

8-9/25

Razor Components  
in Blazor S. 54

Strategien zum Lösen  
von Problemen S. 94

Videodaten mit Java-  
Script abgreifen S. 106

8-9/25

# dotnetpro by DDX

## Verteilte Anwendungen und Infrastruktur

Wie wir verteilte Anwendungen erstellen  
und welche Auswirkungen Design und  
Architektur auf die Systeminfrastruktur  
haben. Ab Seite 6

PROGRAMM AB SEITE 57

.NET DEVELOPER  
CONFERENCE '25

Die Konferenz für .NET-Entwickler

DDX  
EVENT

24. BIS 27. NOVEMBER '25  
Pullman Cologne Hotel in Köln



.NET Aspire • IaC • PowerShell • OpenAI und Fabric • Clean Code • PostgreSQL • Blazor QuickGrid

# Du willst ein Thema durchdringen – ohne Zeitverlust, ohne heiße Luft?

Ein digitales Kompaktformat, das auf den  
Punkt bringt, was du wirklich brauchst:

- **READ:** Finde mit unseren Online-Beiträgen deinen Einstieg in das Spotlight-Fokusthema.
- **LISTEN:** Im Interview ordnen Redaktion und Expert:innen relevante Aspekte für dich ein.
- **LEARN:** Unsere Lernvideos vertiefen das Fokusthema – kompakt und übersichtlich, professionell und praxisnah.



**NEW SPOTLIGHT**

„Azure IoT“

Ab sofort online verfügbar







# Sein oder nicht sein

Die KI verändert ohne Zweifel die Grundfesten der Softwareentwicklung. Doch es greift zu kurz, sie deshalb nur zu verteufeln.

**E**rinnern Sie sich an den Schwerpunkt der vorigen dotnetpro? Darin ging es um die Integration von KI in Low Code – ein in meinen Augen faszinierender Ansatz. Früher hätten wir uns Zeit genommen und entsprechende Lösungen und Tools angeschaut, um beurteilen zu können, ob ihr Einsatz für unsere Projekte Sinn ergibt. In Zeiten, in denen ein KI-Hype den nächsten jagt, ist das aber kaum mehr möglich.

Nun also Agentic AI. Allerdings glaube ich, dass die agentenbasierte Softwareentwicklung uns – anders als bisherige KI-Entwicklungen – länger begleitet, weil sie einen zentralen Aspekt der Softwareentwicklung berührt.

Bisher diente die KI als Pair Programmer oder „Juniorentwickler“, der monotone Arbeiten übernimmt und selbständig erledigt: Refactorings, Tests, Deployments. Sie fungierte als technischer Helfer, sie unterstützte. Indem Agenten nun aber in der Lage ist, autark Software zu bauen, hebt Agentic AI jene Unterstützung auf ein neues Niveau. Mit Agentic AI betritt die KI damit ein Terrain, das bisher als Domäne des Menschen galt – die Analyse und Planung, das Design und die Architektur von Softwaresystemen.

## „Das Bild von Entwickler:innen, die den Bau von Software künftig orchestrieren, gefällt mir.“

Damit bewegt sich auch die KI-Diskussion weg von überwiegend technischen hin zu weit grundsätzlicheren Fragen: Wie sieht die Software der Zukunft aus? Welchen Teil der Software baut die KI und welchen der Mensch? Welche Rolle und welche Aufgaben übernehmen Entwickler:innen darin? Wie und worüber definieren sie in dieser Arbeitswelt ihren eigenen Wert? Es wird spannend sein, diese Veränderungen zu beobachten und sie aktiv zu begleiten.

Passend dazu befassen sich in der aktuellen dotnetpro eine Reihe von Beiträgen mit Themen, die in der Diskussion um das Verhältnis von Mensch und KI an Bedeutung gewinnen könnten. Abseits von all den technischen Aspekten richten sie den Blick auf Themen wie Achtsamkeit im Umgang miteinander, gegenseitige Wertschätzung und Respekt. Diese Artikel möchte ich Ihnen ganz besonders ans Herz legen.

Ich wünsche Ihnen eine schöne Zeit mit der neuen dotnetpro!

Herzlichst

Fernando Schneider  
Chefredakteur dotnetpro



**Dotnet-Doktor**

Holger Schwichtenberg verrät, wie er KI-Tools in Outlook integriert hat, um seine Workflows zu optimieren (S. 77)



**Stefan Mintert**

hebt die Bedeutung von Achtsamkeit und Wertschätzung für den Erfolg von Unternehmen hervor (S. 124)



# INHALT

## Focus

### YAML-Chaos ade!

Cloud-native-Anwendungen komplett in C# entwickeln

6

### Nieder mit dem ClickOps!

Infrastructure as Code für verteilte .NET-Systeme

12

## .NET Stack

### Am Puls der Zeit

Die .NET Developer Conference 2025

18

### Data Access: Schnell mal ein Grid bauen

Microsofts Datentabellen-Steuerelement

QuickGrid für Blazor

20

### Moderne Produktivität

Automatisieren mit Microsoft .NET, Dataverse und Power Automate

28

### PowerShell im Dialog

Messageboxen und Dialoge mit PowerShell darstellen

35

### Ordnung statt Chaos

Den Projektzoo organisieren

42

### Eine Spracherweiterung auf dem Prüfstand

Default Interface Methods in C#

46

### NETirol: Razor Components

Wie Razor-Komponenten und Blazor zusammenarbeiten

54

## Artificial Intelligence

### Das Dreamteam

Low Code und künstliche Intelligenz

62

### Von Daten zu Antworten

Microsoft Fabric als Datenbackbone für GenAI-Anwendungen

66

### KI in Outlook

Fallstudie: KI-Integration in der Kundenkommunikation

77

### Mit Denkmustern brechen

Wie KI die Softwareentwicklung verändert

88

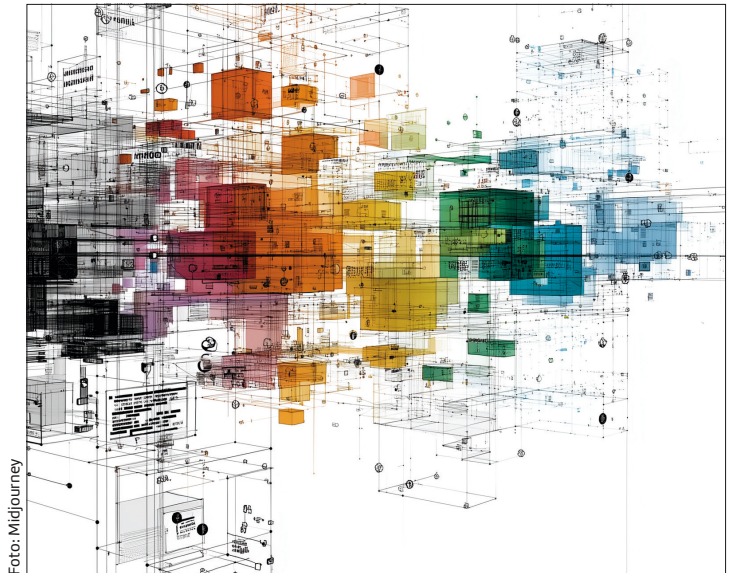
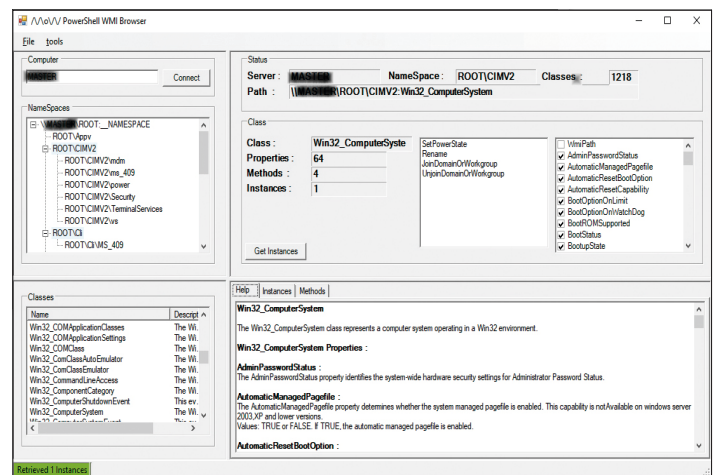


Foto: Midjourney

.NET Aspire und Infrastructure as Code als Grundlage für verteilte Systeme und cloudbasierte Anwendungen.

6



PowerShell: Wie sich Interaktion mit dem Anwender durch .NET-Integration der angezeigten Dialoge optimieren lässt.

35

## Experten in dieser Ausgabe



David Tielke weiß, was passiert, wenn Rollen in der Softwareentwicklung unbesetzt bleiben.

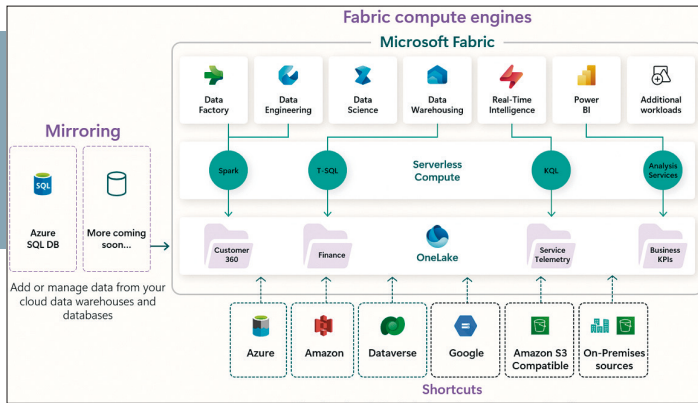
92



Bernhard Pichler plädiert für einen ausbalancierten Weg beim Umgang mit Schuld im Teamwork.

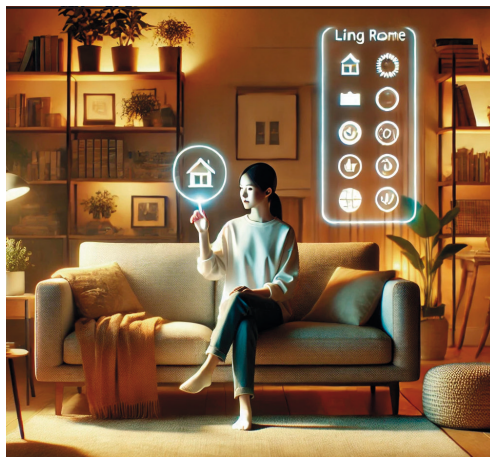
122





**Microsoft Fabric** liefert bereits heute eine leistungsstarke Datenplattform für die Umsetzung von GenAI-Projekten.

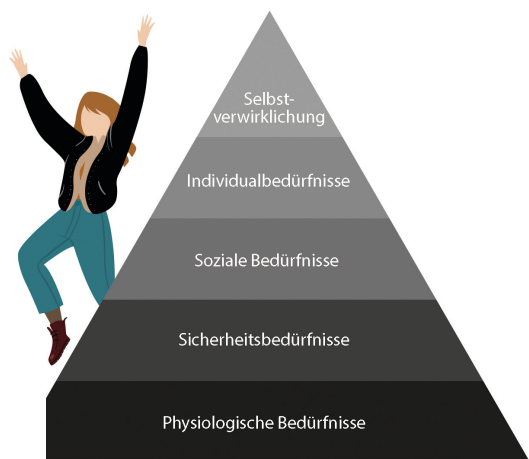
66



### KI und Software-architektur:

Wie sieht die Software der Zukunft aus? Wie verändern sich unsere Rollen, welche Aufgaben erwarten uns? Und wie sollten wir in Zukunft arbeiten?

88



### Bitte, bitte nicht schon wieder!

Warum wir Kolleg:innen oft als die Verursacher von Problemen ansehen – und warum ein zweiter Blick auf die wirklichen Ursachen oft schon die Lösung ist.

131



## Cross Concerns

### Dauids Deep Dive: Das Rollenproblem

In der Softwareentwicklung gibt es keine „unwichtigen“ Rollen

92

### Problemlösen leicht gemacht

Denkanstöße zur Findung geeigneter Strategien

94

### Abhängigkeiten und Architekturmuster

Clean Code und Architektur, Teil 2

98

## What else?

### Browserspionage

Bildschirminhalte aufzeichnen mit JavaScript

106

### Das Concurrency-Modell

Wie PostgreSQL parallele Abfragen mehrerer Benutzer bearbeitet

112

## People & Soft Skills

### Time-out: (K)eine Frage der Schuld

Das Spannungsfeld zwischen systemischen Problemursachen und persönlicher Verantwortung

122

### The Mindful Software Developer

Softwareentwicklung in Krisenzeiten

124

### „Der macht mich rasend ...“

Umgang mit schwierigen Kollegen

131

## Knowledge

### Neuer Standort, noch mehr Leidenschaft

DWX 2025 in Mannheim

140

### Editorial

3

### Inhalt

4

### Arbeitsmarkt

138

### Lernstoff

144

### Impressum / Vorschau / Inserenten

146



**CLOUD-NATIVE-ANWENDUNGEN KOMPLETT IN C# ENTWICKELN**

# YAML-Chaos ade!

.NET Aspire orchestriert verteilte Dienste und Cloud-Ressourcen im Code. Schritt für Schritt bauen Sie damit eine verteilte Anwendung lokal und in Azure auf.

**D**ie Entwicklung verteilter Anwendungen erfordert ein hohes Maß an Koordination und Planung. Es reicht längst nicht mehr aus, nur einen einzelnen Dienst zu schreiben und zu starten. Moderne Anwendungen bestehen aus mehreren Komponenten, die gemeinsam funktionieren müssen: APIs, Benutzeroberflächen, Hintergrundprozesse, Datenbanken, Caches und vieles mehr.

Damit diese Elemente reibungslos miteinander interagieren können, muss sichergestellt sein, dass alle Services korrekt gestartet werden, ihre Konfigurationen zur Verfügung stehen, sie sich gegenseitig finden und aufrufen können (Service Discovery) und sowohl Logs als auch Metriken und Traces erfasst werden.

Hinzu kommt: Die Bedingungen in der Entwicklungsumgebung unterscheiden sich oft stark von der Produktionsumgebung. Lokal wird mit Docker und lokalen Verbindungsdaten gearbeitet, in der Cloud hingegen mit Ressourcen wie Azure SQL, Azure Redis oder Azure Service Bus. Diese Dis-

krepanz erzeugt Aufwände bei der Konfiguration, beim Debugging und beim Deployment.

