HOCHSCHULE DER MEDIEN

Herausgeber Prof. Dr. Arno Hitzges

Schriftreihe Bachelor-Resümee

Forschungsbereich **Low-Code-Technologie**

Entwicklung einer Mobilapplikation mit Hilfe von Microsoft Power Apps und Power Automate

Anwendungsentwicklung zur Unterstützung einer Hochschulvorlesung

Vivienne Wolf

Impressum

Hochschule der Medien

Nobelstrasse 10

70569 Stuttgart

www.hdm-stuttgart.de

0711 8923-0

Autor

Vivienne Wolf

Betreuer

Prof. Dr. Arno Hitzges

Datum

Februar 2023

Wirtschaftsingenieurwesen Medien

www.hdm-stuttgart.de/wing

hitzges@hdm-stuttgart.de

0711/8923-2634

Layout

Jochen Riegg

Fotos und Illustrationen

Innenteil: Vivienne Wolf

Bachelor-Resümee

Entwicklung einer Mobilapplikation mit Hilfe von Microsoft Power Apps und Power Automate

Anwendungsentwicklung zur Unterstützung einer Hochschulvorlesung

Vivienne Wolf

Februar 2023

Vivienne Wolf

Vivienne Wolf begann das Studium Wirtschaftsingenieurwesen Medien an der Hochschule der Medien Stuttgart mit dem Schwerpunkt Digital Publishing Technologies zum Wintersemester 2019. Im Rahmen ihrer Bachelorarbeit entwickelte sie eine Mobilapplikation zur Unterstützung der Vorlesung Content Management Systeme mit Hilfe der Low-Code-Plattform Microsoft Power Apps und wertete in diesem Rahmen die Stärken und Schwächen der Plattform aus.

1. Einleitung

Die steigende Nachfrage nach technischen Lösungen und Softwaresystemen verlangt den Informationstechnologie (IT)-Dienstleistern effiziente und sichere Lösungen in kurzer Zeit, aber dennoch in höchster Qualität ab. Durch den ständigen Kontakt der Nutzer mit digitalen Anwendungen sowohl im Arbeits- als auch im privaten Umfeld besteht die Erwartungshaltung, dass betriebliche Abläufe unmittelbar, einfach und mit hoher Geschwindigkeit ablaufen sollen. Diese Forderung führt dazu, dass die zeitnahe Entwicklung von Geschäftsanwendungen bei den IT-Dienstleistern als größte Herausforderung angesehen wird [1]. Zudem fehlt es für die Digitalisierung der Unternehmen immer mehr an Fachkräften. Die Zahl freier Stellen für IT-Fachkräfte stieg im Jahr 2021 branchenübergreifend auf 96.000, was einen Anstieg von 12 Prozent zum Vorjahr ausmacht [2]. Daraus resultieren weniger Zeit für eine zunehmende Anzahl an Projekten und damit ein verlängerter *time to market*-Prozess für neue Softwaresysteme [3].

Um dem Fachkräftemangel zu begegnen und die benötigten Softwarelösungen bereitzustellen, liegt ein Lösungsansatz darin auch Nicht-Entwickler in die Erstellung von Anwendungen zu integrieren. Aus diesem Grund wächst die Bedeutung am Low-Code-Programmierparadigma. Dieses spiegelt eine Möglichkeit wider, durch minimalen manuellen Programmieraufwand, Software zu entwickeln. Durch die geringe Menge an erforderlichem Quelltext sinkt der Gesamtaufwand für die Entwicklung einer Anwendung sowie die Vorabinvestitionen wie Einrichtung, Schulung und Bereitstellung [4]. Für diese Technologie werden **Low-Code-Entwicklungsplattformen** genutzt, welche Ansätze aus dem Bereich der modellbasierten und der visuellen Softwareentwicklung nutzen [5].

Info

Low-Code-Entwicklungsplattformen ermöglichen eine schnelle Erstellung und Bereitstellung von Businessanwendungen mit sehr geringem Programmieraufwand.
Citizen Developer sind Menschen außerhalb der IT-Abteilung mit wenigen bis gar keinen Programmierkenntnissen, die sich in den Softwareentwicklungsprozess einbringen [6].

