Power Apps und Power Automate
Das umfassende Handbuch

DIE LESEPROBE

Kapitel 1
Digitalisierung mit der Power Platform

Die Power Platform ist eines der führenden Low-Code-Werkzeuge zur Automatisierung und Digitalisierung von Geschäftsprozessen. Mit der Power Platform können alle mit der Entwicklung loslegen und das Umfeld digitalisieren. Was Low-Code bedeutet, wie neu diese Technologie ist, und was das alles für Sie heißt, erfahren Sie in diesem Kapitel.

Um mit diesem Buch arbeiten zu können, benötigen Sie im Grunde nicht mehr als eine gute Portion Neugier und IT-Affinität. Sicherlich ist es hilfreich, wenn Sie sich in der Vergangenheit mit den Anwendungen Microsoft Excel oder Access auseinandergesetzt haben, Voraussetzung ist das allerdings nicht.

In den folgenden Kapiteln zeige ich Ihnen, wie Sie Softwareanwendungen erstellen und wie Sie Prozesse digitalisieren und automatisieren, ohne über tiefgreifende Programmierkenntnisse zu verfügen.

So lernen Sie in Abschnitt 1.1 den Low-Code-Ansatz kennen, durch den jeder zum Softwareentwickler werden kann, und erfahren, warum dieser Ansatz auch IT-Administratoren entspannt in die Zukunft blicken lässt.

In Kapitel 2 gebe ich Ihnen einen Überblick über die Power Platform sowie ihre Bestandteile und Einsatzmöglichkeiten. Sie lernen die wichtigsten Begriffe kennen und erhalten zum Ende einen Einblick in den Lizenzdschungel.

In den beiden folgenden Kapiteln lernen Sie das Maker Portal und das Power Apps Studio kennen. Das Maker Portal aus Kapitel 3 dient Ihnen als Verwaltungsoberfläche und Startpunkt für Ihre Digitalisierungsmaßnahmen, während Sie in Kapitel 4 mit dem Power Apps Studio in Ihre Entwicklungsoberfläche eintauchen.

Kapitel 5 widmet sich der Programmierung mit Power FX. Hier lernen Sie die Grundkonzepte der Programmierung in Power Apps kennen und erhalten einen Überblick über die einzelnen Elemente einer Canvas App sowie deren Anwendung. In Kapitel 6 gehe ich auf die beiden, in das Design Studio integrierten Werkzeuge zum Testen und Analysieren Ihrer App ein: Das Power Apps Test Studio sowie den Power Apps Monitor.

In den Kapitel 7 und 8 zeige ich Ihnen weitere Einsatzbereiche von Canvas Apps. Hier lernen Sie, wie Sie Dataverse for Teams-Apps erstellen, und wie Sie Canvas Apps nutzen um Ihre SharePoint-Formulare zu designen.

In Kapitel 9 und 10 stelle ich den Automatisierungsdienst Power Automate vor. Lernen Sie hier die Unterschiede zwischen den verschiedenen Workflow-Arten kennen, und steigen Sie mit mir direkt in die Entwicklung von Cloud-Flows ein, mit denen Sie zeitgesteuerte, ereignisbasierte oder manuell ausgeführte Workflows realisieren.

Da Sie nun grundlegende Kenntnisse in der Digitalisierung und Automatisierung mit der Power Platform besitzen, zeige ich Ihnen in den Kapiteln 11 und 12 die Anwendung von ausgewählten Konnektoren und eine Auswahl verschiedener Anwendungsfälle.

Bis hierher haben Sie einiges über die Erstellung von Low-Code-Anwendungen mit der Power Platform, insbesondere Power Apps und Power Automate Cloud Flows, erfahren. Nun widmen wir uns in den Kapiteln 13 und 14 den Lösungsmappen und Umgebungen und machen einen kurzen Abstecher in Richtung möglicher Maßnahmen zur Verhinderung von ungewolltem Datenabfluss.

Kapitel 15 setzt den Schlusspunkt mit einer Übersicht über die deutsche Community sowie interessanten Projekten, Blogs und Websites, denen Sie folgen können.

Sprache

Keine Angst, das Buch ist in deutscher Sprache verfasst. Eine große Herausforderung war es allerdings, mit den unterschiedlichen Benennungen innerhalb der Power Platform umzugehen. Ich empfehle grundsätzlich, die englische Umgebungssprache zu nutzen. Da ich in meinen Projekten aber viele Menschen getroffen habe, die der englischen Sprache nicht mächtig sind, habe ich die Entscheidung getroffen, mich an der deutschsprachigen Oberfläche zu orientieren.

Wo es sinnvoll ist, verwende ich zusätzlich zu den deutschen Bezeichnungen auch die englischen Originalbegriffe und hoffe, dass so möglichst verständlich ist, welche Menüs, Elemente und Begriffe gemeint sind.

1.1 Der Low-Code-Ansatz

Wenn Sie sich mit der Power Platform beschäftigen, werden Ihnen rasch die Begriffe *Low-Code* und *No-Code* begegnen – auf Deutsch kann man von *codearm* und *codefrei* sprechen. Damit ist gemeint, dass Sie mit den richtigen Hilfsmitteln Entwicklungsaufgaben lösen können, ohne viel Code zu schreiben. Sie können also ohne viel Programmiererfahrung wiederkehrende Arbeitsschritte automatisieren oder Probleme lösen.

Dieser Ansatz steht im Zentrum der Power Platform. Microsoft verfolgt diese Idee seit einigen Jahren intensiv: *https://powerapps.microsoft.com/de-de/low-code-platform/*.

Obwohl man denken mag, dass die Begriffe No-Code und Low-Code erst mit der Corona-Pandemie in unser Leben traten, handelt es sich dabei um ein altes Konzept: nämlich Software erstellen zu können, ohne ausgebildeter Entwickler zu sein. Mich begleitet dieses Konzept, seit ich mich mit Rechnern und Informatik beschäftige.

- **Die 80er-Jahre (Construction Kits)**
  Ende der 1980er, zur Hochzeit von Commodore 64, Atari und Schneider Colour Personal Computer (CPC), gab es kaum eine Ausgabe meiner damaligen Lieblingszeitschriften »64er« und »Happy Computer«, in der kein Softwarebaukasten beworben wurde. So heißt es in einem Spieletest auf Seite 84 der Januar-Ausgabe 1988 von Happy Computer:

  ... Es ist eine Art Baukasten, um Action-Spiele selber zu schreiben, auch wenn man von Technik und Programmieren nichts versteht.

  Ein weiterer Artikel, den ich leider nicht mehr genau lokalisieren kann, sprach davon, dass man Software irgendwann nicht mehr programmieren, sondern nur noch Bausteine zusammenstecken wird.

  Da ich zu diesem Zeitpunkt dem Parser Commodore Basic V2.0 und dem 6502-Assembler verfallen war, fand ich die Vorstellung, dass man Software ohne Programmierung erstellt, gruselig und utopisch.

- **Die 90er-Jahre (Excel und Access)**
  Die Zeit schritt voran, und ich begann in den 1990ern mein Studium der Wirtschaftsinformatik. Dabei besuchte ich eine Vorlesung zum Thema Produktionsplanung und -steuerung, in der ein Unternehmensplanspiel durchgeführt wurde.

  Ziel war es, durch den geschickten Einkauf von Rohstoffen und Halbfertig-Erzeugnissen eine möglichst gewinnbringende Produktion auf die Beine zu stellen. Die Berechnungen sollten wir den Computer erledigen lassen. Also legten wir los und programmierten unsere erste Unternehmenssoftware auf Basis eines Programms der Firma Microsoft mit dem Namen Excel – Sie haben vielleicht schon davon gehört.

  Das war rückblickend meine erste Low-Code-Entwicklung, denn wir ließen schon damals umfangreiche Berechnungen mit wenigen komplizierten Befehlen ausführen. Anstatt eine For-Schleife zum Aufsummieren von Zahlen zu nutzen, reichte ein simples `=SUMME(A1:A10)`, um die Summe von zehn Feldern zu bilden und das Ergebnis in weiteren Formeln zu verwenden.

- **Die Nuller-Jahre (InfoPath)**
  Nach dem Studium war ich immer noch überzeugt, dass nichts über die echte Programmierung geht, also stieg ich in *.net* ein. Mein Weg führte mich von *VB.net* über *C#* in meine erste Selbstständigkeit. Wir schrieben das Jahr 2005, und meine ersten

Aufträge führten mich zu *MS InfoPath 2003*. Ich war relativ neu in der Technologie, die Projektanforderungen waren aus der Sicht eines Softwareentwicklers hoch und das Projektbudget sehr niedrig.

Ich sollte innerhalb von drei Tagen eine Formularanwendung mit SharePoint-Integration auf die Beine stellen. Mir schien das unlösbar. Auch wenn es ein Code-Fenster für VBScript-Funktionen gab, so ließen sich die meisten Anwendungen tatsächlich in kürzester Zeit realisieren, indem man eine XML-Datenquelle mit Eingabefeldern verband. Die notwendige Logik des Formulars wurde über einen Regel-Editor (kein InfoPath-Entwickler wird die Pop-up-Fenster-Hölle jemals vergessen!) zusammengeklickt. Wenn externe Systeme angebunden werden mussten, so konnte man über einen Assistenten Webdienste einbinden, um zum Beispiel Daten aus SAP in das Formular zu integrieren.

Mit InfoPath gab es jetzt also ein Werkzeug, mit dem man in kürzester Zeit Oberflächen erstellen und ihnen über einen Regel-Editor Leben einhauchen konnte. Auch die Anbindung externer Systeme per SOAP-Webservice war schnell zusammengeklickt. »Richtig« programmiert wurde nur, wenn andere Möglichkeiten nicht mehr ausreichten.

- **Die wundervollen SharePoint-Jahre**
  In der Zwischenzeit entwickelte sich der *SharePoint Portal Server* aka *Microsoft Office SharePoint Server* (*MOSS*) zu einer Erfolgsstory, und der *SharePoint Designer* erblickte das Licht der Welt. Wo bisher kleine Programm-Module entwickelt werden mussten, um auf Änderungen in SharePoint-Listen zu reagieren, reichten nun wenige Klicks, um einen Workflow zu erstellen, der feststellt, ob ein Listeneintrag in einer SharePoint-Liste erstellt oder geändert wurde.
- **Die Transformation vom Anwender zum Digital Citizen**
  Natürlich war man noch weit weg von individuell entwickelten SharePoint-Web-Parts, dennoch gab es im Zusammenspiel von SharePoint und MS InfoPath gefühlt keine Grenzen mehr.

  Mittlerweile war ich hauptberuflicher SharePoint-Entwickler und bemerkte einen Wandel in den Fachabteilungen. Dort, wo ich früher mit Visual Studio für meine Raketenwissenschaft bewundert wurde, fingen die Anwender an, mit mir über meine Architekturvorschläge zu diskutieren. Die Kunden merkten, dass man nicht mehr zwingend einen Entwickler brauchte, um ein Stück Software zu entwickeln, und trauten sich fortan mehr und mehr zu. Langsam, aber sicher bauten sie das technische Know-how im SharePoint-Umfeld auf, das sie für ihre Fachanforderungen benötigten.

  Abteilungen bekamen durch die zentrale IT ihre SharePoint-Site zur Verfügung gestellt und durften sich einen SharePoint Designer installieren. Externe Dienstleister zog man hinzu, wenn die Fachabteilungen an ihre Grenzen kamen. Nun konnte (fast) jeder zum Programmierer oder *Digital Citizen* werden.

1.1.1 Softwareentwicklung ohne Programmierkenntnisse?

Als ein ehemaliger Kollege, während er ein paar Kästchen und Pfeile auf ein Stück Papier schrieb, gefragt wurde, was er da tue, erwiderte er, er schreibe gerade ein Programm. Der Fragesteller war sichtlich verwirrt, da man seiner Meinung nach zur Programmierung einen Computer benötigt. »Den Computer brauche ich zum Coden«, führte mein Kollege weiter aus. »Das Programm ist der Ablaufplan, der Code ist die Sprache, mit der ich das Programm umsetze.«

Seinerzeit musste ich eine Weile darauf rumdenken. Aber es ergab Sinn, denn das eine ist, sich zu überlegen, was ich benötige, um mein Ziel zu erreichen und welche fachlichen Regeln ich berücksichtigen muss – das andere die Umsetzung. Folgende Gedanken sind für mich in der Entwicklung wichtig:

1. Welche Daten benötige ich, also welche Geschäftsobjekte (Auftrag, Person, Kunde)?
2. Wo liegen diese Daten (Dataverse, SQL-Server, SharePoint, Excel, Access)?
3. Wie hängen die Daten zusammen, welche Geschäftsregeln gibt es (Prozesse)?
4. Welche Schritte sind nötig, um das Ziel zu erreichen (Algorithmus)?
5. Welche Rollen/Personen sind an den Prozessen beteiligt?

Wenn ich mein Ziel nicht klar formulieren kann und den Weg und die Regeln nicht kenne, um mein Ziel zu erreichen, komme ich weder mit Programmierung noch mit einer Low-Code-Plattform zum Erfolg. Oder anders gesagt: Wenn ich ein LEGO-Modell bauen möchte, aber nicht weiß, welche Steine ich wie zusammensetzen muss, komme ich über die Kiste mit kleinen bunten Plastiksteinen nicht hinaus. Dasselbe gilt für die Softwareentwicklung. Deshalb beginne ich fast jedes Projekt auf einem Stück Papier. Etwas nachhaltiger sind sicherlich Tools wie *PowerPoint*, *Visio* oder *Figma*.

Geht es an die Umsetzung, zeigen sich schnell die Vorteile des No-Code-/Low-Code-Ansatzes, denn betrachtet man ein typisches kleines bis mittleres Softwareprojekt, so sind die Anforderungen und die Geschäftsregeln relativ schnell klar und können oftmals direkt umgesetzt werden.