Viele Entwickler greifen daher zu Docker Compose, um die lokalen Services in Containern zu orchestrieren. Dabei wird in YAML definiert, welche Container mit welchen Umgebungsvariablen, Ports und Netzwerken gestartet werden sollen. Das funktioniert gut – solange alle Komponenten containerisiert sind und lokal laufen sollen. Doch YAML hat einige Schwächen: Es ist schwer zu warten, bietet keine Typprüfung, lässt sich schlecht testen und skaliert nicht gut bei wachsender Komplexität. Viel entscheidender jedoch: Es entsteht ein Medienbruch: Entwickler schreiben C#-Code für ihre App, aber YAML für die Orchestrierung. Und: Sobald ein Teil der Infrastruktur in Azure laufen soll, etwa der Redis-Cache, müssen zusätzlich ARM-Templates, Bicep-Files oder Terraform-Definitionen gepflegt werden.

.NET Aspire verspricht genau hier eine nahtlose Alternative. Es erlaubt, verteilte Anwendungen komplett in .NET zu

Foto: Midjourney



beschreiben – also in C#. Alle Dienste und deren Abhängigkeiten sowie Konfigurationen werden in einer zentralen Datei orchestriert, die durch den AppHost verwaltet wird. Aspire sorgt dafür, dass die Dienste lokal in der richtigen Reihenfolge gestartet werden, stellt Konfigurationswerte bereit, erzeugt Verbindungsstrings automatisch, startet notwendige Container wie Redis und generiert eine Übersicht über Logs, Health Checks und Traces.

Kurz gesagt: Aspire bringt dieselbe Funktionalität wie Docker Compose, geht aber darüber hinaus. Es ersetzt YAML durch C#-Code, vereinfacht den Zugang zu Azure-Ressourcen, unterstützt Observability direkt und reduziert die Kluft zwischen lokaler Entwicklung und Cloud-Deployment. Für .NET-Entwickler bedeutet das: Eine gewohnte Sprache, eine integrierte Umgebung, ein nahtloser Weg von der IDE bis zur Cloud. Gerade für Neueinsteiger, die erste Erfahrungen mit Cloud-nativen Anwendungen sammeln wollen, bietet Aspire damit einen sanften Einstieg. Ohne Vorwissen zu Docker, Kubernetes oder Azure-ARM-Templates können sie in wenigen Minuten eine verteilte Anwendung starten, die aus mehreren Diensten besteht, diese verbinden, Logs einsehen und sogar verteilte Traces betrachten. .NET Aspire reduziert dabei die Komplexität und liefert einen klaren Mehrwert gegenüber traditionellen Methoden.

## .NET Aspire einrichten

Die Einrichtung eines Aspire-Projekts folgt einem bestimmten Muster, das bewusst auf Klarheit, Modularität und Wartbarkeit ausgelegt ist. Dabei steht der sogenannte AppHost im Zentrum – ein spezielles .NET-Projekt, das alle weiteren Dienste und Ressourcen orchestriert, siehe [1]. In klassischen Multi-Service-Anwendungen wäre das Äquivalent eine Kombination aus mehreren Docker-Compose-Dateien, Shell-Skripten, Konfigurationsdateien und Service-Definitionen. In .NET Aspire dagegen konzentriert sich die gesamte Struktur in einer .NET-Anwendung, die mithilfe der Klasse *DistributedApplicationBuilder* eine deklarative Beschreibung der Zielumgebung vornimmt.

Einsteiger können sich den AppHost wie ein Drehbuch vorstellen, in dem alle beteiligten Akteure genannt werden: Welche Services beteiligt sind, wie sie heißen, in welcher Reihenfolge sie gestartet werden, ob sie miteinander sprechen können und mit welchen Ressourcen sie verknüpft sind. Der große Vorteil: Dieses Drehbuch ist in C# geschrieben. Dadurch ist die Entwicklungsumgebung mitsamt IntelliSense, Tooling, Build-Checks und Debugging-Fähigkeiten vertraut – kein Kontextwechsel nötig, kein Wechsel in YAML oder externe DSLs.

Innerhalb des AppHosts definieren Sie typischerweise ein oder mehrere Projekte, etwa das Frontend, das Backend, einen Worker oder andere Dienste. Diese Projekte sind in Ihrer Lösung enthalten und werden durch den AppHost nicht nur gestartet, sondern auch mit Konfigurationswerten versorgt. Das ist besonders wichtig, da ein Projekt – beispielsweise das Frontend – wissen muss, wie es das Backend erreichen kann. Anstatt selbst URLs, Ports oder Umgebungsvariablen zu konfigurieren, übernimmt .NET Aspire die Service Discovery. Sie schreiben lediglich *WithReference("backend")*, und Aspire

registriert die Abhängigkeit intern so, dass beim Start automatisch die richtigen Verbindungsinformationen in der Konfiguration landen. Das spart Zeit und minimiert Fehler.

Ebenso deklarieren Sie innerhalb des AppHosts die benötigten Ressourcen. Möchte das Backend etwa Redis nutzen, genügt *AddRedis("weather-cache")*. .NET Aspire startet daraufhin im Hintergrund einen Redis-Container, setzt den Connection-String automatisch zusammen und macht diesen für andere Projekte nutzbar. Das Frontend oder das Backend brauchen sich nicht darum zu kümmern, wo Redis läuft, unter welchem Hostnamen oder Port – .NET Aspire abstrahiert diese Details und hält die Umgebung stabil. Selbst ein Wechsel von einem lokalen Redis zu Azure Cache for Redis erfordert nur einen minimalen Eingriff: Sie ersetzen *AddRedis(...)* durch *AddAzureRedis(...)*. Alle anderen Mechanismen bleiben identisch. Kein manueller Umbau nötig, keine zusätzlichen Deployment-Definitionen.

## .NET Aspire Dashboard

Ein häufig übersehener, aber bedeutender Aspekt von .NET Aspire ist das eingebaute Dashboard. Dieses öffnet sich automatisch im Browser, sobald Sie Ihr Aspire-Projekt starten. Dort erkennen Sie visuell, welche Services gerade aktiv sind, welche Ressourcen verbunden sind und ob es Fehler oder Verzögerungen gibt. Es bietet direkte Links zu Logs, Health-Endpunkten und sogar verteilte Traces, die mit OpenTelemetry erstellt wurden. Wenn beispielsweise eine Benutzeranfrage über das Frontend, das Backend und anschließend den Redis-Cache durchläuft, dann erscheint dieser Ablauf als nachvollziehbare Kette in der Trace-Ansicht. Gerade in frühen Phasen der Entwicklung oder bei der Fehlersuche ist dieses visuelle Feedback unschätzbar wertvoll.

Was bislang oft mühsam mit separaten Tools – etwa Jaeger, Zipkin oder Kibana – aufgesetzt werden musste, ist mit .NET Aspire sofort vorhanden. Ohne die Notwendigkeit einer detaillierten Konfiguration funktionieren Logging, Tracing und Konfigurationsweitergabe auf hohem Niveau, was nicht nur das Onboarding neuer Teammitglieder erleichtert, sondern auch eine bessere Stabilität im Betrieb ermöglicht.

## Aufbau eines .NET-Aspire-Projekts

Ein Aspire-Projekt besteht in der Regel aus mehreren strukturierten Bestandteilen, die zusammen eine verteilte Anwendung bilden. Um das besser nachvollziehen zu können, sehen wir uns die Rollen der verschiedenen Projekte und deren Zweck im Detail an:

- **AppHost-Projekt:** Dieses Projekt ist das Herzstück der Aspire-basierten Anwendung. Es handelt sich um eine .NET-Konsolenanwendung, in der mit dem Distributed Application Builder alle anderen Projekte und Dienste eingebunden und orchestriert werden. Sie geben hier in C#-Code an, welche Projekte (wie zum Beispiel Frontend und Backend), externen Ressourcen (wie Redis oder PostgreSQL) und Konfigurationen zur Anwendung gehören. .NET Aspire sorgt beim Start automatisch dafür, dass alle Abhängigkeiten geladen, Container gestartet und Konfigurationswerte wie Verbindungsstrings zur Verfügung gestellt werden. ►

- **.NET-Projekt:** Diese Projekte bilden die Kernlogik Ihrer Anwendung. Das Frontend stellt die Benutzeroberfläche dar. Hier interagieren Nutzer mit der Oberfläche, senden Eingaben ab und erhalten visuelles Feedback, meist in Form von Daten, die durch API-Aufrufe vom Backend geliefert werden. Das Backend wiederum ist ein ASP.NET Core Web API, das Anfragen des Frontends entgegennimmt, Geschäftslogik verarbeitet und Daten – etwa aus einem Cache oder einer Datenbank – zurückliefert. .NET-Projekte sind im AppHost registriert und werden dort mit ihren jeweiligen Abhängigkeiten versehen. .NET Aspire sorgt automatisch dafür, dass zum Beispiel das Frontend den URL des Backends kennt oder dass das Backend den Connection-String für Redis erhält. Für Sie als Entwickler entfällt damit das manuelle Konfigurieren von Ports, IP-Adressen oder Umgebungsvariablen: Die Infrastrukturvernetzung ist deklarativ und fehlerresistent integriert.
- **Externe Dienste (Ressourcen):** Darunter versteht man Komponenten wie Redis, PostgreSQL, Azure Service Bus oder Azure Cosmos DB. Sie müssen diese nicht selbst entwickeln, sondern deklarieren sie einfach im AppHost. .NET Aspire startet sie lokal als Container (sofern verfügbar) oder bindet sie über Azure-Definitionen direkt als Cloud-Dienste ein. Sie können diese Ressourcen genau wie andere Projekte mit *WithReference(...)* verknüpfen und damit automatisch in die Konfiguration anderer Dienste einbinden.

## .NET-Aspire-Projekt konkret

Besonders einfach ist es mit .NET Aspire, neue Ressourcen und Dienste zu integrieren – sowohl lokal mit Docker-Containern als auch in der Cloud über Azure. Diese Erweiterbarkeit folgt einem einheitlichen, durchdachten Muster. Anhand des .NET-Aspire-Starterprojekts zeige ich im Folgenden, wie sich mit wenigen Handgriffen ein Redis-Cache ergänzen lässt.

Die Ausgangsbasis bildet das mit *dotnet new aspire-starter --output EasyProject* erstellte Starter-Template. Dieses enthält typischerweise zwei Projekte: ein Web-API (*apiservice*) und ein Frontend (*webfrontend*). Wie die Anwendung im AppHost-Projekt konfiguriert wird, zeigt **Listing 1**. Dieser Einstieg bietet bereits eine funktionierende Grundstruktur inklusive Health Checks und Service Discovery. Wenn das Host-Projekt gestartet wird, sehen wir das .NET Aspire Dashboard (**Bild 1**).

Möchte man nun beispielsweise einen Redis-Cache ergänzen, folgt man stets demselben, klar strukturierten Vorgehen:

### Listing 1: AppHost Program.cs (Starterprojekt)

```
var builder =
    DistributedApplication.CreateBuilder(args);

var apiService = builder.AddProject<
    Projects.EasyProject_ApiService>("apiservice")
    .WithHttpsHealthCheck("/health");

builder.AddProject<Projects.EasyProject_Web>(
    "webfrontend")
    .WithExternalHttpEndpoints()
    .WithHttpsHealthCheck("/health")
    .WithReference(apiService)
    .WaitFor(apiService);

builder.Build().Run();
```

#### Schritt 1: Dokumentation aufrufen

Die offizielle Microsoft-Dokumentation zu .NET-Aspire-Integrationen steht unter [2] bereit. Hier findet man Informationen zur Hosting- und Client-Integration zahlreicher Dienste, beispielsweise Redis, PostgreSQL, Azure Service Bus oder Elasticsearch.

#### Schritt 2: Hosting-Integration im AppHost

Zunächst wird die gewünschte Ressource im AppHost deklariert. Für Redis genügt zum Beispiel folgender Befehl:

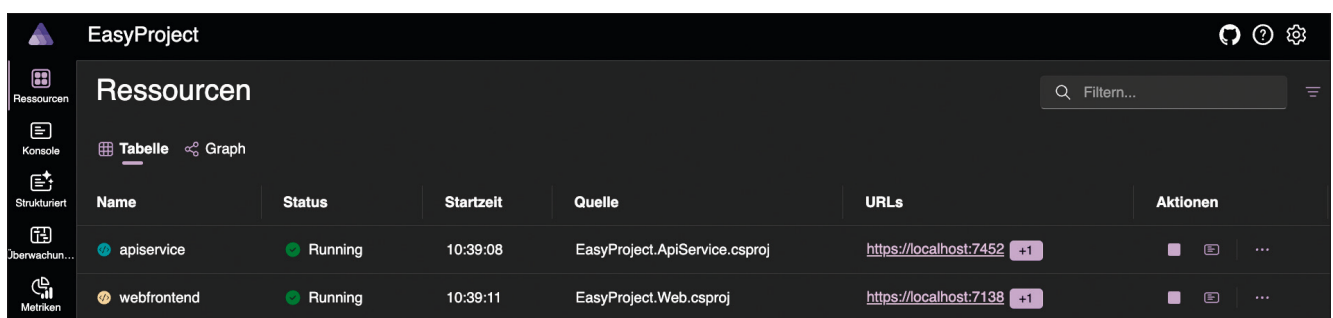
```
dotnet add package Aspire.Hosting.Redis
```

Anschließend kann der Redis-Dienst im Code eingebunden werden:

```
var redis = builder.AddRedis("app-redis");
```

Optional lassen sich Zusatzfeatures wie Redis Commander oder Redis Insights aktivieren:

```
var redis =
    builder.AddRedis("app-redis").WithRedisCommander();
```



The screenshot shows the .NET Aspire Dashboard for a project named 'EasyProject'. The 'Ressourcen' (Resources) tab is active, displaying a table of running services. The table has columns for Name, Status, Startzeit, Quelle, URLs, and Aktionen. Two resources are listed: 'apiservice' and 'webfrontend', both in a 'Running' status.