Neben der geringen Verwendung von Quellcode und der Entwicklung von Anwendungen durch **Citizen Developer**, verspricht die Low-Code-Technologie eine Beschleunigung der Softwareentwicklung. Durch das Low-Code-Prinzip und mit Hilfe von Drag-and-Drop-Funktionen, vorgefertigten Benutzeroberflächen und Modellen für Geschäftsabläufe, verringert sich die benötigte Entwicklungszeit. In der PEAK Matrix 2022 der Everest Group geben 76% der Low-Code-Kunden an, in weniger als sechs Monaten die Anwendung in die Produktionsumgebung einbetten zu können und damit zu einem Go-Live zu gelangen [7]. Die Technologie zeigt jedoch auch ihre Herausforderungen: Die Eigenständigkeit der Fachabteilungen und das schnelle Bereitstellen von Anwendungen für unterschiedliche Aufgaben kann dazu führen, dass die IT-Abteilungen den Überblick über die im Unternehmen eingesetzten Anwendungen verlieren. Dies wiederum begünstigt, dass im Unternehmen Anwendungen genutzt werden, die nicht von der IT geprüft und freigegeben wurden. Zudem kann es zu Problemen kommen, wenn Citizen Developer von aktiv verwendeten Geschäftsanwendungen das Unternehmen verlassen. Unter Umständen kann nicht mehr auf die Anwendung zugegriffen werden, um Wartungen oder Änderungen vorzunehmen [8].

2. Anwendungsfall

Im Studiengang Wirtschaftsingenieurwesen Medien an der Hochschule der Medien (HdM) in Stuttgart werden die Studierenden für eine Karriere im Medien- und Informationszeitalter vorbereitet. Nach Abschluss des Studiengangs sollen die Absolventen als eine Schnittstelle zwischen der kreativen Arbeit von Designern und der technischen Arbeit von Entwicklern agieren [9].

Während des dritten Semesters belegen die Studierenden des Schwerpunktes Informationstechnologie das Modul Contentmanagement Systeme (CMS). Im Rahmen dieser Veranstaltung lernen die Studierenden grundlegendes Wissen über CMS und erhalten Einblicke in Wordpress, Microsoft SharePoint sowie Microsoft Power Automate. Aufgrund der steigenden Relevanz von Low-Code-Entwicklungsplattformen, sollen die Studierenden in Zukunft zusätzlich zu Power Automate weitere Produkte der Microsoft Power Platform kennen lernen. Über die Low-Code-Plattform **Power Apps** sollte ihnen deshalb eine zur Vorlesung CMS begleitende Mobilapplikation zur Verfügung gestellt werden. Die Applikation sollte dabei folgende Handlungsszenarien enthalten: Vorlesungsunterlagen, Dokumentenupload, Quick-Tests sowie Umfragen. Zusätzlich zur Anwendungsentwicklung sollte im Umfang der Arbeit untersucht werden, inwieweit die ermittelten Anforderungen an die Anwendung durch die Low-Code-Entwicklungsplattform realisiert werden konnten. Außerdem wurde kritisch betrachtet, inwiefern die Entwicklung über Power Apps dem Low-Code-Programmierparadigma gerecht wird und welche Stärken und Schwächen im Vergleich zur herkömmlichen Softwareentwicklung festgestellt werden konnten.

Info

Power Apps ist eine von Microsoft realisierte Low-Code-Entwicklungsplattform zum Erstellen von Mobil- und Webanwendungen für die gezielte Problemlösung in Organisationen und Teil der Microsoft Power Platform.
Die **Microsoft Power Platform** bietet Unternehmen die Möglichkeit mit Hilfe von Workflows, Datenvisualisierung und weiteren Tools Lösungen zu entwickeln, die die sich ständig ändernden Anforderungen erfüllen und die geschäftliche Agilität steigern [10].

3. Anforderungsanalyse

Für die Ermittlung der relevanten Anforderungen wurden zunächst die **Stakeholder** identifiziert, um deren Ziele und Erwartungen an die Applikation zu benennen. Dabei konnten für die zu entwickelnde Anwendung vier Stakeholder-Kategorien bestimmt werden: Betreiber, Benutzer, Entwickler und IT-Administratoren. Da zum Betreiber Microsoft kein direkter Kontakt hergestellt werden konnte, wurden dessen Interessen näherungsweise aus einer Online-Recherche ausgelesen. Die Benutzer der App konnten in Gesprächen befragt werden. Die zukünftige Administrator-Rolle konnte vorab nicht definiert werden, weshalb die Bedürfnisse dieses Stakeholders auf Annahmen basieren.