Bei individueller Softwareentwicklung ist die große Herausforderung – besonders in größeren Unternehmen –, einen Sponsor zu finden, der Ihnen das nötige Budget zur Verfügung stellt. Dafür müssen Sie einige Runden mit den Teams aus der IT-Infrastruktur und IT-Sicherheit drehen und schlussendlich einen Entwickler finden, der das Projekt umsetzen kann. Das kann sehr aufwendig sein und sorgt dafür, dass viele gute Ideen im Sande verlaufen, bevor überhaupt mit der Umsetzung angefangen werden kann.

Mit Low-Code-Plattformen wie der Power Platform sparen Sie sich den Umweg über die IT und die Budgetverantwortlichen. Sie wollen digitalisieren, also tun Sie es ein-

fach. Low-Code-Plattformen stellen dem Anwender in der Fachabteilung Bausteine zur Verfügung, die er in einem definierten, sicheren Rahmen verwenden kann, um seine Prozesse selbst zu digitalisieren.

1.1.2 Der traditionelle Entwickler in einem Power-Platform-Projekt

Sind Sie ein Herzblut-Entwickler und haben den Auftrag, eine Power App oder einen Power Automate Cloud Flow zu entwickeln? Seien Sie offen für neue Ansätze, machen Sie sich frei!

Ich schätze, ich lehne mich nicht zu weit aus dem Fenster, wenn ich sage, dass vermutlich jeder Hardcore-Entwickler zunächst die Hände über dem Kopf zusammenschlägt, sobald er sich das erste Mal an einer Low-Code-App versucht.

»Das ist ja furchtbar! Das ist doch keine Entwicklung!«, habe ich nicht selten gehört. Aber man kann das auch anders sehen: Jeder Entwickler hat sein individuelles Repertoire an Tools, die er für seine Arbeit nutzt. So ist auch die Power Platform ein Tool, um Aufgaben bis zu einer bestimmten Anforderungsgrenze umzusetzen.

Nun kommt der Entwickler ins Spiel, denn nicht selten ist es einfacher, eine *Azure Function* zu schreiben, die man als angepasste (oder auch benutzerdefinierte) Konnektoren in einen Flow oder eine App integriert, als Umwege zu suchen, um diese Anforderung ohne Entwickler zu erfüllen.

Erstellen von angepassten (benutzerdefinierten) Konnektoren

Wollen Sie in die Entwicklung von benutzerdefinierten Konnektoren einsteigen, so empfehle ich Ihnen folgende Quelle des Microsoft-Learn-Portals:

https://learn.microsoft.com/de-de/connectors/custom-connectors/define-blank.

Aber auch unabhängig von solchen technischen Herausforderungen lohnt sich für jeden Citizen Developer der Austausch mit einem professionellen Developer, denn auch wenn man es zunächst nicht glauben mag, gibt es doch viele thematische Überschneidungen. Zum Beispiel:

- Anforderungsaufnahme
- Paketierung und Bereitstellung
- Versionierung
- Clean Code
- In-Code-Dokumentation
- Automatisierung
- Erweiterung durch *benutzerdefinierte Konnektoren* oder *PCF-Komponenten*

Sie sind der Experte oder die Expertin für Ihre Prozesse. Die Entwicklerinnen und Entwickler wissen, wie man Software schreibt und diese über die komplette Lebensspanne betreut. Arbeiten Sie zusammen, bilden Sie ein *Fusion Team*. Egal ob Pro-Code, No-Code oder Low-Code: Am Ende haben doch alle dieselben Schmerzen.

Ich selbst komme aus der Entwicklung und bin froh, mich auf das Low-Coden eingelassen zu haben. Ja, es ist »Geklicke«, aber das tut nicht weh! Und eventuell hilft es Ihnen, wenn ich an dieser Stelle wiederhole, dass Low-Code nicht für professionelle Entwickler und Entwicklerinnen gemacht wurde. Low-Code-Plattformen sind nicht dafür da, dem Entwickler maximale Schmerzen zuzufügen, sondern sie sollen Menschen, die kein Entwickler-Know-how besitzen und keine erfahrenen SharePoint-Recken sind, in die Lage versetzen, Anwendungen zu entwickeln, um sich damit den Arbeitsalltag zu erleichtern. Nichtsdestotrotz gilt: Haben Sie die Möglichkeit, einen professionellen Entwickler in Ihre Projekte einzubinden, dann tun Sie es. Nichts ist besser für Ihre Software, als wenn Sie noch mal einen Profi draufschauen lassen.

1.2 Aufgabenbereiche: Wer ist wofür zuständig?

Ziel der Power Platform ist, dass alle, die einen Prozess erkennen, den es zu digitalisieren lohnt, dies selbst umsetzen können. Das heißt: Alle Mitarbeiter und Mitarbeiterinnen Ihres Unternehmens sind von nun an potenzielle Softwareentwicklerinnen und -entwickler, nämlich *Citizen Developer*!

1.2.1 Citizen Developer

Citizen Developer entwickeln Software für sich, ihr Team oder gar das ganze Unternehmen, ohne Softwareentwickler zu sein. Citizen Developer sind nichts Neues. Man findet sie schon lange in sehr vielen Unternehmen. Der Unterschied zwischen gestern und heute ist, dass diese Mitarbeiter jetzt einen Namen haben.

Wenn Sie sich nun fragen, welche Kollegen und Kolleginnen wohl zu dieser geheimnisvollen Gruppe gehören, dann überlegen Sie einmal, wer die ganzen Excel-Sheets, Access-Datenbanken, PowerBI-Reports oder InfoPath-Formulare erstellt hat.

1.2.2 Maker und Owner

Sind Sie leidenschaftlicher Ersteller von Apps und Flows in der Power Platform, so nennt man Sie auch *Maker*!

Erstellen Sie Apps und Flows, dann gehören sie Ihnen. Das heißt, Sie sind *Besitzer* bzw. *Owner* der App bzw. des Flows. Herzlichen Glückwunsch! Natürlich können Sie die Last des Besitzes teilen und Kollegen zu Mitbesitzern bzw. Co-Ownern machen. Wie

das funktioniert, erfahren Sie später im Buch. Wichtig ist zu wissen, dass es bei Software nicht anders ist als bei Immobilien: Besitz verpflichtet!

Die Owner sind letztlich immer verantwortlich für den Betrieb der Apps und Flows. Das heißt, mit dem Abschluss der Entwicklungsarbeit hört die Verantwortung nicht auf. Als Owner sind Sie zuständig für die Stabilität Ihrer Produkte und für die Einhaltung etwaiger Rahmenbedingungen, die durch die Firmen-Compliance und Anforderungen an die Sicherheit vorgegeben sind.

Das mag jetzt abschreckend wirken, aber letztlich passiert hier nichts anderes, als wenn Sie eine Excel-Tapete mit Ihren Kolleginnen und Kollegen teilen und ein Fehler in einer Formel oder einem Makro auftritt. Dann werden Sie sicherlich versuchen, das Problem selbst zu beheben, und nicht ein Ticket für die IT verfassen.

1.2.3 Administratoren: »Die IT«

Eine weitere Gruppe sind die Administratoren, in traditionell strukturierten Unternehmen eher Gegenspieler der Maker als Mitspieler. Über ihnen hängt das Damoklesschwert der Verantwortung für die Sicherheit des Unternehmens. Und wie sorgt man am besten für Sicherheit? Genau, indem man erstmal grundsätzlich alles verbietet. Das läuft der Idee des Citizen Developments diametral entgegen.

Die Power Platform wird allen Seiten gerecht, denn sie erlaubt den Admins, die Leitplanken für die Nutzung durch die Maker zu definieren. So definieren Sie Regeln, die den unerwünschten Datenabfluss verhindern. Mit einer geschickten Umgebungsstrategie schaffen Sie Räume, in denen die Maker wirken können.

So wird aus einem Gegeneinander ein Miteinander!

Das alles läuft unter den Begriffen *Governance* und *Adoption*. Passenderweise hat das Customer Advisory Team (CAT) von Microsoft sich des Themas angenommen und aus Kundenprojekten heraus das *Power Cat Adoption Maturity Model* entwickelt.

Adoption

Mehr zum Thema Adoption und das Power-CAT-Team finden Sie hier:

https://powerapps.microsoft.com/en-us/blog/power-cat-adoption-maturity-model-repeatable-patterns-for-successful-power-platform-adoption/.

Es handelt sich dabei um ein Reifegradmodell, das Sie bei der Einführung der Power Platform in Ihrem Unternehmen begleitet. Es hilft Ihnen zunächst, zu erkennen, wie weit die Einführung in Ihrem Unternehmen fortgeschritten ist, und gibt Handlungsempfehlungen, wie Sie Regeln und Verantwortung sinnvoll entwickeln und den Betrieb der Plattform automatisieren können.

Arbeiten Sie miteinander statt gegeneinander. Sie, die Admins, und Sie, die Influencer im Unternehmen – diejenigen, die vorangehen, die gerne helfen und erklären. Bilden Sie ein Team aus Champions, und identifizieren Sie gemeinsam Maßnahmen, um die Interessen der IT und der Maker optimal zu vertreten.

Das Champions-Programm

Was ein Champion ist und wie Sie einen Champion finden, erfahren Sie hier:
https://learn.microsoft.com/de-de/power-platform/guidance/adoption/champions.

1.3 Geschichten aus dem Projektalltag

1.3.1 Softwareprojekte gestern und heute

Wurden früher aufwendige Lasten- und Pflichtenhefte geschrieben, die dann mühsam in MS-Project-Tapeten gegossen und mit unverrückbaren Terminen versehen wurden, fängt man heute gemeinsam mit dem Kunden einfach mal an.

Nicht selten steht in einem Workshop schon ein erster Prototyp, der nach ein wenig Feinschliff mit ausgewählten Anwendern getestet wird. Besonders große Augen bekommen Ihre Kunden, wenn Sie Anforderungen à la »kann man eigentlich auch …?« direkt umsetzen, während der Wunsch zu Ende formuliert wird.

1.3.2 Softwareentwicklung ohne Risiko?

Aber ist das wirklich alles so easy? Nein, natürlich nicht. Es kommt wie immer darauf an.

Soll eine App zunächst nur eine kleine Access-Datenbank ablösen, die nicht essentiell wichtig ist, sondern nur die Arbeit ein Stück weit vereinfacht, so entwickelt sich daraus im Laufe der Zeit eventuell doch eine Software, die zu einem wichtigen Teil eines geschäftskritischen Prozesses wird.

Schauen Sie in große Unternehmen, die der Power Platform eine strategische Bedeutung zumessen, finden Sie dort aus gutem Grund ein dediziertes Team oder gar eine Abteilung, die sich um den Betrieb und die Weiterentwicklung von Power-Platform-Lösungen für kritische Prozesse auf Abteilungs- oder Unternehmensebene kümmert. Denn wenn die Funktion kritischer Prozesse fehlerhaft ist, muss sich schnell jemand darum kümmern.

Dann spielt es keine Rolle, ob Sie eine Lösung auf herkömmliche Weise oder mit einer Low-Code-Plattform entwickelt haben.

Bewerten Sie also regelmäßig Ihre Entwicklungen nach Ihrer Kritikalität. Faktoren hinsichtlich der Weiterentwicklung und Nutzung Ihrer Apps und Automatisierungen, die Sie dafür heranziehen können, sind:

- Datum der letzten Aktualisierung
- Anzahl Personen, mit der die Software geteilt ist
- verwendete Datenverbindungen und damit die Gefahr von ungewünschtem Datenabfluss
- Anzahl von Zugriffen
- geografische Verteilung der Nutzung
- Offline-Fähigkeit

Das Center of Excellence (CoE)

Das *Center of Excellence* hilft Ihnen dabei, eine Bewertung vorzunehmen und daraus Maßnahmen abzuleiten. Einen grundlegenden Einblick in das CoE finden Sie hier:

https://learn.microsoft.com/de-de/power-platform/guidance/coe/starter-kit.

1.3.3 Ängste der internen IT

An dieser Stelle möchte ich mit ein paar Vorurteilen/Mythen aufräumen, die ich in meinen Projekten wahrgenommen habe:

- **Es wird alles zugemüllt**
  Zuerst sind Sie neugierig, dann gehen Sie die ersten Schritte mit der Power Platform, indem Sie ein paar Tutorials durchgehen und sich auch mal an den einen oder anderen Prozess wagen. Das ist total in Ordnung und soll auch so sein. Wichtig ist nur, dass Sie hinter sich aufräumen und Apps und Flows, die Sie nicht mehr benötigen oder »nur mal so« erstellt haben, wieder entfernen.

  Ob etwas Kunst ist oder wegkann, müssen Sie entscheiden, denn Sie sind der Maker und somit der initiale *Owner*. Auch hierbei hilft das *Center of Excellence* (*CoE*).

Co-Owner

Nehmen Sie immer einen Co-Owner, einen Mitbesitzer, mit in Ihre App. Ich kann nicht mehr zählen, wie viele verwaiste Apps mir in meinen Projekten begegnet sind, da die Besitzerinnen oder Besitzer das Unternehmen verlassen haben.

- **Die IT muss dann wieder die Fehler ausbügeln**
  Sind Sie Administrator? Dann lehnen Sie sich jetzt ganz entspannt zurück, denn derjenige, der die App oder die Automatisierung erstellt hat, ist auch für ihre kor-

rekte Funktion verantwortlich – nicht Sie. Hier gilt ganz klar das Urheberprinzip. Haben Sie vernünftige Regeln definiert, kommen Sie erst wieder ins Boot, wenn zum Beispiel in der App verwendete Dienste nicht mehr funktionieren, die in Ihrem Verantwortungsbereich liegen, oder wenn Lizenzen nicht oder nicht mehr verfügbar sind etc.

Grundsätzlich gilt aber: Helfen darf man immer.

- **Das geht nicht wegen der Datensicherheit**
  In vielen Unternehmen existiert ein latentes Misstrauen der IT gegenüber den übrigen Angestellten, die es zu kontrollieren gilt.

  Der Erfolg der No-Code-Anwendungen liegt auch darin begründet, dass sich die *Maker* keine Gedanken um Nebenwirkungen der Digitalisierung machen müssen.