| Name        | Status  | Startzeit | Quelle                        | URLs                                                           | Aktionen                |
|-------------|---------|-----------|-------------------------------|----------------------------------------------------------------|-------------------------|
| apiservice  | Running | 10:39:08  | EasyProject.ApiService.csproj | <a href="https://localhost:7452">https://localhost:7452</a> +1 | [Stop] [Refresh] [More] |
| webfrontend | Running | 10:39:11  | EasyProject.Web.csproj        | <a href="https://localhost:7138">https://localhost:7138</a> +1 | [Stop] [Refresh] [More] |

**.NET Aspire Dashboard (Starterprojekt) (Bild 1)**



The screenshot shows the .NET Aspire Dashboard for a project named 'EasyProject'. The 'Ressourcen' (Resources) tab is active, displaying a table of running services. The table has columns for Name, Status, Startzeit, Quelle, URLs, and Aktionen. Three resources are listed: 'app-redis' (Running, started at 10:46:01, from docker.io/library/redis:7.4), 'apiservice' (Running, started at 10:45:57, from EasyProject.ApiService.csproj), and 'webfrontend' (Running, started at 10:45:59, from EasyProject.Web.csproj). Each resource has a status icon, a stop button, and a refresh button.

| Name        | Status  | Startzeit | Quelle                                             | URLs                      | Aktionen                |
|-------------|---------|-----------|----------------------------------------------------|---------------------------|-------------------------|
| app-redis   | Running | 10:46:01  | docker.io/library/redis:7.4 -c redis-server --r... | tcp://localhost:59085     | [Stop] [Refresh] [More] |
| apiservice  | Running | 10:45:57  | EasyProject.ApiService.csproj                      | https://localhost:7452 +1 | [Stop] [Refresh] [More] |
| webfrontend | Running | 10:45:59  | EasyProject.Web.csproj                             | https://localhost:7138 +1 | [Stop] [Refresh] [More] |

.NET Aspire Dashboard mit Redis-Cache (Bild 2)

### Schritt 3: Referenzieren im Projekt

Nun verknüpfen Sie die Ressource mit dem konsumierenden Projekt:

```
var apiService = builder.AddProject<Projects.EasyProject>
    _ApiService>("apiservice")
    .WithHttpsHealthCheck("/health")
    .WithReference(redis);
```

Aspire sorgt automatisch dafür, dass die Konfiguration des Redis-Containers im Projekt bereitgestellt wird – inklusive Connection-String.

### Schritt 4: Client-Integration im Code

Damit die Applikation zur Laufzeit mit Redis kommunizieren kann, wird eine Client-Bibliothek benötigt, in unserem Fall ist das StackExchange.Redis:

```
dotnet add package Aspire.StackExchange.Redis
```

In der *Program.cs* des konsumierenden Projekts (zum Beispiel *apiservice*) aktivieren Sie dann die Client-Registrierung:

```
builder.AddRedisClient("app-redis");
```

Die Nutzung erfolgt anschließend über Dependency Injection:

```
public class ExampleService(IConnectionMultiplexer
    connectionMux)
{
    // Redis verwenden ...
}
```

Mit wenigen Schritten wurde ein Redis-Cache vollständig eingebunden – inklusive Hosting, Bereitstellung, Konfiguration, Client-Integration und Observability. Alles typisiert in C#, ohne YAML oder Shell-Skripte. Diese Wiederverwendbarkeit des Integrationsmusters macht .NET Aspire besonders effizient.

Neue Ressourcen wie SQL Server, PostgreSQL oder Azure Services lassen sich nach exakt demselben Prinzip integrieren: NuGet-Paket im AppHost einbinden, Ressource im App-

Host deklarieren, Client-NuGet-Paket referenzieren, Client-Integration im Code.

Ein weiterer nützlicher Aspekt zeigt sich im Aspire-Dashboard. Nach dem Start sehen Sie dort nicht nur alle laufenden Projekte, sondern auch die genaue Quelle jeder Ressource. Im Fall des Redis-Cache erkennen Sie, dass .NET Aspire standardmäßig einen Docker-Container startet und automatisch verwendet. Dieser Container ist direkt mit dem Redis-Projekt im AppHost verknüpft und wird beim Start orchestriert (Bild 2).

Sollten Sie jedoch planen, Ihre Anwendung in der Cloud auszuführen, etwa in Azure, dann können Sie mit minimalem Aufwand die Ressourcendeklaration anpassen. Statt *AddRedis(...)* verwenden Sie beispielsweise *AddAzureRedis(...)*, um einen verwalteten Azure Cache for Redis zu verwenden. Die restliche Konfiguration bleibt unverändert – alle Referenzen, Verbindungseinstellungen und Health Checks funktionieren weiterhin wie gewohnt. So wird aus einem lokalen Entwicklungs-Setup nahtlos eine Cloud-native Lösung, ohne dass Code in den Anwendungen angepasst werden muss.

### Vergleich zu Docker Compose

Um die Vorteile von .NET Aspire vollständig zu verstehen, lohnt sich ein direkter Vergleich mit einem Docker-Compose-Setup. Stellen wir uns vor, wir wollen die einfache Beispielanwendung aus dem vorangegangenen Abschnitt – bestehend aus einem Frontend-Projekt, einem Backend-Projekt und einem Redis-Cache – mit Docker Compose orchestrieren. Eine entsprechende *docker-compose.yml* könnte aussehen wie in Listing 2 gezeigt.

Auf den ersten Blick ist dieses Setup funktional – alle drei Dienste werden gestartet und miteinander verbunden. Doch schnell zeigt sich die Komplexität: Jeder Dienst benötigt ein eigenes Dockerfile, alle Konfigurationswerte wie Ports oder Umgebungsvariablen müssen manuell gesetzt und synchronisiert werden. Zudem gibt es keine automatische Weitergabe von Connection-Strings, keine Health Checks, keine Traces, kein Debugging innerhalb der IDE – und keinerlei Unterstützung für die Azure-Integration.

In .NET Aspire dagegen definieren wir dasselbe Setup in wenigen Zeilen C#-Code, ohne Dockerfiles anpassen oder Container-Logik verstehen zu müssen. ►

Docker Compose bleibt ein bewährtes Werkzeug für die Container-Orchestrierung. Doch sobald es um komplexere .NET-Anwendungen, cloudnahe Entwicklung oder typisierte Service-Integration geht, ist .NET Aspire klar im Vorteil. Es eliminiert Medienbrüche, vereinfacht die Wartung und bringt Struktur in verteilte .NET-Systeme.

## Weitere Vorteile von .NET Aspire

Neben den bereits gezeigten Funktionen wie Service-Orchestrierung, Observability, typisierter Konfiguration und Azure-Integration bietet .NET Aspire noch weitere, oft unterschätzte Möglichkeiten.

Ein besonders spannender Aspekt ist die Unterstützung für modulare Architektur und Wiederverwendbarkeit. Aspire-Projekte lassen sich so strukturieren, dass mehrere Teams an unterschiedlichen Modulen arbeiten können – etwa jeweils mit eigenem AppHost für Testzwecke, der sich problemlos in eine größere verteilte Landschaft einfügt. Die Ressourcen und Dienste sind vollständig kapselbar.

Ein weiterer Vorteil ist die Möglichkeit, kontextsensitives Debugging und Telemetrie direkt im IDE-Workflow zu integrieren. Dienste lassen sich gezielt starten, debuggen oder überwachen – ohne eigene Startskripte, Launch-Profile oder manuelle URL-Weiterleitungen. Auch Szenarien mit Hot Reload, Remote-Debugging oder CI/CD-Vorbereitung profitieren stark von der Integration.

Ein weiterer Vorteil ist die Anbindung an das .NET-Ökosystem. Funktionen wie Polly (Retry-Strategien), Dapr (verteilte Services), Health Checks, OpenTelemetry oder YARP (Reverse Proxy) sind entweder direkt integriert oder lassen sich mit wenigen Zeilen ergänzen. Diese Features erweitern die Architekturmöglichkeiten deutlich über das hinaus, was man mit einfachen Docker-Compose-Files erwarten würde.

.NET Aspire ist ein Werkzeug für Teams mit DevOps-Mindset. Es standardisiert Entwicklungsumgebungen, reduziert Konfigurationsfehler und erlaubt ein einfaches Onboarding neuer Entwickler. Statt stundenlanger Einrichtung – Container starten, Umgebungsvariablen setzen, Hosts konfigurieren – genügt ein `dotnet run`, und die komplette Infrastruktur steht lokal bereit. Das verbessert nicht nur die Produktivität, sondern auch die Teamqualität langfristig.

Kurzum: .NET Aspire bringt nicht nur Ordnung in komplexe Service-Architekturen, sondern schafft einen konsistenten, deklarativen und portablen Ansatz, der sich nahtlos in moderne Softwareentwicklungsprozesse einfügt.

## Azure-Deployment

Nachdem die Anwendung lokal mit .NET Aspire entwickelt und getestet wurde, stellt sich die nächste logische Frage: Wie lässt sich dieses Setup in eine produktionsnahe Umgebung wie Microsoft Azure übertragen? Genau dafür bietet das Aspire-Ökosystem eine einfache und dennoch leistungsfähige Lösung an. Das Stichwort lautet: Azure Developer CLI, kurz *azd*.

Wenn Sie .NET Aspire lokal ausgeführt haben, enthält Ihre Anwendung in der Regel ein zentrales AppHost-Projekt, in dem sämtliche Projekte und Ressourcen beschrieben sind. Um daraus eine laufende Cloud-Anwendung zu machen, ge-

### ● Listing 2: docker-compose.yml (Starterprojekt)

```
services:
  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"

  backend:
    build:
      context: .
      dockerfile: ApiService/Dockerfile
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ConnectionStrings__Redis=redis:6379
    depends_on:
      - redis
    ports:
      - "5001:8080"

  frontend:
    build:
      context: .
      dockerfile: WebFrontend/Dockerfile
    environment:
      - BACKEND_URL=http://backend:8080
    depends_on:
      - backend
    ports:
      - "5002:8080"
```

nügt der Befehl *azd up*. Dieser Schritt analysiert Ihr Aspire-Projekt, erstellt alle benötigten Azure-Ressourcen, verpackt Ihre Projekte in Container, lädt sie in eine Container-Registry und startet sie dann in einer vollständig funktionierenden Umgebung in Azure. Dabei entsteht keine manuelle Arbeit mit YAML, JSON oder ARM-Templates. Alles geschieht automatisch und nachvollziehbar.

Ein konkretes Beispiel: Das Standard-Starterprojekt von .NET Aspire enthält typischerweise zwei .NET-Projekte – ein Web-Frontend und einen API-Dienst – sowie optionale Ressourcen wie einen Redis-Cache. Wird *azd up* aufgerufen, analysiert .NET Aspire zunächst die Projektstruktur und erzeugt eine sogenannte *azure.yaml*-Datei, sofern diese noch nicht vorhanden ist. Anschließend werden die Quellprojekte über das .NET SDK gebaut, als Container verpackt und in eine Azure-Container-Registry hochgeladen. Von dort aus werden die Images dann als Azure-Container-Apps bereitgestellt und ausgeführt. Zusätzlich erstellt *azd* Dienste für Application Insights zur Telemetrie sowie zum Logging und richtet automatisch Verbindungsinformationen sowie Konfigurationsdaten ein, die die Anwendungen zur Laufzeit benötigen.

Nun stellt sich die Frage: Was passiert mit Ressourcen wie Redis, die lokal über Aspire als Container gestartet wurden?

Hier zeigt sich die Flexibilität von Aspire. Wenn im AppHost beispielsweise folgender Code steht:

```
var cache = builder.AddRedis("cache");
```

dann wird beim lokalen Start ein Docker-Container für Redis erzeugt. Beim Deployment über *azd* wird dieser Container ebenfalls erstellt und in Azure als Container-App betrieben. Die Verbindung zum Redis-Container wird automatisch in die Umgebungsvariablen aller abhängigen Projekte eingefügt. Der Code in Ihrer Anwendung bleibt davon unberührt.

Wenn Sie allerdings einen nativen Azure-Dienst verwenden möchten, ist der Umstieg denkbar einfach. Sie ändern lediglich eine Zeile im AppHost:

```
var cache = builder.AddAzureRedis("cache");
```

.NET Aspire erkennt anhand dieser Deklaration, dass es sich um eine Plattformressource handelt, die in Azure direkt verfügbar ist. Beim nächsten Ausführen von *azd up* wird dann kein Container erzeugt, sondern ein Azure Cache for Redis automatisch in einer Subscription bereitgestellt. Die Verbindungsdaten werden wie zuvor allen referenzierenden Projekten zur Verfügung gestellt. Das bedeutet: Die Umstellung auf produktive Infrastruktur erfordert keine einzige Codeänderung in Frontend oder Backend. Auch Logging, Tracing und Health Checks bleiben vollständig erhalten. Ein ähnliches Verhalten gilt für andere Ressourcen wie SQL-Datenbanken, Service Bus, Azure Storage oder PostgreSQL. Lokale Container lassen sich später mit wenigen Anpassungen durch echte Azure-Dienste ersetzen. Diese Austauschbarkeit ist ein Kernvorteil von .NET Aspire und macht es besonders attraktiv für Unternehmen, die schrittweise von lokalen Entwicklungsumgebungen zu Cloud-Infrastrukturen übergehen möchten.

Ein weiterer Punkt, den es beim Deployment zu beachten gilt, betrifft die verwendeten Dateisysteme. Wenn in Ihrem AppHost Ressourcen mit Methoden wie *WithDataVolume()* oder *WithDataBindMount()* versehen sind, etwa zur Persistierung von Redis-Daten, dann funktioniert das nur in lokalen Docker-Umgebungen. In Azure müssen solche Speicheranforderungen durch Dienste wie Azure Files, Azure Blob Storage oder Azure SQL umgesetzt werden. .NET Aspire bietet dabei die notwendige Trennung zwischen Laufzeitumgebung und Speicherinfrastruktur, ohne dass Ihre Anwendung von diesen Details abhängig wird.

Auch die Observability ändert sich nicht. Application Insights wird beim Deployment automatisch aktiviert, sodass Sie weiterhin Logs, Metriken und Traces einsehen können – nur eben in der Cloud statt lokal im Browser. Wenn Sie das .NET Aspire Dashboard gewohnt sind, werden Sie sich auch in der Azure-Variante schnell zurechtfinden.