Info

Stakeholder sind Anspruchsträger. Sie sind Teil der System-Umgebung und haben deshalb direkten Einfluss auf die Anforderungen an die zu entwickelnde Anwendung [11].

Für die weitere Anforderungsanalyse wurde ein Use Case Modell genutzt. Dieses stellte eine Möglichkeit dar, einzelne Funktionen der Anwendung aus der Benutzersicht in logisch zusammengehörige und handliche, funktionale Einheiten zu gliedern [12]. Dies erlaubte eine abstrakte und detaillierte Sicht auf das System, dessen Umgebung und die Nutzerinteraktion. Das Modell besteht aus Use Case Spezifikationen und einem Use Case Diagramm. Während die Use Case Spezifikation jeweils einen Anwendungsfall (Use Case) im Detail beschreibt, ist das Use Case Diagramm eine grafische Darstellung der Menge aller Use Cases, die ein Aktor auslösen kann [12]. Basierend auf den gewünschten Handlungsszenarien der Applikation wurden zwölf Use Cases modelliert. Es handelt sich demnach um Anwendungsfälle und Aktionen, die im Bereich der Vorlesungsunterlagen, dem Dokumentenupload, der Quick-Tests und der Umfragen ausgeführt werden können. Das zugehörige Use Case Diagramm wurde in Abbildung 1 dargestellt. Es veranschaulicht die Menge aller zwölf dokumentierten Use Case Spezifikationen und setzt diese in Zusammenhang zueinander.

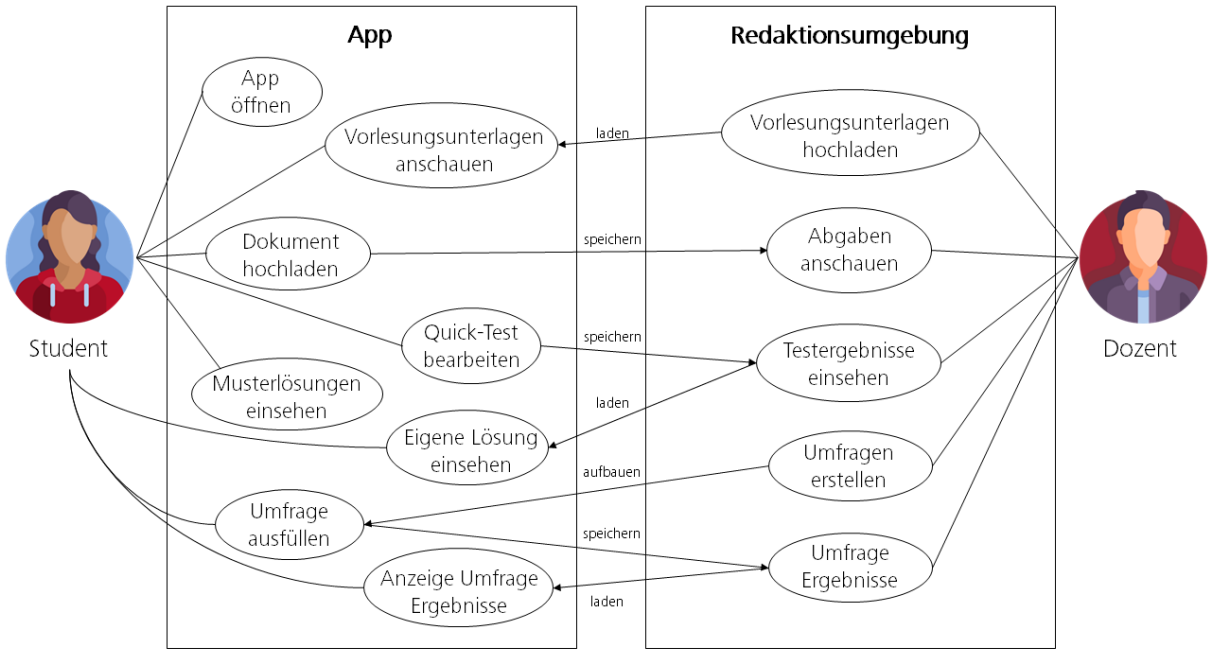

Abbildung 1: Use Case Diagramm aus der Menge aller spezifizierten Anwendungsfälle. (Links): Interaktionen der Studierenden mit der App, (Rechts): Interaktion des Dozenten mit der Redaktionsumgebung.