  Dafür braucht es Leitplanken, die es gemeinsam im Sinne der Anwender und unter Berücksichtigung der Unternehmensrichtlinien aufzusetzen gilt. Definieren Sie eine Umgebungsarchitektur, die den unterschiedlichen Anforderungen gerecht wird. Gibt es für eine Anforderung keine passende Richtlinie, dann nehmen Sie sich die Zeit, und definieren Sie eine, die Ihrem Sicherheitsbedürfnis, aber auch den Anforderungen der Maker gerecht wird.

  Handeln Sie transparent, klären Sie auf, und sensibilisieren Sie Ihre Mitarbeiterinnen und Mitarbeiter für das Thema Sicherheit. Dokumentieren Sie die Einstellungen an einem zentralen Ort.

- **Dann ist Tür und Tor für Schatten-IT offen**
  Die mögliche Existenz von Schatten-IT bereitet jedem IT-Mitarbeiter zu Recht schlaflose Nächte. Aber nicht immer erkennen sie die – eigentlich offenkundige – Ursache dafür. In einer Keynote bei einer IT-Konferenz sagte der Sprecher sinngemäß:

  Je mehr Sie die IT-Infrastruktur abschotten und aus einem falschen Sicherheitsverständnis heraus die Anschaffung/Erstellung von Hard- und Software erschweren, umso höher ist die Wahrscheinlichkeit von Schatten-IT.

  Es wird immer jemanden geben, der einen Weg findet, seine Anforderungen umzusetzen. Im Zweifel bringen sich Kolleginnen und Kollegen sogar ihre eigene Hardware mit.

  Mit der Power Platform stellen Sie sicher, dass sich alle Systeme im Rahmen der Regeln bewegen, die Sie mit einer sauber aufgesetzten Governance manifestieren.

  Nicht häufig genug kann man dabei auf das Center of Excellence (CoE) verweisen, denn am Ende ist dieses Werkzeug das Schmerzmittel für den traditionellen Administrator, der durch den Einsatz der Power Platform einen sicherheitstechnischen Weltuntergang auf sich zukommen sieht.

Wenn ein Mitarbeiter Schaden anrichten möchte, tut er das – egal welche Technologie Sie im Einsatz haben. Der Großteil Ihrer Kolleginnen und Kollegen will aber gar keinen Schaden verursachen, sondern das bisschen Zeit zwischendurch nutzen, um besser arbeiten zu können, weniger Überstunden zu schieben oder die Qualität ihrer Arbeit zu optimieren.

1.3.4 SharePoint und die Power Platform

Im Laufe der Zeit hört man die verrücktesten Dinge über den Zusammenhang zwischen SharePoint und Power Apps. »Power Apps ist doch SharePoint« war dabei der skurrilste Kommentar.

Fakt ist: Power Apps und SharePoint sind zwei eigenständige Produkte. Haben Sie ein M365-Abo, so steht Ihnen in den meisten Plänen eine kostenfreie Lizenz für Power Apps und Power Automate Cloud Flows zur Verfügung, die Ihnen die Benutzung der Standard-Features erlaubt. Nun gehört der Zugriff auf Daten in SharePoint zu den Standard-Features, wohingegen der Zugriff auf Azure SQL, Oracle oder Dataverse eine kostenpflichtige Premium-Lizenz bedingt.

SharePoint ist keine Datenbank

Wie gesagt: Fangen Sie einfach mal an zu entwickeln. Dabei ergibt sich schnell die Notwendigkeit, Daten zu verarbeiten. Was liegt da also näher, als das in SharePoint-Listen zu tun. Die sind ja schließlich so ähnlich wie eine Datenbank, oder?

Lassen Sie mich eines klarstellen: SharePoint ist keine Datenbank! SharePoint ist wundervoll, ein tolles Produkt zum Verteilen von Informationen und eine tolle Dokumentenablage. Aber es ist keine Datenbank!

SharePoint hat einige Einschränkungen, die es von Datenbanken unterscheidet:

- Es ist nicht dafür optimiert, große Datenmengen performant zu verwalten.
- Die Performance liegt weit unter der von »echten« Datenbanksystemen.
- Der Aufbau von relationalen oder gar normalisierten Datenbanken ist nur unter großen Schmerzen möglich.

Ist SharePoint deswegen schlecht? Nein! Es ist halt nur keine Datenbank.

Für den Start Ihrer Power-Platform-Entwickler-Karriere ist SharePoint ein tolles Tool.

Wagen Sie aber hin und wieder einen Blick in die Zukunft, und fragen Sie sich, unter welchen Rahmenbedingungen Ihre App in einem Jahr betrieben wird. Sollen auf einmal nicht nur 10, sondern 2.000 Personen die App bedienen, so kommen Sie mit SharePoint in puncto Datenmengen und Performance schnell an Ihre Grenzen.

Limitierungen von SharePoint

Detaillierte Informationen zu Limitierungen von SharePoint finden Sie hier:

https://learn.microsoft.com/en-us/office365/servicedescriptions/sharepoint-online-service-description/sharepoint-online-limits

Wenn Sie also den Wunsch oder Verdacht hegen, dass die App Wachstumspotenzial hat, kalkulieren Sie die Kosten für die notwendigen Lizenzen, die Migration der App auf die neue Datenbank und die Migration der Daten mit ein.

Wenn Sie aber sagen, dass Ihnen das – Stand heute – egal ist, weil Sie ja jetzt noch nicht wissen, ob sich Ihre App tatsächlich im Arbeitsalltag bewährt und bei den Kolleginnen und Kollegen Akzeptanz findet, ist das vollkommen okay. Wichtig ist nur, immer im Blick zu behalten, dass Sie später möglicherweise andere Datenquellen anbinden müssen, die zusätzliche Lizenzen erfordern. Dann entstehen wahrscheinlich auch zusätzliche Kosten. Lassen Sie sich davon aber bloß nicht abschrecken! Wenn eine App intensiv genutzt wird, weil sie Prozesse verschlankt, stehen zusätzlichen Lizenzkosten oft hohe Einsparungen gegenüber. Und dann tun die Lizenzen gar nicht mehr so weh.

Tatsächlich beginnen viele meiner Projekte auf SharePoint, da dieser halt da ist und keine Diskussionen mit dem Einkauf verursacht. Nicht wenige Apps und Flows werden jedoch später aus den oben genannten Gründen in Richtung SQL oder Dataverse migriert.

Zukunftsfähigkeit

Haben Sie einmal komplexe Datenzusammenhänge in SharePoint-Listen gegossen und dabei auch noch versucht, ein relationales Datenbankmodell umzusetzen? Dann haben Sie Ihre Leidensfähigkeit unter Beweis gestellt, denn dafür ist SharePoint schlicht nicht gemacht.

Je nach Komplexität Ihrer Datenstruktur werden Sie spätestens, wenn Ihr Prozess in weitere EDV-Systeme integriert werden soll, Prozesse schaffen müssen, um Daten aus Ihrer SharePoint-Liste für eine Weiterverwendung aufzubereiten.

Das kann man alles machen, jedoch schreibe ich diese Zeilen nicht ohne Grund, denn eines meiner Projekte wurde aus Kostengründen zunächst auf SharePoint-Listen aufgebaut, obwohl absehbar war, dass es in Zukunft sehr wahrscheinlich von zentraler Bedeutung für das Unternehmen sein würde. Deshalb war zu erwarten, dass Datenmenge und -komplexität stark zunehmen. Und genau so kam es. In der Folge wurde die Dateninfrastruktur von SharePoint in Richtung Azure SQL migriert. Mit Erfolg: Das Projekt ist mittlerweile sehr integrationsfreudig und stellt Daten problemlos für Drittprozesse zur Verfügung.

Verteilbarkeit

Genauso, wie Sie Updates für Ihre Apps und Flows in die verschiedenen Test- und Produktivitätsumgebungen verteilen, müssen auch Ihre Dateninfrastrukturen updatefähig sein. Das bedeutet, Sie müssen in der Lage sein, mit einem Update neue oder geänderte Tabellen, Felder oder Relationen zwischen Tabellen zu transportieren.

Da SharePoint-Listen kein nativer Bestandteil der Power Platform sind, gibt es kein standardisiertes Vorgehen, um SharePoint-Listen, Felder und Relationen mit einem Lösungs-Update auf der Power Platform zu verteilen.

Natürlich können Sie mit *PnP-PowerShell* SharePoint-Sites auslesen, in XML verpacken und an anderer Stelle wieder einspielen, jedoch lässt sich dieses Vorgehen nicht einfach in einen Power-Platform-ALM-Prozess integrieren. Für diejenigen, die dennoch an SharePoint als Dateninfrastruktur festhalten wollen, stelle ich gerne einen Power Automate Cloud Flow zur Verfügung, der Sie bei der Verteilung von SharePoint-Listen unterstützt.

1.3.5 Traditionelle Softwareentwickler und die Power Platform

Die Power Platform ist eine Low-Code-Plattform, die Anwender in den Fachabteilungen befähigt, ohne Entwicklerkenntnisse zu digitalisieren und zu automatisieren. Dieser Abschnitt richtet sich jedoch an die Leserinnen und Leser, die sich als traditionelle Softwareentwickler verstehen und nun in die Verlegenheit kommen, mit der Power Platform entwickeln zu müssen.

Das ist doch keine echte Programmiersprache

Doch! Und Sie hat einen Namen: *Power Fx*! Aber: Sie funktioniert grundlegend anders, denn sie ist in weiten Teilen deklarativ, wohingegen »normale« Programmiersprachen imperativ sind. Kurz zur Verdeutlichung:

Imperative Softwareentwicklung bestimmt, wie etwas zu sein hat. Drücke ich auf einen Knopf, ändere ich einen Status und die Farbe eines Bildschirmelements:

```
Wenn Button geklickt, dann setze Status auf "abgeschlossen" und ändere
Hintergrundfarbe auf Grün.
```

Bei der deklarativen Entwicklung legen Sie die Bedingungen fest, unter denen sich die Farbe des Elements ändern soll.

```
Wenn Button geklickt, setze Status auf "abgeschlossen"
Hintergrundfarbe = Grün, wenn Status = "abgeschlossen", sonst Rot.
```

Jedem, der schon einmal Berechnungen mit Excel-Tabellen vorgenommen hat, sollte dieses Konzept bekannt vorkommen. Denn auch dort folge ich dem deklarativen An-

satz, indem ich für Zelle A5 festlege, dass sie die Summe der Zellen A1 bis A4 bilden soll: Die Formel für Zelle A5 lautet also `Summe(A1;A4)`.

Und diese Ähnlichkeit zieht sich weiter durch, sodass Sie viele Ähnlichkeiten zwischen Power Fx und Excel-Makros feststellen werden.

Alles hat Grenzen

Natürlich kann ich mit der Power Platform nicht die Welt abbilden, aber den Anspruch hat die Plattform auch nicht. Wenn ich das möchte, setze ich auf die individuelle Softwareentwicklung mit all ihren zeitlichen und monetären Rahmenbedingungen.

Power Apps und Power Automate wurden erschaffen, um in kurzer Zeit zu einem Ergebnis zu kommen, ohne Entwickler zu sein. Naturgemäß muss es dann Grenzen geben, an denen man nicht oder nur schwer weiterkommt.

Diese Punkte bemängelten traditionelle Entwickler in meinen Projekten häufig:

- Ich kann keine Stylesheets anpassen.
- Es gibt kein Drag & Drop in der fertigen Lösung.
- Ich kann den Mauszeiger nicht in Bezug auf das darunterliegende Objekt anpassen.
- Es ist nicht möglich, eigenes JavaScript/Libraries in einer Canvas App zu nutzen.
- Die Arbeit mit hoch normalisierten relationalen Datenstrukturen ist nur schwer umsetzbar.
- Es gibt keine nativen Verteilmechanismen, in denen Objekte außerhalb der Power Platform an-/eingebunden werden können, wie zum Beispiel SharePoint-Listen.
- Ich kann keine Breakpoints setzen und den Code in der Ausführung debuggen.

Ja, das ist richtig, wobei man schon Anstrengungen seitens des Herstellers wahrnehmen kann, den einen oder anderen Punkt in Angriff zu nehmen. Nur weil es jetzt nicht funktioniert, heißt es nicht, dass eine Funktion nicht noch kommt.

Zudem darf man sich fragen: Geht das denn im Pro-Development alles so reibungslos? Gerade die Verteilung der Dateninfrastruktur ist immer eine Herausforderung, ganz unabhängig davon, welche Technologie gewählt wird. In der Power Platform habe ich mit Dataverse zumindest die Wahl, eine komplett paketierbare Datenquelle zu verwenden.

So oder so: Mit der Power Platform schlage ich mit Sicherheit jeden Pro-Developer in puncto Time & Budget, wenn es darum geht, eine funktionale App zu erstellen, die problemlos sowohl im Browser als auch auf mobilen Endgeräten lauffähig ist. Zudem lässt sich auf mobilen Endgeräten relativ einfach, in Browsern in Ansätzen, eine Offline-Fähigkeit implementieren.

1.3.6 Das Evergreen-Prinzip

Das Evergreen-Prinzip ist eine phantastische Sache, denn Ihre Software wird von Microsoft quasi rund um die Uhr verbessert und erweitert. Gleichzeitig bedeutet das aber auch, dass sich Benutzeroberflächen verändern und sich Funktionen verschieben oder wegfallen können, wenn sie sich als fehlerhaft herausstellen oder zu wenig genutzt werden.

Freud und Leid

Da ich von Haus aus neugierig und verspielt bin, freue ich mich über jede kleine Neuerung und teste sie umgehend. Nun ist es aber auch mein Job, zu wissen, was wie funktioniert, und technisch auf dem neuesten Stand zu sein.

Problematisch ist das, wenn Sie für Ihre Maker eine Dokumentation für den Umgang mit der Power Platform geschrieben oder sogar Videos gedreht haben und sich die Oberflächen der Software in der Zwischenzeit ändern. Ich erlebe das hautnah mit, während ich diese Zeilen verfasse, da sich die Oberfläche des Power Apps Studios an das Microsoft-Teams-Design angleicht und ich somit ein Kapitel neu schreiben darf.

Grundsätzlich ist das aber kein Phänomen der Power Platform, sondern von Cloud-Diensten im Allgemeinen.