In Kombination mit dem Azure Developer CLI bietet .NET Aspire also eine durchgängige Lösung für das Deployment. Ohne Infrastrukturwissen, ohne separate Skripte und ohne Spezialformate wird Ihre Anwendung bereitgestellt. Nötig sind nur eine Azure Subscription und ein einmaliges Login über *azd login*. Danach ist Ihre Anwendung in wenigen Mi-

nuten erreichbar – genau so, wie sie lokal funktioniert hat. Damit zeigt .NET Aspire, dass moderne Cloud-native Entwicklung nicht kompliziert sein muss. Wer Anwendungen deklarativ und typisiert entwickelt, profitiert gleich mehrfach: schnellere Iterationen, geringere Fehleranfälligkeit und ein nahtloser Weg von der lokalen Entwicklungsumgebung bis zur skalierbaren Bereitstellung in der Cloud.

## Fazit

Dieser Artikel hat einen Einstieg in das .NET-Aspire-Ökosystem geboten. Ziel war es, aufzuzeigen, wie sich verteilte Anwendungen vollständig in C# orchestrieren lassen – ohne dass externe YAML-Dateien, Docker-Compose-Skripte oder Azure-ARM-Templates notwendig sind.

Sie haben gesehen, wie einfach es ist, ein Projekt mit Frontend, Backend und Redis-Cache aufzubauen, lokal zu starten und später mit wenigen Änderungen in Azure zu betreiben. Die Vorteile sind deutlich: automatische Service Discovery, integrierte Observability, flexible Ressourcenanbindung und vor allem ein einheitliches Entwicklungsmodell – vom ersten *dotnet run* bis zum produktionsreifen Deployment.

Besonders hervorzuheben ist dabei, dass .NET Aspire nicht nur eine fixe Lösung vorgibt, sondern ein flexibles Modell anbietet, das sich an individuelle Anforderungen anpassen lässt. Wer mit Azure arbeitet, kann beispielsweise bei der Definition von Cloud-Ressourcen wie Azure Cache for Redis, Azure SQL oder Service Bus auch konkrete Einstellungen wie die verwendete SKU, Größenklassen oder Netzwerkoptionen angeben. .NET Aspire erlaubt es, diese Konfigurationen programmgesteuert vorzunehmen, sodass sich Infrastrukturpräferenzen in einer gewohnten Sprache ausdrücken lassen. Die Möglichkeiten reichen von einfachen Parametern bis hin zu erweiterten Sicherheits- und Netzwerkeinstellungen.

Wenn Sie bisher mit Docker Compose gearbeitet haben oder in Ihrer täglichen Arbeit YAML und Shell-Skripte koordinieren mussten, lohnt sich der Blick auf Aspire ganz besonders. Denn mit Aspire entsteht nicht nur eine technische Vereinfachung, sondern ein struktureller Fortschritt in der Art und Weise, wie moderne Anwendungen geplant, aufgebaut und betrieben werden. ■

- [1] Quickstart: Build your first .NET Aspire solution,  
<https://learn.microsoft.com/en-us/dotnet/aspire/get-started/build-your-first-aspire-app?pivots=dotnet-cli>  
 [2] .NET Aspire integrations overview,  
<https://learn.microsoft.com/en-us/dotnet/aspire/fundamentals/integrations-overview>



**Florian Lenz**

ist Microsoft MVP und Experte für Cloud-native und Serverless-Lösungen in Azure. Er unterstützt Unternehmen dabei, ihre Cloud-Landschaften zu optimieren und Architekturen aufzubauen, die nachhaltigen Erfolg garantieren.

[florian.lenz@florian-lenz.net](mailto:florian.lenz@florian-lenz.net)



## INFRASTRUCTURE AS CODE FÜR VERTEILTE .NET-SYSTEME

# Nieder mit dem ClickOps!

Wer in die Cloud will, muss dort seine Infrastruktur anlegen und pflegen. Das sollte man nicht von Hand erledigen, sondern per Infrastructure as Code.

Wer seine Software bei einem öffentlichen Cloud-Anbieter betreiben möchte, muss dafür in der Regel eine gewisse Infrastruktur anlegen. Das kann auf der Applikationsebene zum Beispiel virtuelle Maschinen, Kubernetes-Cluster, Webapplikationen oder Datenbanken umfassen. Auf der Ebene der Infrastruktur gilt es – ohne Anspruch auf Vollständigkeit – Firewalls ebenso zu konfigurieren wie Netzwerke, DNS-Zonen und Gateways. Alle gängigen Cloud-Anbieter bieten hierzu mehr oder weniger komfortable Webportale an, auf denen man sich das alles zusammenklicken kann – diesen Ansatz nennen wir mal liebevoll ClickOps oder flapsig IoC – Infrastructure on Click.

Das läuft am Anfang auch meistens ganz gut, man kommt schnell vorwärts und sieht auch schnell erste Ergebnisse. Dann kommt der Tag, an dem man die eigene Applikation den auftraggebenden Kunden zugänglich machen will. Also legt man alles noch mal in einer Produktivumgebung an – hat man ja alles schon mal gemacht, also kein Problem. So lange, bis es doch zu einem Problem wird, weil man den einen kleinen Extra-Klick vergessen hat, der dazu geführt hat, dass im Dev-System endlich alles funktioniert hat. Vielleicht hat man diesen Klick auch gar nicht selbst vorgenommen, sondern das war ein Kollege, was die Sache nicht einfacher macht.

Wenn man das Deployment dann endlich hinbekommen hat, lassen Änderungen an der Infrastruktur nicht lange auf sich warten. Komponenten müssen neu angelegt, verschoben oder (leider seltener) gelöscht werden. Regeln müssen angepasst werden, sei es für das Routing, die Firewall oder für das Alerting. Jedes Mal muss man lernen, wie das auf der entsprechenden Plattform funktioniert, jedes Mal muss man das Ganze dann für alle Umgebungen durchziehen, jedes Mal wird man dabei das eine oder andere vergessen, was dann jedes Mal zu Ärger führt – bei einem selbst oder im schlimmeren Fall bei den Kunden.

Es liegt also im Grunde auf der Hand, dass ClickOps keine gute Idee ist – aber was dann? Die Antwort lautet IaC – Infrastructure as Code. Kleine Änderung im Akronym mit großer Wirkung. Statt mit der Maus arbeitet man dann mit der Tastatur, statt zu klicken tippt man Befehle oder Deklarationen ein. Hört sich für jemanden, der sonst gerne Code schreibt, doch gar nicht so übel an, oder? „Uff, muss ich dann auch noch ‚Fancy-IoC-Skriptsprache-XYZ‘ lernen?“, mag der eine oder andere im Geiste stöhnen. Ja, muss man. Erstens muss man das ClickOpsen ja auch lernen, man schlägt sich also mit den Eigenheiten der Webportale der Cloud-Anbieter herum. Und das macht beileibe nicht immer Spaß. Zweitens

kann man eine solche formale Definition der gewünschten Infrastruktur leicht oder zumindest leichter wiederholen – auf anderen Umgebungen oder einfach so, weil man irgendwas vermurkst hat. Nebenbei wird die Infrastruktur damit auch ein Teil des Codes und fügt sich daher in die Versionskontrolle ein, wo sie sogar per Pull Request abgenommen werden kann, wenn man das gerne möchte. So verpasst man auch die kleinen Tricks und Kniffe der Kollegen nicht. Wer noch ein bisschen weiter in eine utopische Zukunft blickt, sieht am Horizont sogar schon automatisierte Deployment-Prozesse für die Infrastruktur, ganz so wie für die Software, die auf derselben läuft.

Die Vorteile liegen also klar auf der Hand – klarer Fall von „So machen wir es!“. Nur wie? Hierfür gibt es – das kann man als Fluch oder Segen auffassen – wie so oft im Leben von IT-Fachkräften viele verschiedenen Möglichkeiten mit Vor- und Nachteilen, aber keine „Silver Bullet“, die alle Probleme löst. Denn jede technische Lösung beseitigt Probleme und bringt dafür neue mit.

## Welche Möglichkeiten gibt es?

Wir sammeln mal ein paar Ideen in einer Strichliste und sortieren und gruppieren das Ganze im Anschluss daran:

- ARM Templates
- AWS Cloud Formation
- Google Deployment Manager
- az-CLI
- aws-CLI
- gcloud-CLI
- Bicep
- Terraform
- OpenTofu
- Crossplane
- Pulumi

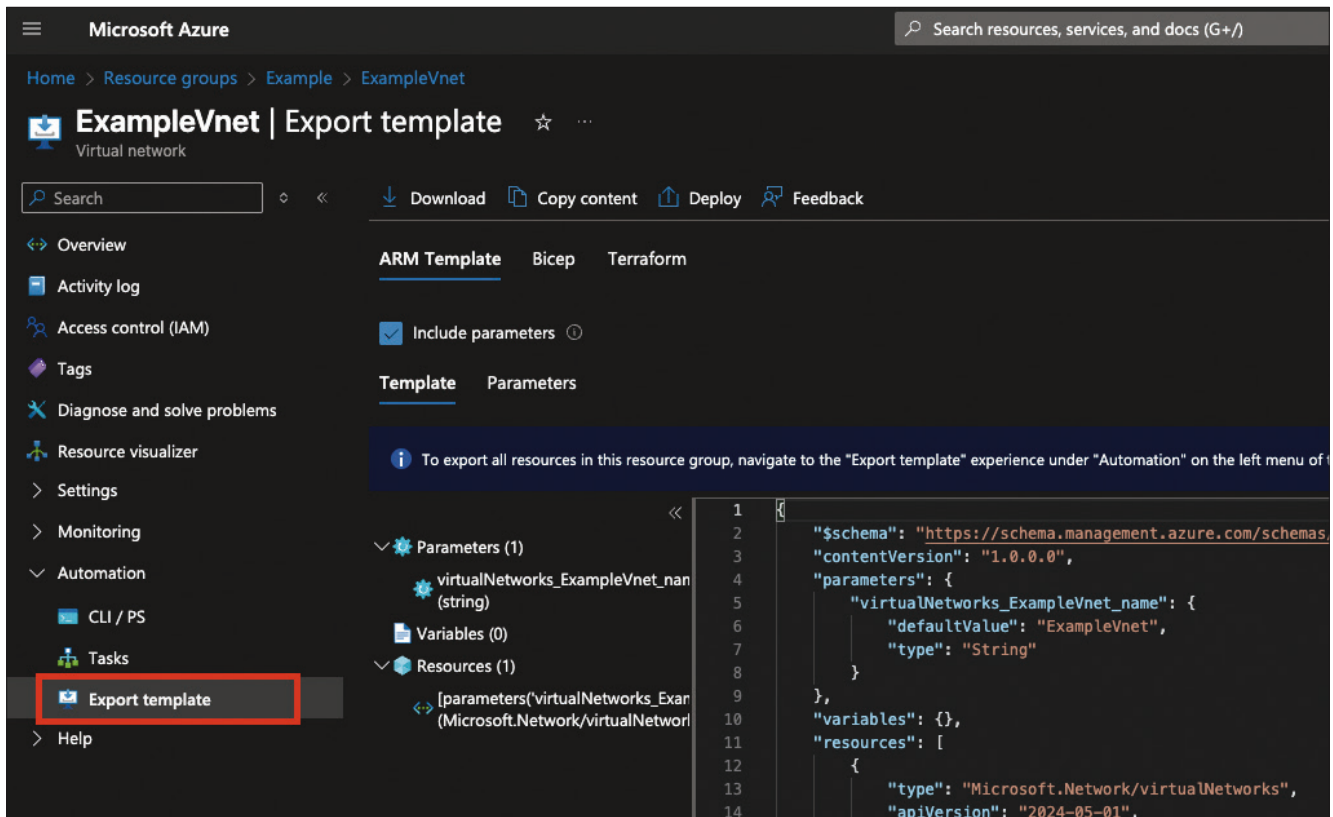
Diese Liste wurden aus mehreren Webseiten kuratiert (darunter [1], [2] und [3]) und so ausgewählt, dass sich die Kandidaten auch primär um die Erstellung von Infrastruktur in Cloud-Umgebungen drehen. Sie ist aber nicht vollständig. Stattdessen findet man grob vier unterschiedliche Ansätze für IaC, und für jede dieser Gruppen lassen sich wohl auch noch weitere Vertreter finden.

- Die erste Gruppe bilden die nativen Deployment-Mechanismen der jeweiligen Cloud-Provider, zum Beispiel ARM-Templates in Azure. Diese setzen auf den nativen Deployment-Mechanismus der Plattform auf. Alle diese Ansätze

sind klar auf eine bestimmte Cloud zugeschnitten, und es gibt keine Erleichterung, um zwischen den Anbietern zu wechseln. Wer von Azure zu AWS wechseln möchte, muss von ARM auf den entsprechenden Mechanismus in AWS umlernen. Des Weiteren können auch Skripte oder Resource Templates schnell unübersichtlich und obskur werden.

Einsatz, mit dem Änderungen geplant und ausgeführt werden können.

- Die vierte Gruppe, hier vertreten durch Crossplane und Pulumi, bedient sich bereits vorhandener Sprachen/DSLs und erweitert diese um die Bereitstellung von Infrastruktur. Crossplane nutzt zum Beispiel Kubernetes Resource Defi-



Ein ARM-Template im Azure Portal (Bild 1)

- Die zweite Gruppe bilden die Shell-Skripte. Hier setzt man auf Mechanismen, die mehr oder weniger schon vorhanden sind – man muss eine Shell auswählen, die auf allen Zielsystemen zur Verfügung steht. Vor allem wenn Entwickler unter Windows arbeiten und auf Linux deployen, muss man dafür sorgen, dass man nicht alle Skripte zweimal schreiben muss. Außerdem braucht man in der Regel noch ein CLI für den Cloud-Provider, also zum Beispiel `az` – das Azure CLI.
- Hier will die dritte Gruppe Abhilfe schaffen: In dieser finden sich IaC-Tools, die eine eigenen Domain-Specific Language definieren, wie zum Beispiel Terraform oder dessen Open-Source-Ableger OpenTofu. Diese DSLs sollen leicht zu erlernen und zu lesen sein. Außerdem verfolgen diese Tools im Gegensatz zu Shell-Skripten einen deklarativen Ansatz: Man sagt nicht, was zu tun ist, sondern wie es werden soll. Im Fall von Terraform/OpenTofu kommt noch dazu, dass man mit derselben DSL Ressourcen für verschiedene Cloud-Plattformen definieren kann. Für Bicep gilt das nicht, da es ausschließlich auf Azure ausgelegt ist. Für die Ausführung der Deklarationen kommt ein eigenes CLI zum

nitionen zum Erzeugen von Infrastruktur, Pulumi diverse Programmiersprachen. Hier kann man von vorhandenem Plattformwissen profitieren, das dann nur noch um die infrastrukturenspezifischen Aspekte erweitert werden muss.

