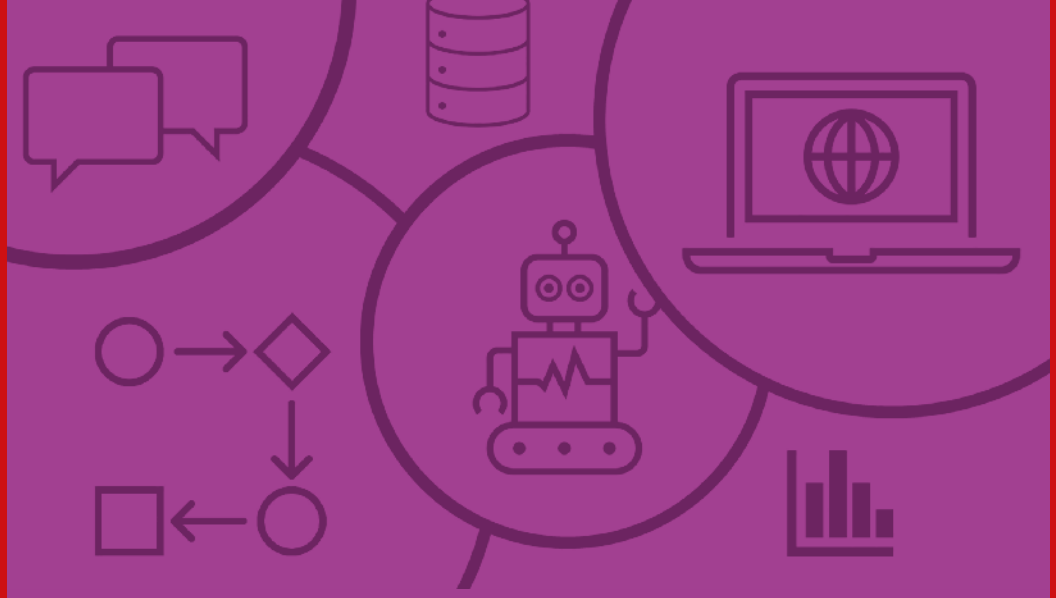

Eckhard **HAUENHERM**
Lukas **HILLESHEIM**
Dirk **LARISCH**

MICROSOFT POWER PLATFORM

IM CITIZEN DEVELOPMENT

Geschäftsanwendungen mit NoCode/LowCode entwickeln

HANSER

Disclaimer zur Barrierefreiheit

Der Carl Hanser Verlag unternimmt große Anstrengungen, um seine Produkte barrierefrei zu machen. Dazu gehört auch, dass Bilder oder Tabellen für blinde und sehbehinderte Menschen zugänglich gemacht werden. Dies geschieht durch zusätzliche beschreibende Texte (Alternativtexte), die in den Daten integriert sind. Die Alternativtexte können von assistiven Technologien (z. B. Screenreadern) vorgelesen werden. Bei der Erstellung dieser Texte kommt eine KI zum Einsatz. Die inhaltliche Verantwortung liegt weiterhin bei den Lektor:innen und Autor:innen.

Hauenherm/Hillesheim/Larisch

Microsoft Power Platform im Citizen Development

Eckhard Hauenherm
Lukas Hillesheim
Dirk Larisch

Microsoft Power Platform im Citizen Development

Geschäftsanwendungen mit NoCode/LowCode entwickeln

HANSER

Print-ISBN: 978-3-446-48081-0
E-Book-ISBN: 978-3-446-48133-6
E-Pub-ISBN: 978-3-446-48134-3

Die allgemein verwendeten Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet unter http://dnb.d-nb.de abrufbar.

Vilshofener Straße 10 | 81679 München | info@hanser.de
www.hanser-fachbuch.de
Lektorat: Sylvia Hasselbach, Kristin Rothe
Herstellung: Grazyna Lada
Covergestaltung: Thomas West
Titelmotiv: © Eckhard Hauenherm
Satz: Eberl & Koesel Studio, Kempten
Druck: Elanders Waiblingen GmbH, Waiblingen
Printed in Germany

Inhalt

1
Was ist Citizen Development?

1.1 Die Idee hinter Citizen Development

Um eine Anwendung zu entwickeln, die Nutzer bei der Verarbeitung von Informationen hilft, benötigt man zweierlei Arten von Kenntnissen. Erstens ist zu verstehen, wie die Nutzer die Informationen verarbeiten und zu welchem Zweck, zweitens benötigt man das Wissen darüber, wie die dafür erforderlichen Funktionen und Abläufe in der Programmiersprache, in der die Anwendung geschrieben wird, umgesetzt werden können.

Während die erste Art des Wissens typischerweise aufseiten der Anwender existiert – zugegebenermaßen häufig nicht sehr strukturiert – findet sich die zweite Art von Kenntnissen typischerweise bei Softwareentwicklern. Diese unterschiedliche Wissensverteilung stellt eine der größten Herausforderungen in der Softwareentwicklung dar, insbesondere in der Entwicklung organisationsspezifischer Applikationen.

Für Anwender ist es häufig sehr schwierig, ihre Tätigkeiten und Abläufe so zu beschreiben, dass sie in eine strukturierte Anwendung überführt werden können. Auf Entwicklerseite fällt es dagegen häufig schwer, den Aufbau und die Struktur einer Applikation so zu beschreiben, dass die Anwender ihre Tätigkeiten darin wiederfinden.

Die Idee hinter Citizen Development, auf Deutsch gerne als Fachbereichsentwicklung bezeichnet, ist, diese Lücke zu schließen, indem die Entwicklung näher an den Fachbenutzer herangeführt wird. Im Idealfall werden die Anwender selbst in die Lage versetzt, die von Ihnen benötigten Anwendungen zu erstellen.

