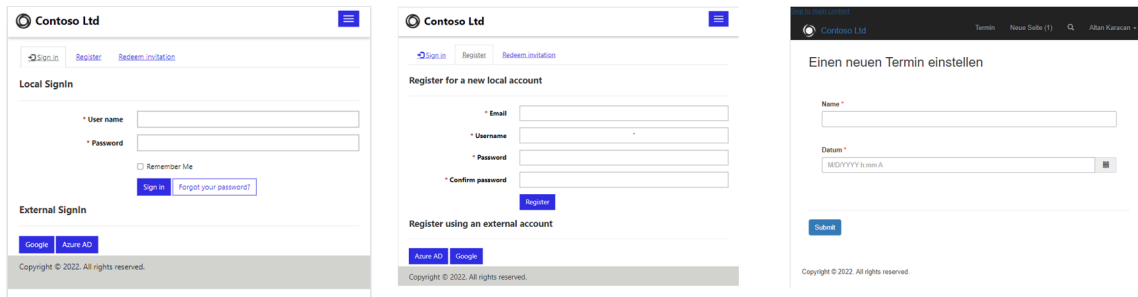


Abbildung 5.19: Benutzeroberflächen der Portal-Anwendung mit Power Apps

Für Power Apps-Portallösungen wurden keine Überwachungstools gefunden. Das Monitoring-Tool von Power Apps kann nur für Canvas-Apps und modellgesteuerte Apps verwendet werden. Abgesehen von der Erfassung der Dauer eines Ereignisses gab es in diesem Tool keine Möglichkeit, das Laufzeitverhalten zu analysieren.[65]

Power Apps verfügt über ein Tool namens ‚Test Studio‘, das ausschließlich für End-to-End-Tests der Benutzeroberflächen von Canvas-Apps verwendet werden kann[66]. Zur Evaluation des Tools wurden einige Tests für die Canvas-App aus der ersten Phase durchgeführt. Ein Testfall besteht aus mehreren Testschritten, in denen Aktionen definiert sind, die auf der Benutzeroberfläche ausgeführt werden. Abbildung 5.20 zeigt einen Beispieltest, der sicherstellt, dass sowohl der Antragsteller als auch der Fahrzeughalter nicht in der Datenbank enthalten sind.

5 Umsetzung und Evaluation

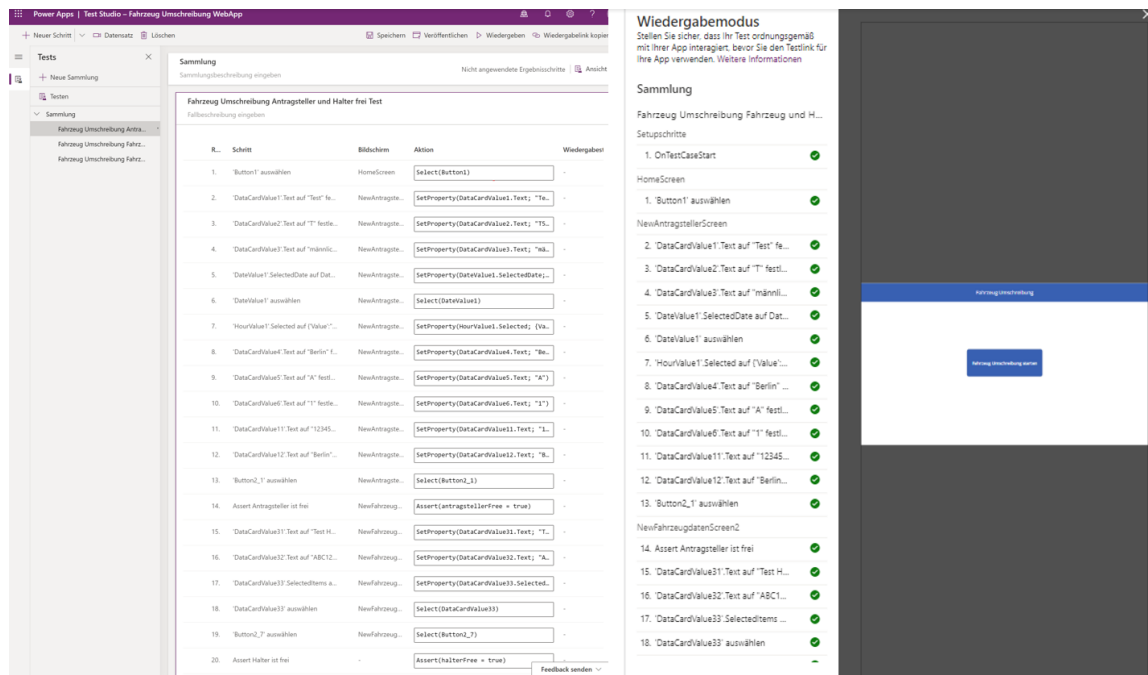


Abbildung 5.20: Beispiel eines End-to-End-Tests in Power Apps Test Studio

Eine Aktion wird in Form von Funktionen in der Power Apps Ausdruckssprache geschrieben. Um einen Test zu validieren, werden Testassertionen geschrieben, um das erwartete Ergebnis mit dem tatsächlichen Testergebnis abzugleichen.

5.2.3 Evaluation

Zeitersparnis / Zeitverlust

Abbildung 5.21 zeigt den Zeitvergleich für die Implementierung verschiedener Funktionalitäten der Anwendung mit den verschiedenen Low-Code-Plattformen und der klassischen Softwareentwicklung mit dem Spring Framework. Die Zeit, die in die Einrichtung der Anwendung und die Suche nach Lösungen investiert wurde, wurde bei dem Zeitvergleich ebenfalls berücksichtigt.

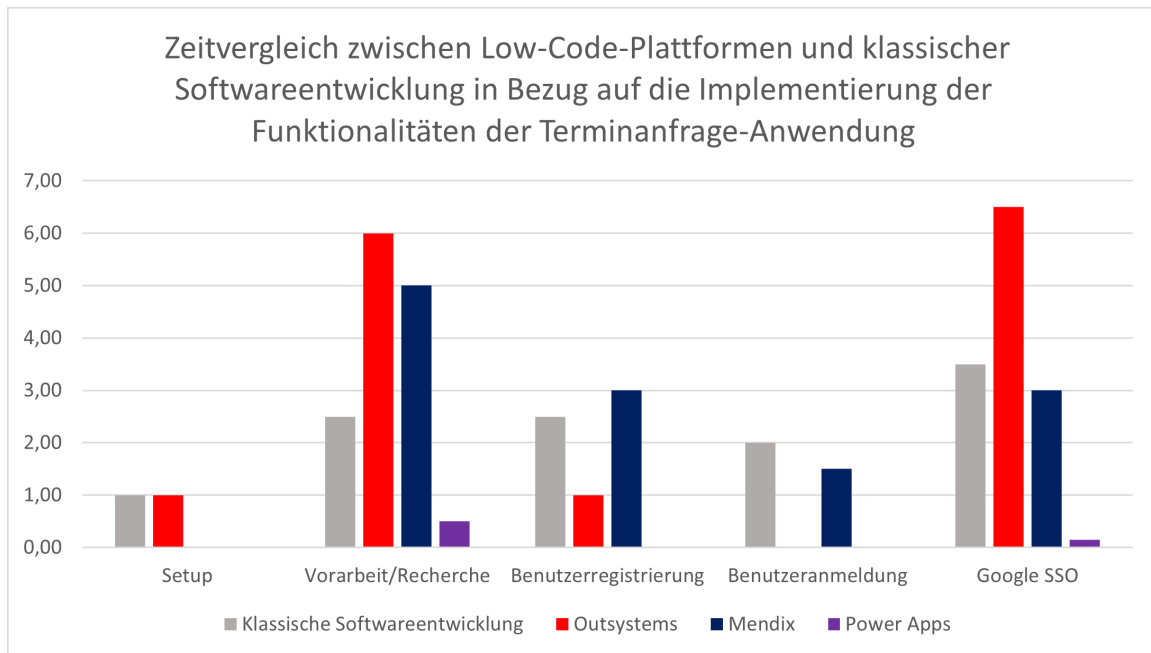


Abbildung 5.21: Zeitvergleich zwischen Low-Code-Plattformen und klassischer Softwareentwicklung in Bezug auf die Implementierung der Funktionalitäten der Terminanfrage-Anwendung

Im Durchschnitt benötigten die Low-Code-Plattformen in etwa neun Stunden für die Implementierung der Anforderungen und waren damit insgesamt zeiteffizienter als die klassische Softwareentwicklung mit knapp zwölf Stunden. Bei der Einzelbetrachtung lag Power Apps mit einem größeren Unterschied an der Spitze, gefolgt von der klassischen Softwareentwicklung. Mendix war nur geringfügig zeitaufwändiger als die klassische Softwareentwicklung. Den höchsten Zeitaufwand hatte jedoch OutSystems aufgrund von Vorarbeiten und der eigenständigen Implementierung der OAuth2.0-Authentifizierung.

Die Einrichtung der Anwendung mit Low-Code-Plattformen, mit Ausnahme von Outsystem, erforderte im Vergleich zur klassischen Softwareentwicklung nur wenige Schritte, da nach der Erstellung keine weitere Konfiguration erforderlich war. Da bei Outsystem ein Wechsel von der reaktiven zur traditionellen Webanwendung notwendig war, kam es zu einer Verzögerung. Der klassische Entwicklungsansatz war bei der Einrichtung der Anwendung ebenfalls nicht sehr zeitaufwändig, da ein vorgefertigtes Tool wie ‚Spring Initializr‘ verwendet wurde, um die Anwendung mit ihren Abhängigkeiten zu erstellen. Darüber hinaus musste eine MySQL-Datenbank angelegt und eine JDBC-Verbindung eingerichtet werden, was ebenfalls kaum Zeit in

Anspruch nahm. Insgesamt wurden hier nur sehr geringe Zeitunterschiede zwischen den beiden Entwicklungsansätzen festgestellt.

Bei OutSystems und Mendix war es zeitaufwändig, bestehende Lösungen für die Implementierung der OAuth2.0-Authentifizierung zu recherchieren und Kenntnisse darüber zu erlangen, wie Berechtigungen in Low-Code-Anwendungen geregelt werden. Bei der klassischen Softwareentwicklung verlief die Recherche wesentlich besser, da es eine Vielzahl von Lösungen gab, so dass eine Lösung nach Belieben ausgewählt werden konnte. Im Vergleich dazu gab es für die Low-Code-Plattformen nur wenige Lösungen.

Die Benutzeranmeldung war in allen Low-Code-Plattformen standardmäßig enthalten. Allerdings musste die Benutzerregistrierung in allen Plattformen außer Power Apps manuell implementiert werden. Im Vergleich dazu wurde für Spring Boot keine Abhängigkeit verwendet. Trotz der manuellen Implementierung eines Datenmodells, der Benutzerrollen, der Anwendungssicherheit und der Anmelde- und Registrierungslogik war der Aufwand hier geringer als bei Mendix.

Trotz der Anleitung war die Implementierung der OAuth2.0-Authentifizierung in OutSystems komplex und zeitaufwändig, da sie vollständig implementiert werden musste. An dieser Stelle ist es jedoch wichtig zu erwähnen, dass die externen Module, die in der reaktiven Webanwendung nicht funktionierten, nach der Änderung des Anwendungstyps nicht erneut untersucht wurden. Im Fall von Mendix war die Implementierung der OAuth2.0-Authentifizierung trotz weniger Konfigurationen nur gering zeitaufwändiger als mit Spring Boot. Einer der Gründe dafür war, dass die Dokumentation des Moduls an einigen Stellen nicht ausführlich genug war. Für den klassischen Weg wurde eine Abhängigkeit in Spring Boot verwendet, mit der nur wenig Code benötigt wurde, um das Feature zu implementieren. Mit Ausnahme von OutSystems benötigten die Low-Code-Plattformen insgesamt weniger Zeit für die Implementierung dieser Funktion als die traditionelle Softwareentwicklung.