Des einen Freud ist des anderen Leid. Aber klar ist, dass es nicht die Aufgabe der Mitarbeiter ist, technisch immer auf dem Laufenden zu bleiben. In kleineren Unternehmen wird es sich dennoch nicht verhindern lassen. In größeren Unternehmen sollte es Kümmerer geben, also eine Gruppe von Menschen, die sich den Herausforderungen stellen und die notwendigen Informationen für die Maker aufbereiten, ähnlich wie es eine Abteilung für Kommunikation mit Unternehmensinformationen tut. Sehen Sie diese Personen als eine Art Feel-Good-Manager, die sich nicht um das Seelenheil der Beschäftigten im Allgemeinen, sondern in Bezug auf die Power Platform im Speziellen kümmern.

Einzelne Features einer Power App, die in Zukunft zurückgezogen werden, finden Sie im Einstellungsmenü des Power Apps Studios (siehe Abbildung 1.1). Weitere Informationen zu den Einstellungsmöglichkeiten einer App finden Sie in Abschnitt 4.2, »Einstellungen«.

Das Schöne ist, dass Microsoft diesen kontinuierlichen Update-Prozess transparent gestaltet, denn die aktuellen Informationen zu neuen oder geänderten Funktionen findet man in der Beschreibung der anstehenden Release Waves. Release Waves sind gleichbedeutend mit den Servicepacks oder Featurepacks, die Sie vermutlich aus der On-Premises-Welt mit lokal installierten Softwaresystemen kennen.

Abbildung 1.1 Zurückgezogene Funktionen

Abbildung 1.2 zeigt einen Ausschnitt aus den neuen und geplanten Funktionen der Release Wave 2 des Jahres 2022, den Sie auch unter dem folgenden Link abrufen können: *https://learn.microsoft.com/en-us/power-platform-release-plan/2022wave2/power-apps/planned-features*.

Collaborative workloads

Collaborative workloads use app experiences and data, allowing users to collaborate with team members using existing tools like Teams.

Feature	Enabled for	Public preview	Early access*	General availability
Try cards for Power Apps	Users by admins, makers, or analysts	✔ Oct 2, 2022	-	-
Skills match for Power Apps makers	Admins, makers, marketers, or analysts, automatically	✔ Oct 9, 2022	-	-
Form UX improvements	Users, automatically	-	✔ Aug 1, 2022	✔ Oct 1, 2022
Power BI quick reports in Power Apps	Admins, makers, marketers, or analysts, automatically	✔ Jul 24, 2022	-	✔ Oct 2, 2022
Filter grid data, save results as view	Users, automatically	-	✔ Aug 1, 2022	✔ Oct 2, 2022
Model-driven app client toasts moved to the toast stack	Users, automatically	-	✔ Aug 1, 2022	✔ Oct 2, 2022
Modern advanced find turned on by default	Users, automatically	-	✔ Aug 1, 2022	✔ Oct 2, 2022
Form component enhancements - support for command bar, business process flow, header, and tabs	Users by admins, makers, or analysts	Dec 2022	-	Dec 2022
Grid control support for grouping, aggregation, nested grids	Users by admins, makers, or analysts	-	-	Dec 2022
Use Teams chat with others in app	Users by admins, makers, or analysts	Dec 2022	-	Mar 2023

Abbildung 1.2 Neue und geplante Funktionen

Anstehende Änderungen finden

Suchen Sie in Ihrer favorisierten Suchmaschine nach den Begriffen »Power Platform Release Wave«, um zur Dokumentation der neusten Änderungen zu gelangen.

Das Babylon-Problem

Ein großes Vergnügen ist das Thema Mehrsprachigkeit in der Power Platform, da wirklich jeder Begriff, jeder Menüpunkt, wirklich alles in die jeweilige Landessprache übersetzt wurde. So kam es dazu, dass man eine Automatisierung mit »Microsoft Fluss« vornahm oder einen Klickpfad in Power Apps »gedatensatzt« hat.

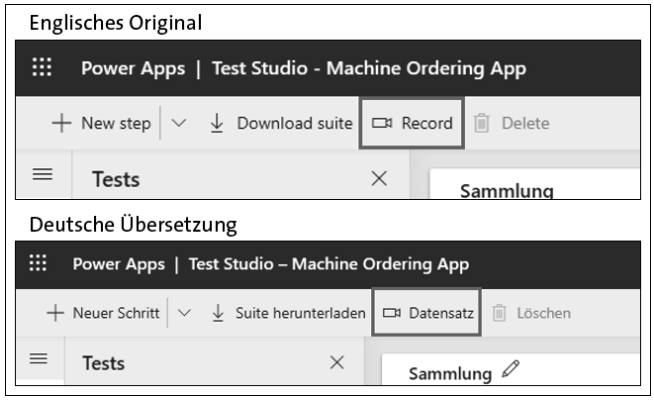

Abbildung 1.3 Merkwürdige Übersetzungen

Um Ihre Neugier zu befriedigen: Abbildung 1.3 zeigt die etwas misslungene Übersetzung des Befehls RECORD zur Aufnahme eines Klickvideos. Die korrekte Übersetzung ins Deutsche ist »Aufzeichnen«. Record ist allerdings auch die Übersetzung des Wortes »Datensatz«, was hier zu einem witzigen Übersetzungsfehler führte. Microsoft hatte ein Einsehen und ist meinem Vorschlag gefolgt.

Sie werden sich sicherlich daran gewöhnen, wenn Sie ausschließlich in einer Umgebungssprache arbeiten. Wenn Sie jedoch öfter zwischen deutschen und englischen Versionen wechseln, fühlen Sie sich wie ein blutiger Anfänger, denn Sie finden erst einmal nichts wieder. Das Gemeine ist, dass Microsoft die Übersetzungen stetig nachbessert, was zur Folge hat, dass kuriose, aber dennoch lieb gewonnene Übersetzungen plötzlich korrigiert werden.

1.3.7 Eine App kommt selten allein

Sie haben Ihre App gebaut und für Ihre Kolleginnen und Kollegen freigegeben, prompt bekommen Sie die ersten Meldungen, dass die App nicht funktioniert. Woran liegt's?

Die App-Entwicklung ist das eine, die Berechtigungen auf genutzte Ressourcen sind das andere: Denken Sie daran, alle Nutzerinnen und Nutzer auf sämtliche verwendeten Datenquellen zu berechtigen. Übliche Verdächtige sind *OneDrive* und *SharePoint*, also die Datenquellen, die in den meisten Canvas Apps verwendet werden, die mir untergekommen sind.

Denken Sie also immer daran, dass Sie alle SharePoint-Webseiten, Listen und Bibliotheken sowie OneDrive-Ordner freigeben, sofern Sie dort Daten lagern, die für den Betrieb der App notwendig sind.

1.4 Von der Idee zur App

Dieser Abschnitt widmet sich dem Prozess von der Idee bis hin zur ersten Version Ihrer App.

1.4.1 Motivation

Bevor Sie sich die Arbeit machen, eine App oder einen Flow zu entwickeln, überlegen Sie, ob sich der Aufwand tatsächlich lohnt. Natürlich entwickeln Sie mit der Power Platform rasend schnell, jedoch haben auch Sie keine Zeit zu verschwenden. Deshalb möchte ich Ihnen an dieser Stelle einige Anhaltspunkte mit auf den Weg geben, in welchen Fällen sich die Digitalisierung/Automatisierung lohnt:

- Es handelt sich um immer wiederkehrende Arbeitsschritte.
- Es lassen sich Kosten sparen.
- Die Pandemie-Sicherheit von Prozessen ist nicht gegeben.
- Die Qualität ist verbesserungswürdig und -fähig.
- Die Zeit für die Durchführung von Prozessen lässt sich verkürzen.
- Die Teilhabe lässt sich damit ermöglichen oder verbessern.

Wie sich durch die Digitalisierung eines Prozesses die Durchlaufgeschwindigkeit und Qualität eines Prozesses verbessern lässt, soll das folgende Beispiel aus meinem Projektalltag verdeutlichen:

Anlegen einer SharePoint-Teamsite vor Automatisierung

1. Eine Benutzerin oder ein Benutzer schreibt ein Ticket und fordert eine neue SharePoint-Teamsite an.
2. Ein Support-Mitarbeiter öffnet das Ticket und fragt gegebenenfalls noch einmal nach, weil Informationen fehlen.
3. Die Site wird durch den Support angelegt und konfiguriert und das Ticket an die Person weitergegeben, die sich um die Berechtigungen kümmert.

4. Diese informiert die Person, die die Site angelegt hat, um die Site in die Navigation aufzunehmen.
5. Sobald das geschehen ist, informiert diese Person wiederum die Person, die die Berechtigung vorgenommen hat, um die antragstellende Person zu informieren, dass die Site eingerichtet ist.

   Dieser Prozess dauert, wenn es schnell geht, drei Tage, in der Regel aber eine bis anderthalb Wochen. Die eigentliche Arbeit sind etwa 15 Minuten.

   Hierbei handelt es sich um einen klassischen Fall eines dringend zu digitalisierenden Prozesses, der dann so ablaufen könnte:

   Anlegen einer SharePoint-Teamsite nach Automatisierung

   1. Die Person füllt ein mit Pflichtfeldern versehenes Formular in MS Forms aus, womit die Informationen automatisch komplett sind.
   2. Nach dem Versenden empfängt ein Cloud-Flow die Daten, legt die Teamsite an, setzt die notwendigen Berechtigungen und informiert den Support per Teams-Nachricht.
   3. Der Support prüft noch einmal und bestätigt das Anlegen in Teams, woraufhin die antragstellende Person informiert wird.

Das Formular aus Abbildung 1.4 war innerhalb von fünf Minuten erstellt. Falls Ihnen Microsoft Forms in puncto Bedienung und Datenvalidierung nicht zusagen, können Sie eine individuelle Power App vermutlich innerhalb von 30 Minuten erstellen.

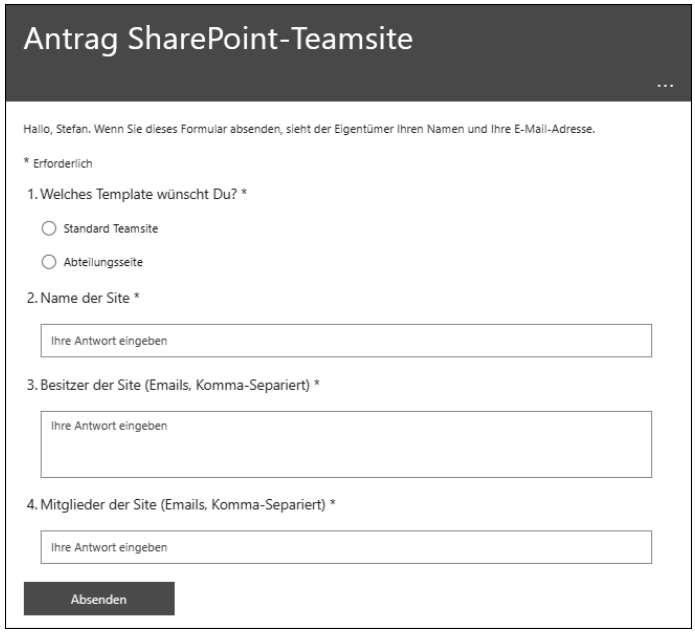

Abbildung 1.4 Antragsformular für eine neue SharePoint-Teamsite

1.4.2 Aufwand und Nutzen

Auch wenn Sie mit der Power Platform einen enormen Geschwindigkeitsvorteil gegenüber der traditionellen Softwareentwicklung besitzen, haben Sie trotzdem Aufwände für Planung, Kommunikation, Umsetzung und Fehlerbehebung. Spätestens, wenn die Umsetzung Ihrer Idee noch den Erwerb weiterer Lizenzen erfordert, sollten Sie eine Kostenkalkulation vornehmen.

Überlegen Sie sich, wie in anderen Projekten auch, ob sich der Aufwand und eventuell anfallende Lizenzkosten lohnen. Denn wenn die Digitalisierung/Automatisierung Ihres Prozesses die Kosten nicht wieder einspielt, dann lassen Sie es. Digitalisierung ist kein Selbstzweck.

Sharing is caring

Schauen Sie dabei auch über den Tellerrand, denn meist ist es doch so, dass uns immer wiederkehrende und ähnliche Prozesse nerven, langweilen und Zeit rauben. Erzählen Sie bei einem Treffen mit Freunden von Ihrem Arbeitstag, so verspreche ich Ihnen, wird der Großteil stöhnend mit dem Kopf nicken und bestätigen, dass das bei ihnen genauso läuft.

Denken Sie vor der Digitalisierung also auch an Ihre Kolleginnen und Kollegen. Sprechen Sie mit ihnen in der Kaffeeküche oder am Mittagstisch, und identifizieren Sie Mitmenschen, die ähnliche Probleme haben wie Sie.

Laden Sie diese Menschen ein, Ihre App mitzunutzen, oder geben Sie ihnen eine Kopie, die einfach an andere Prozesse angepasst werden kann, so werden Sie sehen, dass sich eine einfache Digitalisierung sehr schnell rentiert.

Nicht das Rad neu erfinden

Genauso, wie Sie mögliche Synergien bedenken sollten, sollten Sie zunächst in Erfahrung bringen, ob es Ihren Wunschprozess schon gibt oder sich ein vorhandener Prozess anpassen lässt. Falls ja, bitten Sie die Owner um einen Export, und spielen Sie diesen bei sich ein. Hier empfehle ich Ihnen auch einen Blick in Abschnitt 4.2, in dem Sie lernen, mit einer der vielen phantastischen Vorlagen zu starten.

In einer idealen Welt haben Sie in Ihrem Unternehmen sogar ein Team, das sich genau darum kümmert, die Erkenntnisse und Resultate aller Maker zu strukturieren und auf einer zentralen Plattform zu veröffentlichen. In diesem Fall spricht man vom unternehmensinternen *Center of Excellence* bzw. dem *Champions Team*, also Menschen, denen es am Herzen liegt, dass jeder Maker in Ihrem Unternehmen maximale Unterstützung bei der Entwicklung erfährt.

Haben Sie noch kein solches Team, dann fangen Sie doch genau jetzt damit an, eines zu gründen.