1.2 Die Grundlagen von Citizen Development

Moderne Low-Code-/No-Code-Plattformen bieten die Möglichkeit, Applikationen ohne oder mit geringen Kenntnissen von Programmiersprachen zu entwickeln. Diese Plattformen zeichnen sich dadurch aus, dass sie auf der einen Seite vordefinierte Programm- oder Funktionsbausteine verwenden. Dadurch sind keine spezifischen Programmierkenntnisse notwendig, um zum Beispiel einen Programmablauf zu gestalten oder auf eine Schnittstelle zuzugreifen. Typischerweise werden einzelne Funktionen so abgebildet, dass der Anwender über eine einfache Auswahl und Eingabe von Parametern spezifische Einstellungen vornehmen kann.

Auf der anderen Seite verwenden diese Plattformen grafische Entwicklungswerkzeuge. Darin werden Ablaufstrukturen und Benutzeroberflächen für Anwender anschaubarer und die Applikation verständlicher. Die Ersteller der Anwendung können sich damit stärker auf die Funktionalität konzentrieren statt auf das Schreiben des korrekten Codes.

Seit dem Aufschwung der KI mit der Veröffentlichung von ChatGPT 3 im Jahr 2022 bieten viele dieser Plattformen zusätzlich eine entsprechende Unterstützung durch KI-gestützte Agenten an. Diese arbeiten in der Regel so, dass eine Beschreibung der gewünschten Funktionalität einen ersten Programmentwurf liefert, der dann schrittweise verfeinert werden kann.

1.3 Erfolgsfaktoren im Citizen Development

Der aus unserer Sicht am meisten unterschätzte Erfolgsfaktor im Citizen Development ist das Verständnis der Anwendungslogik. Die Marketingbroschüren der Anbieter von Low-Code-/No-Code-Plattformen vermitteln in der Regel den Eindruck, dass damit jeder Anwender in die Lage versetzt wird, gute Applikationen zu erstellen. Das ist nach unserer Erfahrung aber nur bei wirklich simplen Anwendungen oder Workflows der Fall. Zu verstehen, was in der Anwendung passieren soll und welche Funktion dafür am besten geeignet ist, ist essenziell für die erfolgreiche Erstellung der Anwendung.

Allein schon, um eine Idee dafür zu entwickeln, was das fertige Programm machen soll, muss der Ersteller ein Verständnis davon haben, wie in seinem Arbeitsprozess auf Daten zugegriffen wird, wie sie verarbeitet werden und wie sie am effizientesten zu strukturieren sind. In der Umsetzung benötigt er darüber hinaus ein nicht minder tiefes Wissen darüber, welche Funktionen in den genutzten Schnittstellen zur Verfügung stehen und wie diese arbeiten. Grundsätzlich sind drei Wissensbereiche von zentraler Bedeutung für den erfolgreichen Einsatz von Citizen Development:

- Wissen über die zu verarbeitenden Daten
  - Welche Daten nutzt die Anwendung?
  - Wie sind diese Daten strukturiert und abgespeichert?
  - Wie werden diese Daten in der Anwendung verarbeitet?
- Wissen über den Verarbeitungsprozess
  - In welchen Schritten werden die zugrunde liegenden Daten verarbeitet?
  - Was passiert in dem jeweiligen Schritt mit den Daten?
  - Welche Eingaben von Benutzern sind dabei erforderlich?
  - Gibt es Variationen im Prozess, die abgebildet werden müssen?
- Wissen über die Struktur des Programms
  - Welche Funktionen sind erforderlich?
  - Wie sind die Funktionen in der Plattform abgebildet?
  - Wie kann ein Benutzer mit der Anwendung interagieren?

Auch wenn die KI uns einen Teil dieses Wissens ersetzen kann, ist Citizen Development immer noch da am erfolgreichsten, wo es von technisch versierten Anwendern wie zum Beispiel Administratoren oder auch Softwareentwicklern als Unterstützung für die Entwicklung unternehmensspezifischer Anwendungen eingesetzt wird. Hier kann der Ansatz seine Vorteile ausspielen und den Entwicklungsaufwand deutlich reduzieren, allein dadurch, dass das Schreiben des Codes entfällt oder zumindest stark reduziert wird.

1.4 Einsatz im Unternehmen

In Zeiten eines Fachkräftemangels in den IT-Abteilungen verspricht der Einsatz von Citizen Development in Unternehmen einen deutlichen Effizienzgewinn. Werden Anwender in die Lage versetzt, ihre Anforderungen hinsichtlich der Automatisierung und Digitalisierung von Arbeitsprozessen selbst in Anwendungen umzusetzen, scheint sich der Arbeitsaufwand in der IT zu verringern. Die Frage ist aber, ob dieser Effekt tatsächlich so linear ist, wie er auf den ersten Blick erscheint. Der Einsatz von Citizen Development bringt nämlich auch einige Herausforderungen mit sich.

1.4.1 Die Gefahr einer Schatten-IT

Schon vor der Idee des Citizen Developments gab es in Organisationen vielfach Anwender, die sich ihre eigenen Anwendungen erstellt haben. Die Gründe dafür waren vielfältig. Manchmal wollten sie einfach nicht auf eine Umsetzung ihrer Anforderun-

gen durch die IT-Abteilungen warten, manchmal fehlte selbst in der IT-Abteilung das entsprechende Know-how. Oder die Anforderungen waren so spezifisch, dass sich die IT der Organisation dafür nicht zuständig fühlte.