Laufzeitverhalten

Im Wesentlichen verfügen alle drei Low-Code-Plattformen über Monitoring-Tools, mit denen die Ausführungszeiten von Aktionen gemessen werden können. Im Gegensatz zu Mendix verwendet OutSystems eine serverseitige Lösung, so dass kaum Overhead in der Anwendung entsteht. Über die Effektivität der Monitoring-Tools von OutSystems und Mendix kann an dieser Stelle keine Aussage getroffen werden, da sie in dieser Phase nicht eingesetzt wurden.

Weiterentwicklung

Die Weiterentwicklung der Anwendung um die Funktionalität der Terminanfrage war auf allen Plattformen erfolgreich. Sowohl Mendix als auch Power Apps hatten jedoch die generelle Schwierigkeit, dass einem Benutzer von einem externen Identitätsanbieter nur manuell eine Benutzerrolle zugewiesen werden konnte. Hier war OutSystems klar im Vorteil, da direkte Anpassungen an den Aktionen möglich waren.

Test-Frameworks

Sowohl Mendix als auch OutSystems verfügen über Test-Frameworks, die für Unit-Tests verwendet werden können. Das BDD-Framework von OutSystems kann als Modul in eine Anwendung integriert werden. Ein Test mit diesem Framework kann nur für öffentliche Serveraktionen einer Anwendung durchgeführt werden. Es verfügt über visuelle Werkzeuge zur Erstellung der erforderlichen Testkomponenten, so dass die Testdurchführung nur wenige Schritte erfordert. Insgesamt war es ein hilfreiches Framework, um die Korrektheit wichtiger Funktionalitäten der Anwendung sicherzustellen. Power Apps hingegen verfügt über ein Test-Framework für End-to-End-Tests, das bereits für alle Canvas-Anwendungen verfügbar war. Es ist möglich, Interaktionen mit der Anwendung aufzuzeichnen und als Grundlage für einen Testfall zu verwenden, so dass der Aufwand für die Testdurchführung als gering eingestuft werden kann. Insgesamt war es auch ein hilfreiches Framework, das einige neue Fehler in der Anwendung aufdeckte.

Wiederverwendbarkeit

Sowohl Mendix als auch OutSystems bieten auf ihren App-Marktplätzen eine Vielzahl von wiederverwendbaren Lösungen an, die sowohl für die Gestaltung der Benutzeroberfläche, als auch für wiederkehrende Probleme verwendet werden können. Es kann jedoch nicht ausgeschlossen werden, dass Lösungen von Drittanbietern fehlerhaft sind. Es wird daher empfohlen, Lösungen von offiziellen Entwicklern von Low-Code-Plattformen oder von den Plattformen empfohlene Lösungen zu verwenden. In der Power Apps-Portallösung können dagegen nur die bereitgestellten Tools verwendet werden. Insgesamt ist die Power Apps-Portallösung weniger flexibel in Bezug auf wiederverwendbare Lösungen.

5.3 3. Phase: Bewerbermanagement

5.3.1 Ziele

Mit der dritten Phase der Evaluation stieg die Erfahrung in der Low-Code-Entwicklung, weshalb die Komplexität des letzten Anwendungsbeispiels erhöht wurde. Da die Automatisierung von Geschäftsprozessen ein Anwendungsfall für die Low-Code-Entwicklung ist, wurde in der dritten Phase eine solche Anwendung umgesetzt. In diesem Anwendungsfall ging es vor allem darum, administrative Aufgaben in einem Unternehmen durch automatisierte digitale Lösungen zu ersetzen, um die betriebliche Effizienz zu steigern. Zu diesen administrativen Aufgaben gehört unter anderem die Pflege großer Mengen von Unternehmensdaten, die in unterschiedlichen Dateiformaten gespeichert sind.

In dieser Phase sollte ein webbasiertes Bewerbermanagement-Tool entwickelt werden, der potenziell im Personalwesen eingesetzt werden kann. Dabei sollte die Anwendung die Möglichkeit bieten, größere Datenmengen in die Anwendung zu importieren und tabellarisch darzustellen. Außerdem sollte es möglich sein, bestehende Daten zu pflegen und neue Daten anzulegen. Ein weiteres Ziel war es, zu untersuchen, ob aus den Daten automatisch aussagekräftige Dashboards mit Kennzahlen, Diagrammen und Berichten generiert werden können. Da größere Datenmengen oft ein komplexeres Datenmodell erfordern, sollte an dieser Stelle geprüft werden, inwieweit sich komplexe Datenmodelle umsetzen lassen. In der dritten Phase war außerdem von Interesse, ob die Funktionalitäten der Anwendung durch benutzerdefinierten Code erweitert werden können. Darüber hinaus sollte geprüft werden, ob die Anwendung auch mit externen Versionskontrollsystemen verknüpft werden kann. Dabei war unter anderem von Interesse, ob eine CI/CD-Pipeline bereits von den Plattformen bereitgestellt wird und wie aufwändig es wäre, eine Pipeline mit externen DevOps-Tools zu erstellen.

5.3.2 Umsetzung

Power Apps

Ein komplexes Datenmodell einer Anwendung konnte so dargestellt werden, dass alle möglichen Kardinalitäten im Datenmodell abgebildet wurden und möglichst viele Entitäten aus dem Datenmodell miteinander verknüpft wurden. Für die Darstellung

eines Bewerbungsprozesses wurde ein solches komplexes Datenmodell modelliert, das in Abbildung 5.22 dargestellt ist.

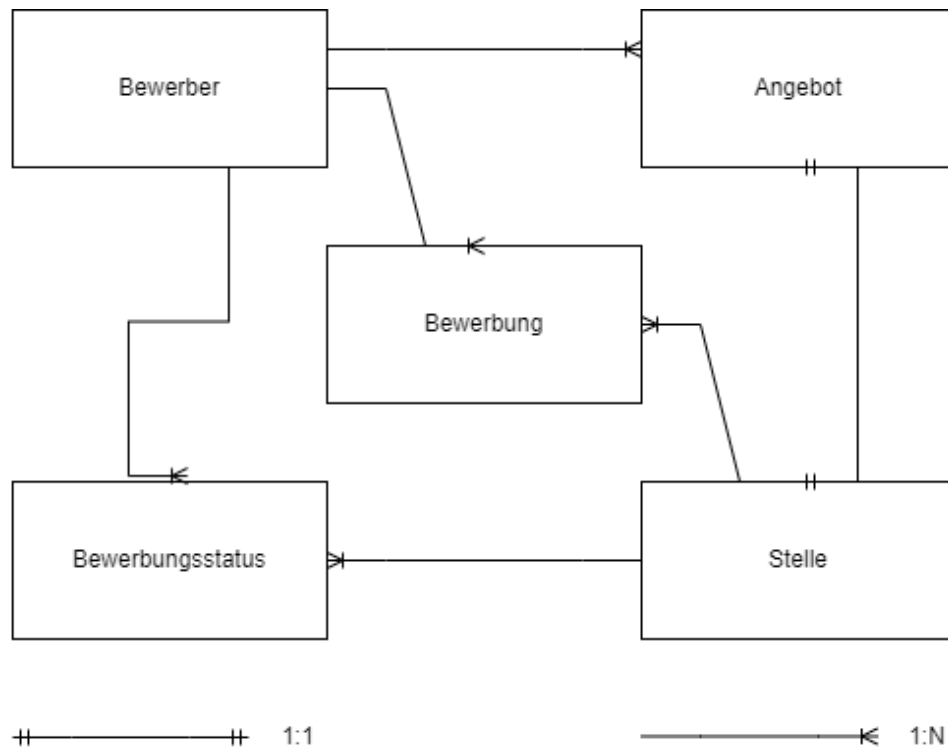


Abbildung 5.22: Datenmodell der Bewerber-Management-Anwendung

Ein Bewerbungsprozess wurde dadurch initiiert, dass sich ein Bewerber auf eine oder mehrere Stellen in verschiedenen Abteilungen bewirbt. Nach Eingang der Bewerbung wurde dem Bewerber ein Bewerbungsstatus zugewiesen. Damit wurde festgehalten, ob bereits ein Telefoninterview, ein Vorstellungsgespräch oder ein Angebot erfolgt war. Schließlich wurde ein Angebot unterbreitet, wenn der Bewerber die Anforderungen für die Stelle erfüllte. Auf der Grundlage dieses Datenmodells wurden anhand von Beispieldaten sowohl die Bewerber mit ihren persönlichen Informationen, als auch die Stellen mit den zugehörigen Informationen in Excel-Dateien abgebildet, die für den Import verwendet wurden.

Zu Beginn wurde das Datenmodell in Dataverse umgesetzt. Es galt zunächst herauszufinden, wie das aus der ersten Phase bekannte 1:1-Kardinalitätsproblem zu lösen ist. Nach längerer Recherche und durch die Unterstützung der Online-Community von Power Apps stellte sich heraus, dass diese Kardinalität so umgesetzt werden kann, dass in einer der beiden miteinander verknüpften Tabellen eine neue Spalte angelegt

und ihr der Datentyp ‚Suche‘ zugewiesen werden muss. Diese Spalte wurde dann als Suchfeld im Formular abgebildet. Für alle anderen Kardinalitäten gab es bereits eine fertige Lösung über das Dropdown-Menü. Für die n:m-Beziehung zwischen den Tabellen ‚Bewerber‘ und ‚Stelle‘ war es nicht notwendig, die zusätzliche Tabelle ‚Bewerbung‘ manuell anzulegen, da sie von Dataverse bereits automatisch erstellt wurde.

Nachdem das Datenmodell implementiert war, bestand der nächste Schritt darin, die Tabellen mit Daten aus der erstellten Excel-Datei zu befüllen. Dataverse bietet an dieser Stelle zwei Optionen für einen Import an. Bei der ersten Option können die Daten aus verschiedenen Konnektoren wie einer Excel-Datei oder einer PostgreSQL-Datenbank importiert werden. Ist der Import erfolgreich, kann der Benutzer anschließend Attribute aus einer Tabelle manuell einem Datensatz zuweisen. Bei der zweiten Option wird eine Excel-Datei, die bereits die Dataverse-Tabellenstruktur enthält, auf das lokale Dateisystem heruntergeladen, die dann manuell mit Daten befüllt werden muss. Diese enthält eine Authentifizierungsfunktion, um sich mit der persönlichen Power Apps Umgebung zu verbinden. Schließlich werden die Daten über einen Veröffentlichungsbutton importiert. Die erste Option war fehlerhaft, da es nicht möglich war, die Daten einigen Attributen der Tabelle zuzuordnen. Bei der zweiten Option hat der Import schließlich funktioniert.

Nach der erfolgreichen Schaffung einer Datenbasis konnte die Anwendung erstellt werden. Dazu war es notwendig, im Vorfeld herauszufinden, welcher Anwendungstyp in Power Apps die Anforderungen der Anwendung am besten erfüllen kann. Nach einem genaueren Vergleich der verschiedenen Anwendungstypen fiel die Wahl auf die modellgetriebene Anwendung. Ein Grund für die Entscheidung war, dass sich damit Prozesse mit höherer Datendichte besser verwalten lassen. Zum anderen harmonisieren modellgetriebene Anwendungen sehr stark mit Dataverse, so dass Formulare, Diagramme und Tabellenansichten am einfachsten integriert werden können.