1.4.3 Prototyping

Sie sind sicher, dass es sich lohnt, Ihre Idee kommt bei den Kollegen und Kolleginnen super an, und Sie haben die Zeit für die Umsetzung. Dann kann es jetzt losgehen:

Der einfachste Weg, eine Idee auf Machbarkeit hin zu überprüfen, ist, einen Prototyp zu erstellen. Ein *Prototyp* oder auch eine Machbarkeitsstudie verfügt über die wichtigsten Funktionalitäten, die das Endprodukt besitzen sollte. Ein Prototyp ist aber auch nicht mehr als das – lassen Sie sich also nicht drängen, Nice-to-have-Funktionen gleich mitzuentwickeln!

Prototypen sind keine Produktivsysteme

In der Regel werden Prototypen schnell und ohne Rücksicht auf Verluste entwickelt, entsprechen also in der Regel keiner Programmierrichtlinie. In diesem Fall werfen Sie die App oder den Flow bitte einfach weg und starten eine frische, saubere Entwicklung!

Funktionsumfang

Wichtig ist es, dass Sie sich anfangs ein Set an Funktionen überlegen, die Sie abbilden wollen. Sobald Sie diese realisiert haben, steht Ihre Version 1.0.

Bleiben Sie bei der Entwicklung so eng wie möglich an Ihrem geplanten Funktionsumfang, denn sonst werden Sie nicht fertig.

Natürlich kann es beim Prototyping immer mal einen wichtigen Einfall geben, der Ihren Prototyp aufwertet oder sogar wichtig für eine spätere Version ist. Die Erfahrung zeigt jedoch, dass Sie nicht zum Ende kommen, wenn Sie versuchen, den Funktionsumfang eines fertigen Produkts in einem Prototyp umzusetzen. Dem zu widerstehen, fällt oft schwer, ist aber enorm wichtig.

Ein Weg wäre, diese Anforderungen in einem Backlog für eine Version 2.0 oder 3.0 zu erfassen. Auf diese Weise haben Sie die zusätzlichen Ideen und Wünsche dokumentiert und können diese für neue Releases berücksichtigen.

Ein Beispiel: Sie planen eine App für den innerbetrieblichen Beschaffungsprozess. Sie müssen also klären, welche Anforderungen (grob) beachtet werden müssen, welche Systeme beteiligt sind und wie die wichtigsten Funktionen aussehen sollen. Fangen wir mit den Anforderungen an.

Grobe Anforderungen

- Es soll einen Produktkatalog geben.
- Produkte haben eine Kostenart, einen Titel, Beschreibungen, Bilder und Dokumente.
- Produkte sind nach Kategorien sortiert.

- Der Katalog soll nach Preisen sortierbar sein.
- Es soll eine Produktsuche geben.
- Die bestellende Person befüllt einen Warenkorb mit den Produkten.
- Vor der Bestellung muss eine Kostenstelle ausgewählt werden.
- Genehmigungen sollen in Abhängigkeit vom Warenkorb-Preis, Kostenstellen-Limits und separat nach Kostenarten erfolgen.
- Besteller und Genehmiger sollen per E-Mail und/oder Chat-Nachricht informiert werden.
- Produkte sollen bewertbar sein (1 bis 5 Sterne).

Ich würde mir nun ein Stück Papier zur Hand nehmen und grob skizzieren, wie ich die Daten in meinem Prozess am besten verarbeiten kann. Dabei geht es vor allem um Schnittstellen. Egal, was Sie entwickeln, Sie müssen Daten in Ihr System hinein-, aber auch wieder herausbekommen.

Das kann auf zwei Arten geschehen: entweder durch Menschen über eine ansprechende Benutzeroberfläche oder über Drittsysteme wie zum Beispiel OneDrive, SharePoint, Dataverse oder Adobe, die sich durch Datenverbinder (Konnektoren) in die Power Platform integrieren lassen.

Dann entwerfen Sie das grobe Look & Feel Ihrer App und ermitteln die beteiligten Systeme, die Sie für Ihr Vorhaben benötigen.

Beteiligte Systeme

- Sie planen, die Produktdaten in *SharePoint* zu verwalten.
- Die erste Genehmigung soll durch den Vorgesetzten erfolgen, der im *Active Directory* gepflegt wird.
- Kostenstellen und Kostenarten sollen inklusive der genehmigenden Personen ebenfalls in SharePoint gehalten werden.
- Für die Kommunikation benötigen Sie Zugriff auf den *Outlook*- sowie den *Teams*-Konnektor.

Nun überlegen Sie, welches die Knackpunkte in dem System sein könnten, und identifizieren die wichtigsten (!) Funktionen, über die Ihre App verfügen muss, damit Sie Ihre Idee komplett umsetzen können. Dieser Punkt bedarf sicherlich einiger Erfahrung, aber lassen Sie sich davon nicht abschrecken. Wir haben alle mal klein angefangen!

Die wichtigsten Funktionen

- eine kleine Produktauswahl mit Filter und Suche
- ein Warenkorb
- ein Bestell- und Genehmigungsprozess

- genehmigen soll der Vorgesetzte aus dem Active Directory
- Bestätigungs-E-Mail und Chat-Nachricht

Nun kann es losgehen!

Sicherheit

Prüfen Sie als Erstes, ob Sie Zugriff auf die beteiligten Systeme haben, indem Sie eine App erstellen und die notwendigen Konnektoren hinzufügen.

Überlegen Sie sich gleich, wo die Daten liegen, die Sie in Ihrem Prozess verwalten wollen. Denn unter Umständen ist die Kombination der für Ihre Idee notwendigen Schnittstellen für Ihre Zielumgebung nicht zugelassen. Suchen Sie in diesem Fall den Kontakt zu den zuständigen Personen im Unternehmen, und bitten Sie sie, Ihnen entweder eine passende Umgebung freizugeben oder eine neue Umgebung bereitzustellen, in der Sie den gewünschten Prozess umsetzen können.

Nichts ist demotivierender, als während der Entwicklung festzustellen, dass Sie mit der IT noch eine Runde drehen müssen, da Ihnen entweder Zugänge fehlen oder Sicherheitsrichtlinien die gemeinsame Nutzung der von Ihnen präferierten Datenverbindungen verhindern.

Grundsätzlich können Sie mit der Power Platform nicht viel mehr tun, als Sie es ohne dieses technische Hilfsmittel tun könnten. Haben Sie keine Berechtigung auf Ihren SharePoint, so können Sie die Daten darin auch mit der Power Platform nicht nutzen oder bearbeiten.

Was bedeutet das? Die Power Platform verfügt über eine Fülle von Schnittstellen, mit denen der Maker einen Datenaustausch betreiben kann. Es fällt nicht schwer, sich vorzustellen, was passieren mag, wenn in einer App oder einem Flow die gleichzeitige Verwendung einer Verbindung zu *SQL*, *SharePoint* oder *Dataverse* sowie *Twitter, Facebook* oder *LinkedIn* möglich ist.

Es hindert Sie technisch nichts daran, Unternehmensdaten per Hand auf beliebigen sozialen Netzwerken zu verteilen, doch wäre das sehr mühsam. Durch die Automatisierung können Sie das nun sehr viel schneller mit sehr viel mehr Daten tun, und darin liegt die Gefahr. Die Lösung für das Problem sind *Sicherheitsrichtlinien*, also Regeln, welche Schnittstellen miteinander reden dürfen und welche nicht.

[!]

Data Loss Preventions gegen ungewollten Datenabfluss

Mit Data Loss Preventions (DLPs) steuern Sie, welche Konnektoren in einer Umgebung gemeinsam in einer App oder einem Flow genutzt werden dürfen.

Die Standardeinstellung lässt leider alles zu, weshalb Sie zumindest in der Default-Umgebung, die im Standard immer vorhanden ist und auf die jeder Benutzer Zugriff hat, um Apps zu entwickeln, maximale Einschränkungen vornehmen sollten.

Benutzeroberflächen

Nehmen Sie nun Ihre Aufzeichnungen zur Hand, und übertragen Sie Ihre Idee vom Blatt in Ihre App. Erstellen Sie Bildschirm für Bildschirm, und realisieren Sie einen ersten einfachen Klickpfad durch die App, indem Sie die Navigationselemente wie Schaltflächen oder Symbole mit der *Navigate*-Funktion versehen.

Nun ist es an der Zeit, der App Leben einzuhauchen und die Elemente ihrer Bildschirme mit den Datenquellen zu verbinden.

Programmieren einer lokalen Datenstruktur

Falls Ihnen die Datenquellen noch nicht zur Verfügung stehen, Sie aber schon anfangen wollen, dann ist der der Aufbau einer Datenstruktur im Ereignis `App.OnStart` eine Möglichkeit. Hier können Sie beliebig komplexe Datenstrukturen aufbauen, die Ihre späteren SharePoint-Listen, Dataverse-Tabellen oder Planner-Aufgaben simulieren. Definieren Sie einfach in `App.OnStart` Ihre Daten durch die Verwendung von Collections (siehe Abbildung 1.5), und benutzen Sie sie wie jede andere Datenquelle (siehe Abbildung 1.6).

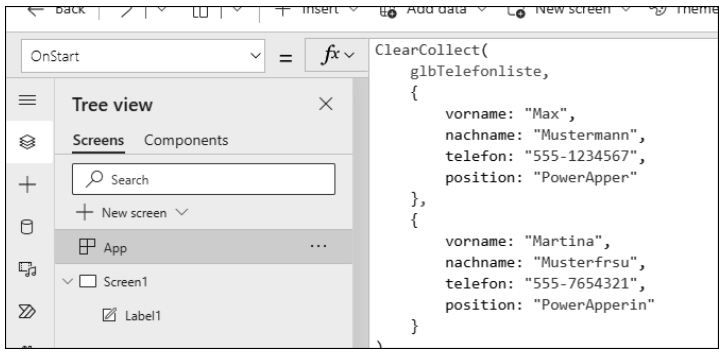

Abbildung 1.5 Manuelles Erstellen einer Datenstruktur

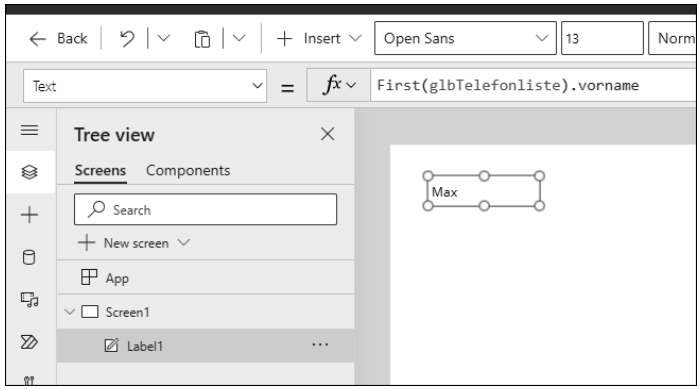

Abbildung 1.6 Benutzung der lokalen Daten

Welchen Weg Sie beschreiten, liegt letztlich bei Ihnen. Sofern ich noch keine Strukturen habe, auf denen ich aufsetzen kann, starte ich mit lokalen Daten, die ich mir in `App.OnStart` anlege.

Alternativ erstelle ich mir auf die Schnelle eine SharePoint-Site und erzeuge dort eine vorläufige Datenstruktur mit SharePoint-Listen, die man über den SharePoint-Konnektor sehr schnell integriert hat.

1.4.4 Der erste MVP

Ihr Prototyp ist fertig, und Sie sind sicher, dass Sie Ihre Idee umsetzen können. Sie haben die App in der Zwischenzeit auch Ihren Kolleginnen und Kollegen gezeigt und dabei weitere Funktionen identifiziert, die Sie in *Nice-to-have* und *Must-have* unterteilen.

Also ist es an der Zeit, aus dem Prototyp ein Produkt zu entwickeln. Auch hier ist es wichtig, sich auf einen ersten Funktionsumfang, bestehend aus den Must-haves, zu verständigen, also einen Umfang, der unbedingt nötig ist, damit die App grundsätzlich einsatzfähig ist. Ein solches Produkt mit einem minimalen Funktionsumfang nennt man auch *MVP* oder *Minimum Viable Product*.

Menschen freuen sich über Regenbögen und Einhörner

Ich lasse immer noch Raum für ein kleines hübsches Extra, denn die Bedienung der App soll Spaß und Lust auf mehr machen.

Eine gute Vorbereitung ist das A und O

Je mehr Mühe Sie sich bei der Erstellung des Prototyps gegeben haben, umso einfacher fällt Ihnen die Erstellung des MVP.

Nun ist es an der Zeit, das MVP gründlich zu testen und in den Produktivbetrieb zu überführen. Je nach *Reifegrad* der Power Platform in Ihrem Unternehmen kopieren Sie die App per Hand oder aber überführen sie in einen automatisierten Application-Lifecycle-Management-Prozess (*ALM*).

Kapitel 2
Die Power Platform im Überblick

Es hat ein bisschen gedauert, bis die Power Platform das geworden ist, was sie jetzt ist. Sie unterliegt zwar einer stetigen Veränderung, aber einige Grundkonzepte bleiben unverändert. In diesem Kapitel lernen Sie die wichtigsten Konzepte und Begriffe kennen. Diese bilden letztlich die Grundlage für die Vertiefung in den nachfolgenden Kapiteln.

Die Power Platform gehört zu den weltweit führenden Technologien für Low-Code-/No-Code-Lösungen. Obwohl einzelne Bestandteile schon längere Zeit auf dem Markt sind, nahm die Platform besonders in den letzten Jahren Schwung auf.

Marktführerschaft

Wenn Sie sich für die Bedeutung der Power Platform interessieren, schauen Sie am besten in der entsprechenden Gartner-Studie nach:

https://powerapps.microsoft.com/en-us/blog/microsoft-named-a-leader-in-2023-gartner-magic-quadrant-for-enterprise-low-code-application-platforms/

2.1 Die Power Platform vorgestellt

Die Power Platform besteht aus mehreren Bestandteilen, die ich Ihnen schrittweise vorstellen möchte.

2.1.1 Power Apps

Power Apps erblickte im Herbst 2016 das Licht der Welt, als von der *Power Platform* noch keine Rede war. Nach Access Web Forms und MS LightSwitch war Power Apps der erste legitime Nachfolger der Erfolgssoftware *Microsoft InfoPath*.