Dabei entstanden in vielen Unternehmen Access-Datenbanken, Makro-gesteuerte Excel-Mappen oder auch einfache Webanwendungen, in denen Unternehmensdaten verarbeitet wurden. Wie die Anwendung arbeitet, welche Daten verarbeitet wurden und wie die Zugriffsrechte gesteuert werden, wurde in solchen Fällen allein durch die Anwender selbst entschieden und lag häufig außerhalb der zentralen Steuerung durch die IT. Vielfach wurden die Anwendungen der IT erst bekannt, wenn Probleme auftraten.

Citizen Development kann denselben Effekt haben. Auf Anwenderseite werden Programme erstellt, von denen die IT erst mal nichts mitbekommt. Wenn die Plattform dafür aber im Unternehmen generell bereitgestellt ist, erwarten Anwender auch hier sehr häufig bei Problemen eine Unterstützung der IT.

Ein zweiter negativer Effekt ist, dass es dadurch zu Parallelentwicklung kommen kann. Verschiedene Anwender oder Unternehmensabteilungen bilden weitgehend gleiche Anforderungen in unterschiedlichen, selbstentwickelten Anwendungen ab.

Um diesen Gefahren entgegenzuwirken, muss die IT-Abteilung des Unternehmens oder der Organisation darüber im Bild sein, welche Anwendungen die Anwender entwickeln. Bestenfalls sollte ein Prozess etabliert werden, der klar definiert, unter welchen Bedingung Anwender selbst Applikationen erstellen können, welche Prüfungen vorab erfolgen und wie die Betreuung der Anwendung im Einsatz erfolgt. Zumindest sollte aber bei der Auswahl der Plattform darauf geachtet werden, dass ein zentraler Überblick über alle in der Organisation auf der Plattform entwickelten Anwendungen verfügbar ist.

1.4.2 Aufrechterhalten eines Anwendungslebenszyklus

Mit der Gefahr der Schatten-IT ist insbesondere der Mangel eines gesteuerten Anwendungslebenszyklus verbunden. Unter einem Anwendungslebenszyklus (Application Lifecycle) versteht man einen gesteuerten Prozess zur Betreuung einer Anwendung von ihrer Entstehung, über Betrieb, Wartung und Updates bis hin zur Stilllegung. Dazu gehören insbesondere Richtlinien für die einzelnen Schritte und entsprechende nachvollziehbare Dokumentationen.

Sind die Anwendungen aber nur einzelnen Benutzern und nicht der IT im Unternehmen bekannt, kann der Prozess nur unzureichend gesteuert werden. Eine der großen Gefahren ist, dass die Anwender, die die Anwendung erstellt haben, das Unternehmen verlassen und es niemand anderes gibt, der weiß, wie die Anwendung arbeitet. Ändert sich der Prozess oder die Datenstruktur hinter der Anwendung, muss erst aufwendig die Anwendung analysiert werden.

Auf zwei Ebenen wird dieser Gefahr bei Citizen Development entgegengewirkt. Da Citizen-Development-Anwendungen in der Regel nicht mit individuellem Code erstellt werden, sondern über einfache Funktionsbausteine zusammengesetzt sind, sind sie häufig einfacher zu analysieren und zu ändern.

Zweitens bieten die meisten Low-Code-/No-Code-Plattformen eine zentrale Verwaltungsoberfläche mit einem Überblick über alle darin entwickelten und laufenden Anwendungen. Da sie sehr häufig auch entsprechende benutzerbasierte Lizenzmodelle bieten, lässt sich zumindest auch grundlegend steuern, wo und durch wen Applikationen erstellt werden können.

Ohne ein detailliertes Know-how der IT über die Plattform selbst, gestaltet sich der Betreuungsprozess der Anwendungen aber auch in solchen Plattformen schwierig. Unternehmen sollten auch über organisatorische Strukturen bei der Einführung der Plattform den einzuhaltenden Anwendungslebenszyklus definieren und insbesondere die Rollen zwischen Fachabteilungen und IT klären.

1.4.3 Datensicherheit

Anders als lokale Anwendungen in der früheren Schatten IT, sind die meisten Low-Code-/No-Code-Plattformen heutzutage Cloud-basiert. Einer der großen Vorteile liegt dabei in der Möglichkeit, über entsprechende Schnittstellen und Connectoren nicht nur auf unternehmensintern Ziele zuzugreifen, sondern auch auf eine große Zahl externer Quellen und Ziele. Naturgemäß unterliegen diese nicht der Steuerung des Unternehmens. Viele Plattformen begrenzen die Nutzung der Schnittstellen nur über die Lizenzen, aber nicht auf Basis von Funktionalitäten. Werden dann vom Unternehmen keine Einschränkungen hinsichtlich der Nutzung dieser Schnittstellen vorgenommen, können interne Daten i sehr leicht an Stellen gespeichert und verarbeitet werden, an denen sie nicht den Unternehmensvorgaben entsprechend geschützt sind.

Hier ist bei der Einführung einer Citizen-Development-Strategie und bei der Auswahl der Plattform darauf zu achten, dass die Anwender nur solche Schnittstellen nutzen können, die den Sicherheitsbedürfnissen des Unternehmens entsprechen. Zwar zieht die Strategie der meisten Softwarehersteller heutzutage Unternehmen sehr stark in die Cloud, einige Anbieter bieten aber auch on-premises einsetzbare Versionen ihrer Plattform an.