Für die Implementierung der Anwendung wurde der klassische App Designer verwendet. Dabei werden die verschiedenen Komponenten der Anwendung innerhalb der Strukturen des App Designers entworfen, so dass dieser automatisch eine Anwendung aus dem Entwurf generiert. In der Siteübersicht wurde für jede Entität aus dem Datenmodell ein Navigationselement erstellt, das über eine Sidebar in der Anwendung aufgerufen werden kann. In der Entitätsansicht wurden alle zugehörigen Tabellen des Bewerbungsprozesses aus Dataverse hinzugefügt, deren zugehörige Formulare, Ansichten und Diagramme in der Anwendung über die entsprechenden

Sidebar-Elemente aufgerufen werden konnten. Hierfür war es jedoch notwendig, die Formulare, Ansichten und Diagramme der zugehörigen Tabellen mit den von Dataverse bereitgestellten Lösungen zu gestalten. Jede Lösung verfügt über eine eigenständige Ansicht. Formulare werden in einer Strukturansicht gestaltet. Hier werden alle Spalten der Tabelle angezeigt, so dass die Auswahl einer Spalte automatisch das Formularfeld im Formular erzeugt. Anpassungen wie die Umbenennung des entsprechenden Formularfeldnamens oder die Anordnung der Formularfelder sind möglich. Auf diese Weise wurden die Formulare für die Tabellen erstellt, um neue Datensätze zu erstellen. Für Tabellen, die mit einer anderen Tabelle verknüpft sind, z.B. in einer 1: n-Beziehung, kann das entsprechende Formularfeld jedoch nur über eine Unterraster-Komponente in das Formular integriert werden. Eine Tabellendatenansicht wird hier als Listenansicht mit dem gleichen Ansatz wie bei der Formulargestaltung entworfen.

Dataverse ermöglichte ebenso die Erstellung von grafischen Darstellungen der Datensätze einer Tabelle, z. B. in Balken-, Flächen-, Liniendiagrammen. Als Beispiel wurde ein Kreisdiagramm erstellt, das die Anzahl der verschiedenen Abteilungen von allen offenen Stellen in Kreissektoren zeigt. Außerdem wurde ein Säulendiagramm erstellt, das in den Säulen die Anzahl aller angenommenen oder abgelehnten Angebote zeigt. Abbildung 5.23 zeigt die Ansicht zur Erstellung von Diagrammen in Dataverse.

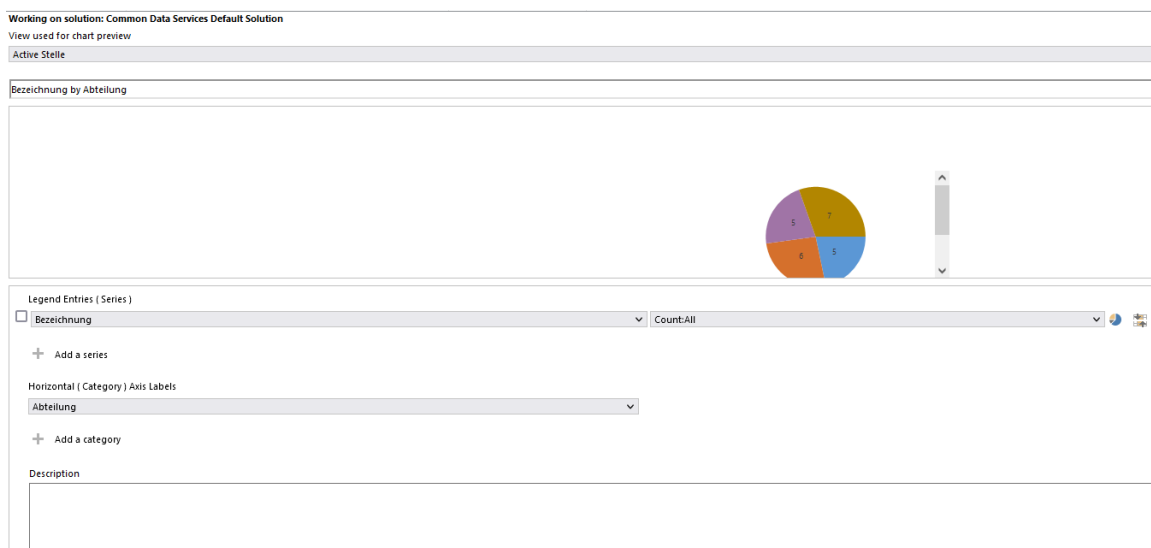


Abbildung 5.23: Ansicht zur Erstellung eines Diagramms in Dataverse

5 Umsetzung und Evaluation

Um ein Diagramm zu erstellen, musste zunächst die Ansicht der Tabelle ausgewählt werden. Außerdem musste die Tabellenspalte angegeben werden, die für Legendeneinträge verwendet werden soll. Auf diesen Datensatz dieser Spalte konnte eine Aggregatfunktion angewendet werden. Für die Beschriftung der horizontalen Achse wurde ebenfalls eine Tabellenspalte angegeben.

Die Integration von benutzerdefiniertem Code in modellgesteuerte Anwendungen ist in Form von clientseitigem JavaScript möglich, welches bei einem Formularereignis wie z.B. ‚onLoad‘ aufgerufen wird. Clientseitiges Scripting kann z. B. als Geschäftsregel zur Validierung von Formularfeldern verwendet werden.[67] Allerdings gibt es hier Einschränkungen, wie z.B. die Einbindung eines IFRAME-Formulars in ein von Dataverse bereitgestelltes Formular[68]. Aufgrund dieser Erkenntnis und des Fehlens eines clientseitigen Anwendungsfalls wurde keine Kennzahlenberechnung über benutzerdefinierten Code durchgeführt.

Nach der endgültigen Gestaltung und Veröffentlichung der Komponenten wurde die Bewerbermanagement-Anwendung automatisch generiert, deren Benutzeroberflächen in Abbildung 5.24 dargestellt sind.

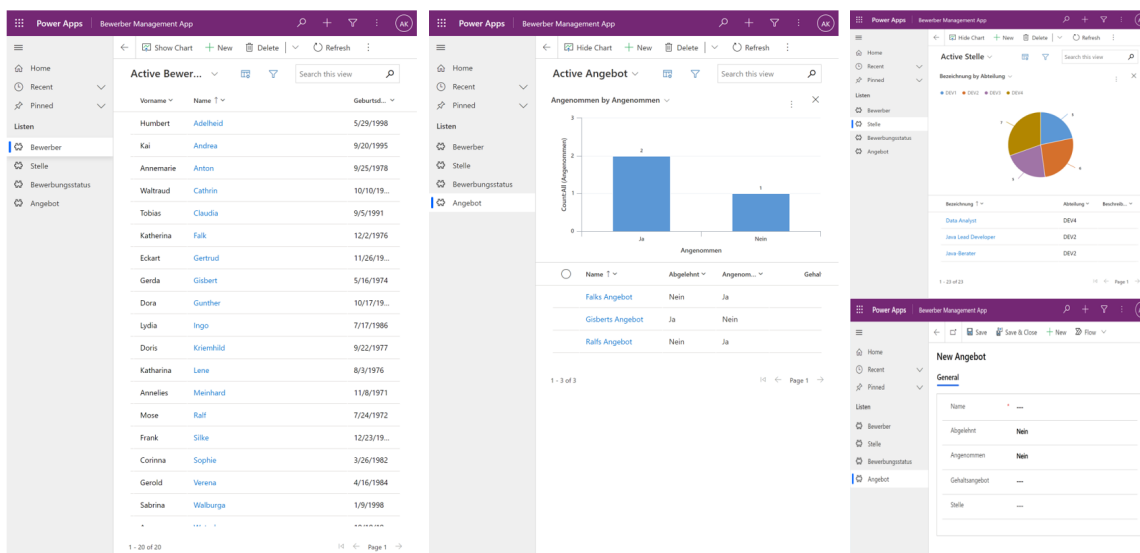


Abbildung 5.24: Benutzeroberflächen der Bewerbermanagement-Anwendung mit Power Apps

Die Listenansichten der verschiedenen Tabellen konnten über die Einträge in der Seitenleiste aufgerufen werden. Eine Anpassung der Listenspalten, sowie eine Filterung der Daten wurde hier ebenfalls unterstützt. Bei Auswahl eines Eintrags aus der Liste konnten verknüpfte Daten eingesehen oder neue Verknüpfungen erstellt werden.

Weiterhin war es möglich, über die erstellten Formulare neue Datensätze anzulegen oder bestehende zu entfernen. Auch die Anzeige von erstellten Diagrammen war hier möglich.

Eine CI/CD-Pipeline kann mit Azure DevOps erstellt werden. Der Visual Studio Code Marketplace bietet viele Power Platform-spezifische Erweiterungen für Azure DevOps, die Build- und Bereitstellungsaufgaben automatisieren. Diese Erweiterungen helfen unter anderem dabei, eine Verbindung mit der persönlichen Power Platform-Umgebung herzustellen.[69]

OutSystems

OutSystems ermöglicht die Erstellung von Tabellen in Service Studio durch den Import einer Excel-Datei. Beim Import der Beispieldaten für Bewerber und Stellen wurden die beiden Tabellen mit ihren Strukturen und den Datensätzen automatisch angelegt. Alle anderen Entitäten des Datenmodells wurden manuell angelegt. Die vollständige Implementierung des Datenmodells, insbesondere der Beziehungen zwischen den Tabellen, erforderte zunächst die Untersuchung der Darstellung der Kardinalitäten 1:1 und n:m im Entitätsdiagramm. Die Untersuchung ergab, dass im Falle der n:m-Beziehung zwischen den beiden Tabellen ‚Stelle‘ und ‚Bewerber‘ die zusätzliche Tabelle ‚Bewerbung‘ manuell angelegt werden muss. Eine 1:1-Beziehung wurde dargestellt, indem dem eindeutigen Identifikator der verknüpften Tabelle ein ‚Entity Identifier‘ als Datentyp zugewiesen wurde.

Durch Drag-and-Drop der Tabellen in den MainFlow aus der Benutzerschnittschicht wurden automatisch die zugehörigen Detailseiten für die Erstellung eines Datensatzes sowie Listenansichten erstellt. Darüber hinaus wurde ein Dashboard erstellt, das aussagekräftige Diagramme und Kennzahlen anzeigt. Die Kennzahlen stellten die Anzahl der angenommenen, abgelehnten und offenen Angebote dar. Ihre Berechnung erfolgte mit einer mit Integration Studio erstellten Aktion, die als Erweiterung in Service Studio installiert wurde. Der zugehörige Code der Aktion wurde mit Visual Studio Code entwickelt. Ursprünglich sollte diese Aktion als Eingabeparameter die Liste aller Angebote erhalten, die zur Berechnung der Kennzahlen verwendet wird. Da in Integration Studio für die Parameter nur generische Datentypen vergeben werden können, war der einzig mögliche Datentyp ‚List Record‘, um die Liste aller Angebote darzustellen. Dieser Datentyp erforderte jedoch die zusätzliche Definition der entsprechenden Entität. Es war auch nicht möglich, eine bestehende Entität aus Service Studio auf eine Entität in Integration Studio abzubilden. Die Aktion wurde

daher so angepasst, dass ihr die Anzahl aller Angebote sowie die zu berechnenden Kennzahlen als Parameter übergeben wurden, um daraus die prozentuale Verteilung innerhalb der Angebote berechnen zu können. Es handelte sich also nicht um eine klassische Funktion mit einem Rückgabewert, sondern um eine Funktion mit mehreren Rückgabewerten, die als Ausgabeparameter in der Aktion definiert sind. Nachdem die Aktion definiert war, wurde die entsprechende C#-Funktion im Visual Studio-Code aufgerufen und die Berechnungen implementiert. Die Schaltfläche ‚1-Click Publish‘ in der Symbolleiste von Integration Studio wurde verwendet, um den Code zu synchronisieren, zu validieren und schließlich die Aktion als Erweiterung zu veröffentlichen. Diese Erweiterung wurde als eine Abhängigkeit in Service Studio installiert. Sie wurde als Serveraktion dargestellt, die in einer Datenaktion im Sichtbarkeitsbereich des Dashboard-Bildschirms aufgerufen wurde. Die folgende Datenaktion aus Abbildung 5.25 zeigt ihren Aufruf zur Berechnung der Kennzahlen.