Sehen wir uns nun jeweils einen Vertreter dieser vier Gruppen genauer an, speziell im Hinblick auf Charakteristika sowie Vor- und Nachteile.

### Nativer Deployment-Mechanismus des Anbieters

Jeder Cloud-Provider hat einen nativen Deployment-Mechanismus. In Azure ist das ARM – der Azure Resource Manager. Jedes Tool aus der obigen Liste verwendet diesen Mechanismus unter der Haube, um Ressourcen zu erzeugen oder zu löschen. Warum sollte man das also nicht selbst tun?

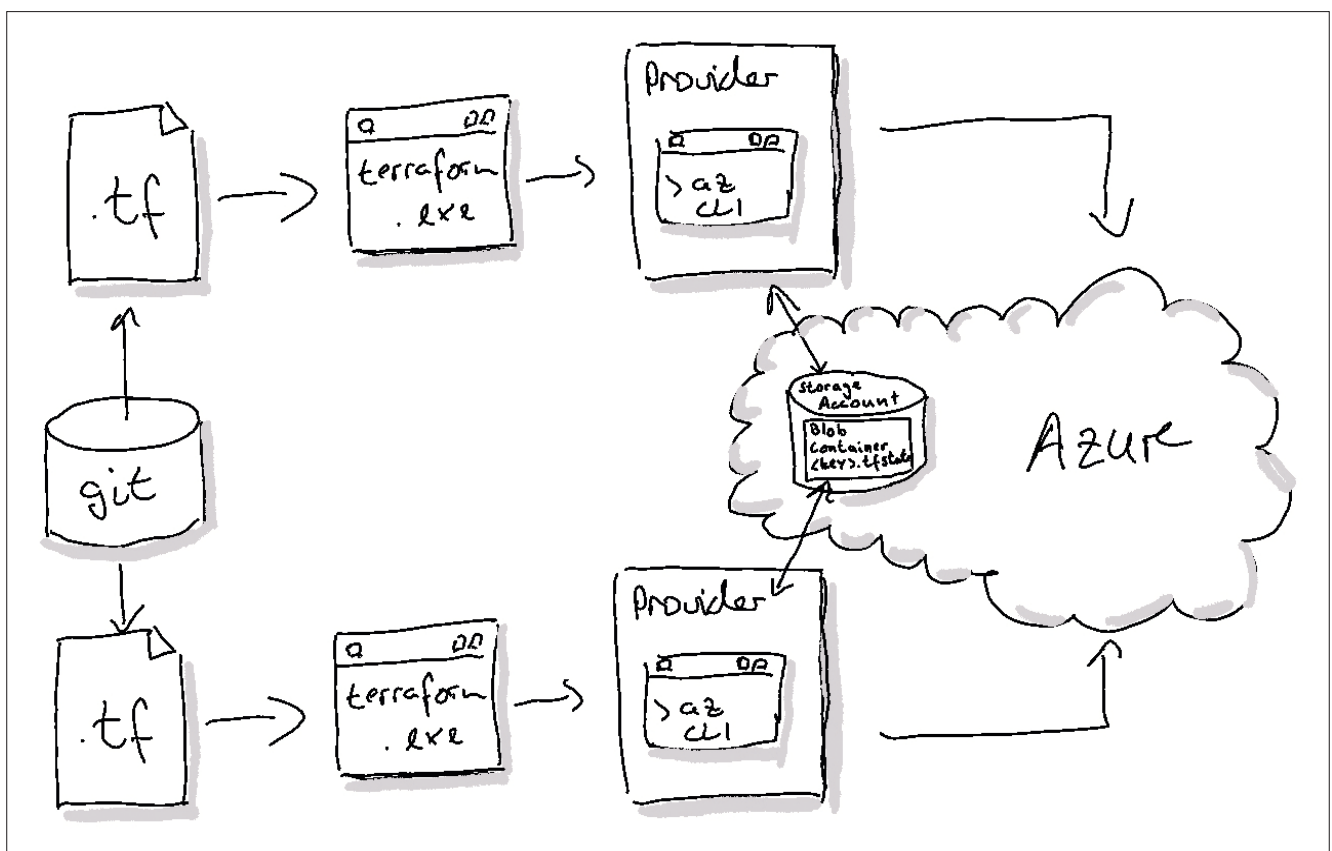
In Azure findet man eine Möglichkeit für die Erzeugung von ARM-Templates in jeder Ressource unter *Automation* (vergleiche Bild 1). Man kann das Template natürlich auch von Hand schreiben. Später kann man das Template dann wieder deployen. Über das Azure Portal ist das etwas versteckt: ►

Man muss zunächst eine neue Ressource anlegen, sucht dann nach *template*, wählt die Ressource *Template Deployment* aus, klickt auf *Build your own template in the editor* und kann dann sein Template hochladen. Über das az-CLI gelingt das einfacher über die Befehlsgruppe *az deployment*.

Der Vorteil dieses Ansatzes besteht darin, dass er immer die aktuellsten Ressourcen erstellen kann. In allen anderen Tools müssen Neuerungen des Providers erst implementiert werden.

Was Azure betrifft, enden die Vorteile dann aber bereits. Was folgt, ist eine beachtliche Liste von Nachteilen: Ein Template von Hand zu schreiben ist so aufwendig, dass es sich schlichtweg nicht lohnt. Der Weg, ein Template aus vorhan-

drauf achten, dass diese Shell sowohl für die Entwickler auf deren lokalen Maschinen zur Verfügung steht als auch in der CI/CD-Umgebung, sonst muss man alle Skripte mehrfach schreiben, was doppelten Aufwand bedeutet. Wer kein Freund der etwas obskuren bash-Syntax ist, kann zum Beispiel auf PowerShell Core zurückgreifen, da diese sowohl unter Windows als auch für macOS und Linux zur Verfügung steht. Des Weiteren benötigt man das CLI-Tool des Cloud-Providers der Wahl – im Fall von Azure ist dies das az-CLI. Dann besteht die Aufgabe darin, alle Ressourcen mithilfe der passenden Shell-Befehle zu erzeugen. Dabei gilt es darauf zu achten, dass die Namen so parametrisierbar sind, dass man die Ressourcen in allen Umgebungen erzeugen kann.



**Die Provider-Architektur von Terraform (Bild 2)**

denen Ressourcen zu erstellen, erspart einem viel von dieser Arbeit, aber die Nachbearbeitung und Parametrisierung ist ebenfalls sehr aufwendig, denn die Syntax der Skripte ist nicht wirklich selbsterklärend, und gleichzeitig werden die Templates schnell sehr groß und unübersichtlich.

Nicht umsonst gibt es schließlich für jeden großen Cloud-Anbieter mindestens eine Abstraktionsschicht über dem nativen Mechanismus – im einfachsten Fall wird dieser in Form eines CLI des Anbieters bereitgestellt.

### Shell-Skripte plus CLI des Cloud-Providers

Die erste Frage, die man in diesem Szenario beantworten muss, ist die, welche Shell man nutzen möchte. Man muss da-

Der Vorteil dieses Ansatzes besteht darin, dass man die hier verwendeten Tools sehr wahrscheinlich ohnehin bereits bis zu einem gewissen Grad beherrscht, weil diese auch in anderen Automatisierungsprozessen zum Einsatz kommen – zum Beispiel in CD-Pipelines. Man muss also weder viele Dinge installieren noch grundlegend neue Dinge lernen.

Der Nachteil besteht hier aber im imperativen Stil des Ansatzes. Erstens kann es je nach CLI vorkommen, dass Befehle nicht ohne Weiteres wiederholbar sind, wenn sie bereits erfolgreich ausgeführt wurden – teilweise deswegen, weil die entsprechenden Ressourcen bereits erzeugt wurden und nicht ein zweites Mal erzeugt werden können. Daher muss man jeden Befehl durch eine vorherige Abfrage absichern.



Mittlerweile ist dies nicht mehr bei allen Objekten der Fall – bei manchen, etwa bei Ressourcengruppen oder Netzwerken, kann wiederholt der gleiche Befehl ausgeführt werden, ohne dass es zu einem Fehler kommt. Trotzdem sollte man das vorher pro Ressource prüfen, damit man keine bösen Überraschungen erlebt. Außerdem muss man für die Zerstörung der Ressourcen ein eigenes Skript schreiben, da die Befehle für das Anlegen und Löschen unterschiedlich sind. Ein weiterer Nachteil der Skripte besteht darin, dass man keine klare Übersicht darüber bekommt, was das Skript nun tatsächlich tun wird – man kann es nur ausführen und dann sehen, was passiert.

Um zum Beispiel eine Ressourcengruppe und darin ein virtuelles Netzwerk mit dem `az`-CLI anzulegen, benötigt man folgende Befehle:

```
az group create -name Example -location westeurope
az network vnet create `
  --resource-group Example `
  --location westeurope `
  --name ExampleVnet `
  --address-prefix 10.1.4.0/22
```

Die Ressourcengruppe muss dafür vorher existieren. Um das Netzwerk wieder zu löschen, benötigt man das `delete`-Verb.

## Eine neue Sprache lernen

Um die Schwächen des Shell-Skript-Ansatzes zu reduzieren, gibt es spezialisierte Tools für das Anlegen von Infrastruktur. Die prominenteren Vertreter dieser Gattung sind Terraform, dessen Open-Source-Pendant OpenTofu oder Bicep.

Diesen Tools ist gemeinsam, dass sie eine eigene Domain-Specific Language (DSL) für die Definition von Infrastruktur definieren, die von einem eigenen CLI interpretiert und ausgeführt wird. Dieser Ansatz hat einige Vorteile. Besonders bemerkenswert ist der Umstand, dass diese Art der Infrastrukturdefinition deklarativ ist – im Gegensatz zu einem imperativen Shell-Skript. Das bedeutet, dass nicht gesagt wird, mit welchem Befehl eine Ressource angelegt werden soll – wie zum Beispiel `az vnet create` –, sondern nur, dass überhaupt eine Ressource benötigt wird.

Betrachtet man Terraform als Beispiel, wird diese Deklaration in einer eigenen Datei mit der Endung `.tf` untergebracht und vom Terraform-CLI verarbeitet. Dieses CLI analysiert die Ressourcendefinitionen und vergleicht diese mit der vorhandenen Infrastruktur, die als State gespeichert ist. Dieser State kann zum Beispiel in einem Azure Blob Storage gespeichert werden, damit mehrere Entwickler parallel an der Infrastruktur arbeiten können. Die Ressourcendefinition bildet den Soll-Zustand, der State den Ist-Zustand des Systems ab. Diese beiden Stände kann das CLI vergleichen und daraus einen Plan ableiten, welche Ressourcen zu erzeugen, zu modifizieren oder zu löschen sind. Dieser Vergleich führt noch nicht zu einer Änderung am System. Man hat also Gelegenheit, zunächst den Plan zu kontrollieren und zu prüfen, ob dieser keine unerwarteten Auswirkungen mitbringt. Ein Beispiel: „Um die Leistung der Datenbank zu erhöhen, muss ich diese

löschen und neu anlegen.“ Wenn der Plan in Ordnung ist, kann dieser durch einen weiteren Befehl in die Tat umgesetzt werden. Dies geschieht über einen Provider, der die Ressourcendefinitionen in Befehle für die Zielplattform übersetzt. Diese Provider kapseln gewissermaßen die CLI-Tools des Cloud-Anbieters. In **Bild 2** ist diese Architektur im Überblick dargestellt.

Terraform unterstützt die drei Cloud-Riesen Azure, AWS und Google Cloud, aber auch weitere kleinere Cloud-Anbieter wie Alibaba oder Oracle. Um eine Ressourcengruppe und darin ein VNet in Azure zu erstellen, braucht man folgende Ressourcendefinitionen, geschrieben in HCL (HashiCorp Configuration Language):

```
resource "azurerm_resource_group" "rg" {
  name      = "Example "
  location  = "West Europe"
}

resource "azurerm_virtual_network" "vnet" {
  name                = "example-vnet"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  address_space       = ["10.0.0.0/16"]
}
```

Das Beispiel stellt keine komplette Definition dar, die sofort ausgeführt werden könnte – zugunsten der Übersichtlichkeit wurde einiges vom Grundaufbau weggelassen. Ressourcen werden in Terraform durch sogenannte *resource*-Blöcke definiert. Die Deklaration enthält die Art des Blocks, zum Beispiel `azurerm_resource_group`, und dessen Namen. Der Body definiert dann die Eigenschaften, die zur Erzeugung der Ressource notwendig sind. Es gibt weitere Typen von Blöcken, unter anderem zur Definition von Variablen oder für Abfragen des Status einer Ressource. Wie man an der `location`-Eigenschaft der VNet-Ressource erkennen kann, ist es möglich, sich bei den Eigenschaften der Ressource auf bereits definierte Ressourcen zu beziehen.

Das Planen, Ausführen und gegebenenfalls Zerstören der Infrastruktur wird durch das Terraform-CLI umgesetzt. Dieses unterstützt entsprechende Verben für die Umsetzung der Definition in existierende Infrastruktur. Leicht vereinfacht dargestellt folgt Terraform folgendem Zyklus von Init–Plan–Apply:

```
terraform init
# Initialisierung, z.B. Herunterladen des Providers
terraform plan -out="plan.out"
# Planen der Änderungen
terraform apply plan.out
# Ausführen der Änderungen
```

Terraform hat von einiger Zeit seine Lizenz von der Mozilla Public License auf eine Business Source License geändert, weshalb im Rahmen bestimmter Nutzungsszenarien nun ►

Lizenzgebühren an den Hersteller HashiCorp gezahlt werden müssen. Das ist in der Entwicklercommunity nicht gut angekommen, sodass ein Open-Source-Fork von Terraform namens OpenTofu ins Leben gerufen wurde [4]. Dieser kann als „Drop in“-Ersatz für Terraform fungieren, OpenTofu versteht also vorhandene Terraform-Skripte.