1.4.4 Vorteile aus Unternehmenssicht

Sind sich Unternehmen dieser Gefahren bewusst und steuern die Einführung von Citizen Development mit einem entsprechend strukturierten Ansatz, überwiegen nach unserem Dafürhalten die Vorteile der Strategie. Ein gesteuerter Einsatz von Citizen Development bietet folgende Vorteile:

- Leichtere Integration
- Schnellere Umsetzung
- Geringere Kosten
- Flexibilität
- Stabilität
- Zukunftssicherheit
- Governance

1.4.5 Integrierbarkeit

In der Regel verwenden No-Code-/Low-Code-Plattformen einen servicebasierten Architekturansatz (Service Based Architecture). Dabei werden über standardisierte Schnittstellen (normalerweise Web-Services mit Rest-APIs) und Standardsprachen wie JSON, Anwendungen und Datenprovider abgefragt bzw. angesprochen. Da sich dies inzwischen zu einem anwendungsübergreifenden Standard entwickelt hat, können diese Plattformen Verbindungen zu einer großen Anzahl von Applikationen bieten. Allein in der Power Platform existieren inzwischen über 900 Connectoren zu anderen Diensten.

Dem Anwender werden diese Schnittstellen, basierend auf der Definition des Anbieters, als einfache Funktionsbausteine präsentiert und notwendige Einstellungen werden über Parameter der jeweiligen Funktion abgefragt. Der Anwender benötigt also kein spezifisches Wissen über die Arbeitsweise der jeweiligen Schnittstelle.

Selbst wenn noch kein Connector für eine Zielapplikation vorliegt, ist es in der Regel möglich, über eine standardisierte Abfrage den Connector selbst zu definieren und wiederzuverwenden. Eine Neuprogrammierung entfällt damit.

Wird eine Verbindung zu einer Anwendung benötigt, die keine Standardschnittstelle bietet, können viele Plattformen oder zumindest darin erstellte Anwendungen um eigenen Code erweitert werden.

Der Integration der Anwendungen in eine größere IT-Landschaft, insbesondere wenn sie Cloud-gehosted ist, sind damit nahezu keine Grenzen gesetzt. Große Plattformen wie die Power Platform bieten zudem die Möglichkeit, aus der Cloud heraus auf On-

premises-Anwendungen zuzugreifen. Sind diese nicht direkt über aus dem Internet erreichbar, können dafür sogenannte Gateways implementiert werden, die die Anfragen aus der Cloud annehmen und an die Zielanwendung weiterleiten – natürlich unter Berücksichtigung von Sicherheitsanforderungen wie Authentifizierung und Verschlüsselung.

1.4.6 Schnelle agile Umsetzung

Da in den meisten Fällen im Citizen Development kein Programmcode geschrieben werden muss, wird nicht nur der eigentliche Erstellungsprozess beschleunigt. Auch Aufgaben wie die Bereinigung des Codes (Refactoring) und aufwendige Testverfahren zur Entdeckung von Fehlern im Code entfallen dadurch. Das Testing im Citizen Development findet nur noch auf der funktionalen Ebene statt, prüft also nur, ob die Anwendung das macht, wofür sie geplant ist.

Zugleich vereinfacht sich die Anpassung der Applikation. Neue Funktionen können einfach wie in einem Bausteinsystem hinzugefügt werden oder vorhandene Funktionen durch Einstellen einzelner Parameter angepasst werden. Dadurch kann ein sehr agiler Entwicklungsprozess eingesetzt werden. Ein solcher startet in der Regel mit einer reduzierten, aber funktionalen Erstversion (häufig als Minimal Viable Product, kurz MVP, bezeichnet). Dieser Prototyp kann dann schrittweise um neue Funktionen erweitert werden, bis die volle Funktionalität der Zielanwendung erreicht ist. Sehr häufig werden dabei ursprünglich gewünschte, aber für den Einsatz nicht notwendige Funktionen deutlich. Dadurch reduziert sich nicht nur der Umfang der Anwendung, sondern auch der damit verbunden Entwicklungsaufwand.

Aus strategischer Sicht kann der Einsatz von Citizen Development darüber hinaus die Agilität des Unternehmens insgesamt erhöhen. Bei einem unternehmensweiten Einsatz und einer fundierten Strategie werden Abteilungen und Geschäftsbereiche damit in die Lage versetzt, Lösungen auch in der Interaktion mit Geschäftspartnern und Kunden einfach und schnell selbst aufzubauen und zu betreiben.

1.4.7 Reduzierte Kosten

Schon allein aufgrund des reduzierten Aufwands bietet Citizen Development die Möglichkeit, Kosten zu sparen. Zusätzlich schlägt zu Buche, dass der Bedarf an teurem Spezialwissen zur Erstellung von Programmcode, zur Definition von Schnittstellen oder zur Exploration der Zielanwendung entfällt. Selbst der Aufwand für das Design von Benutzerschnittstellen (UI-Design) wird dadurch reduziert, dass in der Regel ein einfaches, ebenfalls auf vorgefertigten Bausteinen basierendes, Designmodell eingesetzt wird. Die Gestaltung der Anwendung ähnelt damit eher der Gestaltung einer Präsentationsfolie als einem UI-Design-Prozess.

1.4.8 Flexibilität

Schon in Abschnitt 1.4.6 haben wir über die Flexibilität eines agilen Entwicklungsprozesses gesprochen. In erster Linie basiert die Flexibilität auf der Verwendung von Funktionsbausteinen, die einfach anzupassen und auszutauschen sind. Damit bieten Citizen-Development-Anwendungen auch einen hohen Grad an Wiederverwendbarkeit. Anwendungen könne einfach kopiert und an neue Systeme oder Prozesse angepasst werden.

Um dabei der Gefahr eines Wildwuchses zu begegnen, sollten Unternehmen auch für diese Fälle einen Steuerungsprozess implementieren.