Abbildung 5.25: Aufruf der Erweiterung zur Berechnung von Dashboard-Kennzahlen in einer Datenaktion

Die beschriebenen Eingabeparameter der Aktion wurden über Aggregate ermittelt und als Parameter der Serveraktion ‚AngeboteQuote‘ übergeben. Die Rückgabewerte der Serveraktion wurden lokalen Variablen zugewiesen, deren Werte über ‚Expression‘-Widgets im Dashboard angezeigt wurden.

Für die Erstellung von Diagrammen wurde das Modul ‚OutSystems Charts‘ verwendet, das verschiedene Diagramme wie Flächen-, Balken-, Liniendiagramme und viele mehr als fertige Widgets zur Verfügung stellt, die jedoch über den Eigenschaftseditor für die Anzeige von Daten konfiguriert werden müssen. Zu diesem Zweck wurde ein Balkendiagramm erstellt, um die verschiedenen Gehaltsangebote nach Bewerbungen darzustellen. Bei der Konfiguration des Diagramms wurde die Liste der Angebote als Datenquelle, die Bezeichnungen der Bewerbungen als Beschriftung der Balken und die Gehaltsangebote als Werte der Balken angegeben. Ein weiteres Kreisdiagramm wurde erstellt, um die verschiedenen Abteilungen aller Stellen darzustellen. Hier reichte es nicht aus, eine statische Liste als Datenquelle anzugeben, da ein Kreisdiagramm in mehrere Kreissektoren unterteilt ist. Daher mussten die einzel-

5 Umsetzung und Evaluation

nen Kreissektoren iterativ berechnet werden. Zu diesem Zweck wurde die folgende Serveraktion aus Abbildung 5.26 entwickelt.

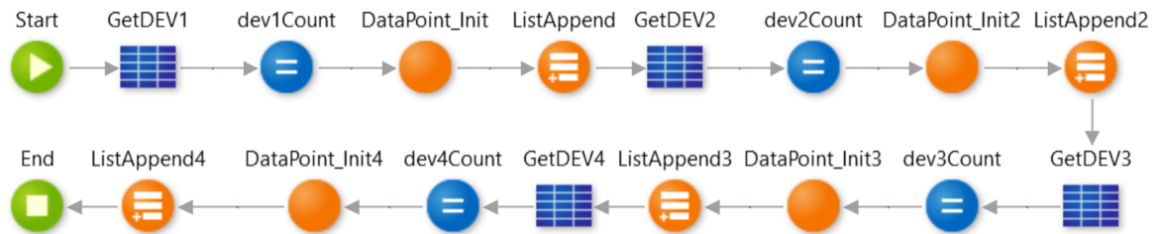


Abbildung 5.26: Serveraktion zur Berechnung der Kreissektoren eines Kreisdiagramms in OutSystems

Ein Datensatz eines Diagramms wurde in diesem Modul mit der speziellen Datenstruktur ‚DataPoint‘ dargestellt. Dieser wurde mit der vom Modul bereitgestellten Serveraktion ‚DataPointInit‘ erzeugt. Hier war es notwendig, iterativ die Listen der Abteilung (DEV1, DEV2, DEV3 und DEV4) aus allen Stellen über Aggregate zu ermitteln und der Serveraktion ‚DataPointInit‘ als Eingabeparameter zu übergeben, um die jeweiligen Kreissektoren zu erzeugen. Da das Kreisdiagramm eine Liste von DataPoints erwartete, wurden alle Teillisten zu einer Liste zusammengeführt. Die resultierende Anwendung ist in Abbildung 5.27 dargestellt.

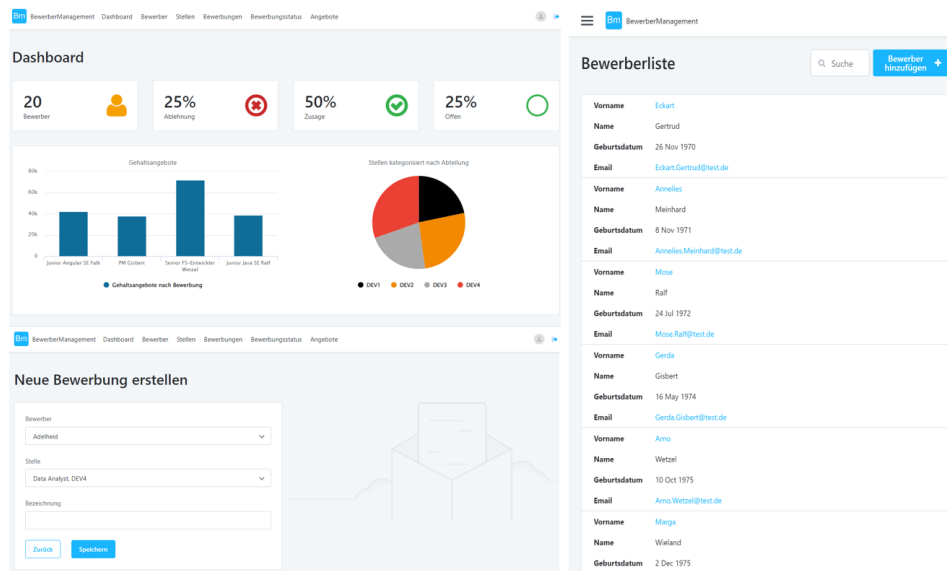


Abbildung 5.27: Benutzeroberflächen der Bewerbermanagement-Anwendung mit OutSystems

Über das Menü in der Kopfzeile lassen sich die Listenansichten der Tabellen aufrufen,

in denen neue Datensätze angelegt werden können. Außerdem gibt es einen weiteren Menüpunkt zum Aufrufen des Dashboards, in dem die Kennzahlen und Diagramme angezeigt werden.

OutSystems verfügt bereits über eigene Tools für die automatische Bereitstellung neuer Versionen in den verschiedenen Umgebungen[70]. Eine automatisierte Bereitstellungs-pipeline außerhalb der OutSystems-Cloud für eine OutSystems-Anwendung kann entweder mit Jenkins oder Azure DevOps erstellt werden. Die Erstellung einer Pipeline sowohl mit Jenkins als auch mit Azure DevOps erfordert die Generierung eines Authentifizierungstokens des LifeTime-Servers als Anmeldeinformation für die CI/CD-Plattformen, was mit der Testlizenz nicht möglich ist.[71][72] Durch Hinzufügen eines weiteren Schritts zum CI/CD-Pipeline-Prozess kann eine OutSystems-Anwendung mit einem externen Versionskontrollsystem wie GitHub integriert werden[73].

Prinzipiell kann eine Anwendung in OutSystems mit der Testversion nicht skaliert werden. OutSystems gewährleistet jedoch sowohl eine vertikale als auch eine horizontale Skalierung in der OutSystems-Cloud. Die vertikale Skalierung wird durch die Erhöhung der Hardware-Ressourcen für den Frontend- und Datenbankserver erreicht. OutSystems stellt hierfür ein klassenbasiertes Skalierungsschema zur Verfügung, in dem die verfügbaren Ressourcen festgelegt sind. Für den Frontend-Server können bis zu 8 virtuelle Kerne mit 16 GB RAM und für den Datenbank-Server bis zu 8 virtuelle Kerne mit 61 GB RAM konfiguriert werden.[74] Die horizontale Skalierung wird durch den Einsatz zusätzlicher Frontend-Server sowie durch die Verwendung eines Load Balancers zur effizienten Verteilung des Datenverkehrs auf mehrere Frontend-Server erreicht[75]. Alternativ kann eine OutSystems-Anwendung in einer Container-Umgebung bereitgestellt und über verschiedene Orchestrierungssoftware wie Kubernetes oder andere skaliert werden[76].

Mendix

Mendix Studio Pro bietet standardmäßig keine Möglichkeit, eine Anwendung auf Basis von Daten zu erstellen. Daher wurde auf Empfehlung von Mendix das Modul ‚Excel Importer‘ installiert und konfiguriert, das den Import einer Excel-Datei gewährleistet. Das Modul bietet eine Konfigurationsseite in der Anwendung, auf der die Excel-Datei hochgeladen wird, um die importierten Daten anschließend einer Entität zuzuordnen. Da der Importer keine Entitäten aus den Excel-Tabellen erstellt, wurden sie manuell im Domänenmodell angelegt. Anschließend wurden die Beispieldaten

über den Importer in die Bewerber- und Stellentabellen importiert. Allerdings wurden die Beispieldaten in zwei separaten Excel-Dateien gehalten, da der Importer nur Daten aus einer Tabelle der Excel-Datei lesen kann. Mit der Beschaffung einer Datengrundlage wurden die Beziehungen zwischen allen Tabellen ohne zusätzliche Recherche umgesetzt, da die verschiedenen Arten von Beziehungen im Entitätsdiagramm durch eine konfigurierbare Verbindungslinie zwischen zwei Entitäten realisiert werden können.

Der nächste Schritt war die Implementierung der Benutzeroberfläche der Anwendung. Mendix bietet als Grundlage verschiedene Vorlagen wie Formulare, Listen etc. an, deren Struktur sich automatisch an die zugeordnete Datenquelle anpasst. Für die Listenansichten der Tabellen wurde jedoch keine passende Vorlage gefunden, so dass sie manuell mit einem ‚Data Grid‘-Widget erstellt wurden, das eine Listenansicht darstellt, in der Datensätze auch gleichzeitig gesucht, erstellt, bearbeitet und gelöscht werden können. Zum Anlegen und Bearbeiten eines Datensatzes wurden die Listenansichten mit Template-basierten Formularen verknüpft. Es waren jedoch manuelle Anpassungen im Stellenformular erforderlich, da die Zuweisung mehrerer Bewerber zu einer Stelle nicht über die Vorlage abgebildet wurde. Um dieses Problem zu lösen, wurde das externe ‚Bootstrap Multi Select‘-Widget verwendet, das dem Formular ein Dropdown-Menü mit mehreren Auswahlmöglichkeiten hinzufügt.

Mendix ermöglicht die Entwicklung von benutzerdefiniertem Code über eine sogenannte ‚Java Action‘, die eine Java-Methode darstellt[77]. Diese wurde verwendet, um die Kennzahlen für das Dashboard zu berechnen. In einer Java Action werden die Eingabeparameter und der Rückgabewert der zugehörigen Java-Methode definiert. Um die Kennzahlen in einem Objekt zu speichern, wurde im Domänenmodell eine nicht-persistente Entität definiert, die als Rückgabewert der Java-Aktion festgelegt wurde. Des Weiteren wurde hier die Liste von Angeboten, sowie die Liste von Bewerbern als Eingabeparameter festgelegt. Um auf den Quellcode der Java-Aktion zuzugreifen, wurde der lokale Projektordner über Eclipse geöffnet. Dort befand sich die zugehörige Methode der Java-Aktion in einer Java-Klasse. Nachdem die Berechnungen implementiert waren, wurde die Java-Aktion in einem Microflow aufgerufen, der in Abbildung 5.28 dargestellt ist.

5 Umsetzung und Evaluation

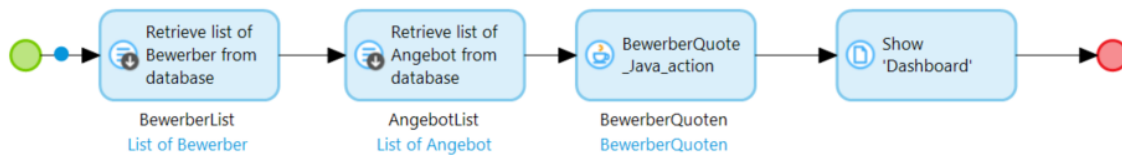


Abbildung 5.28: Benutzeroberflächen der Bewerbermanagement-Anwendung mit Mendix

Die Objektaktivität ‚Retrieve‘ wurde verwendet, um die Listen der Bewerber und Angebote abzurufen, die als Eingabeparameter an die Java-Aktion übergeben wurden. Das von der Java-Aktion zurückgegebene Objekt wurde an die Dashboard-Seite übergeben, um die Kennzahlen mithilfe des Widgets ‚dataview‘ anzuzeigen.