Bicep ist ein IaC-Tool von Microsoft, das eine eigene DSL definiert. Anders als Terraform, das durch sein Providermodell unterschiedliche Cloud-Anbieter unterstützt, funktioniert Bicep nur für die Azure-Cloud. Dafür ist es hier sehr tief integriert: Es benötigt zum Beispiel kein *State*-File, das man irgendwo unterbringen muss, sondern prüft direkt in der bestehenden Infrastruktur, wie der Ist-Stand des Systems ist. Des Weiteren sind alle neuen Azure-Objekte direkt in Bicep verfügbar, während man in Terraform oft warten muss, bis neue Features im entsprechenden Provider implementiert sind.

Terraform und Co. bringen viele Vorteile mit: Die DSLs sind klar strukturiert und relativ einfach zu verstehen. Bevor Änderungen durchgeführt werden, können diese zunächst geplant und anhand dieser Planung überprüft werden. Es werden alle Arten von Änderungen – Anlegen, Modifizieren und Löschen – ausgeführt, die notwendig sind, um den Ist- in den Soll-Zustand zu überführen. Auch das Zerstören der gesamten Infrastruktur ist auf Basis der Ressourcendefinition möglich, was sehr praktisch ist, wenn man einen Test oder einen Proof of Concept erstellt hat, den man später wieder loswerden möchte.

Leider gibt es wie immer in der Softwareentwicklung nicht nur Licht, sondern auch Schatten. Auf der Contra-Seite der Medaille steht zum einen eine Lernkurve, denn eine neue DSL und der Umgang mit einem weiteren CLI erlernen sich nicht von allein. Selbst wer zum Beispiel Terraform einsetzt, kommt wahrscheinlich dadurch nicht ohne das az-CLI aus. Man hat also ein zusätzliches Tool im Stack. Außerdem muss man sich für ein Produkt entscheiden. Solche Entscheidungen können weitreichende Konsequenzen haben, wie zum Beispiel die Änderung der Lizenz von Terraform zeigt.

Um die Einstiegshürde etwas zu reduzieren, bietet das Azure Portal seit einiger Zeit die Möglichkeit, aus bestehenden Ressourcen Terraform oder Bicep-Skripte zu erstellen. Wer noch einmal mit Argusaugen **Bild 1** studiert, entdeckt unterhalb der Symbolleiste die Reiter *ARM Template*, *Bicep* und *Terraform*. Hier kann man sich initiale Definitionen in der jeweiligen DSL erstellen lassen. Ohne tiefere Kenntnisse der gewählten Technologie kommt man aber auch dann nicht aus, denn diese Start-Skripte müssen noch nachbearbeitet werden, um wirklich „production ready“ zu sein.

## Eine Sprache sprechen, die man schon kann

Der Ansatz, eine eigene DSL für die Konfiguration der Infrastruktur zu nutzen, bietet viele Vorteile. Als Nachteil muss sich dieser Ansatz jedoch gefallen lassen, dass er den Nutzern aufbürdet, eine neue Sprache zu erlernen. Hier versprechen Tools wie Pulumi und Crossplane Abhilfe, die anstelle einer eigenen Sprache auf solche setzen, die die Nutzer bereits kennen, nämlich zum Beispiel auf Kubernetes Resource Definitions oder auf gängige Programmiersprachen.

Für Crossplane spricht das einheitliche Interface über Kubernetes Resource Definitions. Wer also Kubernetes einsetzt, für den kann sich ein Blick auf Crossplane lohnen. Allerdings bringt es einen höheren Installationsaufwand mit sich, da man erst einmal einen Cluster braucht – und wie hat man den erstellt? Obwohl es durchaus reizvoll ist, bleibt es eher eine Nischenlösung.

Pulumi kommt nahbarer daher. Es bietet auf seiner Webseite Beispiele in TypeScript, Python, Go, C# und Java an. Auch YAML ist möglich. Ähnlich wie Terraform unterstützt Pulumi Azure, AWS und Google Cloud und reicht damit also nicht ganz an die Vielfalt von Terraform heran. Außerdem erfordert es neben der Sprach-Runtime, in der man die eigene Konfiguration definieren möchte, auch noch ein eigenes CLI, das für alle gängigen Betriebssysteme zur Verfügung steht. Unter Windows kann man Chocolatey oder Winget für die Installation heranziehen. Möchte man – wie im folgenden Beispiel – C# als Sprache verwenden, benötigt man das .NET SDK. Zu guter Letzt muss man noch beim eigenen Cloud-Provider angemeldet sein – ein Service-Account tut es hier natürlich auch.

Bei der ersten Nutzung verlangt Pulumi einen Login in die eigene Webpräsenz. Hier kann man später optional und dann auch kostenpflichtig den State hinterlegen. Alternativ kann man diesen auch in einem Blob Store parken. Das ist im Grunde identisch zu den Optionen, die auch Terraform bietet, aber durch den erzwungenen Login und die damit verbundene Notwendigkeit, einen Account bei Pulumi anzulegen, etwas aufdringlicher.

Das Pulumi-CLI leitet gekonnt durch die Erstellung eines neuen Projekts, das neben der erwarteten *cspjproj*-Datei noch eine proprietäre YAML-Projektdatei enthält. Außerdem gibt es YAML-Dateien mit Werten für spezifische Umgebungen, die als Stacks bezeichnet werden. Hauptbestandteil der Infrastrukturdefinition ist die *Program.cs*, in der die gewünschten Ressourcen beschrieben sind. Für eine Ressourcengruppe und ein VNet sieht das wie folgt aus:

```
var resourceGroup = new ResourceGroup("example");

var vnet = new VirtualNetwork(
    "example-vnet",
    new VirtualNetworkArgs
    {
        ResourceGroupName = resourceGroup.Name,
        AddressSpace = new AddressSpaceArgs {
            AddressPrefixes = new[] { "10.0.0.0/16" } },
        Location = resourceGroup.Location,
    }
);
```

Der Code stellt sich erfreulich selbsterklärend dar. Was nicht direkt ersichtlich wird, ist, dass der erste Parameter in den Ressourcen der logische Name ist. Dieser muss nicht mit dem physischen Namen der Ressource in Azure übereinstimmen. Der physische Name kann manuell über das jeweilige Argumentobjekt gesetzt werden. Um diese Ressourcen in Azure

anzulegen, startet man das Projekt nicht in der IDE, sondern greift auf das Pulumi-CLI zurück, indem man im Projektverzeichnis folgenden Befehl gibt:

```
pulumi up
```

Dieser wird von Pulumi mit einem Plan darüber beantwortet, was das Tool zu tun gedenkt. Dem kann man zustimmen oder auch nicht. Im Fall der Zustimmung verrichtet das CLI über den gewählten Cloud-Provider dann sein Werk. Änderungen an der Konfiguration lassen sich mit dem gleichen Befehl planen und ausführen. Wer seiner Ressourcen überdrüssig ist, kann diese mit

```
pulumi destroy
```

wieder abreißen.

Pulumi ähnelt von der Nutzung stark Terraform, obwohl das eigene CLI für die Alltagsaufgaben mit etwas weniger Befehlen auskommt, die auch aufgeräumter wirken. Die Nutzung der eigenen Lieblingssprache ist angenehmer, man bleibt innerhalb des eigenen bekannten Toolings und kann die Stärken der Sprache nutzen. So werden Fehler in der Konfiguration direkt in der IDE erkannt, und man kann Intellisense ebenso nutzen wie alle üblichen Kontrollstrukturen der Sprache wie Schleifen oder Bedingungen.

Auf der anderen Seite stimmt es leider nicht, dass man nichts Neues lernen muss – denn alle Objekte, die Pulumi für das Anlegen der Ressourcen verwendet, muss man kennenlernen. Das hält sich vom Aufwand her die Waage mit Terraform, das allerdings an Boden verliert, wenn es um Bedingungen und Schleifen geht, die in HCL durchaus gewöhnungsbedürftig sind. Zur Reduzierung der Einstiegshürde ermöglicht Pulumi die Konvertierung eines ARM-Templates in den Code einer unterstützten Sprache.

Wenn die Pflege der Infrastruktur von den gleichen Personen vorgenommen wird, die auch die Infrastruktur betreuen, ist Pulumi sicher einen Blick wert. Bei einem eigenen Infrastrukturteam lassen die Vorteile der freien Sprachwahl nach, und das Argument der weiteren Verbreitung von Terraform/OpenTofu oder Bicep gewinnt die Überhand.

## Einer, zwei oder drei? Wie viele Cloud-Provider dürfen es denn sein?

Von den hier vorgestellten Tools versprechen Terraform/OpenTofu und Pulumi, cloudbübergreifend zu sein. Das sind sie auch tatsächlich, denn sie unterstützen mehrere Cloud-Anbieter. Daraus darf allerdings nicht geschlossen werden, dass man dadurch automatisch Multi-Cloud-fähig ist. Denn die konkrete Konfiguration, die man erstellt, ist immer für einen spezifischen Cloud-Anbieter geschrieben. Die Struktur und die Namen der Ressourcen hängen stark vom Provider ab und lassen sich nicht ohne manuelle Änderungen auf einer anderen Cloud-Plattform deployen. So kann eine HCL-Konfiguration für den Azure Provider nicht auf der Google Cloud ausgeführt werden. Um einen Vendor-Lock-in zu vermeiden, müssen die Konfigurationen manuell übertragen werden.

Natürlich kann man für die Übertragung sein Glück auch mit ChatGPT und Co. versuchen; da die Konfigurationen aber durchaus komplex werden können, sollte man das Ergebnis auf jeden Fall mit gesundem Sachverstand überprüfen.

Auf der anderen Seite fällt es sicherlich leichter, eine in HCL geschriebene Konfiguration von Azure auf AWS zu übertragen, als dies anhand von nativen Templates oder Shell-Skripten zu tun – insofern sichert die Verwendung dieser Tools eine gute Ausgangslage für den Wechsel in eine andere Cloud. Zum Selbstläufer wird dies aber leider nicht.

## Fazit

Dieser Artikel hat eine Vielzahl an IaC-Tools in der Übersicht vorgestellt. Zu Terraform und Bicep wurde in der dotnetpro bereits an anderer Stelle in der Tiefe berichtet [5][6].

Welches Tool man einsetzt, hängt von vielen Faktoren ab. Wichtig ist die Frage, welche Cloud-Anbieter man nutzen möchte. Wer sich auf Azure festlegt, für den ist Bicep eine gute Wahl. Wer sich mehr Optionen offenhalten möchte, setzt besser auf Terraform oder OpenTofu – selbst wenn das Bedienen mehrerer Clouds auch hier kein Zuckerschlecken ist.

Die Größe und die Wichtigkeit des Projekts spielen ebenfalls eine Rolle. Für kleine Projekte kann man mit Shell-Skripten starten und später auf eine der „größeren“ Varianten umsteigen. Des Weiteren stellt sich die Frage, ob die Infrastruktur von einem separaten Team erstellt werden soll oder ob dies von den Entwicklern miterledigt wird. Im ersten Fall kann man auf die Platzhirsche wie Terraform und Co setzen, im zweiten Fall kann man die Einstiegshürden niedrig halten, indem man eine Lösung wie Pulumi einsetzt. Diese ist zwar gut dokumentiert, aber etwas weniger weit verbreitet.

Im Endeffekt ist die Frage eigentlich weniger, für welchen Ansatz man sich entscheidet. Wirklich wichtig ist nur: Die Erstellung der Infrastruktur muss automatisiert erfolgen. Egal wie. Erspart euch schlaflose Nächte und viel Ärger. Nieder mit dem ClickOps! ■

[1] Env0, <https://www.env0.com/blog/best-infrastructure-as-code-tools-and-terraform-alternatives#pulumi>

[2] Spacelift, <https://spacelift.io/blog/terraform-alternatives>

[3] Bluelight, <https://bluelight.co/blog/best-infrastructure-as-code-tools>

[4] OpenTofu, <https://opentofu.org/>

[5] Tobias Richling, *Gedächtnis für das Team*, dotnetpro 5/2022, Seite 48 ff.

[6] Tobias Richling, *Muskelpaket für Azure*, dotnetpro 7/2022, Seite 62 ff.



### Tobias Richling

arbeitet als Softwareentwickler und betreut die B2B-Plattformen der apetito AG. Er sucht dabei stets nach Möglichkeiten zur Verbesserung von Funktionalität und Performance. Sie erreichen ihn unter [tobias.richling@apetito.de](mailto:tobias.richling@apetito.de) oder [@trichling](https://twitter.com/trichling).



## DIE .NET DEVELOPER CONFERENCE 2025

# Am Puls der Zeit

Jedes Jahr pilgern .NET-Entwickler:innen nach Köln, um sich vier Tage lang zusammen mit Expert:innen über Neuerungen in Sachen .NET, Cloud, AI und mehr auszutauschen.

Wenn sich vom 24. bis 27. November 2025 die deutschsprachige .NET-Community im Pullman Cologne versammelt, weiß man: Es ist wieder DDC-Zeit. Die .NET Developer Conference hat sich seit ihrer Erstausgabe 2011 als einer der wichtigsten Treffpunkte professioneller .NET-Entwickler:innen etabliert – und das aus gutem Grund: Die DDC ist kein anonymer Mega-Kongress, sondern ein Get-together, das vom Austausch auf Augenhöhe lebt.

Der Fokus ist klar: Konkrete Tipps und aktuelle, praxisnahe Inhalte rund um das .NET-Ökosystem prägen die Konferenz. Ob nun .NET/C# oder Blazor, MAUI oder KI, Azure, DevOps oder Container, Softwarearchitektur oder Projektmethoden – wer im .NET-Umfeld beruflich unterwegs ist, findet auf der DDC eine Reihe von Lösungen für die Herausforderungen im Projekt, neue Ansätze für Lösungen beim Softwarebau – und eine Menge Gleichgesinnte.