1.4.9 Stabilität

Da im Citizen Development Anwendungen nicht von Grund auf neu entwickelt, sondern aus vorgefertigten Bausteinen zusammengesetzt werden, haben sie auch einen anderen Betriebsmodus. Sie laufen auf einer Basisplattform, die durch den Betreiber der Umgebung zur Verfügung gestellt wird. Damit übernimmt der Betreiber auch die Sicherstellung des Betriebs. Die zu erwartende Verfügbarkeit ist in den Lizenzmodellen der Betreiber typischerweise im Rahmen von Service Level Agreements (SLAs) definiert.

Allerdings sind dabei auch einige Einschränkungen hinsichtlich der Performance zu berücksichtigen. In der Regel handelt es sich um Multiuser-Plattformen, auf denen viele Anwendungen gleichzeitig laufen. Jede Anwendung wird aber auf der Plattform unabhängig von anderen Anwendungen ausgeführt. Fehler in den Anwendungen, soweit sie denn auftauchen, haben keinen Einfluss auf die Stabilität anderer Anwendungen. Einzig Fehler in den Connectoren selbst oder Änderungen an den Schnittstellen aufseiten der Zielsysteme können die Stabilität der Anwendungen beeinträchtigen.

Aber auch Änderungen an den Schnittstellen selbst können häufig über die Provider durch Aktualisieren der Connectoren abgefangen werden. Damit entfällt in vielen Fällen die aufwendige Anpassung des Codes bzw. der genutzten Funktionen.

1.4.10 Zukunftssicherheit

Cloudplattformen – die meisten Citizen-Development-Plattformen sind Cloudplattformen – mit ihren agilen Entwicklungsmodellen bergen natürlich immer die Gefahr, dass Änderungen durch den Provider Anpassungen an den darin laufenden Anwendungen erfordern. Grundsätzlich wird kein Provider die Garantie geben, dass eine Citizen-Development-Anwendung auf seiner Plattform ohne Anpassungen unendlich betrieben werden kann.

Die Provider sind daran interessiert, die Plattformen kontinuierlich den technischen Möglichkeiten anzupassen, neue Funktionen zu implementieren und vorhandene Funktionen zu verbessern bzw. zu verändern sowie natürlich auch die Lizenzmodelle umzugestalten, um die Profitabilität für sie zu erhöhen. Es ist also davon auszugehen, dass Citizen-Development-Anwendungen eine mehr oder weniger kontinuierliche Betreuung oder zumindest ein Monitoring erfordern, um sie auf Dauer lauffähig zu halten.

Da, wie oben schon erläutert, CD-Anwendungen nicht monolithisch aufgebaut sind, sondern aus Funktionsbausteinen zusammengesetzt werden, können notwendige Anpassungen, wie zum Beispiel der Wechsel zu einer aktuelleren Version eines Bausteines, einfach vorgenommen werden. Manchmal sind dabei interne Abhängigkeiten zu berücksichtigen und ebenfalls anzupassen, der Zugriff auf externe Schnittstellen bleibt aber davon unberührt.

Allerdings sollte bei der Auswahl des Betreibers die Marktstabilität mitberücksichtigt werden. Dabei können folgende Fragen helfen:

- Wie lange existiert die Plattform?
- Wie groß ist der Betreiber?
- In welchem Marktumfeld bewegt er sich?
- Welche eigenen Anwendungen betreibt er unabhängig von der Plattform?
- Auf welcher technischen Basis läuft die Plattform und wem gehört sie?

1.4.11 Governance

Der Plattformansatz im Citizen-Development-Umfeld bietet auch die beste Voraussetzung, der Gefahr einer Schatten-IT durch einen fundierten Governance-Ansatz zu begegnen. Da alle Anwendungen des Unternehmens in derselben Umgebung laufen, ist es einfach, einen Überblick über alle entwickelten, laufenden oder stillgelegten Applikationen zu bekommen. Bietet die Plattform entsprechende Monitoringwerkzeuge, kann die IT sich jederzeit die gewünschten Informationen beschaffen.

Plattformen mit einem höheren Reifegrad verfügen in der Regel auch über Authentifizierungs- und Autorisierungsmechanismen, über die der Zugriff auf Erstellung und Nutzung der Plattform teilweise bis auf die Ebene der Einzelfunktionen gesteuert werden kann.

Wichtiger noch als die Zugriffssteuerung ist für eine Governance aber, dass das Unternehmen Grundregeln für die Nutzung der Plattform und die Erstellung und Betreuung der Anwendungen definiert. Dies kann zum Beispiel dadurch geschehen, dass im Unternehmen ein einheitliches Wissen über Schulungen und Communities aufgebaut und gestreut wird. Ein Governance-Prozess kann den Zugriff auf die Plattform dann

auch an den erfolgreichen Nachweis des Wissens oder zumindest den Besuch entsprechender Fortbildungen binden. Grundsätzlich sollte die IT Zugriff auf laufende Applikationen bekommen können und entsprechende Eingreifmöglichkeiten besitzen. Dies erfordert fundierte Kenntnisse über die Plattform selbst.

Der Anwendungslebenszyklus sollte klar definiert sein und es sollten entsprechende Rollen beschrieben werden, mit den Verantwortlichkeiten für Erstellung, Betrieb, Übergabe und Stilllegung von Anwendungen. Insbesondere ist daran zu denken, dass Anwendungen auch weiterlaufen müssen, wenn die Ersteller eventuell das Unternehmen schon verlassen haben. Prozesse wie diese sind in der Governance zu definieren.

Im Gegensatz zur klassischen Schatten-IT, bei der die „Anwendungen“ sehr häufig lokal auf Computern der Anwender liefen, bietet das zentrale Hosting die Möglichkeit, die Einhaltung der Governance zu überprüfen und den Plattformzugriff entsprechend zu steuern.