Aus den Zielen und Erwartungen der Stakeholder sowie den zwölf Use Cases ließen sich 57 konkrete Anforderungen an die Applikation ableiten. Diese sind sowohl funktionaler als auch nichtfunktionaler Art. Sie wurden in einer Anforderungsdokumentation gemäß IEEE Standard ausformuliert und stakeholdergerecht zusammengestellt [13]. Im letzten Schritt der Analyse wurden die Anforderungen priorisiert. Von den insgesamt 57 Anforderungen wurden 45 mit der Priorität hoch, acht mit mittel und vier mit niedrig bewertet. Die Umsetzung von Anforderungen der Priorität hoch stand dabei an erster Stelle. Anforderungen mittlerer Bewertung entsprachen im Projekt sogenannten Wunschanforderungen. Die Umsetzung dieser Anforderungen war gewünscht, jedoch nicht zwingend notwendig. Niedrig priorisierte Anforderungen waren für den Erfolg des Projektes nicht relevant.

4. Entwurfskonzept

Der Entwurf der App war der Teil des Projektes, in welchem konkretisiert wurde, was in der prototypischen Realisierung auf Grundlage der Informationen aus der Anforderungsanalyse entwickelt werden sollte. In dieser Phase wurden das User Interface (UI) Design sowie die Architektur des Datenflusses vorbereitet. Das konzeptionelle Design bildete dabei den Gesamtentwurf der App ab. Darin enthalten war der Navigationsfluss der Applikation sowie die Definition der Benutzeroberflächen durch erste Entwurfsskizzen und Wireframes. Für die Nutzernavigation wurde die Hierarchiestruktur gewählt. Diese besteht aus einer Menüseite und weiteren Unterseiten. Durch das Anklicken von Icons oder Schaltflächen gelangt der Nutzer innerhalb eines Funktionsbereichs stets weiter. Der Vorteil dieser Navigationsstruktur ist, dass der Nutzer sein Ziel schnell erreicht. Außerdem ist diese Art der Navigation bereits aus beispielsweise Ordnerstrukturen auf dem Desktop oder anderen Mobilapplikationen bekannt [14]. Die Wireframes auf Basis erster händischer Entwurfsskizzen nehmen Bezug auf jede einzelne Ansicht in der App sowie die detaillierte Beschreibung der Funktionen. Insgesamt entstanden 16 Wireframes, welche die Vorlage für die Umsetzung des UI-Designs in Power Apps bildeten. Abbildung 2 zeigt die schematische Darstellung der Funktion *Umfragen* in Form von Wireframes.

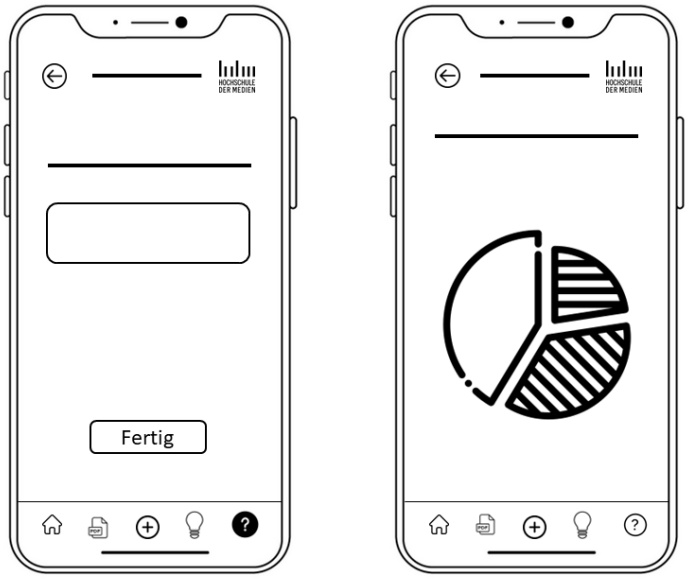

Abbildung 2: Schematische Darstellung der Funktion Umfragen in Form von Wireframes.