InfoPath war eine Formularsoftware, mit der sich Eingabeformulare auf Basis von XML-Definitionen entwerfen ließen. Ein Formel-Designer half dabei, eine UI-Logik im Formular zu implementieren, sodass man einzelne Felder ein- und ausblenden oder Standardwerte definieren konnte.

Datenmengen

Bei der Verwendung von `SaveData` im Browser bzw. in Teams ist die Datenmenge auf 1 MB begrenzt. Das ist nicht viel, aber damit weiß man, was man hat. Auf mobilen Endgeräten ist die Datenmenge durch den verfügbaren Speicher begrenzt.

Ich empfehle Ihnen, und das gilt auch für viele andere Anwendungsfälle, dass Sie den Worstcase mit dem langsamsten und ältesten Gerät durchspielen, das real genutzt werden kann. Durch die Automatisierungsmöglichkeiten ist die massenhafte Erzeugung von Daten ein Kinderspiel.

5.9 Programmierrichtlinien

Nun geht es um sauberes Programmieren, einen Teil der Softwareentwicklung, der meistens ignoriert wird. Dabei ist es egal, ob Sie im No-Code-, Low-Code- oder Pro-Code-Umfeld aktiv sind.

Etwas überspitzt ist es egal, welche Richtlinien Sie verfolgen, Hauptsache, Sie halten sich an ein gewisses Set an Regeln.

5.9.1 Elemente

Wenn Sie mit der Entwicklung von Power Apps beginnen, werden Sie feststellen, dass gleichlautende Steuerelemente mit einer nachgelagerten Nummerierung versehen werden.

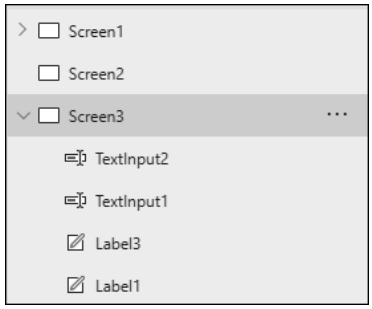

Abbildung 5.230 Steuerelemente werden automatisch durchnummeriert.

Natürlich funktioniert die App, egal wie Sie die Elemente darin benennen, aber wartbar ist anders. Wenn Sie auf dem Bildschirm zum Löschen von Daten die Feldbeschriftung für den Nachnamen ändern wollen, haben Sie bei der Nomenklatur, wie in Abbildung 5.230 zu sehen, keine Chance, außer sich durchzuwühlen.

Benennen Sie die Elemente aber nach ihrem Zweck, wie in Abbildung 5.231 zu sehen, so haben Sie es deutlich einfacher.

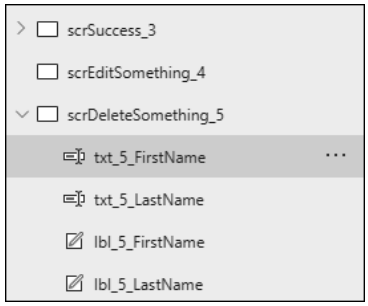

Abbildung 5.231 Zweckbezogene Benennung hilft bei der Wartung.

Diese Benennungsrichtlinie folgt einem ganz einfachen Muster. Es gibt ein Präfix für den Elementtyp, nämlich:

- scr: Bildschirme (Screen)
- txt: Textboxen (TextInput)
- lbl: Bezeichner (Label)

Da es ein Label für einen Nachnamen auf mehreren Bildschirmen geben kann, würden Sie eine Eindeutigkeit wieder über eine Nummerierung erreichen, denn ein Name muss in der kompletten App eindeutig sein.

Also versehe ich hier jeden Bildschirm noch mit einer Nummer als Suffix und verwende diese Nummer zusätzlich in den Elementnamen.

Folgen Sie diesem Muster, haben Sie es auch mit dem Auffinden einfacher, denn nutzen Sie die Suche aus Abschnitt 4.1.3, um eine bestimmte Textbox zu finden, so kommen Sie deutlich schneller ans Ziel (siehe Abbildung 5.232).

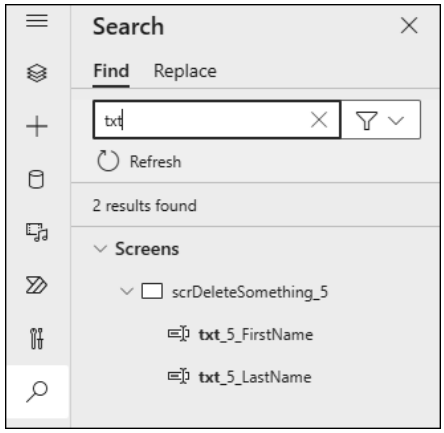

Abbildung 5.232 Suchen bei Verwendung von Nomenklaturen

Tabelle 5.46 können Sie die Präfixe für Bildschirmelemente entnehmen, wie ich sie in meinem Projektgeschäft benutze.

Element	Präfix
Bildschirm/Screen	`scr`
Textfeld/TextInput	`txt`
Beschriftung/Label	`lbl`
Schaltfläche/Button	`btn`
Katalog/Gallery	`gal`
DataCard (In Forms oder Scrollable Screens)	`dc`
Checkboxen	`cbx`
RadioButton	`rad`
DropDowns	`dd`
ComboBoxen	`cb`
Bilder/Images	`img`
Icons	`ico`
Rechtecke (spezielles Icon)	`rec`
Komponenten	`cmp`
Timer	`tim`

Tabelle 5.46 Präfixe von Bildschirmelementen

5.9.2 Variablen

Auch wenn es in Abschnitt 5.1.8 über Variablen schon angesprochen wurde, möchte ich der Vollständigkeit halber hier noch einmal auf sinnvolle Namensrichtlinien für Variablennamen eingehen.

Ich verwende üblicherweise die CamelCase-Schreibweise, sodass Variablen klein anfangen und jedes weitere Teilwort mit einem großen Buchstaben beginnt, wie Sie dem folgenden Beispiel entnehmen können:

- `txtVornameKunde`
- `txtOrder`
- `btnSendOrder`

Auf Variablen bezogen ergibt sich folgende Schreibweise:

Variablentyp	Schreibweise und Deklaration
Collection	`Collect(colOrders,{})`
globale Variablen	`Set(glbCurrentUser,)`
lokale Variablen	`UpdateContext({locNumberOfDays:42})`

Tabelle 5.47 Beispiele für Variablennamen

5.9.3 Code-Kommentare

Power Fx verfügt über die Möglichkeit, Teile eines Codes auszukommentieren: So kommentieren Sie mit // eine Zeile und mit /**/ einen Bereich aus. Nutzen Sie diese Funktionalität, um die Idee hinter Ihrem Code und seiner Funktionsweise zu dokumentieren (siehe Abbildung 5.233). Es dauert nicht sehr lange, und Sie haben auch die einfachsten Konzepte Ihrer App vergessen. Wenn Sie diese nicht dokumentieren, müssen Sie sich im Falle der Wartung oder Erweiterung wieder umständlich einarbeiten.

Sie brauchen keine Angst zu haben, dass zu viele Kommentare die App verlangsamen, denn Kommentare werden zur Laufzeit nicht an den Browser weitergegeben.

```
/*
Solange das Teeam noch keine fünf Mitglieder besitzt, werden dem Team ColAttendees
die Personen aus der ComboBox cbxTeamattendee_1 hinzugefügt.
Anschliessend wird auf den Bildschirm zur Terminvereinbarung weitergeleitet
*/
If(
    CountRows(cbxTeamattendee_1.SelectedItems) <= 6, //Es sind maximal sechs Mitglieder erlaubt
    ForAll(
        cbxTeamattendee_1.SelectedItems,
        Collect(
            ColAttendees,
            {
                Name: ThisRecord.DisplayName,
                Email: ThisRecord.Mail,
                Firstname: ThisRecord.Surname,
                Lastname: ThisRecord.GivenName,
                Active: true
            }
        );
    );
    //Wechsele zur Terminberainbarung
    Navigate(
        scrSetSchedule,
        Fade
    );

)
```

Format text | Remove formatting | Find and replace

Abbildung 5.233 Kommentare im Code erleichtern das Leben.

Automatische Dokumentation Ihrer App per Power Apps Docstring

Ein hervorragendes Werkzeug zur Dokumentation Ihrer App finden Sie im GitHub-Repository von Sebastian Muthwill:

https://github.com/sebastian-muthwill/powerapps-docstring

Power Apps Docstring ist kostenfrei verwendbar und erlaubt die nahezu lückenlose Dokumentation Ihrer App, basierend auf den verwendeten Kommentaren.

5.10 Ausgewählte einfache Anwendungsfälle

Die folgenden Seiten führen Sie durch häufig wiederkehrende Anwendungsfälle, die die Basis für viele Apps bilden. Erstellen Sie sich dazu bitte eine leere Canvas App wie in Abschnitt 3.3.4 beschrieben. Die folgenden Beispiele basieren auf einer App im Tablet-Modus. Wenn Sie lieber im Smartphone-Format arbeiten möchten, müssen Sie die einzelnen Schritte entsprechend anpassen.

5.10.1 Anzeige des angemeldeten Benutzers

Wir starten mit der `User()`-Funktion. Sie beinhaltet grundlegende Informationen zum Benutzer der App, ist parameterlos und gibt folgende Werte zurück:

- `FullName`: Vor- und Zuname des angemeldeten Benutzers
- `Email`: E-Mail-Adresse des angemeldeten Benutzers
- `Image`: Bild des angemeldeten Benutzers

Diese Werte eignen sich hervorragend, um eine schöne Kopfzeile zu gestalten.

1. Wählen Sie einen beliebigen Bildschirm in Ihrer Strukturansicht.
2. Wählen Sie aus dem Menüpunkt EINFÜGEN in der oberen Befehlsleiste folgende Steuerelemente, und stellen Sie sie wie folgt ein:

X	0
Y	0
Höhe/Height	50
Breite/Width	1366

Tabelle 5.48 Ein Rechteck aus dem Abschnitt »Formen«

X	1291
Y	0
Höhe/Height	75
Breite/Width	75
Randradius	45

Tabelle 5.49 Ein Bild aus dem Abschnitt »Medien«

X	1091
Y	0
Höhe/Height	35
Breite/Width	200
Farbe/Color	White
Abstand rechts/PaddingRight	10
Textausrichtung/Align	Right

Tabelle 5.50 Beschriftung 1

X	986
Y	35
Höhe/Height	35
Breite/Width	305
Farbe/Color	White
Abstand rechts/PaddingRight	10
Textausrichtung/Align	Right

Tabelle 5.51 Beschriftung 2

Der Kopfbereich Ihres Bildschirms sollte nun so aussehen wie in Abbildung 5.234.

Abbildung 5.234 Zwischenschritt Kopfbereich

3. Selektieren Sie das BILD und die Eigenschaft IMAGE aus der Eigenschaftenliste wie in Abbildung 5.235.

Abbildung 5.235 Die Image-Eigenschaft der User()-Funktion

4. Nun selektieren Sie das obere Beschriftungselement und die Eigenschaft Text. Befüllen Sie die Eigenschaft mit dem Code `User().FullName`.
5. Im zweiten Beschriftungselement befüllen Sie die Eigenschaft mit User().Email.

Wenn die Daten Ihres Users gut gepflegt sind, sollte der Kopfbereich Ihres Bildschirms nun wie in Abbildung 5.236 aussehen.

Abbildung 5.236 Kopfbereich mit Daten der User()-Funktion

5.10.2 Navigieren zwischen Bildschirmen

Bei der Navigation zwischen Bildschirmen hilft Ihnen der `Navigate`-Befehl:

```
Navigate(Zielbildschirm; ScreenTransition;{Name:Wert;Name2:Wert2;…})
```

`Zielbildschirm` legt den Namen des Bildschirms fest, auf den Sie springen wollen. Der optionale Wert `ScreenTransition` gibt die Art der Überblendung an, mit `{Name:Wert}` legen Sie die Werte fest, die Sie an den Zielbildschirm übermitteln wollen.

Mit den folgenden Schritten erstellen Sie auf einem Bildschirm eine Schaltfläche, die Sie per Klick auf einen weiteren Bildschirm leitet. Sie ändern außerdem den Effekt des Seitenübergangs und übergeben am Ende Parameter von einem Bildschirm auf den anderen.

1. Wählen Sie einen beliebigen Bildschirm in Ihrer Strukturansicht, indem Sie ihn dort selektieren (siehe Abbildung 5.237).
2. Fügen Sie nun eine Schaltfläche hinzu, indem Sie aus der oberen Befehlsleiste den Menüpunkt EINFÜGEN und dort das Element SCHALTFLÄCHE auswählen (siehe Abbildung 5.238).

Abbildung 5.237 Bildschirm auswählen

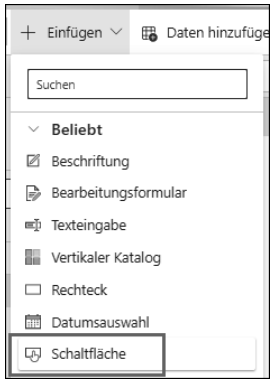

Abbildung 5.238 Schaltfläche auswählen

Symbole als Bedienelemente

Anstatt einer Schaltfläche können Sie auch Symbole (Icons) nutzen, um Funktionen auszuführen oder zwischen Bildschirmen zu navigieren.