2
Einführung in die Power Platform

Mit der Power Platform bietet Microsoft in seiner Microsoft-365-Umgebung eine Sammlung von Werkzeugen des Citizen Development an, mit denen die sowieso schon mächtige Umgebung einfach um eigene Applikationen erweitert werden kann. Die Power Platform bietet Werkzeuge zur Erstellung digitaler Workflows (Power Automate), mobiler Apps (Power Apps), Webportale (Power Pages), Business-Intelligence-Dashboards (Power BI) sowie von Chatbots (früher Power Virtual Agents, heute Copilot Studio). Ergänzt wird die Plattform um die Möglichkeit der Nutzung einer internen Datenbank (Dataverse) sowie zahlreiche Werkzeuge und Einstellungen für die Steuerung der Governance in der Plattform (Power Platform Admin Center). Zusätzlich enthält sie nicht erst seit der Einführung von Microsoft Copilot eine KI-Komponente zur Erkennung und Verarbeitung von Dokumenteninhalten. Natürlich ist heutzutage eine weitreichende KI-Unterstützung bei der Erstellung von Applikationen über Microsofts ChatGPT-Instanz (MS Copilot) vorhanden.

Die Power Platform ist vollständig in Microsoft 365 eingebunden. Die Berechtigungssteuerung erfolgt über Entra ID. Sie verfügt über Schnittstellen zu allen in Microsoft 365 betriebenen Applikationen wie SharePoint Online, Exchange Online, Teams, Business Central, OneDrive, Office Online etc. Damit bildet sie für Organisationen, die weitestgehend auf die Microsoft Cloud setzen, die ideale Basis für die Einführung und Nutzung von Citizen Development. Zusätzlich lassen sich Drittanbieter über zahlreiche Connectoren anbinden, von denen die Power Platform inzwischen über 1300 bietet. Selbst wenn kein vordefinierter Connector existiert, lassen sich eigene Connectoren erstellen, im einfachsten Fall einfach über eine veröffentlichte API-Beschreibung des Anbieters.

Schauen wir uns die einzelnen Säulen der Power Platform im Detail an.

2.1 Power Automate

Power Automate dient innerhalb der Power Platform dazu, Arbeitsaufgaben zu automatisieren. Die Einsatzmöglichkeiten reichen dabei von einfachen Benutzeraktionen, die in derselben Weise wiederholt ausgeführt werden müssen, bis hin zur Abbildung komplexer Geschäftsprozesse in modellgetriebenen (Model-driven) Apps mit mehrschrittigen Benutzereingaben.

Die Automatisierungen in Power Automate werden allgemein als „Flows“ bezeichnet. Ein Flow besteht minimal aus einem Auslöser, dem sogenannten „Trigger“ und einer oder mehreren Aktionen. Für die Verbindung zu den unterschiedlichen Datenquellen und Anwendungen stehen die über 1300 Connectoren zur Verfügung. Einige dieser Connectoren werden von Microsoft als Premium-Connectoren bezeichnet. Das heißt, für deren Verwendung bedarf es zusätzlicher Lizenzen, die Standardlizenzen aus den O365- und M365-Plänen bieten nur Zugriff auf die Standardconnectoren.

Neben der Möglichkeit, Automatisierungen in der Microsoft Cloud ablaufen zu lassen (sogenannte Cloud-Flows), bietet Power Automate auch einen Desktop-Client, der es ermöglicht, Automatisierungen auf einem lokalen Computer laufen zu lassen. Cloud-Flows und Desktop-Flows können miteinander interagieren und Daten austauschen. Mithilfe der Desktop-Flows können auch Anwendungen in die Automatisierung eingebunden werden, die dafür keine programmatische Schnittstelle bieten. Der Desktop-Flow kann für diesen Fall einfach die Benutzerinteraktion auf der Anwendungsoberfläche simulieren, ähnlich wie es über den Makrorekorder in Excel möglich ist.

Neben dem Interagieren mit Daten und Anwendungen bietet Power Automate eine reichhaltige Auswahl an Aktionen für die Steuerung des Flows selbst. Neben der Möglichkeit, mit Schleifen, Bedingungen und Schaltern zu arbeiten, stehen auch Variablen zur Verfügung, die Daten zur Laufzeit des Flows halten können. Zusätzlich bietet die Formelsprache PowerFX einen reichhaltigen Funktionskanon zur Ad-hoc-Bearbeitung von Daten, wie z. B. Konvertierungsfunktionen oder Datenextraktionen.

Der Datenaustausch innerhalb eines Flows erfolgt mittels JSON (JavaScript Object Notation), einer Standardsprache zur Beschreibung von Datenstrukturen. Die grundlegende Struktur von JSON besteht immer aus Schlüssel-Wert-Paaren. Diese können hierarchisch ineinander gegliedert werden, sodass sich beliebig komplexe Daten damit abbilden lassen. Jede Aktion eines Flows erzeugt ein JSON-Statement. Die darin beschriebenen Objekteigenschaften können als dynamische Inhalte in anderen Aktionen weiterverarbeitet werden. Darüber hinaus gibt es Basisfunktionen zum Auslesen der JSON-Daten, für den Fall, dass nicht alle Eigenschaften aus einem JSON-Statement angeboten werden.