Ergebnis des darauffolgenden architektonischen Entwurfs war die Datenlogik sowie -struktur der App. Die Begrenzung auf SharePoint als einziger Konnektor für den Datenfluss hilft dabei die Anwendung einfach und somit die benötigte Rechenleistung auf dem Mobilgerät, auf welchem die App ausgeführt wird, möglichst gering zu halten. SharePoint dient neben der Datenspeicherung als Redaktionsumgebung für den Dozenten der Vorlesung. Die App selbst stellt lediglich das Interface dar. Alle Nutzer der App verfügen innerhalb dieser über dieselben Rechterollen.

5. Prototypische Realisierung

In diesem Teil der Arbeit wurde die Anwendung auf Grundlage der vorangegangenen Ergebnisse mit Hilfe der Low-Code-Plattform Power Apps entwickelt. Das Ergebnis der prototypischen Realisierung war eine Mobilapplikation mit den vier Handlungsszenarien Vorlesungsunterlagen, Dokumentenupload, Quick-Tests und Umfragen. Im Folgenden wird konkret auf die Umsetzung der Funktion *Vorlesungsunterlagen* und den Einsatz von **Power Fx**-Formeln in der Power Apps-Entwicklungsumgebung eingegangen.

Info

Power Fx ist die deklarative und funktionale Programmiersprache der Microsoft Power Platform, welche in lesbarem Text ausgedrückt wird. Durch ihre prägnante und einfache Art soll Power Fx auch Citizen Developern die Chance geben, Apps entwickeln zu können [15].

Für den Zugriff auf die Vorlesungsunterlagen innerhalb der App wurde zunächst eine SharePoint Bibliothek erstellt. Diese bietet dem Dozenten die Möglichkeit Vorlesungsskripte im PDF hochzuladen. Mit Hilfe eines vertikalen Dialogs wurden die in der Bibliothek vorhandenen Dokumente in der App als Liste dargestellt (siehe Abbildung 3).

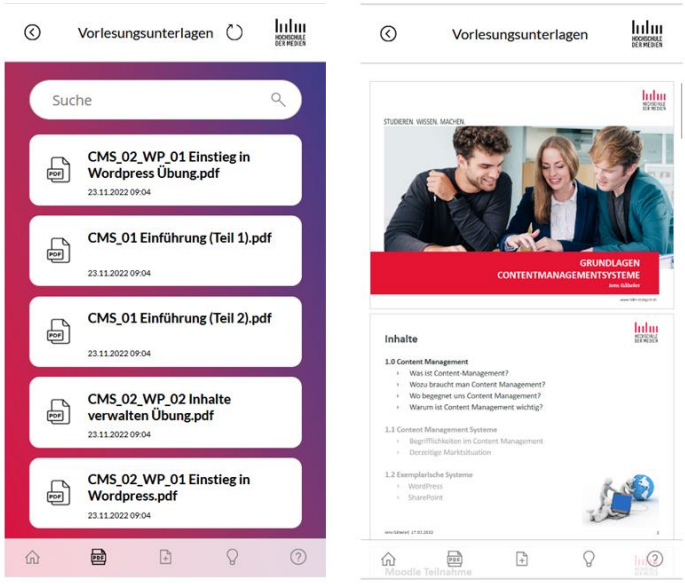

Abbildung 3: Beispielhafte Darstellung der Funktion Vorlesungsunterlagen in der CMS-Vorlesungsapp. (Links): PDF-Übersichtsliste, (Rechts): PDF-Viewer.

Die Skripte sollten in der App außerdem als PDF angezeigt werden können. Hierfür wurde das Steuerelement PDF-Viewer der Entwicklungsplattform Power Apps genutzt. Der PDF-Viewer unterstützt nur PDF-Dateien. Die Sicherheitsarchitektur von Power Apps erfordert außerdem, dass dieses Steuerelement ausschließlich HTTPS-Links unterstützt, um auf ein Dokument zuzugreifen und es in einer Ansicht darzustellen [16]. Dies bedeutet, dass zu jedem in der SharePoint Bibliothek hochgeladenen Vorlesungsskript ein Link erstellt werden musste, damit die PDF in der App geöffnet werden kann. Für diese Funktion gibt es verschiedene Lösungsansätze. Für die Implementierung in der CMS-

Vorlesungsapp wurde der Ansatz von Paul Murana [17] als Vorlage genutzt. Hierbei wurde ein Power Automate Flow erstellt, welcher beim Hochladen eines neuen Vorlesungsskriptes diesem ein Uniform Resource Identifier (URI) erstellt und zuordnet. Dafür wurde der SharePoint Bibliothek zunächst eine neue Spalte hinzugefügt, in welche der URI gespeichert werden kann. Anschließend wurde in Microsoft Power Automate ein Flow erstellt. Der Flow wird immer dann ausgelöst, wenn ein neues Dokument hochgeladen wird. Sobald der Flow ausgelöst wurde, wird der Dateiinhalt des jeweiligen Dokuments ausgelesen.