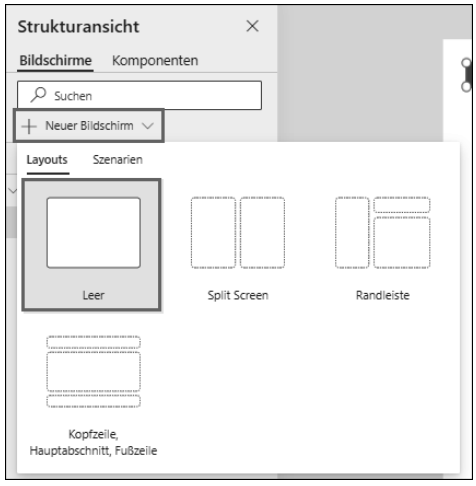

Abbildung 5.239 Neuen Bildschirm hinzufügen

3. Jetzt erstellen Sie den Bildschirm, zu dem Sie navigieren wollen: Wählen Sie dazu den Befehl NEUER BILDSCHIRM in der Strukturansicht, und suchen Sie sich einen leeren Bildschirm aus (siehe Abbildung 5.239).
4. Benennen Sie den Bildschirm in `scrZielBildschirm` um.
5. Nun selektieren Sie die Schaltfläche aus Schritt 2 und wählen in der Eigenschaftenliste das Ereignis `OnSelect`.
6. Löschen Sie dort den Standardwert `false`, und ersetzen Sie ihn durch folgenden Code (siehe Abbildung 5.240):

   ```
   Navigate(scrZielBildschirm)
   ```

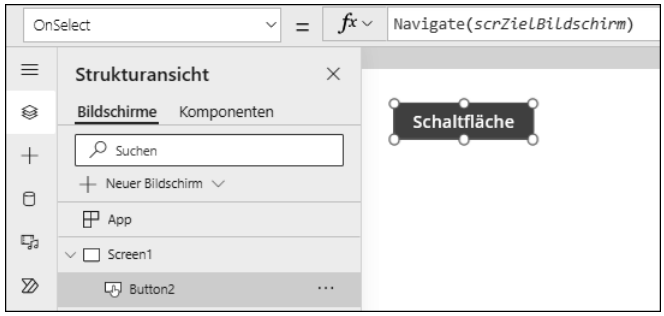

Abbildung 5.240 Navigate-Befehl

7. Stellen Sie nun sicher, dass Sie sich auf dem Bildschirm mit Ihrer soeben angelegten Schaltfläche befinden, und starten Sie Ihre App, indem Sie das Abspielen-Symbol in der Befehlsleiste betätigen.
8. Selektieren Sie jetzt die Schaltfläche auf Ihrem Bildschirm, und Sie werden zu Ihrem Zielbildschirm weitergeleitet.

   Weiter geht es mit der Anpassung des Bildschirmübergangs.
9. Dazu schließen Sie den Ausführungsmodus und selektieren wieder das `OnSelect`-Ereignis Ihrer Schaltfläche auf dem Ausgangsbildschirm.
10. Dort ergänzen Sie den `Navigate`-Befehl um den Parameter der `ScreenTransition`, also des Bildschirmübergangs, indem Sie hinter dem Parameter `scrZielBildschirm` Ihren Wunschübergang angeben (siehe Abbildung 5.241).

    ```
    Navigate(scrZielBildschirm;ScreenTransition.Fade)
    ```

Abbildung 5.241 Navigation mit Transition

Probieren Sie gerne auch die weiteren Effekte aus, die oben in der Parameter-Erklärung aufgeführt sind, und schauen Sie sich das Ergebnis durch Ausführung von Punkt 7 an.

Wenn Sie mit den Effekten zufrieden sind, geht es an die Übergabe der Werte.

11. Möchten Sie Werte von einem Bildschirm auf den nächsten übergeben, nutzen Sie dafür die Werteliste im `Navigate`-Befehl. Selektieren Sie dafür wieder das `OnSelect`-Ereignis Ihrer Schaltfläche, und ergänzen Sie den dritten Parameter des `Navigate`-Befehls um folgenden Code (siehe Abbildung 5.242):

```
Navigate(
    scrZielBildschirm;
    Fade;
    {
        locMeinName:User().FullName
    }
)
```

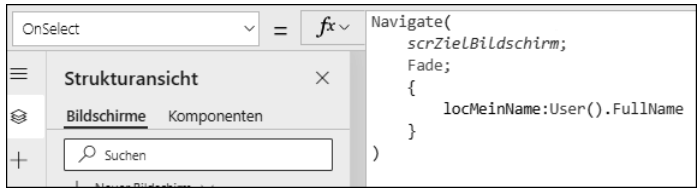

Abbildung 5.242 Parameterübergabe im Navigate-Befehl

Auf der Zielseite wird nun eine lokale Variable erzeugt, die den Wert aus `User().FullName` enthält.

Das `User()`-Objekt haben Sie im vorherigen Beispiel kennengelernt. Es stellt Ihnen verschiedene Eigenschaften des angemeldeten Benutzers zur Verfügung.

12. Wechseln Sie nun in der Strukturansicht auf Ihren Zielbildschirm `scrZielBildschirm`, und platzieren Sie dort eine Beschriftung aus dem EINFÜGEN-Menü der oberen Befehlsleiste (siehe Abbildung 5.243).

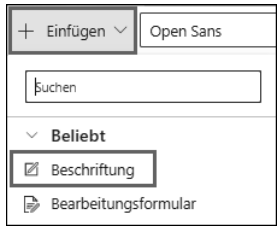

Abbildung 5.243 Beschriftung einfügen

13. Platzieren Sie das Element auf dem Bildschirm, und wählen Sie die Eigenschaft `Text`. Ersetzen Sie den Standardwert `Text` nun durch den Wert, den Sie über die `Navigate`-Funktion empfangen (siehe Abbildung 5.244):

    ```
    "Mein Name ist: " & locMeinName
    ```

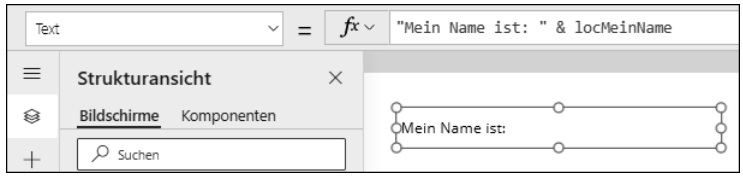

Abbildung 5.244 Wert auf Zielseite empfangen und ausgeben

    Gegebenenfalls erhöhen Sie die Breite des Beschriftungselements durch Setzen der Eigenschaft `Width` in der Eigenschaftenauswahl oder der Eigenschaft `Breite` im Eigenschaftenbereich, damit Sie ausreichend Platz haben.

14. Nun fahren Sie mit Schritt 7 fort, um das Ergebnis zu überprüfen. Wenn alles funktioniert, werden Sie nach Betätigung Ihrer Schaltfläche zum Zielbildschirm geleitet und lesen dort Ihren Namen (siehe Abbildung 5.245).

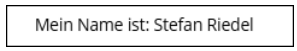

Abbildung 5.245 Es hat funktioniert.

5.10.3 Zum vorherigen Bildschirm navigieren

Oftmals werden Bildschirme von verschiedenen Punkten Ihrer App angesprungen. Um also zum aufrufenden Bildschirm zurückkehren zu können, benötigen Sie eine Art »Zurück«-Funktion. Passenderweise verfügt Power Fx über die Funktion `Back`.

`Back` ist parameterlos und führt Sie auf den vorherigen Bildschirm, wie Sie es auch von Ihrem Browser kennen.

Wenn Sie das Beispiel aus dem vorherigen Abschnitt 5.10.2 entwickelt haben, können Sie die Back-Funktion ganz einfach kennenlernen.

1. Wählen Sie den Bildschirm `scrZielBildschirm`.
2. Fügen Sie dem Bildschirm nun eine *Schaltfläche* oder alternativ ein *Symbol* Ihrer Wahl aus dem Menüpunkt EINFÜGEN der oberen Befehlsleiste hinzu, und ändern Sie die Eigenschaft `Text` in »Zurück«. Haben Sie ein Symbol gewählt, können Sie keine `Text`-Eigenschaft setzen!
3. Wählen Sie nun das Ereignis `OnSelect` aus der Eigenschaftenliste, und ersetzen Sie den vorhandenen Code durch die Funktion `Back` (siehe Abbildung 5.246).
4. Wählen Sie nun den ersten Bildschirm aus dem vorigen Beispiel, und starten Sie die App.

5. Ein Klick auf die vorhandene Schaltfläche sollte Sie auf den Bildschirm SCRZIELSCREEN weiterleiten. Klicken Sie nun auf die soeben hinzugefügte Schaltfläche oder das Icon, werden Sie auf den Ausgangsbildschirm geleitet.

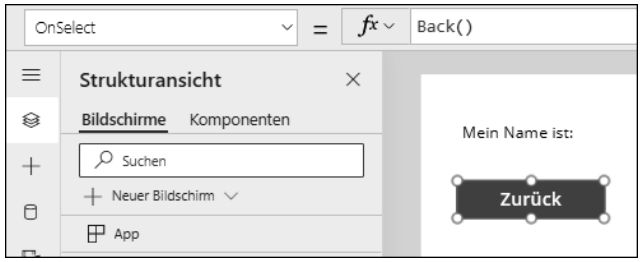

Abbildung 5.246 Die Back-Funktion

5.10.4 Auswahl eines Datensatzes aus einer Liste von Datensätzen

Ein klassisches Beispiel für den Einsatz eines Katalogs ist die Anzeige von Datensätzen, aus denen man einen Datensatz auswählt, um diesen entweder zu bearbeiten, zu löschen oder zu editieren.

1. Erstellen Sie dazu eine neue leere App im Tablet-Format, und nennen Sie sie »Fahrzeugverwaltung«.

   Sie befinden sich nun im Power Apps Studio und können damit beginnen, eine Datenquelle zu importieren.

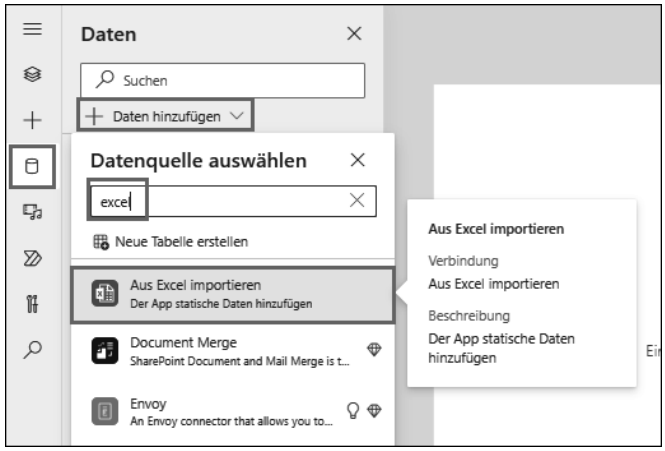

Abbildung 5.247 Daten aus Excel importieren

2. Wählen Sie nun den Datenbereich aus dem App-Erstellungs-Menü (siehe Abbildung 5.247).

3. Suchen Sie nach dem Konnektor Aus Excel importieren, und selektieren Sie den Konnektor.
4. Wählen Sie die Excel-Datei *Datenstruktur.xlsx*, und bestätigen Sie die Wahl mit Öffnen. Auf der rechten Bildschirmseite öffnet sich ein Bereich mit allen verfügbaren Tabellen der Excel-Liste. Wählen Sie hier die Liste »Fahrzeuge«, und bestätigen Sie Ihre Wahl mit Verbinden.

   Danach geht es um die Wahl des Katalogs.

Abbildung 5.248 Tabelle auswählen

5. Wählen Sie den Bildschirm Screen1 in der Strukturansicht, und benennen Sie ihn in »scrKatalog« um.
6. Nun wählen Sie aus dem Bereich Layout des Menüpunkts Einfügen (obere Befehlsleiste) das Element Leerer vertikaler Katalog.

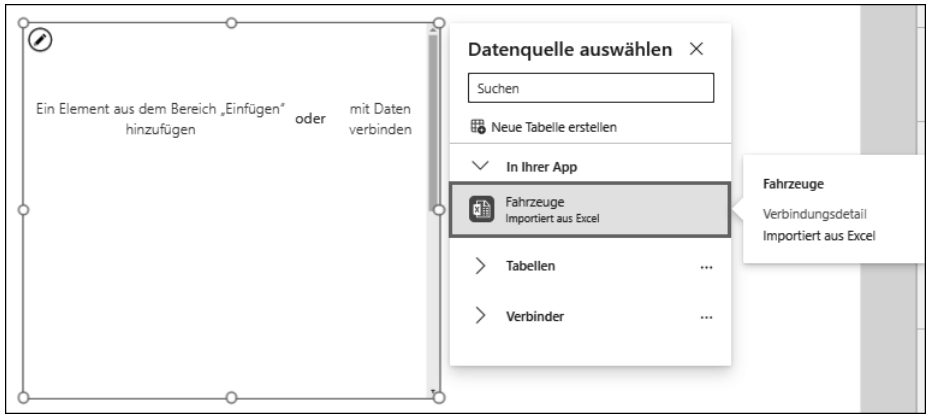

Abbildung 5.249 Datenquelle binden

7. Bestätigen Sie die angebotene Datenquelle »Fahrzeuge«. Die Datenquelle ist nun an den Katalog gebunden.
8. Legen Sie für den Katalog folgende Eigenschaften fest:

Eigenschaft	Wert
X	0
Y	60
Breite/Width	640
Höhe/Height	`Parent.Height-Self.Y`

Tabelle 5.52 Die Eigenschaften des Katalogs

9. Markieren Sie den Katalog nun in der Strukturansicht, und nennen Sie ihn »galFahrzeuge«.

   Dann geht es mit dem Formatieren der Datensatzvorlage weiter.

10. Markieren Sie den Katalog in der Strukturansicht, oder wählen Sie EIN ELEMENT AUS DEM BEREICH »EINFÜGEN« HINZUFÜGEN, um die Vorlage in den Bearbeitungsmodus zu setzen (siehe Abbildung 5.250).