Einige grundlegende Einschränkungen sind bei der Arbeit mit Cloud-Flows in Power Automate zu berücksichtigen. Erstens läuft jeder Flow im Sicherheitskontext eines Benutzers ab, abhängig von der Art des Flows entweder im Kontext des Flow-Erstel-

lers oder im Kontext des Benutzers, der den Flow auslöst. Das ist bei der Planung der Datenzugriffe zu berücksichtigen. Zweitens haben Cloud-Flows nur eine begrenzte Lebensdauer von 30 Tagen. Das heißt, spätestens nach 30 Tagen wird eine Flow-Instanz beendet, unabhängig davon, ob schon alle Aktion durchgeführt wurden. Aufgrund dieser Einschränkung ist es häufig notwendig, Flows regelmäßig neu zu instanziieren, um längere Bearbeitungszeiten abzubilden. Diese Anforderung unterscheidet Power Automate Flows von anderen Automatisierungstools, wie zum Beispiel SharePoint Designer Workflows, die eine unbegrenzte Lebensdauer haben. Auf der anderen Seite bietet Power Automate mit seinen Connectoren unendlich mehr Möglichkeiten der Automatisierung, schon allein dadurch, dass die Flows über mehrere Anwendungen hinweg arbeiten können.

2.2 Power Apps

Power Apps ist die Komponente der Power Platform, in der Anwendungsoberflächen und mobile Anwendungen gestaltet werden können. Dabei werden zwei grundlegend verschiedene Arten von Apps unterstützt. Auf der einen Seite die sogenannten Canvas-Apps, die mittels Connectoren auf Datenquellen zugreifen und mit diesen interagieren können, und auf der anderen Seite die sogenannten Model-driven Apps, die ihre eigene Datenstruktur im Microsoft Dataverse, der Power-Platform-Datenbank, verwenden.

Eine Canvas-App besteht aus einem oder mehreren Screens, auf denen wie auf einer leeren Leinwand Steuerungselemente angeordnet werden können. Die Steuerungselemente, auch Controls genannt, können Daten darstellen und mit Aktionen belegt werden. Obwohl es sich um eine Art Bausteinsystem handelt, lassen sich die einzelnen Elemente frei anordnen und gestalten. Das Grundlayout der App wird am Anfang festgelegt, entweder vertikal als sogenanntes Telefon-Layout oder horizontal ausgerichtet als Tablet-Layout.

Während der Erstellungsprozess bei einer Canvas-App mit der Gestaltung der Oberfläche beginnt, startet die Erstellung einer Model-driven App mit der Definition der Datenstrukturen. Das heißt in der Datenbank, dem Microsoft Dataverse, werden Tabellen erstellt, diese mit Beziehungen hinterlegt, Ansichten und Formulare erstellt und eine Business-Logik darüber gelegt. Die Model-driven App besteht dann ebenfalls aus mehreren Seiten, auf denen typischerweise Ansichten der Tabellen hinterlegt sind. Die Gestaltung der Oberfläche ist im Vergleich zu einer Canvas-App deutlich eingeschränkt. Auch die Funktionsbausteine sind in der Regel schon über die Datenstruktur und zum Beispiel die Standardformulare der Tabellen vordefiniert.

2.3 Power Pages

Power Pages waren ursprünglich unter dem Namen Power Apps Portals Bestandteil der Power Apps. Der alte Name zeigt schon die Richtung des Einsatzzwecks. Bei Power Pages geht es darum, einfache interaktive Websites zu gestalten, die ähnlich mit Daten interagieren, wie die übrigen Anwendungen der Power Platform. Die Einbindung in die Power Platform zeigt aber auch, dass es bei Power Pages nicht etwa um das Gestalten öffentlicher Webauftritte geht, sondern um den Aufbau interaktiver Portale für die Einbindung von Kunden, Lieferanten und anderen Geschäftspartnern in interne Geschäftsprozesse. Typische Beispiele hierfür sind Buchungswebsites oder Lieferantenportale für den sicheren Austausch von Geschäftsinformationen.

Die Gestaltung der Webseiten orientiert sich stark an dem von SharePoint bekannten Baukastensystem mit seinen Webparts. Einzelnen Funktionseinheiten wie Textfelder, Bilder, Formulare etc. liegen als fertige Bausteine bereit und werden über einen grafischen Editor auf den Seiten positioniert. Die Seitenstruktur selbst wird in der Regel über eine einfache Vorlagenauswahl vordefiniert und lässt sich dann schrittweise erweitern.

Innerhalb der Website können Daten in den Tabellen des Dataverse gespeichert und verwaltet werden. Externe Daten können über Verbindungen auf Basis der Power-Platform-Connectoren als externe Tabellen in die Datenstruktur eingebunden werden. Zusätzlich können Cloud-Flows aus Power Automate in die Website integriert werden, um Informationsflüsse zu anderen Anwendungen abzubilden.

Für die Authentifizierung stehen fertige Connectoren zu vielen Identitätsanbietern wie Google, Facebook, LinkedIn oder X zur Verfügung. Für einen gesicherten Informationsaustausch sind somit nicht zwingend interne Konten erforderlich. Gerade die Einbindung mehrerer Indentitätsanbieter bietet die Möglichkeit, mit Kunden und Geschäftspartnern zusammenzuarbeiten, zu denen ansonsten keine intensive Partnerschaft besteht.

2.4 Power BI

Power BI ist eine Datenauswertungskomponente in der Power Platform. Kernaufgabe des Werkzeugs ist es, Anwendern einen Self Service für die visuelle Auswertung von Geschäftsdaten zu bieten. Das Ergebnis sind interaktive Dashboards und Berichte, die in sogenannten Arbeitsbereichen in der Cloud veröffentlicht werden und darüber in andere Anwendungen wie SharePoint oder Power Apps eingebunden werden können.