Der nächste Schritt des Flows dient dazu die Dateieigenschaften zu aktualisieren. Mit Hilfe des Dateiinhalts wird ein URI erstellt, welcher in der neuen Spalte der SharePoint Bibliothek gespeichert wird (siehe Abbildung 4).

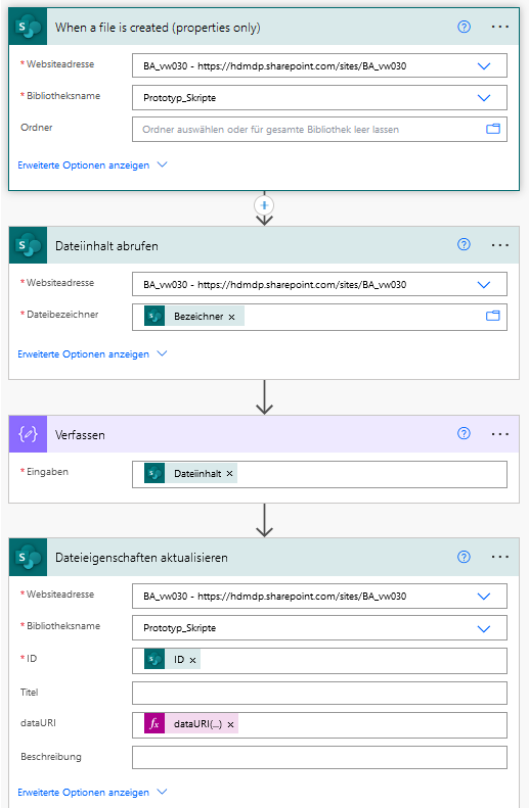

Abbildung 4: Beispielhafte Darstellung zur Aktualisierung der PDF-Dateien in der SharePoint Bibliothek durch einen Power Automate Flow. Angelegt wurden dabei vier nacheinander ausgeführte Schritte.

Durch den Flow erstellten URI konnte in Power Apps die Dokumenteneigenschaft des PDF-Viewers in der Bearbeitungsleiste auf Folgendes gesetzt werden:

```
_GalleryVorlesungsunterlagen.Selected.dataURI
```
(4.1)

Die *_GalleryVorlesungsunterlagen* ist dabei die Liste aller Skripte in der App. *Selected* gibt an, dass das vom Benutzer in der Liste angeklickte und damit ausgewählte Dokument als PDF angezeigt werden soll. *dataURI* verweist auf die Spalte in der SharePoint Bibliothek, in welcher sich der durch den Flow erstellte URI befindet. Über den URI kann das Steuerelement PDF-Viewer in Power Apps das Dokument öffnen und als PDF anzeigen (siehe Abbildung 3).

6. Auswertung

Nach den ersten Beta-Versuchen der App traten einige Herausforderungen auf. Diese konnten zunächst im Bereich der Vorlesungsunterlagen festgestellt werden. Dabei handelte es sich um lange Ladezeiten beim Öffnen der Skripteansicht. Basierend auf der verwendeten Liste konnten drei Einflussfaktoren als Ursache für die langen Ladezeiten identifiziert werden. Der Power Automate Flow hatte keinen signifikanten Einfluss auf die Ladezeit. Der Einfluss der Menge und der Größe der Dateien wurde anhand unterschiedlicher Anzahl und Größen von Dokumenten in der Bibliothek überprüft. Dabei konnte ein signifikanter Einfluss der Dokumentengröße auf die Ladezeit festgestellt werden, sobald sich mehr als fünf Dokumente in der Bibliothek befanden. Dies konnte durch die Datenlogik der App begründet werden. Als weitere Herausforderung wurde die Optimierung der Umfragenfunktion identifiziert. Die Beschränkung, jeweils nur eine Fragestellung in der SharePoint Liste eintragen zu können, erwies sich im Hinblick auf die kontinuierliche Anwendung der App in der CMS-Vorlesung als hinderlich. Dies konnte durch das Hinzufügen einer weiteren Spalte in der SharePoint Liste behoben werden. Die neue Spalte gibt den Status der Fragestellung mit den Werten *aktiv* und *inaktiv* an. So entstand die Möglichkeit mehrere Umfragen gleichzeitig in der Liste anlegen zu können und mit dem Status steuern zu können, welche Frage in der App angezeigt werden soll. Um die problemlose Verwendung der Funktion gewährleisten zu können, mussten zusätzliche Anpassungen in der Programmierung der App stattfinden, sodass stets die aktive Fragestellung ausgefiltert werden kann.