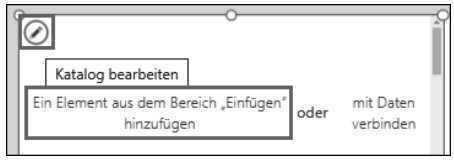

Abbildung 5.250 Die Katalogvorlage bearbeiten

11. Fügen Sie nun vier BESCHRIFTUNGEN ein, wählen Sie für jedes Element die Eigenschaft TEXT aus der Eigenschaftenliste, und binden Sie sie wie in Abbildung 5.251 jeweils an die folgenden Felder:
   - `ThisItem.Nummer`
   - `ThisItem.Marke`
   - `ThisItem.Modell`
   - `ThisItem.Laufleistung`

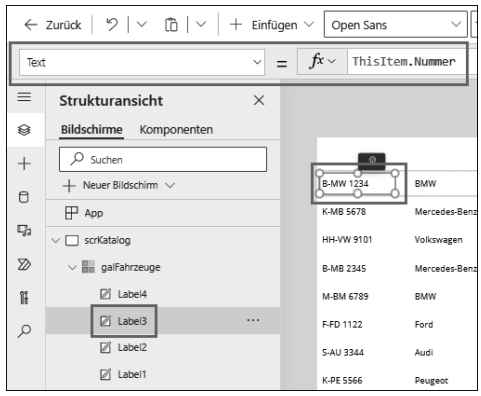

Abbildung 5.251 Binden eines Felds an ein Beschriftungselement

Reihenfolge der Beschriftungselemente

Ich habe die Beschriftungen in Abbildung 5.251 umsortiert, weshalb bei Ihnen die Reihenfolge der Elemente zu der gebundenen Spalte unterschiedlich sein kann.

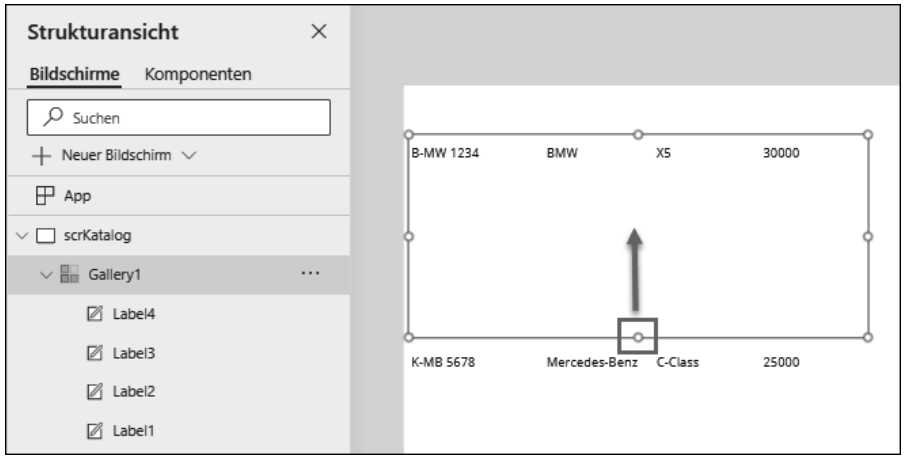

Abbildung 5.252 Beschriftungen einfügen und binden

12. Anschließend verkleinern Sie die Katalogvorlage, bis jeder Datensatz den Raum hat, den er benötigt (siehe Abbildung 5.252). Nun sollte der Katalog mit den Fahrzeugen in etwa aussehen wie in Abbildung 5.253.

Abbildung 5.253 Der fertige Katalog

Nun geht es darum, das selektierte Element in der Datenliste hervorzuheben, indem wir die Hintergrundfarbe ändern. Zudem soll der ausgewählte Datensatz in einem Formular angezeigt werden.

13. Markieren Sie jetzt den Katalog GALFAHRZEUGE in der Strukturansicht, und wählen Sie die Eigenschaft `TemplateFill` aus der Eigenschaftenliste. Schreiben Sie nun den folgenden Code in die Bearbeitungsleiste, und starten Sie Ihre App in der Vorschau:

```
If(ThisItem.IsSelected;Color.LightGray;Color.White)
```

Kapitel 15
Community

Hier dreht sich alles um die Power-Platform-Community und damit um Menschen, die anderen pro bono beim Umgang mit der Power Platform helfen.

Eine Community ist zunächst eine Gruppe von Menschen, die sich für ein gemeinsames Thema, in diesem Fall die Power Platform, interessiert und die in mehr oder weniger regelmäßigen Meetups oder Konferenzen, vor Ort oder online zusammenkommt. Üblicherweise sind diese Zusammentreffen kostenfrei durch Community-Leader organisiert. Konferenzen nehmen auch schon mal ein kleines Eintrittsgeld, um einen Teil der Kosten zu decken. Online-Veranstaltungen sind aber üblicherweise kostenfrei zu besuchen.

Diese Treffen leben zum einen von den Sprecherinnen und Sprechern, die ihr Wissen kostenfrei teilen, aber auch vom interessierten Publikum, das sich an Diskussionen beteiligt, die üblicherweise nach einem Vortrag stattfinden. Letztere ist auch der Grund, warum die meisten Meetups ein offenes Ende mit viel »Community-Time« haben.

Neben Meetups und Konferenzen teilen aktive Community-Mitglieder ihr Wissen auch in sozialen Netzwerken wie LinkedIn und Twitter/X sowie in Blogs oder stellen kostenfreie Apps, Flows, Erweiterungen und Dokumente auf GitHub zur Verfügung.

Das Wichtigste aber ist, dass das alles aus der puren Lust am Teilen von Wissen geschieht, ohne eine monetäre Gegenleistung dafür zu erwarten.

Und wenn Sie sich fragen, was Ihnen das bringen soll, dann habe ich hier ein paar Argumente für Sie:

- In der Community finden Sie Expertinnen und Experten zum Anfassen. Sie haben eine Frage und bekommen eine Antwort. Wenn Sie ein kniffeligeres Problem haben, stürzen sich unter Umständen gleich mehrere Community-Mitglieder darauf, um es gemeinsam mit Ihnen zu lösen.
- Sie wollen über neue Features informiert werden? Nichts ist für die Community spannender als Neues auszuprobieren. Wenn Sie wissen wollen, was funktioniert oder was nicht funktioniert, dann erfahren Sie es bei uns, der Community, in Form

von Blogs, Posts auf LinkedIn oder Twitter/X oder in einem Vortrag oder Diskussion bei einem Event.

- Bis auf wenige Ausnahmen, ich erwähnte es bereits, können Sie kostenfrei Vorträgen lauschen, Expertenrat einholen und sich ein Netzwerk aufbauen. Nur hier bekommen Sie Best-Practices frei Haus.
- Sie haben eine Idee, die Sie alleine nicht umsetzen können? Ich wette, Sie finden in der Community passende Mitstreiter.
- Sie wollen sich selbst als Sprecherin oder Sprecher versuchen? Dann hilft Ihnen die Community und gibt Ihnen Tipps und die Möglichkeit, sich auszuprobieren. Wir haben alle mal angefangen. Natürlich helfen wir Ihnen auch, wenn Sie eine eigene lokale Community gründen und aufbauen wollen.

Sprechen Sie uns einfach an

Grundsätzlich verdienen die aktiven Community-Mitglieder ihr Geld mit ihrem Wissen. Falls Sie irgendwo nicht weiterkommen oder in einem Blog-Artikel etwas nicht verstehen, dann sprechen Sie die Personen einfach freundlich an. Ich habe schon mehrfach gehört, dass sich Menschen nicht trauen uns anzusprechen, weil die Frage zu trivial sei. Das ist Quatsch! Trauen Sie sich. Wir können zwar nicht alles beantworten und eventuell dauert es etwas, bis wir reagieren. Möglicherweise geht auch mal eine Frage unter. Das ist dann kein böser Wille, denn auch wir haben alle einen Job und Familie.

Ich selber betreibe mit einem guten Freund drei Communities, von denen sich zwei monatlich bzw. quartalsweise als Hybrid-Event in Hannover bzw. online treffen, sowie eine wöchentliche Online-Experten-Runde: das *Power Atelier*, das jeden Freitag von 10 Uhr bis 11 Uhr stattfindet. Dort probieren wir Dinge in der Power Platform aus, aber auch versuchen auch, jede Frage zu beantworten, die uns die Teilnehmer stellen.

15.1 Was sind Microsoft MVPs?

Oftmals finden Sie die drei Buchstaben MVP in Verbindung mit IT-Expertinnen und -Experten. Diese drei Buchstaben stehen für »Most Valuable Professional«. Der Titel wird jedes Jahr von Microsoft an ausgewählte Personen verliehen, die sich in einem technischen Bereich mit ihren Pro-Bono-Aktivitäten für die Community besonders engagiert haben. Falls Sie einen Experten suchen, nutzen Sie die Suche auf *mvp.microsoft.com*. Dort können Sie eine Person suchen oder Profile nach Land, Kategorie oder Technologiebereich filtern. Für die Power Platform wählen Sie dort die Kategorie Business Applications bzw. direkt Power Apps oder Power Automate als Technologiebereich.

15.2 Wie organisiert sich die Community?

Online-Inhalte wie Blog-Posts oder Beiträge in sozialen Netzwerken erscheinen nach Zeit, Lust und Laune der Verfasser. Manche schreiben wöchentlich, manche alle paar Monate.

Meetups und Konferenzen finden üblicherweise regelmäßig statt und werden über Plattformen wie MeetUp oder Sessionize organisiert. Manche nutzen dafür auch LinkedIn-Gruppen.

15.3 Wie werde ich Mitglied der Community?

Grundsätzlich gibt es keine Aufnahmeanträge oder Ähnliches. Wenn Sie einfach passiv dabei sein wollen, ist das absolut in Ordnung. Damit Sie von regelmäßigen Treffen erfahren, sollten Sie wenigstens einen LinkedIn-Account besitzen bzw. sich auf Meetup.com registrieren. Dort können Sie passende Gruppen suchen. Eine Auswahl an Communities, Konferenzen, Blogs, Videos und Hashtags sollte Ihnen helfen, die richtigen Infos und die passende Umgebung zu finden.

Wollen Sie aktiv dabei sein, ist die wichtigste Voraussetzung, dass Sie grundsätzlich bereit sind Ihr Wissen zu teilen. Das können Sie zum Beispiel in Form von Diskussionsbeiträgen auf einem lokalen Community-Event, einem Vortrag oder einem Blog-Post tun. Besuchen Sie einfach ein Community-Event in Ihrer Nähe, aber seien Sie achtsam, denn wer einmal in der Community ist, bleibt dort meist für immer. Noch nie habe ich erlebt, dass jemand die Community verlassen hat.

15.4 Meetups, Online-Communities, Projekte und Hashtags

In diesem Abschnitt finden Sie eine Auswahl verschiedener Communities, die Sie vor Ort oder online besuchen können. Darunter ist auch eine kleine Auswahl freier Projekte aus der deutschsprachigen Community. Den Abschluss bildet eine Liste von Hashtags, die Sie in sozialen Netzwerken wie LinkedIn oder Twitter/X zur Suche nach Informationen rund um die Power Platform nutzen können.

Meetups und Communities

- *Power Platform User Group Hannover*: *https://www.meetup.com/de-DE/power-apps-power-automate-usergroup-hannover/*
- *Power Platform User Group Cologne*: *https://www.meetup.com/de-DE/power-platform-ug-cologne/*

- *Power Platform User Group München*: *https://powerusers.microsoft.com/t5/PPMUG-Power-Platform-Munich-User/gh-p/pp_PPMUG_usergroup*
- *Power Platform Hamburg User Group*: *https://www.meetup.com/de-DE/pphhug/*
- *Microsoft Dynamics Meetup Deutschland*: *https://www.meetup.com/de-DE/Microsoft-Dynamics-Meetup-of-Berlin/*
- *Das Power Atelier*, die wöchentliche Expertenrunde zum Fragen, Zuschauen und Mitmachen. Jede Woche Freitag online von 10:00 – 11:00 Uhr. Mitteilung über Power Platform User Group Hannover.

Konferenzen

- *DACH Powerthon*: *https://powerthon.info/*

  Eine jährliche Veranstaltung rund um die Power Platform, die meist an einem Wochenende in der zweiten Jahreshälfte von Freitag bis Samstag stattfindet.
- *Global Power Platform Bootcamp*: *https://www.powerplatformbootcamp.com/*

  Das GPPBC ist eine weltweit stattfindende Konferenz, die Ende Februar von den lokalen Power Platform Communities, wie z. B. Hamburg oder Hannover, durchgeführt wird.
- *European Collaboration Summit*: *https://www.collabsummit.eu/*

  Die wohl größte Konferenz, veranstaltet von der Community für die Community.

Projekte

- *Powerapps Docstring* (Sebastian Muthwill): *https://github.com/sebastian-muthwill/powerapps-docstring*

  Dieses Werkzeug von Sebastian Muthwill liest die Kommentare innerhalb Ihrer App aus und fertigt daraus eine Dokumentation an.
- *PowerDocu* (Rene Modery): *https://github.com/modery/PowerDocu*

  Dieses Tool erzeugt Dokumentationen aus exportierten Flow- und App-Paketen sowie Lösungen.
- *Power Apps Design Toolkit* (Luise Freese und Robin Rosengrün): *https://pnp.github.io/blog/post/material-design-component-library-for-power-apps/*

  Luise Freese und Robin Rosengrün stellen hier eine kostenfreie Komponentenbibliothek mit hochwertigen Steuerelementen zur Verfügung.
- *Power Apps Helper* (Michael Megel): *https://github.com/megel/powerapps-helper*

  Der Power Apps Helper von Michael Megel sind ein Add-On für Visual Studio Code, um Lösungen und Apps zu packen und zu entpacken.

Blogs und Videos

- Stefan Riedel: *https://stefanriedel.wordpress.com/*
- Tomislav Karafilov: *https://tkarafilov.wordpress.com/*
- Robin Rosengrün: *https://www.r2power.de/blog*
- Michael Megel: *https://never-stop-learning.de/*
- Marcel Lehmann: *https://lehmann.ws/*
- Luise Freese: *https://www.m365princess.com/*
- Tomislav Karafilov und Stefan Riedel: *https://www.youtube.com/@Cloud-CouchRocks*
- April Dunnam: *https://www.youtube.com/c/aprildunnam*
- Reza Dorrani: *https://www.youtube.com/@RezaDorrani*
- Shane Cows: *https://www.youtube.com/@ShanesCows*

Hashtags

Mit den folgenden Hashtags finden Sie auf LinkedIn bzw. Twitter/X Experten und Community-Aktivitäten rund um die Power Platform:

- *#PowerAddicts*
- *#SharingIsCaring*
- *#CommunityRocks*
- *#NeverStopLearning*
- *#PowerApps*
- *#PowerAutomate*
- *#PowerPlatform*