Mendix verfügt über vorgefertigte Widgets für die Anzeige verschiedener konfigurierbarer Diagramme wie Flächendiagramme, Liniendiagramme, Blasendiagramme und andere. Die Konfiguration eines Diagramms erfordert die Angabe der Datentabelle sowie der Datenattribute für die Anzeige der x-Achse und der y-Achse des Diagramms. Für die y-Achse sind jedoch nur Datenattribute mit numerischen Datentypen zulässig. Es ist aber auch möglich, die Datenreihen des Diagramms durch einen Microflow zu spezifizieren, oder die Datensätze einer Tabelle mit Constraints zu filtern. Die Benutzeroberflächen der resultierenden Anwendung ist in Abbildung 5.29 dargestellt.

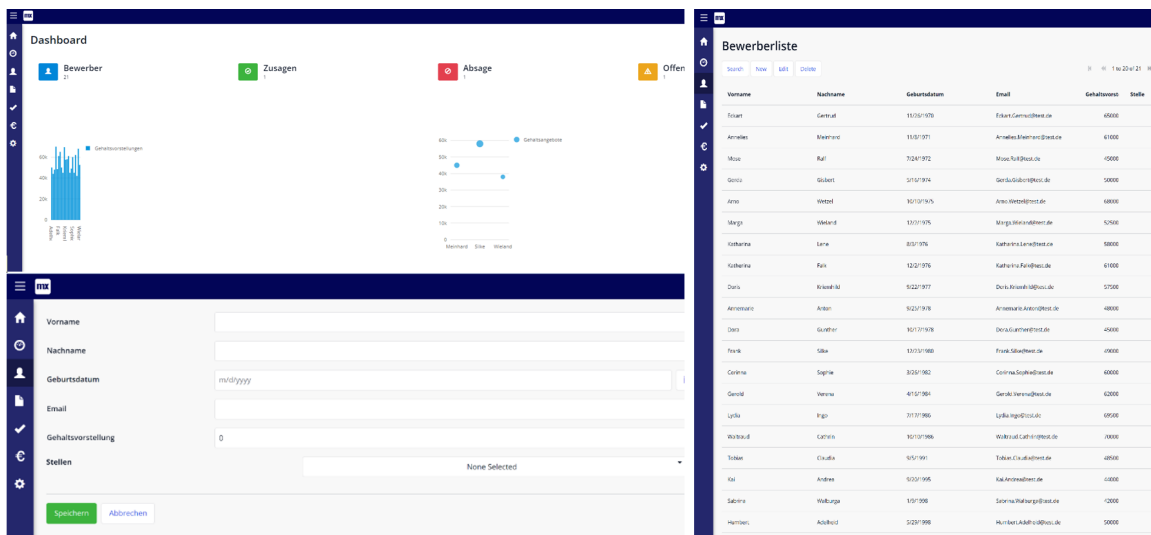


Abbildung 5.29: Aufruf der Java-Aktion in einem Microflow zur Berechnung von Dashboard-Kennzahlen

Auf dem Dashboard wurde ein Balkendiagramm erstellt, um die unterschiedlichen Gehaltsvorstellungen der Bewerber darzustellen und ein Blasendiagramm, um die unterschiedlichen Gehaltsangebote an die Bewerber aufzuzeigen. Außerdem wurden in einer linken Sidebar Navigationselemente platziert, die die verschiedenen Listenansichten aufrufen. Jede Listenansicht ist auch mit den entsprechenden Formularen zum Erstellen und Bearbeiten eines Datensatzes verknüpft.

Eine CI/CD-Pipeline wird bereits durch die von Mendix im Mendix Developer Portal bereitgestellten Tools unterstützt. Das Erstellen externer CI/CD-Pipelines, z.B. mit Jenkins, wird durch die APIs der Mendix-Plattform ermöglicht.[78] Dies ist jedoch mit der Testlizenz aufgrund der fehlenden Mendix-API-Rechte nicht möglich[79]. Mendix Studio Pro ermöglicht es jedoch, die Anwendung mit einem externen Versionskontrollsystem wie GitLab zu verknüpfen. Dies wäre eine alternative Lösung zur Erstellung einer CI/CD-Pipeline über die bereitgestellten Werkzeuge des Versionskontrollsystems.

Grundsätzlich können nur lizenzierte Mendix-Anwendungen skaliert werden. Allerdings wird die vertikale und horizontale Skalierung auf verschiedenen Mendix Cloud Versionen unterstützt. ‚Mendix Cloud v3‘ unterstützt nur die vertikale Skalierung, während ‚Mendix Cloud v4‘ sowohl die vertikale als auch die horizontale Skalierung unterstützt. Die horizontale Skalierung erfolgt durch das Hinzufügen weiterer Serverinstanzen und die vertikale Skalierung durch die Erweiterung des Arbeitsspeichers pro Instanz. Es können jedoch maximal 16 GB RAM auf die Instanzen verteilt werden.[80]

ServiceNow

Zur Implementierung der Anwendung wurde zunächst der ‚Guided Application Creator‘ von ServiceNow eingesetzt, mit dem die Tabellen für Bewerber und Stellen auf Basis der Beispieldaten aus den Excel-Dateien erstellt werden konnten. Eine Anpassungsmöglichkeit der Tabellenstrukturen war nach dem Import der Daten gegeben. Alle anderen Tabellen aus dem Datenmodell wurden manuell erstellt. Für die Erstellung der Tabellenbeziehungen war ein Umstieg auf ServiceNow Studio IDE notwendig, da sie bereits eine fertige Lösung zur Umsetzung von n:m-Beziehungen in der Datenmodellkomponente zur Verfügung stellt. In dieser Lösung genügte es, die beiden in Beziehung stehenden Tabellen, sowie eine Bezeichnung für die Referenztable anzugeben. Für eine 1:n-Beziehung wurde in der entsprechenden Datentabelle manuell ein Referenzattribut mit einem Verweis auf die verknüpfte Tabelle erstellt.

5 Umsetzung und Evaluation

Allerdings wurde die Umsetzung einer 1:1-Beziehung insgesamt nicht unterstützt.

Für die Gestaltung der Benutzeroberflächen der Anwendung wurde eine Experience-Vorlage verwendet, die automatisch Listenansichten zur Anzeige der Daten und Entity-Formulare zur Erstellung und Bearbeitung von Datensätzen anhand der Datentabellen generiert. Wie bereits aus der ersten Phase bekannt, kann benutzerdefinierter Code sowohl serverseitig für eine Geschäftsregel als auch clientseitig für Formulare verwendet werden. Da dies die Kennzahlenberechnung als Anwendungsfall nicht abdeckt, wurde als alternative Lösung die Komponente ‚Data visualization‘ als Widget im UI-Builder verwendet, die automatisch verschiedene Darstellungsformen wie Diagramme und Kennzahlen aus dem Datensatz einer Datentabelle generiert. Hier war es auch möglich, den Datensatz zu filtern, so dass die Kennzahlen über diesen Weg berechnet werden konnten. Die Benutzeroberflächen der implementierten Anwendung können der Abbildung 5.30 entnommen werden.

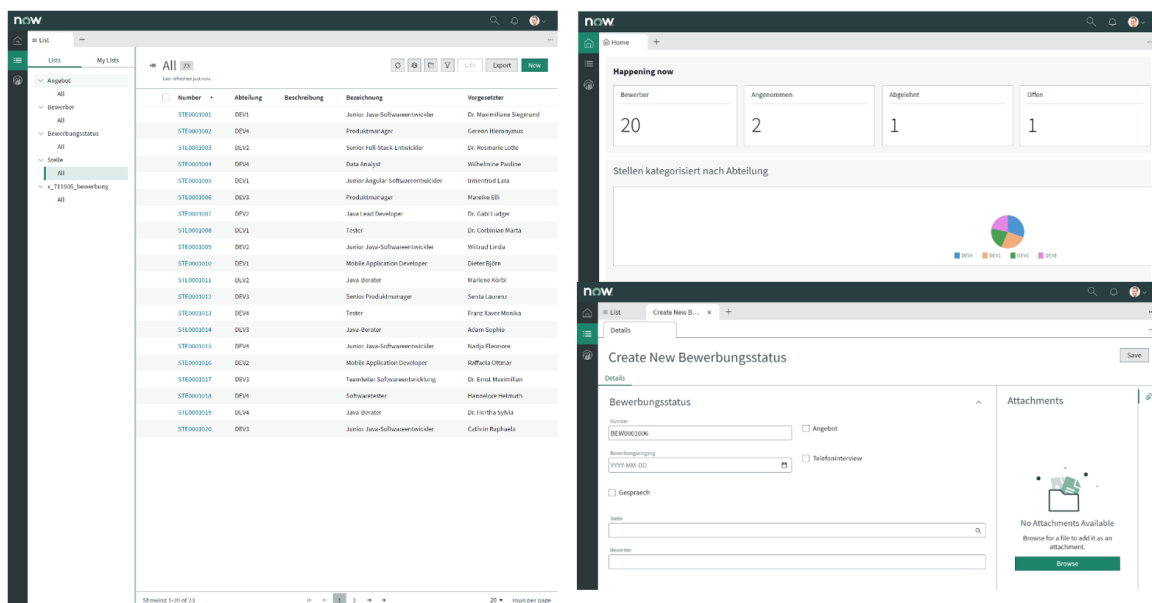


Abbildung 5.30: Benutzeroberflächen der Bewerbermanagement-Anwendung mit Service-Now

Mit dem in Service Studio IDE integrierten Versionskontrollsystem war es möglich, die Anwendung mit einem Repository eines externen, Cloud-basierten Versionskontrollsystems wie GitLab zu verbinden. Mit den von GitLab bereitgestellten Tools wurde eine erste einfache CI/CD-Pipeline erstellt.

In ServiceNow kann nur die Produktionsinstanz vertikal bis zu 2 GB RAM für zwei Frontend-Server skaliert werden[81].

5.3.3 Evaluation

Komplexe Datenmodelle

Insgesamt konnte das komplexe Datenmodell mit allen Low-Code-Plattformen mit Ausnahme von ServiceNow vollständig implementiert werden. Allerdings hebt sich Mendix von den anderen Plattformen dadurch ab, dass die Umsetzung des Datenmodells am benutzerfreundlichsten ist. Power Apps hat ebenfalls eine benutzerfreundliche Darstellung, allerdings kann es verwirrend sein, dass eine 1:1 Kardinalität nicht direkt abgebildet wird, sondern über einen Workaround erstellt werden muss. Auch die automatische Generierung der Referenztabelle für eine n:m-Beziehung gibt Mendix und Power Apps einen klaren Vorteil gegenüber OutSystems.

Datenverarbeitung

Der Import von größeren Datenmengen wird grundsätzlich von allen Plattformen unterstützt. Trotz manueller Dateneingabe hebt sich Power Apps von den anderen Plattformen dadurch ab, dass es über eine große Anzahl von Konnektoren verfügt, die den Import aus verschiedenen Datenquellen ermöglichen. Auf allen Plattformen wurde kein automatischer Mechanismus zur Erstellung von Dashboards mit aussagekräftigen Diagrammen festgestellt. Mendix und OutSystems bieten jedoch einen Vorteil gegenüber Power Apps bei der manuellen Erstellung von Diagrammen, da sie die Filterung der anzuzeigenden Daten ermöglichen.