### Vielfalt: die Themen

Der inhaltliche Rahmen ist auch 2025 wieder breit gefächert und gliedert sich in acht eigenständige Themen-Tracks. Mit dabei sind „Software Architecture, Testing and Quality“, wo es um Clean Code, Refactoring und Microservices geht. „Frontend Development“ beleuchtet alles rund um Blazor, WPF, MAUI und moderne Konzepte für zeitgemäße User Interfaces. Ebenfalls vertreten ist „GenAI & Agentic“, das sich mit Technologien wie Semantic Kernel/Flow, Azure AI oder Microsoft Copilot Studio beschäftigt. Weitere Tracks widmen sich den Bereichen Cloud & DevOps, Data & Analytics, Projektmethoden sowie dem effektiven Einsatz der Microsoft Developer Tools. Unterm Strich ergibt dies ein Themenspektrum, das nicht nur aktuelle technologische Entwicklungen abdeckt, sondern diese auch systematisch vertieft.

### Eingängig: die Struktur

Der zeitliche Ablauf der Konferenz folgt dem bewährten Aufbau der DDC-Vorgängerausgaben:

- Los geht es am Montag, dem 24. November, mit den Workshops, intensiven Deep-Dive-Formaten, die über acht Stunden hinweg die jeweiligen Themen in kleinen Gruppen fundiert und praxisnah vermitteln.

- Am Dienstag und Mittwoch (25./26. November) folgt das klassische Konferenzprogramm mit 50 Sessions à 60 Minuten, eingeleitet von zwei Keynotes und flankiert von ausreichend Pausen, Diskussionen und einem abendlichen Rahmenprogramm. Klare Intention ist hier, das intensive Networking der Community zu ermöglichen.
- Donnerstag, der 27. November schließlich ist den DevSessions gewidmet – vierstündige Formate mit Workshop-Charakter, bei denen die Referent:innen ihre Themen den Teilnehmenden kompakt erläutern, unterbrochen von nur einer kurzen Pause.



Teilnehmer der DDC während eines Vortrags

### Sessions, Workshops und DevSessions: die Tiefe

Ein Blick ins Programm der DDC zeigt die große thematische Bandbreite, die vom Advisory Board sorgsam kuratiert wurde. So widmet sich etwa einer der ganztägigen Workshops der Frage, wie sich Clean Architecture und Microservices sauber und skalierbar in .NET-Projekten umsetzen lassen. Ein weiterer geht der Frage nach, wie Blazor und MAUI in modernen Cross-Platform-Projekten harmonisieren. Besonders spannend wird „Azure AI und Copilot Studio“, das zeigt, wie sich Agentic-AI-Lösungen konkret in die eigene Codebasis integrieren lassen. Und wer es technisch besonders fundiert mag, findet im Workshop zur Testautomatisierung mit NUnit und CI/CD wertvolle Impulse für den Build-Alltag.

Auch die Vorträge zeigen, dass es nicht um Buzzwords geht, sondern um Lösungen. Im Track zu Softwarearchitektur etwa wird erklärt, wie Architekturentscheidungen die langfristige Wartbarkeit beeinflussen – anhand realer Projekte. Im Frontend-Track steht das Zusammenspiel von UI-Patterns mit Blazor und WPF im Zentrum, inklusive Erfahrungsberichten aus produktiven Anwendungen. In „GenAI & Agentic“ zeigt einer der Talks, wie sich Semantic Kernel und Copilot Studio effizient in .NET-Projekten nutzen lassen. In „Cloud Development“ geht es unter anderem um CI/CD-Pipelines mit Azure DevOps und Infrastructure as Code, während in „Data & Analytics“ konkrete Methoden zur Visualisierung von Echtzeitdaten gezeigt werden. Der Track zu „Approaches“ thematisiert, wie sich agile Methoden mit technischer Exzellenz vereinen lassen, etwa in hybriden Teams. Wer nach Tools sucht, die den

Alltag erleichtern, erhält fundierte Einblicke in den Einsatz von GitHub Copilot. Und in Sachen Architektur zeigt eine der Sessions, wie sich große Legacy-Systeme Schritt für Schritt in eine moderne, modulare Struktur überführen lassen.

Am Donnerstag, dem letzten Tag der Konferenz, geht es schließlich noch mal ans Eingemachte. In den DevSessions dreht sich alles um praktische Umsetzung und tieferes Verständnis. Beispiel(e) gefällig? Wer etwa lernen will, wie man eine performante Datenpipeline in .NET umsetzt, ist hier ebenso gut aufgehoben wie all diejenigen, die einen auto-



**DDC-Speaker Thomas Claudius Huber** während seines Workshops

nomen KI-Assistenten mit Azure AI realisieren oder mittels Azure und Container ihre Cloud-Performance maximieren möchten. Strategien fürs automatisierte Testen werden ebenso behandelt wie pragmatische Refactoring-Techniken für gewachsene Legacy-Codebasen. Ergänzt wird das Angebot durch DevSessions zu .NET Aspire in Action, Avalonia und Kubernetes-Cluster mit ASP.NET Core und Blazor.

## Experten: die Sprecher:innen

Für die inhaltliche Expertise sorgen mehr als 40 Sprecher:innen und Sprecher, darunter viele bekannte Namen aus der deutschsprachigen .NET-Szene, aber auch einige neue Gesichter, die frischen Wind in die Konferenz bringen. Sie alle einzeln vorzustellen würde den Rahmen dieses Artikels bei Weitem sprengen. Daher folgt hier die Hall of Fame der diesjährigen DDC-Sprecher:innen: Mark Allibone, Florian Bader, Emanuele Bartolesi, Harald Binkle, Gregor Biswanger, Taswar Bhatti, Ruby Jane Cabagnet, Andrea Díez, Dennis Doomen, Dr. Damir Dobric, Kajetan Duszyński, Ina Einemann, Francesco Esposito, Kevin Gosse, Liuba Gonta, Miriam Hunger, Teena Idnani, Yuliya Khadasevich, Florian Lenz, Niels Laute, Codrina Merigo, Sofía Platas, Georg Poweleit, Dr. Holger Schwichtenberg, Connell Sharp, Johan Smarius, Michael Staib, Torsten Strauß, Oliver Sturm, Sander ten Brin-

ke, Thomas Tomow, Ivo Verburgh, Christian Wenz, Florian Winkelbauer.

Sie alle sorgen – zusammen mit den Track Chairs Christian Giesswein, Thomas Claudius Huber, Thorsten Kansy, Michael Kaufmann, Roland König, Marc Müller, Jörg Neumann und David Tielke, die als Referenten ihre Tracks eröffnen – für die hohe fachliche Qualität, die die Teilnehmer:innen der DDC seit Jahren so sehr schätzen.

## Mehrwert: Was habe ich davon?

Verständliche Frage. Meine Antwort? Eine Menge. Denn wer als Entwickler:in in Sachen .NET am Puls der Zeit bleiben möchte, um für die Herausforderungen des Projektalltags gewappnet zu sein, findet an den vier Tagen in Köln eine Reihe von Themen und Impulsen, die sie beziehungsweise ihn dabei unterstützen – sei es in Fragen der KI-Integration, zu Frontend und Backend, rund um Softwarearchitektur und Softwarequalität oder mit Blick auf Cloud-Strategien.

Damit aber nicht genug, denn ebenso wichtig ist das Praxiswissen, das nicht nur die Workshops und DevSessions liefern, sondern auch in den Vorträgen aufblitzt und sich direkt im Projektalltag umsetzen lässt. Diese Mischung ermöglicht den Teilnehmer:innen und Teilnehmern gezielte Weiterbildung, denn mit den unterschiedlichen Ticketoptionen lässt sich das Programm nach individuellen Bedürfnissen zusammenstellen, von einzelnen Tagen bis hin zum All-inclusive-Pass. Ein besonderes Plus: der Community-Aspekt, denn auf der DDC steht der Aus-



**Konferenzteilnehmer** im Austausch

tausch unter Gleichgesinnten im Vordergrund – nicht nur unter Teilnehmer:innen und Teilnehmern, sondern vor allem mit den Sprecher:innen und Sprechern.

Und dann ist da noch ein Punkt, der bisweilen unterschätzt wird: Die DDC versucht Vielfalt bestmöglich zu leben. Durch das gezielte Setzen wichtiger Themen und die sorgsame Abwägung bei der Auswahl der Inhalte und Sprecher:innen sorgt die DDC für ein Konferenzklima, das alle mitnimmt und vier Tage lang eint. Community eben. Und auf diese offene Atmosphäre, so meine Meinung, kommt es bei einer Konferenz wie der DDC am Ende an – fachlich wie menschlich. ■



## Fernando Schneider

begleitet die Fort- und Weiterbildung für Softwareentwickler:innen seit den frühen 1990er-Jahren. Als Chefredakteur der dotnetpro und Head of Content für die DWX ist er für die programmatische Ausrichtung der developer Unit verantwortlich.



# DATA ACCESS

Die Datenzugriffsberatung mit DR. HOLGER SCHWICHTENBERG

## MICROSOFTS DATENTABELLEN-STEUERELEMENT QUICKGRID FÜR BLAZOR

# Schnell mal ein Grid bauen

Microsoft liefert nur wenige Steuerelemente bei Blazor mit, aber mittlerweile immerhin ein DataGrid inklusive RAD-Werkzeugen in Visual Studio.

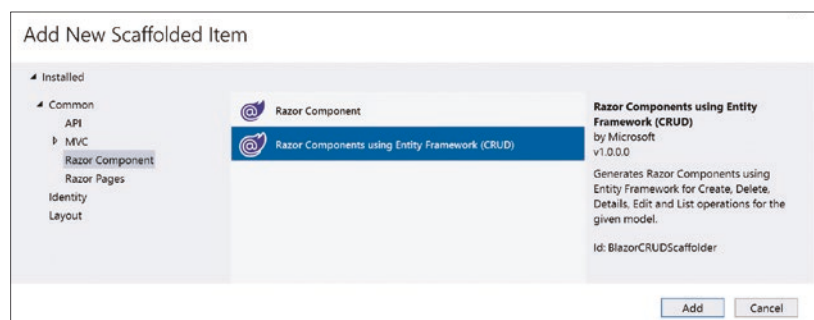
**D**ataGrid-Steuerelemente für Blazor gab es vor Blazor 7.0 nur von kommerziellen Drittanbietern (zum Beispiel Developer Express, Syncfusion, Telerik, Infragistics, Radzen, GrapeCity/Component One) oder aus der Community (zum Beispiel MudBlazor). Radzen ist ein kommerzieller Anbieter, der aber das Grid und andere Steuerelemente kostenfrei verteilt.

### Das QuickGrid von Microsoft

Erstmals seit .NET 7.0 Preview 6 liefert Microsoft für Blazor selbst ein DataGrid. Das neue Steuerelement `<QuickGrid>` steckt im NuGet-Paket `Microsoft.AspNetCore.Components.QuickGrid`, hat aber auch in Blazor 7.0 den Status „Preview“. Erst seit Blazor 8.0 ist das Paket einsatzreif.

Mit dem Namen „QuickGrid“ wollen die Entwickler bei Microsoft betonen, dass man mit dem Steuerelement eine „performance baseline for anyone building Blazor data grid components“ [1] schaffen wolle. Eine Livedemonstration des QuickGrid-Steuerelements in einer Blazor-WebAssembly-Anwendung findet man unter [2].

Das QuickGrid ist auch Stand .NET 9.0 und 10.0 auf die Darstellung von Daten (inklusive Filtern, Sortieren von Zeilen, Paging und Virtualisierung) ausgerichtet. Es gibt von



Einstieg in den Codegenerator (Bild 1)

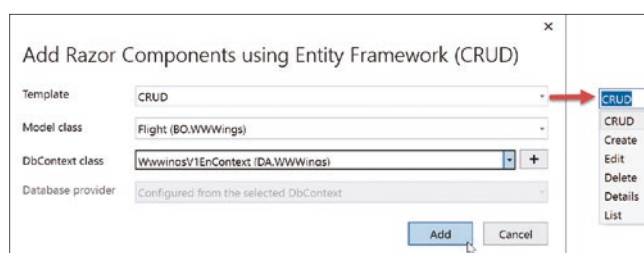
Haus aus keinen Eingabemodus für Daten, was kommerzielle Datengitter-Steuerelemente normalerweise können. Man kann das QuickGrid aber mit etwas Eigenarbeit auch zur Dateneingabe verwenden [3].

Auch andere in Drittanbieter-Steuerelementen übliche Funktionen wie das Umsortieren der Spalten fehlen im QuickGrid.

### QuickGrid erstellen per Codegenerierung in Visual Studio

Microsoft bietet seit dem 14. November 2023 (Visual-Studio-Version 2022 Version 17.9 Preview 1) in Visual Studio erstmals überhaupt eine Codegenerierung für Razor Components auf Basis bestehender Entitätsklassen an.

Die Generierung funktioniert auf Basis der Projektvorlage *Blazor Web App* mit den Interaktivitätsmodi *None* (also Static Server-Side Rendering) und *Server* (also Blazor Server), aber nicht mit *WebAssembly* und *Auto*, denn die Codegenerierung erstellt eine 2-Tier-Architektur mit direktem Zugriff auf ein Datenbankmanagementsystem via Entity Framework Core. Das ist bekanntlich in Blazor WebAssembly aufgrund der Browser-Sandbox nicht möglich (allenfalls eine lokale SQLite-Datenbank im Browserspeicher ist nutzbar, diesen Sonderfall berücksichtigt der Codegenerator aber nicht). Mi-



Schritt 2 des Assistenten (Bild 2)





Microsofts Codegenerator bietet leider keine 3-Tier-Option mit Zugriff auf einen Webservice (zum Beispiel OData oder GraphQL).

Zum Start der Codegenerierung rufen Entwicklerinnen und Entwickler im Solution Explorer in einem Projekt im Kontextmenü **Add | New Scaffolded Item** auf und wählen dort die Vorlage **Razor Components using Entity Framework (CRUD)** (siehe Bild 1).