Insgesamt konnten von 57 Anforderungen, aufgeteilt in 48 funktionale und neun nicht-funktionale Anforderungen, 53 erfüllt werden. Davon sind 42 mit hoher Priorität. Drei der Anforderungen konnten nur zum Teil oder mit Einschränkungen und eine Anforderung mittlerer Priorität nicht erfüllt werden.

7. Ergebnis und Ausblick

Die Ergebnisse der Arbeit bestätigen, dass eine App inklusive Datenverbindung zu SharePoint ohne weitere IT-Fach- und Programmierkenntnisse möglich ist. Trotzdem ist es hilfreich ein fundiertes IT-Verständnis mitzubringen, wenn über Power Apps eine Anwendung programmiert werden soll. Für komplexere Anwendungen sind Grundlagen des IT-Fachwissen sowie Programmierkenntnisse erforderlich, jedoch müssen keine kompletten Programmiersprachen beherrscht werden. Im Vergleich zu einer Anwendungsentwicklung der herkömmlichen Softwareentwicklung erfordert die Entwicklung einer Low-Code-Anwendung trotz den aufgeführten Punkten bedeutend weniger IT-Fachwissen sowie Programmierkenntnisse. Die Low-Code-Entwicklungstechnologie gilt damit trotz der erforderlichen IT-Erfahrung als schneller erlernbar als eine klassische Programmiersprache inklusive zusätzlicher Frameworks.

Die Arbeit konnte des Weiteren zeigen, dass die Anwendungsentwicklung über eine Low-Code-Plattform eine gewisse Zeitersparnis mit sich bringt. Die Plattform bietet alle erforderlichen Grundlagen, wodurch der Aufbau der einzelnen Seiten der Anwendung sowie kleinere Funktionen in kurzer Zeit entwickelt werden konnten. Die prototypische Realisierung hat jedoch auch gezeigt, dass Low-Code-Anwendungen weniger flexibel sind als Anwendungen der herkömmlichen Softwareentwick-

lung. Neben der vollen Kontrolle über alle Elemente und die individuelle und flexible Anpassung dieser, hat ein Entwickler der klassischen Softwareentwicklung die Wahl welcher Programmiersprache er sich bedienen möchte. Bei der Low-Code-Entwicklung ist der Entwickler auf die bestehenden Elemente sowie die Sprache der Plattform beschränkt. Benutzerdefinierter Code kann mit Hilfe von Power Fx-Formeln nur für bestehende Eigenschaften von vorgefertigten Komponenten geschrieben werden. Dies bedeutet, ist eine Funktion in der Low-Code-Umgebung nicht durch eine bestehende UI-Komponente umsetzbar oder ein Objekt besitzt nicht die vom Entwickler benötigte Eigenschaft, bleibt diese Funktion nicht umsetzbar. Ein Hinzufügen von Elementen oder Eigenschaften durch manuellen Quellcode ist nicht möglich.

Folgend wird ein kurzer Ausblick für Weiterentwicklungsmöglichkeiten der App gegeben. Damit eine Behebung der Herausforderung der langen Ladezeit im Bereich der Vorlesungsunterlagen möglich wird, müsste eine zusätzliche Unterteilung der Skripte nach Themenbereichen stattfinden. So würde die Möglichkeit entstehen mehrere SharePoint Bibliotheken mit jeweils weniger als sechs Dateien anzulegen. Die Dateigröße würde dann keinen Einfluss mehr auf die Ladezeit haben und könnte vernachlässigt werden. Weiter wäre eine Ergänzung der Quick-Test Funktion vorstellbar. Bisher bestehen in der Anwendung fünf Tests, die von den Nutzern bearbeitet werden können. Zur Weiterentwicklung der App könnten weitere Tests implementiert werden.