Benutzerdefinierter Code

Mendix und OutSystems verfolgen zwei unterschiedliche Ansätze bei der Erstellung von benutzerdefiniertem Code. Bei Mendix ist er ein fester Bestandteil der Anwendungslogik, der dadurch die Angabe spezifischer Datentypen erlaubt. OutSystems hingegen verwendet ihn als eigenständige Erweiterung, die theoretisch in jedes Modul einer Anwendung integriert werden kann. Insgesamt hebt sich Mendix jedoch von OutSystems dadurch ab, dass die Interaktion des benutzerdefinierten Codes reibungsloser verlief. Power Apps erlaubt im Vergleich zu den anderen Plattformen keine funktionale Erweiterung der modellgesteuerten Anwendungen, da nur clientseitiges Scripting möglich ist.

CI / CD

Sowohl Mendix als auch OutSystems verfügen bereits über integrierte Tools in ihren

Clouds, die die Bereitstellung von Anwendungen in verschiedenen Umgebungen automatisieren, während Power Apps eine manuelle Einrichtung erfordert. Die Einrichtung einer CI/CD-Pipeline auf privaten Servern ist prinzipiell bei allen Plattformen möglich. Bei OutSystems ist der Aufwand jedoch höher einzustufen, da Service Studio keine intuitive Möglichkeit bietet, die Anwendung mit einem externen Versionskontrollsystem zu verbinden.

Skalierbarkeit

OutSystems hebt sich von allen Plattformen durch seine flexiblere und ressourcenintensivere Skalierungsstrategie ab. Durch den Einsatz eines Load Balancers kann zudem eine effiziente Bearbeitung der Anfragen an den Server gewährleistet werden. Im Falle eines Serverausfalls kann er sicherstellen, dass keine Daten verloren gehen. Über die Skalierbarkeit von Power Apps kann keine Aussage getroffen werden, da die Recherchen, wie u.a. auf der Dokumentationsseite, keine Ergebnisse brachten.

6 Fazit

Dieses Kapitel fasst die in den drei Phasen erzielten Ergebnisse und die Erkenntnisse über die Low-Code-Plattformen zusammen. Darüber hinaus werden auf der Grundlage der Ergebnisse Empfehlungen zu verschiedenen Anwendungsbereichen gegeben, für die die untersuchten Low-Code-Plattformen am besten geeignet erscheinen, so dass sie als Leitfaden für die Entwicklung entsprechender Anwendungen genutzt werden können. Schließlich werden die Low-Code-Plattformen mit der klassischen Softwareentwicklung verglichen, um ihre Vor- und Nachteile zu verdeutlichen.

6.1 Zusammenfassung der Ergebnisse

Grundsätzlich ist bei allen Plattformen ein **Zeitersparnis** bei der Einrichtung der Anwendung und der Konfiguration der von den Plattformen bereitgestellten Lösungen im Vergleich zur Einrichtung einer Anwendung mit klassischen Softwareentwicklungswerkzeugen zu erwarten. Bei der Umsetzung komplexerer Funktionalitäten ist jedoch mit einem **Zeitverlust** zu rechnen, da oft nur wenige Lösungsideen von der Online-Community oder den Entwicklern der Plattformen bereitgestellt werden.

Die **Integration von Diensten** wird in OutSystems, Mendix und Power Apps durch die von den Plattformen bereitgestellten Lösungen unterstützt. Für ServiceNow bleibt diese Forschungsfrage offen.

OutSystems ermöglicht die Entwicklung von anwendungsunabhängigen Serveraktionen, die mit C#-Code entwickelt werden und als funktionale Erweiterungen in beliebigen Anwendungen eingesetzt werden können. In Mendix hingegen kann anwendungsspezifische Funktionalität alternativ zu einem Microflow über eine Java-Action entwickelt werden. Power Apps schränkt die Entwicklung mit **benutzerdefiniertem Code** in modellgesteuerten Anwendungen stark ein, verhindert funktionale Anpassungen und erlaubt nur die Entwicklung von clientseitigen Geschäftsregeln mit Hilfe der bereitgestellten Methoden. ServiceNow ermöglicht, dass sowohl serverseitige als auch clientseitige plattformspezifische Funktionen wie Geschäftsregeln mit JavaScript-Code unter Verwendung der bereitgestellten Methoden entwickelt werden. Insgesamt mangelt es allen Plattformen an Flexibilität, wenn es um größere Anpassungen durch benutzerdefinierten Code geht.

OutSystems stellt seinen Premium-Anwendern im LifeTime und Service Center

verschiedene Analyse- und Monitoring-Tools zur Verfügung, mit denen das **Laufzeitverhalten** von Aktionen analysiert werden kann. Mendix bietet die Lösung ‚Performance Tool‘ als externes Modul zur Messung der Ausführungszeit von Microflows an. Power Apps verfügt auch über ein eigenes Überwachungstool für Canvas- und modellgesteuerte Anwendungen, das die Ausführungszeit der in der Anwendung durchgeführten Aktionen misst.

Sowohl Mendix als auch OutSystems bieten visuelle Datenmodellimplementierungstechniken für die Umsetzung **komplexer Datenmodelle**. Allerdings erfordert die Implementierung der 1:1- und n:m-Beziehungstypen in OutSystems eine manuelle Definition in den Entitäten. In Power Apps hingegen erfolgt die Erstellung der Entitäten vollständig manuell. Die Erstellung der verschiedenen Beziehungstypen erfolgt jedoch automatisch durch Auswahl des vordefinierten Beziehungstyps aus einer Dropdown-Liste in Dataverse. Eine 1:1-Beziehung wird in dieser Dropdown-Liste jedoch nicht angeboten, so dass eine manuelle Definition eines Datenfeldes als Suchfeld erforderlich ist. In ServiceNow werden sowohl Entitäten als auch der 1:n-Beziehungstyp manuell definiert. Für eine n:m-Beziehung gibt es auch eine vorgefertigte Lösung, die die Beziehung automatisch durch Angabe der verknüpften Tabellen erstellt. Eine 1:1-Beziehung wird jedoch insgesamt nicht abgebildet.

Darüber hinaus kann eine Anwendung mit allen Plattformen ohne großen Aufwand weiterentwickelt werden. Es ist jedoch nicht auszuschließen, dass die Verwendung externer Abhängigkeiten die **Weiterentwicklung** einschränkt.

Mendix und OutSystems decken die **Testbarkeit** einer Anwendung vorzugsweise mit Unit-Tests über externe Module ab. Insgesamt können bei OutSystems verschiedene Tests ohne großen Aufwand entwickelt werden. Der Aufwand zur Durchführung eines Unit-Tests bleibt für Mendix offen, da keine Untersuchung durchgeführt wurde. Power Apps hingegen bietet ein erweitertes Tool für Canvas-Apps, das einen End-to-End-Test realisiert. Auch hier kann der Aufwand als gering eingestuft werden, da die beispielhafte Interaktion mit der Benutzeroberfläche aufgezeichnet und als Grundlage verwendet werden kann.

Auf allen Plattformen können neue Versionen mit nur einem Klick in den Entwicklungsumgebungen der Server bereitgestellt werden. Mendix und OutSystems bieten auf ihren Servern integrierte Tools für die **automatische Bereitstellung** neuer Versionen auf allen Umgebungen. Im Hinblick auf die Erstellung einer CI/CD-Pipeline mit externer Software wird der Aufwand von OutSystems als hoch eingestuft, verglichen mit den anderen Plattformen, die entweder bereits über ihre IDEs Verknüpfungsmög-

lichkeiten mit externen Versionskontrollsystemen anbieten oder eigene Lösungen bereitstellen, die die Einrichtung stark vereinfachen können.

OutSystems und Mendix bieten **vertikale** und **horizontale Skalierung**, während ServiceNow nur vertikale Skalierung bietet. Darüber hinaus bietet OutSystems sowohl die umfangreichste Skalierung in Bezug auf die Hardwareressourcen insgesamt als auch die einzige Plattform, die durch den Einsatz eines Load Balancers **Ausfallsicherheit** bietet.

6.2 Diskussion zur Einschätzung der geeignetsten Plattform

Für diese Diskussion wird ServiceNow aufgrund mangelnder Entwicklungserfahrung und unzureichender Ergebnisse ausgeschlossen.

Bei der Erstellung funktionaler Mockups zur Präsentation verschiedener Lösungsvorschläge für eine geplante Anwendung empfiehlt sich die Verwendung der Power Apps Canvas-Lösung, die die Entwicklung mit vorgefertigten UI-Komponenten und einem insgesamt sehr geringen manuellen Entwicklungsaufwand unterstützt. Obwohl Mendix und OutSystems ebenso viele vorgefertigte UI-Widgets mit bereits eingebauter Logik anbieten, zeigen die Ergebnisse, dass an den meisten Stellen immer noch manuelle Entwicklung erforderlich ist.

Für die Entwicklung einer externen Webanwendung mit einer hohen Funktionsdichte empfiehlt sich der Einsatz von OutSystems oder Mendix, da diese eine bessere Gesamtkontrolle über die Anwendungskomponenten bieten. Die Ergebnisse zeigen außerdem, dass die Portallösung von Power Apps weniger flexibel in Bezug auf die Erweiterung der Funktionalität ist, da nur die mitgelieferten Tools verwendet werden können. Es ist nicht auszuschließen, dass die mitgelieferten Werkzeuge den Anforderungen neuer Funktionen nicht genügen. In einer komplexen Anwendung ist auch das Auftreten von Fehlern zu erwarten, was für die Notwendigkeit eines Debugging-Werkzeugs und die Sicherstellung der Korrektheit von funktionalen Anwendungskomponenten durch automatisierte Tests spricht. Mendix und Outsystem erfüllen diese Anforderung mit ihren ausgereiften Debugging-Tools und Test-Frameworks.

Andererseits zeigen die Ergebnisse, dass alle drei Low-Code-Plattformen gut für datenzentrierte Anwendungen mit komplexen Entitätsbeziehungen geeignet sind. Während sich die modellgetriebene Anwendung von Power Apps stark auf die Mo-

dellierung der einzelnen Komponenten der Anwendung konzentriert und ein Standarddesign für die resultierende Anwendung verwendet, können in Mendix und OutSystems individuelle Benutzeroberflächen gestaltet werden.

Zusammenfassend lässt sich sagen, dass sowohl Mendix als auch OutSystems eine solide Erfahrung in der Anwendungsentwicklung voraussetzen. Auch wenn Power Apps funktionsähnliche Formeln verwendet, sind sie insgesamt weniger komplex, als die visuellen Programmier Techniken von Mendix und OutSystems. Insgesamt bietet Microsoft in seiner Plattform viele Out-of-the-Box-Funktionen, was den manuellen Entwicklungsaufwand geringhält.

6.3 Diskussion zum Vergleich mit der klassischen Softwareentwicklung

Aus den Ergebnissen lässt sich ableiten, dass Low-Code-Anwendungen insgesamt weniger flexibel sind als Anwendungen, die mit klassischer Softwareentwicklung entwickelt werden, da in den Low-Code-Plattformen hauptsächlich vorgefertigte Werkzeuge verwendet werden, während die klassische Softwareentwicklung die volle Kontrolle über jede Softwarekomponente bietet und individuelle Anpassungen ermöglicht. Die Einschränkung der Programmiersprache für die Verwendung von benutzerdefiniertem Code in den Plattformen ist ebenfalls ein Nachteil gegenüber der klassischen Softwareentwicklung, die keine Grenzen für die Wahl der Programmiersprache setzt.

Ein wichtiges Werkzeug in der klassischen Softwareentwicklung ist ein Versionskontrollsystem, das in vielen klassischen IDEs bereits komfortabel eingebunden ist. Die meisten Low-Code-Plattformen verfügen auch über ein eigenes Versionskontrollsystem in der IDE, allerdings nicht mit der vollen Bandbreite an Werkzeugen. Zum Beispiel ist das temporäre Speichern von Änderungen in Mendix Studio Pro nicht möglich.