Zur Definition der Datenverbindungen und Transformation der Daten für die Auswertung nutzt Power BI eine Desktopkomponente, Power BI Desktop. Hierin lassen sich verschiedene Datenquellen anbinden und auch umfangreiche Datentransformationen gestalten. Das damit generierte Datenmodell wird in Power BI als „Semantisches Modell“ bezeichnet. Es generiert sozusagen die Bedeutung der Daten.

Auf Basis des semantischen Modells lassen sich dann visuelle Berichte gestalten. Dafür steht eine große Anzahl von Visualisierungen zur Verfügung, von einfachen Diagrammen bis hin zu geografischen Auswertungen. Die Berichte selbst können wiederum Bestandteil eines Dashboards werden, das über eine zusätzliche Komponente zur natürlichsprachlichen Abfrage der Daten verfügt. Das Dashboard, die Berichte und die dazugehörigen semantischen Modelle werden anschließend in einem Power BI Workspace veröffentlicht. Nach der Veröffentlichung kann das semantische Modell mit einer automatischen Aktualisierungsrate versehen werden, sodass die Daten regelmäßig aktualisiert werden und in Berichten und Dashboards immer aktuelle Daten angezeigt werden.

2.5 Copilot Studio (Power Virtual Agents)

Copilot Studio ergänzt die Power Platform um ein Werkzeug zur Erstellung interaktiver Chatbots, in der Power Platform als „Agenten“ bezeichnet. Ein Chatbot bietet Anwendern die Möglichkeit, über eine einfache Texteingabe in einen Dialog einzutreten, um sich schrittweise einer Problemlösung zu nähern. Typischerweise werden Agenten genutzt, um Anwender in Prozessen interaktiv zu unterstützen, zum Beispiel bei Bestellungen. Neben dem Präsentieren von Antworten bieten die Agenten in Copilot Studio auch die Möglichkeit, über Workflows Aktionen für die Anwender durchzuführen.

Für die Erstellung eines Chatbots sind im ersten Schritt Themen („Topics“) zu definieren. Dabei handelt es sich um vordefinierte Dialogverläufe, die typischerweise durch Triggerbegriffe in der Benutzereingabe aufgerufen werden. Die Dialogverläufe bieten weitere Abfragen an den Benutzer, reagieren auf die Antworten, bieten Auswahloptionen, können Workflows anstoßen oder Informationen über Datenquellen abrufen und dem Benutzer präsentieren.

Ein Chatbot kann mehrere Themen beinhalten. Der Wechsel zwischen den Themen kann entweder direkt vom Benutzer angestoßen werden oder basierend auf spezifischen Benutzerreaktionen in einem Dialog erfolgen. Typischerweise sind grundlegende Begrüßungs- und Verabschiedungsthemen im Chatbot vordefiniert. Diese dienen einfach der Aktivierung bzw. Deaktivierung des Agenten.

Mit der Einbindung von Microsoft Copilot ist für die Agenten eine KI-Komponente verfügbar, bei der Antworten mittels künstlicher Intelligenz aus hinterlegten Wissensquellen, wie zum Beispiel SharePoint-Sites, Datenbanken oder auch einfach einer Sammlung von Dateien, generiert werden können.

Die fertigen Chatbots lassen sich als Apps in Teams-Kanäle einbinden, auf SharePoint-Seiten veröffentlichen oder auch in Power Pages oder Power Apps integrieren.

2.6 Das Dataverse

Wenn die in der Power Platform erstellten Anwendungen Daten selbst persistent halten müssen, anstatt über Connectoren auf externe Datenquellen zuzugreifen, kann eine erweiterbare Datenbank innerhalb der Platform eingerichtet werden. Ursprünglich als Common Data Service bezeichnet, verwendet Microsoft inzwischen den Namen Dataverse dafür. Das Datenbankmodell basiert auf der Dynamics Datenbank und bringt schon viele Tabellen und Funktionen mit, die für den Betrieb von Geschäftsanwendungen sinnvoll sind. Die Datenbank ist aber in allen Bereichen erweiterbar. Vorhandene Tabellen können mit zusätzlichen Spalten oder weiteren Ansichten und Formularen ergänzet werden. Für spezifische Funktionen lassen sich eigene Tabellen mit allen benötigten Funktionen erstellen. Zur Abbildung der Datenlogik lassen sich zusätzliche Beziehungen zwischen Tabellen erstellen oder vorhandene Beziehungen erweitern. Damit lässt sich bei Bedarf ein komplett eigenes Datenmodell für komplexe Lösungen aufbauen. Technisch gibt es auch keine Einschränkungen hinsichtlich der Speicherkapazität der Datenbank. Allerdings ist die verfügbare Speicherkapazität für Daten, Dateien und Protokolle lizenzabhängig.

Nicht für alle Einsatzszenarien der Power Platform ist aber eine Datenbank erforderlich. Reine Canvas-Apps oder ein Großteil der Funktionen in Power Automate erfordern keine Datenhaltung in der App selbst. Andere Anwendungsbereiche, wie zum Beispiel Model-driven Apps oder Genehmigungsworkflows mittels Approvals, setzen zwingend eine Datenbank voraus, um Daten speichern zu können.

Zur Steuerung eines Anwendungslebenszyklus und zur Trennung von Produktiv- und Testumgebungen können in der Power Platform Umgebungen genutzt werden. Die Datenbanken sind dabei pro Umgebung zu gestalten. Diese bietet die Möglichkeit, bei Nutzung der Datenbank unter anderem Produktiv- und Testdaten voneinander zu trennen. Ein weiterer Vorteil bei der Nutzung verschiedener Umgebungen besteht in der Berechtigungstrennung. Jede Umgebung bietet ihre eigene Berechtigungsverwaltung, sodass Zugriffrechte pro Umgebung vergeben werden können.