8. Referenzen

[1] J. Rymer und R. Koplowitz, „Low-Code Development Platforms for AD&D Professionals - The 13 Providers That Matter Most And How They Stack Up," The Forrester Wave, 2019.

[2] „Bitkom Research," 3 Januar 2022. [Online]. Available: https://www.bitkom-research.de/de/pressemitteilung/it-fachkraefteluecke-wird-groesser-96000-offene-jobs. [Zugriff am 26 Oktober 2022].

[3] JobRouter, „Whitepaper | Low-Code für digitale Prozesse in Unternehmen," 20 Februar 2020. [Online]. Available: https://www.jobrouter.com/de/download/whitepaper-low-code-fuer-digitale-prozesse-in-unternehmen/.

[4] D. Di Ruscio, „Low-code development and model driven engineering: Two sides of the same coin?," SpringerLink, 2022.

[5] R. Waszkowski, „Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine.," 2019. [Online]. Available: https://doi.org/10.1016/j.ifacol.2019.10.060.

[6] C. Rentrop und S. Augsten, „Low-Code Entwicklung - Ansatz und Tools. Dev-Insider.," 22 Oktober 2018. [Online]. Available: https://www.dev-insider.de/low-code-entwicklung-ansatz-und-tools-a-763889/. [Zugriff am 26 Oktober 2022].

[7] Everest Group, „Low-code Application Development Platforms PEAK Matrix Assessment 2022.," Everest Group, 2022.

[8] M. Koller, „Low Code - Wie sinnvoll ist das Konzept wirklich?," 21 Januar 2021. [Online]. Available: https://www.upgreat.ch/blog/low-code-wie-sinnvoll-ist-das-konzept-wirklich. [Zugriff am 4 November 2022].

[9] H. Stuttgart, „Studienschwerpunkt Informationstechnologie," o.D.. [Online]. Available: https://www.hdm-stuttgart.de/wing/studieninteressierte/studieninhalte/studieninhalte_schwerpunkt_dpt.html. [Zugriff am 8 November 2022].

[10] M. Learn, „Was ist Microsoft Power Platform?," o.D.. [Online]. Available: https://learn.microsoft.com/de-de/training/modules/introduction-power-platform/2-what-is-power-platform. [Zugriff am 19 Dezember 2022].

[11] H. Tremp, „Agile objektorientierte Anforderungsanalyse: Planen - Ermitteln - Analysieren - Modellieren - Dokumentieren - Prüfen (erfolgreich studieren)," Springer Vieweg, 2022, p. 48.

[12] T. Cziharz, Juli 2022. [Online]. Available: https://www.ireb.org/redirect-download/?lang=de&file=ireb_cpre_handbuch_requirements_modeling_advanced_level_de_v2.0&redirect=https://www.ireb.org/content/downloads/19-handbook-cpre-advanced-level-requirements-modeling/ireb_cpre_handbuch_requirements_mod. [Zugriff am 8 Dezember 2022].

[13] IEEE, IEEE Recommended Practice for Software Requirements Specifications, IEEE Standard 830-1998, 1998.

[14] J. Semler und K. Tschierschke, App-Design: Das umfassende Handbuch., Rheinwerk Design, 2019.

[15] G. Lindhorst, „Microsoft Learn. Microsoft Power Fx Überblick - Power Platform.," 7 Juli 2022. [Online]. Available: https://learn.microsoft.com/de-de/power-platform/power-fx/overview. [Zugriff am 22 Dezember 2022].

[16] C. Moncayo, „Microsoft Learn. PDF-Viewer Steuerelement in Power Apps.," 1 April 2022. [Online]. Available: https://learn.microsoft.com/de-de/power-apps/maker/canvas-apps/controls/control-pdf-viewer. [Zugriff am 19 Dezember 2022].

[17] P. Murana, „YouTube. Super easy method to Display PDF Files Stored in SharePoint in Power Apps PDF Viewer," 21 Juli 2021. [Online]. Available: https://www.youtube.com/watch?v=GFfAkk3g1y8. [Zugriff am 19 Dezember 2022].