In den Entwicklungssitzungen mit den Low-Code-Plattformen wurde jedoch festgestellt, dass Plattformversionen und Abhängigkeiten automatisch und ohne manuelle Anpassungen aktualisiert werden, während bei der klassischen Entwicklung Aktualisierungen in der Regel zeitaufwändig sind und zusätzliche Entwicklung erfordern. Im Vergleich zur klassischen Softwareentwicklung, die komplexe und zeitaufwändige Prozesse zur Bereitstellung neuer Versionen erfordert, erfolgt dies bei der

Low-Code-Entwicklung mit nur einem Klick über die IDE. Während die klassische Softwareentwicklung den manuellen Aufbau einer CI/CD-Pipeline erfordert, sind Mendix und OutSystems mit ihren automatisierten Deployment-Tools klar im Vorteil.

Bei der Recherche nach Lösungsansätzen für die Implementierung der Anwendungsbeispiele zeigte sich auch, dass die Online-Community von Low-Code-Plattformen nicht vergleichbar groß ist wie die der klassischen Softwareentwicklung. Dies kann dazu führen, dass dem Entwickler bei komplexeren Problemen keine Lösungshilfe zur Verfügung gestellt wird. Der Einsatz von externen Lösungen hat auch gezeigt, dass diese teilweise fehlerhaft und auch unzureichend dokumentiert waren, was bei der klassischen Softwareentwicklung seltener der Fall ist.

Im Hinblick auf die Implementierung der Geschäftslogik einer Low-Code basierten Anwendung mittels visueller Programmiertechniken sind einige Unterschiede, aber auch Gemeinsamkeiten zu grundlegenden Softwareentwicklungskonzepten festzustellen, die hier näher beleuchtet werden. Die Initialisierung einiger Datenstrukturen, wie z.B. Listen, erwies sich in Mendix als umständlich. Sie erfordert die Definition einer nicht-persistenten Entität im Domänenmodell, die Erstellung von Objekten dieser Entität, die Zuweisung einer Zeichenkette an eine Variable, die Zuweisung dieses Variablenwerts an das Attribut eines Objekts und das Hinzufügen eines Objekts zu der Liste dieser Entität. Andererseits wurde auch festgestellt, dass einige wichtige Konzepte der objektorientierten Programmierung bereits in Mendix und OutSystems umgesetzt sind. Wie in der zweiten Phase angedeutet, kann Vererbung im Domänenmodell von Mendix verwendet werden. Ein gewisses Maß an Abstraktion durch die Verwendung von Unterprogrammen ist in beiden Plattformen ebenfalls gegeben. Dies wird z.B. in OutSystems dadurch gewährleistet, dass die Implementierung von Teilproblemen auf verschiedene Serveraktionen verlagert wird, so dass die Implementierung in der aufrufenden Aktion verborgen bleibt.

Zusammenfassend lässt sich aus persönlicher Entwicklungserfahrung über die drei Phasen hinweg sagen, dass Low-Code-Entwicklungswerkzeuge trotz der erforderlichen Mindestprogrammiererfahrung insgesamt schneller erlernt werden können als eine herkömmliche Programmiersprache mit zusätzlichen Frameworks.

6.4 Ausblick

Einige Forschungsfragen blieben insbesondere für die ServiceNow-Plattform bei dieser Arbeit offen, die in zukünftigen Arbeiten untersucht werden könnten. Darüber hinaus wäre es sinnvoll, auch die Werkzeuge der Plattformen zu untersuchen, die mit kostenpflichtigen Lizenzen versehen sind. Dazu gehören unter anderem die automatisierten Deployment-Tools von Outsystem und Mendix, aber auch der Aufbau von CI/CD-Pipelines über externe Software für verschiedene Umgebungen, die üblicherweise in einem klassischen Softwareentwicklungsprozess eingesetzt werden. Ein weiterer Vorschlag wäre, die verschiedenen Skalierungsstrategien der Plattformen im Rahmen der kostenpflichtigen Lizenz zu untersuchen, indem hohe Lasten auf eine Anwendung simuliert und die Antwortzeiten des Anwendungsservers gemessen werden.

Für künftige Arbeiten wäre es auch interessant, Werkzeuge zur Prozessautomatisierung in Form von Workflows zu untersuchen, die hier aufgrund der eingeschränkten Zugänglichkeit, aber auch des fehlenden Anwendungsfalls in den Anwendungsbeispielen nicht verwendet wurden. Unter anderem wurden verschiedene Tools der Power-Plattform wie Power Automate oder Power BI erwähnt, deren Nutzen ebenso in zukünftigen Arbeiten untersucht werden könnte.

A Anhang

A.1 Katalog der Logikwerkzeuge für Aktionen in OutSystems









Symbol	Beschreibung
	Start: Gibt an, wo ein Flow mit der Ausführung beginnt.
	Server-Aktion: Führt eine Aktion mit Logik aus, die auf dem Server ausgeführt wird.
	Entscheidung: Stellt eine Entscheidung dar, die entweder wahr oder falsch sein kann.
	Zuweisung: Weist einer Variablen einen Wert zu.
	Aggregat: Ruft Daten aus der Datenbank ab.
	Ziel: Leitet den Flow zu einem anderen Bildschirm um.
	Nachricht: Gibt dem Endbenutzer eine Rückmeldung.
	Ende: Beendet den Flow. Der Flow kann an dieser Stelle mit einem Rückgabewert beendet werden.

Tabelle A.1: Katalog der Logikwerkzeuge für Aktionen in OutSystems[82]

A.2 Katalog der Toolbox-Elemente für Microflows in Mendix











Symbol	Beschreibung
	Start Event: Ort, an dem der Microflow gestartet wird.
	Parameter: Ist das Eingabeobjekt für den Microflow und kann in jeder Aktivität des Microflows eingesetzt werden.
	Entscheidung: Stellt eine Entscheidung dar, die entweder wahr oder falsch sein kann.
	Objektaktivität-Create-Object: Erstellt ein Datenobjekt.
	Objektaktivität-Retrieve: Ein oder mehrere Objekte können entweder über eine Assoziation oder über eine Datenbankabfrage aufgerufen werden.
	Objektaktivität-Change-Object: Ändert ein vorhandenes Datenobjekt oder Eigenschaften dieses Objekts.
	Integrationsaktivität-Call-REST-Service: Ruft den REST-Service auf.
	Clientaktivität-Show-Page: Zeigt dem Endnutzer eine Seite an.
	Clientaktivität-Show-Message: Zeigt dem Endnutzer eine Nachricht an.
	End Event: Ort, an dem der Microflow beendet wird.

Tabelle A.2: Katalog der Toolbox Elemente für Microflows in Mendix[53]

A.3 Repositories

- Protokolle
 - <https://git.imp.fu-berlin.de/altanmk93/master-thesis>
- Wunschkennzeichen prüfen
 - <https://github.com/karacaltan/wunschkennzeichenpruefen>
- Kfz Zulassungsanwendung
 - <https://git.imp.fu-berlin.de/altanmk93/terminfinder-kfzumschreibung>
- Bewerbermanagement Anwendung
 - <https://git.imp.fu-berlin.de/altanmk93/bewerbermanagement-servicenow>

Literaturverzeichnis

- [1] F. Ulrich, M. Pierre und B. Alexander, *Low code platforms: Promises, concepts and prospects. A comparative study of ten systems*. ICB Research Reports, 2021.
- [2] F. K. Kaiser, M. Wiens und F. Schultmann, *Use of digital healthcare solutions for care delivery during a pandemic-chances and (cyber) risks referring to the example of the COVID-19 pandemic*. Springer, 11:1125–1137, 2021.
- [3] R. Waszkowski, *Low-code platform for automating business processes in manufacturing*. IFAC-PapersOnLine, 52(10):376–381, 2019.
- [4] P. Vincent, Y. Natis, K. Iijima, J. Wong, S. Ray und A. L. Akash Jain, *Magic Quadrant for Enterprise Low-Code Application Platforms*. Gartner, 2020.
- [5] Gartner, *Enterprise Low-Code Application Platforms (LCAP) Reviews and Ratings*, <https://www.gartner.com/reviews/market/enterprise-low-code-application-platform/>, 2022. Aufgerufen am 24.01.2022.
- [6] D. D. Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi und M. Wimmer, *Low-code development and model-driven engineering: Two sides of the same coin?* Springer, 437–446, 2022.
- [7] T. Stahl, M. Völter, S. Efftinge u. a., „Einführung in MDSD“, in *Modellgetriebende Softwareentwicklung*, 2007, S. 11–25.
- [8] D. Degenhardt, *Low-Code-Plattformen und deren Ursprünge*. 2020.
- [9] P. Beynon-Davies, C. Carne, H. Mackay und D. Tudhope, *Rapid application development (RAD): an empirical review*. European Journal of Information System, 8:211–223, 1999.
- [10] A. Sahay, A. Indamutsa, D. D. Ruscio und A. Pierantonio, *Supporting the understanding and comparison of low-code development platforms*. IEEE, 2020.
- [11] Y. Luo, P. Liang, C. Wang, M. Shahin und J. Zhan, *Characteristics and Challenges of Low-Code Development: The Practitioners’ Perspective*. Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering und Measurement (ESEM), 12:1–11, 2021.
- [12] D. Dahlberg, *Developer Experience of a Low-Code Platform: An exploratory study*. Master thesis, Umea University, 2020.

- [13] OutSystems, *It began with a vision*, <https://www.outsystems.com/de-de/evaluation-guide/it-began-with-a-vision/>, Aufgerufen am 27.01.2022.
- [14] Wikipedia, *OutSystems*, <https://en.wikipedia.org/wiki/OutSystems>, Aufgerufen am 27.01.2022.
- [15] OutSystems, *OutSystems Preise*, <https://www.outsystems.com/de-de/pricing-and-editions/>, Aufgerufen am 27.01.2022.
- [16] OutSystems, *Die Low-Code-Plattform für hochwertige Applikationen*, <https://www.outsystems.com/de-de/platform/>, Aufgerufen am 27.01.2022.
- [17] OutSystems, *Components and Tools*, <https://www.outsystems.com/training/lesson/2183/components-and-tools>, Aufgerufen am 27.01.2022.
- [18] OutSystems, *OutSystems development and management tools*, <https://www.outsystems.com/evaluation-guide/development-and-management-tools/>, Aufgerufen am 27.01.2022.
- [19] OutSystems, *Extending with Custom Code*, <https://www.outsystems.com/evaluation-guide/extending-with-custom-code/>, Aufgerufen am 27.01.2022.
- [20] H. Henriques, H. Lourenço, V. Amaral und M. Goulão, *Improving the Developer Experience with a Low-Code Process Modelling Language*. MODELS '18: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages und Systems, 200–210, 2018.
- [21] OutSystems, *OutSystems Platform Services*, <https://www.outsystems.com/evaluation-guide/platform-services/>, Aufgerufen am 27.01.2022.
- [22] OutSystems, *Service Studio Overview*, https://success.outsystems.com/Documentation/11/Getting_started/Service_Studio_Overview, Aufgerufen am 30.01.2022.
- [23] OutSystems, *Choose the right app for your project*, https://success.outsystems.com/Documentation/11/Getting_started/Choose_the_right_app_for_your_project, Aufgerufen am 31.01.2022.
- [24] OutSystems, *How does OutSystems enable team collaboration?*, <https://www.outsystems.com/de-de/evaluation-guide/how-does-outsystems-enable-team-collaboration/>, Aufgerufen am 03.02.2022.
- [25] OutSystems, *Generate and Distribute Your Mobile App*, https://success.outsystems.com/Documentation/11/Delivering_Mobile_Apps/Generate_and_Distribute_Your_Mobile_App, Aufgerufen am 03.02.2022.

- [26] OutSystems, *Use Site Properties to Configure Behaviors at Runtime*, https://success.outsystems.com/Documentation/11/Developing_an_Application/Use_Data/Use_Site_Properties_to_Configure_Behaviors_at_Runtime, Aufgerufen am 03.02.2022.
- [27] OutSystems, *UI Flows*, https://success.outsystems.com/Documentation/11/Developing_an_Application/Design_UI/Navigation/UI_Flows, Aufgerufen am 03.02.2022.
- [28] OutSystems, *Integration Studio*, https://success.outsystems.com/Documentation/11/Reference/Integration_Studio, Aufgerufen am 06.02.2022.
- [29] Microsoft, *Was ist Microsoft Power Platform?*, <https://docs.microsoft.com/de-de/learn/modules/introduction-power-platform/2-what-is-power-platform>, Aufgerufen am 25.02.2022.
- [30] Microsoft, *Power Apps – Preise*, <https://powerapps.microsoft.com/de-de/pricing/>, Aufgerufen am 03.04.2022.
- [31] Microsoft, *Was sind Canvas-Apps?*, <https://docs.microsoft.com/de-de/power-apps/maker/canvas-apps/getting-started>, Aufgerufen am 03.03.2022.
- [32] Microsoft, *Grundlagen zu Power Apps Studio*, <https://docs.microsoft.com/de-de/power-apps/teams/understand-power-apps-studio>, Aufgerufen am 25.02.2022.
- [33] Microsoft, *Was sind modellgesteuerte Apps in Power Apps?*, <https://docs.microsoft.com/de-de/power-apps/maker/model-driven-apps/model-driven-app-overview>, Aufgerufen am 03.03.2022.
- [34] Microsoft, *Erstellen einer modellgesteuerten App-Sitemap mit dem Sitemap-Designer*, <https://docs.microsoft.com/de-de/power-apps/maker/model-driven-apps/create-edit-app>, Aufgerufen am 03.03.2022.
- [35] Microsoft, *Was sind Power Apps-Portale?*, <https://docs.microsoft.com/de-de/power-apps/maker/portals/overview>, Aufgerufen am 03.03.2022.
- [36] Microsoft, *WYSIWYG-Editor*, <https://docs.microsoft.com/de-de/power-apps/maker/portals/compose-page>, Aufgerufen am 04.03.2022.
- [37] Microsoft, *Power Apps-Portalstudio*, <https://docs.microsoft.com/de-de/power-apps/maker/portals/portal-designer-anatomy>, Aufgerufen am 04.03.2022.

- [38] Microsoft, *Übersicht über die Portalverwaltungs-App*, <https://docs.microsoft.com/de-de/power-apps/maker/portals/configure/configure-portal>, Aufgerufen am 06.03.2022.
- [39] N. Doelman, *Overview of the PowerApps Portals Management App*, <https://readyxrm.blog/2019/08/16/overview-of-the-powerapps-portals-management-app/>, Aufgerufen am 06.03.2022.
- [40] C. Creative, *What is ServiceNow?*, https://community.servicenow.com/community?id=community_question&sys_id=cdc2afe6db2590505ed4a851ca9619da, Aufgerufen am 09.03.2022.
- [41] ServiceNow, *IT Service Management*, https://docs.servicenow.com/de-DE/bundle/sandiego-it-service-management/page/product/it-service-management/reference/r_ITServiceManagement.html, Aufgerufen am 09.03.2022.
- [42] ServiceNow, *Preisgestaltung für ServiceNow IT Service Management*, <https://www.servicenow.de/lpgp/pricing-itsm.html>, Aufgerufen am 09.03.2022.
- [43] ServiceNow, *Guided Application Creator*, <https://docs.servicenow.com/en-US/bundle/sandiego-application-development/page/build/guided-app-creator/concept/guided-app-creator.html>, Aufgerufen am 09.03.2022.
- [44] ServiceNow, *App Engine Studio*, <https://docs.servicenow.com/de-DE/bundle/sandiego-application-development/page/build/app-engine-studio/concept/exploring-aes.html>, Aufgerufen am 03.04.2022.
- [45] ServiceNow, *UI Builder*, <https://docs.servicenow.com/de-DE/bundle/sandiego-application-development/page/administer/ui-builder/concept/ui-builder-overview.html>, Aufgerufen am 09.03.2022.
- [46] ServiceNow, *What is an Experience?*, https://developer.servicenow.com/dev.do#!/learn/courses/rome/app_store_learnv2_uibuilder_rome_ui_builder/app_store_learnv2_uibuilder_rome_create_pages_in_ui_builder/UCP_WhatIsAnExperience_rome, Aufgerufen am 08.03.2022.
- [47] ServiceNow, *ServiceNow Studio*, https://docs.servicenow.com/de-DE/bundle/sandiego-application-development/page/build/applications/concept/c_ServiceNowStudio.html, Aufgerufen am 13.04.2022.
- [48] Mendix, *What Is Mendix?*, <https://www.mendix.com/evaluation-guide/what-is-mendix/>, Aufgerufen am 13.02.2022.

- [49] Mendix, *Why Was Mendix Founded?*, <https://www.mendix.com/evaluation-guide/why-founded/>, Aufgerufen am 13.02.2022.
- [50] Mendix, *Flexible Pricing for Teams of All Sizes*, <https://www.mendix.com/pricing/>, Aufgerufen am 13.02.2022.
- [51] Mendix, *Developer Portal Guide*, <https://docs.mendix.com/developerportal/>, Aufgerufen am 18.02.2022.
- [52] Mendix, *Studio 9 Guide General Info*, <https://docs.mendix.com/studio/general/>, Aufgerufen am 18.02.2022.
- [53] Mendix, *Microflows*, <https://docs.mendix.com/studio/microflows/>, Aufgerufen am 18.02.2022.
- [54] Mendix, *Studio Pro Overview*, <https://docs.mendix.com/refguide/studio-pro-overview/>, Aufgerufen am 19.02.2022.
- [55] Microsoft, *Einen benutzerdefinierten Connector auf der Grundlage einer OpenAPI-Definition erstellen*, <https://docs.microsoft.com/de-de/connectors/custom-connectors/define-openapi-definition#import-the-openapi-definition-for-power-automate-and-power-apps>, Aufgerufen am 04.03.2022.
- [56] ServiceNow, *Business rules*, https://docs.servicenow.com/en-US/bundle/sandiego-application-development/page/script/business-rules/concept/c_BusinessRules.html, Aufgerufen am 27.01.2022.
- [57] Microsoft, *Grundlagen der Delegierung in einer Canvas-App*, <https://docs.microsoft.com/de-de/power-apps/maker/canvas-apps/delegation-overview>, Aufgerufen am 12.03.2022.
- [58] Mendix, *App Security*, <https://docs.mendix.com/refguide/project-security/>, Aufgerufen am 07.04.2022.
- [59] Mendix Technology BV 2022, *APM 1 Reference Guide*, <https://docs.mendix.com/addons/apd-addon/rg-one-apm/>, 2022. Aufgerufen am 08.04.2022.
- [60] Mendix, *Performance Tool*, <https://docs.mendix.com/addons/apd-addon/rg-one-performance-tool/>, Aufgerufen am 08.04.2022.
- [61] Mendix, *Three tools to test your Mendix application*, <https://www.mendix.com/blog/three-tools-to-test-your-mendix-application/>, Aufgerufen am 08.04.2022.

- [62] OutSystems, *User Roles*, https://success.outsystems.com/Documentation/11/Developing_an_Application/Secure_the_Application/User_Roles, Aufgerufen am 11.04.2022.
- [63] OutSystems, *View the Environment Logs and Status*, https://success.outsystems.com/Documentation/11/Managing_the_Applications_Lifecycle/Monitor_and_Troubleshoot/View_the_Environment_Logs_and_Status, Aufgerufen am 12.04.2022.
- [64] OutSystems, *Monitoring OutSystems applications*, https://success.outsystems.com/Documentation/Best_Practices/Performance_and_Monitoring/Monitoring_OutSystems_applications, Aufgerufen am 12.04.2022.
- [65] Microsoft, *Monitor-Übersicht*, <https://docs.microsoft.com/de-de/power-apps/maker/monitor-overview>, Aufgerufen am 12.04.2022.
- [66] Microsoft, *Test Studio*, <https://docs.microsoft.com/de-de/power-apps/maker/canvas-apps/test-studio>, Aufgerufen am 12.04.2022.
- [67] Microsoft, *Anwenden von Geschäftslogik mit Client-Skripting in modellgesteuerten Anwendungen mit JavaScript*, <https://docs.microsoft.com/de-de/power-apps/developer/model-driven-apps/client-scripting>, Aufgerufen am 19.04.2022.
- [68] Microsoft, *Erste Schritte mit modellgesteuerter App-Anpassung durch Code*, <https://docs.microsoft.com/de-de/power-apps/developer/model-driven-apps/supported-customizations#unsupported-customizations>, Aufgerufen am 19.04.2022.
- [69] Microsoft, *Microsoft Power Platform Build Tools für Azure DevOps*, <https://docs.microsoft.com/de-de/power-platform/alm/devops-build-tools>, Aufgerufen am 19.04.2022.
- [70] OutSystems, *Integrating OutSystems with your ecosystem*, https://success.outsystems.com/Documentation/Best_Practices/Development/Integrating_OutSystems_with_your_ecosystem#DevOps_and_CI_CD_integration, Aufgerufen am 19.04.2022.
- [71] OutSystems, *Building an OutSystems pipeline with Azure DevOps*, <https://github.com/OutSystems/outsystems-pipeline/wiki/Building-an-OutSystems-pipeline-with-Azure-DevOps>, Aufgerufen am 19.04.2022.

- [72] OutSystems, *Building an OutSystems pipeline with Jenkins*, <https://github.com/OutSystems/outsystems-pipeline/wiki/Building-an-OutSystems-pipeline-with-Jenkins>, Aufgerufen am 19.04.2022.
- [73] OutSystems, *Integrate OutSystems with external version controls*, https://success.outsystems.com/Documentation/How-to_Guides/DevOps/Integrate_OutSystems_with_external_version_controls, Aufgerufen am 19.04.2022.
- [74] OutSystems, *How does OutSystems provide vertical scalability?*, <https://www.outsystems.com/evaluation-guide/how-does-outsystems-provide-vertical-scalability/>, Aufgerufen am 19.04.2022.
- [75] OutSystems, *How does OutSystems provide horizontal scalability?*, <https://www.outsystems.com/evaluation-guide/how-does-outsystems-provide-horizontal-scalability/>, Aufgerufen am 19.04.2022.
- [76] OutSystems, *Scalability*, <https://www.outsystems.com/evaluation-guide/scalability/>, Aufgerufen am 19.04.2022.
- [77] Mendix, *Java Actions*, <https://docs.mendix.com/refguide/java-actions/>, Aufgerufen am 24.04.2022.
- [78] Mendix, *CI/CD*, <https://www.mendix.com/evaluation-guide/app-lifecycle/cicd/>, Aufgerufen am 20.04.2022.
- [79] Mendix, *Implement a Simple CICD Pipeline with Mendix APIs*, <https://docs.mendix.com/howto/integration/implement-cicd-pipeline/>, Aufgerufen am 20.04.2022.
- [80] Mendix, *Scaling Your Environment in Mendix Cloud v4*, <https://docs.mendix.com/developerportal/deploy/scale-environment/>, Aufgerufen am 20.04.2022.
- [81] ServiceNow-Valor, *Instance Performance*, https://community.servicenow.com/community?id=community_question&sys_id=bb8c8fa5db9cdbc01dcaf3231f9619a8, Aufgerufen am 24.04.2022.
- [82] OutSystems, *OutSystems Logic*, https://success.outsystems.com/Documentation/11/Reference/OutSystems_Language/Logic, Aufgerufen am 03.02.2022.