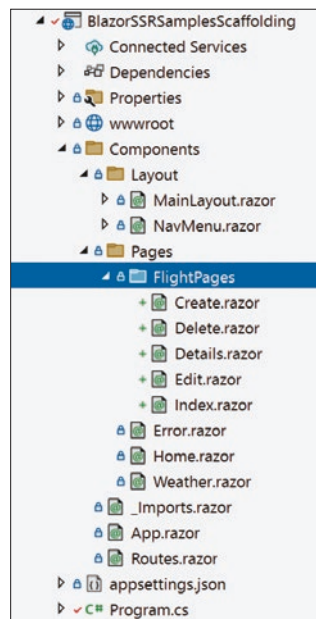
In Blazor-WebAssembly-Projekten beziehungsweise dem Client-Teil von Blazor-Auto-Projekten gibt es den Menüpunkt **New Scaffolded Item** erst gar nicht – er erscheint nur vor dem ersten Kompilieren, und ein Versuch, den Assistenten dann auszuführen, resultiert in einem Fehler.

Im nächsten Dialog (siehe Bild 2) wählt man eine der Vorlagen aus, wobei **CRUD** bedeutet, dass alle fünf Einzelvorlagen (**Create**, **Edit**, **Delete**, **Details** und **List**) gemeinsam zum Einsatz kommen. Es entstehen dann fünf Razor Components. Eine Möglichkeit, davon mehrere, aber nicht alle Teile zu wählen (zum Beispiel nur **List** und **Details**), gibt es nicht. Dazu muss man den Generator entweder mehrfach aufrufen oder **CRUD** wählen und dann löschen, was man nicht benötigt.

In diesem Schritt wählt man auch eine bestehende Modellklasse (alias Datenklasse, Entitätsklasse) und eine bestehende Entity-Framework-Core-Kontextklasse. Die Entitätsklasse und die Kontextklasse müssen nicht in dem Webprojekt liegen. Sie können auch in referenzierten Assemblies existieren, was empfehlenswert ist.

In diesem Schritt kann man auch per Klick auf das Pluszeichen eine neue Entity-Framework-Core-Kontextklasse für SQL Server, SQLite oder PostgreSQL anlegen. Das Anlegen der Kontextklasse an dieser Stelle ist aber nicht zu empfehlen, denn die Kontextklasse würde direkt im Blazor-Projekt und nicht in einer getrennten Data-Access-Schicht landen.

Was hier im Dialog leider fehlt, ist eine Auswahl von Standort und Namen der anzulegenden Komponenten.



**Ergänzte Razor Components** für die Entitätsklasse „Flight“ und die Vorlage „CRUD“ (Bild 3)

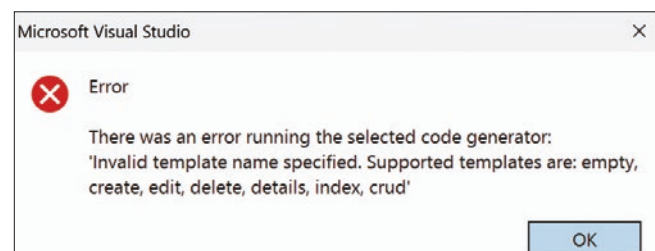
Visual Studio wählt als Standort immer `/Component/Pages/[Entitätsklassenname]Pages` (also zum Beispiel für die Entitätsklasse *Flight* den Ordner `/Component/Pages/Flight-Pages`) und feste Namen für die Razor Components: *Create.razor*, *Delete.razor*, *Details.razor*, *Edit.razor* und *Index.razor*.

Visual Studio ändert nach Klick auf **Add** das Blazor-Projekt wie folgt (siehe auch Bild 3):

- Es werden in der Projektdatei die NuGet-Pakete *Microsoft.AspNetCore.Components.QuickGrid.EntityFrameworkAdapter* und *Microsoft.VisualStudio.Web.CodeGeneration.Design* ergänzt (jeweils in Version 9.0.0, nicht in der zurzeit aktuellen Version 9.0.5). Daher ist hier noch ein manuelles NuGet-Paketupdate erforderlich.
- Im Ordner `/Components/Pages/EntitätsklassennamePages` werden eine oder mehrere Razor Components entsprechend der gewählten Vorlage ergänzt.

Neben den bereits erwähnten Herausforderungen zeigen sich bei der Codegenerierung folgende Schwächen (Stand Visual Studio 2022 17.14.5):

- Die Codegenerierung stürzt manchmal ab mit dem Fehler, dass die gewählte Vorlage nicht gefunden werden konnte. In diesem Fall ist ein Neustart von Visual Studio notwendig (siehe Bild 4).
- Für Timestamp-Spalten wird in *Create.Razor* und *Edit.Razor* auch ein Eingabefeld mit `<InputText>` erzeugt. Das funktioniert nicht, wenn die Timestamp-Spalten als Byte-Array angelegt sind. Dann sind die Eingabefelder für die Timestamp-Spalten aus den Formularen zu entfernen, und der Timestamp muss gegebenenfalls getrennt gemerkt werden (mehr dazu siehe weiter unten).
- Wenn man die Anwendung dann startet, beschwert sich Blazor zuerst, dass es keine injizierten Dienste für die Kontextklasse gibt. Man muss den folgenden Code in der *Program.cs* ergänzen: ▶



**Diese Fehlermeldung** erscheint häufiger bei der Codegenerierung. Dann muss man Visual Studio neu starten (Bild 4)

| FlightNo | Departure | Destination | FlightDate          | NonSmokingFlight | Seats | FreeSeats | PilotPersonid |
|----------|-----------|-------------|---------------------|------------------|-------|-----------|---------------|
| 100      | Hamburg   | Paris       | 02.09.2022 03:10:54 | False            | 200   | 5229      | 24420         |
| 101      | Berlin    | Seattle     | 04.05.2022 13:10:54 | False            | 200   | 88        | 24420         |
| 102      | Hamburg   | Prague      | 04.01.2022 23:17:54 | False            | 250   | 3         | 24470         |
| 103      | Prague    | Chicago     | 16.05.2022 14:34:54 | False            | 250   | 134       | 24375         |
| 104      | Chicago   | Oslo        | 16.01.2022 00:34:54 | False            | 250   | 14        | 24344         |
| 105      | Rome      | Berlin      | 27.05.2022 15:51:54 | False            | 250   | 145       | 24408         |
| 106      | London    | Berlin      | 27.01.2022 01:51:54 | False            | 250   | 25        | 24385         |
| 107      | Paris     | Madrid      | 07.06.2022 17:08:54 | False            | 250   | 156       | 24249         |
| 108      | Munich    | Milan       | 08.02.2022 03:15:54 | False            | 250   | 37        | 24323         |
| 109      | Milan     | Prague      | 18.06.2022 18:25:54 | False            | 250   | 167       | 24511         |

Rendering der Flugliste mit QuickGrid-Komponente in der generierten Seite „/FlightPages/Index.razor“ (Bild 5)

```
builder.Services.AddDbContextFactory<
    WwwingsV1EnContext>(options => options.UseSqlServer(
        builder.Configuration.GetConnectionString(
            "WWWings")));
}, ...
```

- In der *Program.cs* fehlt zudem der Aufruf *builder.Services.AddQuickGridEntityFrameworkAdapter()*.

Außerdem ist die Verbindungszeichenfolge in *AppSettings.json* abzulegen:

```
{
  "ConnectionStrings": {
    "WWWings": "Data Source=IhrServer;Initial
Catalog=WWWingsV1_EN;Integrated Security=True;Trust
Server Certificate=True;Command Timeout=300"
```

Sind diese Punkte alle abgearbeitet, sollte man tatsächlich die Daten sehen können, indem man den relativen URL aufruft, den man in */FlightPages/Index.razor* findet, zum Beispiel */flights*.

Den generierten Quellcode für die Index-Seite zeigt **Lis-ting 1**. Hier kommt das QuickGrid-Steuerelement von Microsoft für die Tabellendarstellung aller Datensätze zum Einsatz.

### • Listing 1: Generierter Code „/FlightPages/Index.razor“ mit <QuickGrid>-Steuerelement

```
@page "/flights"
@using Microsoft.AspNetCore.Components.QuickGrid
@inject BO.WWWings.WwwingsV1EnContext DB
@using BO.WWWings

<PageTitle>Index</PageTitle>

<h1>Index</h1>

<p>
  <a href="/flights/create">Create New</a>
</p>

<QuickGrid Class="table" Items="DB.Flights">
  <PropertyColumn Property="flight =>
    flight.Airline" />
  <PropertyColumn Property="flight =>
    flight.Departure" Sortable="true" />
  <PropertyColumn Property="flight =>
    flight.Destination" Sortable="true" />
  <PropertyColumn Property="flight =>
    flight.FlightDate" Sortable="true" />
  <PropertyColumn Property="flight =>
    flight.NonSmokingFlight" />
  <PropertyColumn Property="flight => flight.Seats" />
  <PropertyColumn Property="flight =>
    flight.FreeSeats" />
  <PropertyColumn Property="flight =>
    flight.PilotPersonId" />
  <PropertyColumn Property="flight => flight.Memo" />
  <PropertyColumn Property="flight =>
    flight.Timestamp" />

  <TemplateColumn Context="flight">
    <a href="@($"/flights/edit?flightno={
      flight.FlightNo})">Edit</a> |
    <a href="@($"/flights/details?flightno={
      flight.FlightNo})">Details</a> |
    <a href="@($"/flights/delete?flightno={
      flight.FlightNo})">Delete</a>
  </TemplateColumn>
</QuickGrid>
```

Die Datenspalten werden per `<PropertyColumn>` erzeugt, die Befehlsspalten per `<TemplateColumn>`. Ein Paging ist leider noch nicht implementiert.

Bei Bedarf können Entwicklerinnen und Entwickler selbst einen Eintrag in `/Components/Layout/NavMenu.razor` vornehmen, damit die neue Seite im Menü erscheint. Ebenso wird der Primärschlüssel der Datenbanktabelle nicht in der Benutzeroberfläche angezeigt. Dafür ergänzt man eine Zeile in `Index.razor` innerhalb der Komponente `<QuickGrid>`, in diesem Beispiel:

```
<PropertyColumn Property="flight => flight.FlightNo" />
```

Im Screenshot in **Bild 5** ist beides schon erledigt.

Codegenerierung per Kommandozeilenwerkzeug

Theoretisch kann man die Codegenerierung nicht nur in Visual Studio, sondern auch per Kommandozeile starten: mit dem Kommandozeilenwerkzeug `dotnet scaffold`, das allerdings nicht Teil des .NET-SDK ist, sondern dort erst als Zusatz installiert werden muss:

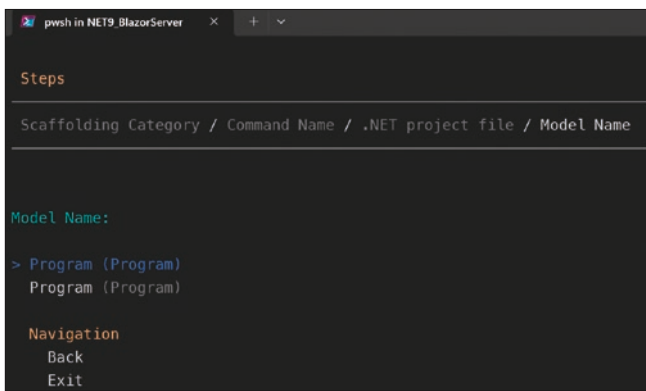
```
dotnet tool install --global Microsoft.dotnet-scaffold
```

Dann kann man innerhalb eines Blazor-Projekts `dotnet scaffold` aufrufen. Allerdings wird man feststellen, dass das Kommandozeilenwerkzeug Modellklassen nur im Blazor-Projekt selbst sucht und nicht in referenzierten Assemblies (Stand Version 9.0.1, die am 14.06.2025 aktuell ist), siehe **Bild 6**. Damit ist das Werkzeug für die Praxis ungeeignet.

QuickGrid anpassen

Nun kann man weitere Funktionen der QuickGrid-Komponente ergänzen, wie zum Beispiel Vorlagenspalten, Sortieren, Filtern und Blättern (Paging). Wichtig: Viele dieser Zusatzfunktionen funktionieren nur, wenn die Seite mit Blazor Server läuft, nicht bei Blazor Static Server-Side Rendering. Daher sieht man in **Listing 2** die Zeile `@rendermode InteractiveServer`.

Es gibt in der QuickGrid-Komponente nur zwei Spaltentypen: `<PropertyColumn>` und `<TemplateColumn>`. Der Codegenerator erstellt immer `<PropertyColumn>`. Vorlagenspalten mit beliebigem Inhalt erstellt man mit dem Tag `<TemplateColumn>`. Innerhalb von `<TemplateColumn>` greift man auf den aktuellen Datensatz via `@context` zu, zum Beispiel so:



**dotnet scaffold** findet die Modellklassen in referenzierten Assemblies nicht (**Bild 6**)

```
<TemplateColumn Title="Nichtraucher-Flug">
  <input type="checkbox" autofocus
    @bind="@context.NonSmokingFlight" />
</TemplateColumn>
```

Der Codegenerator vergibt bei `<PropertyColumn>` keine expliziten Spaltentitel, sondern verwendet im Standard den Feldnamen. Man kann dazu das Attribut `Title` ergänzen:

```
<PropertyColumn Title="Freie Plätze" Property="flight =>
  flight.FreeSeats" />
```

Damit eine Spalte sortierbar ist, verwendet man `Sortable="true"`:

```
<PropertyColumn Title="Freie Plätze" Property="flight =>
  flight.FreeSeats" Sortable="true" />
```

Die Implementierung von Paging und Filtern ist in **Listing 2** gezeigt. Es sind mehrere Dinge zu tun:

- Eingabefeld für den Filter-Suchbegriff per `<ColumnOption>` bei der entsprechenden Spalte (der Suchbegriff könnte aber auch irgendwo außerhalb der QuickGrid-Komponente eingegeben werden).
- Das Attribut `Items` im Tag `<QuickGrid>` verweist nun nicht mehr auf die Property `FlightSet` in der Kontextklasse, sondern auf eine Property mit Getter in der Razor Component, die den Filter anwendet, wenn es dort eine Eingabe gibt. ►

| FlugNr | Abflugort | Zielort             | Nichtraucher-Flug                   | Plätze | Freie Plätze | Pilot                 |
|--------|-----------|---------------------|-------------------------------------|--------|--------------|-----------------------|
| 104    | ch        | 16.01.2025 00:34:54 | <input checked="" type="checkbox"/> | 250    | 0            | Frank-Walter Seehofer |
| 108    |           | 08.02.2025 03:15:54 | <input checked="" type="checkbox"/> | 250    | 37           | Horst Özdemir         |
| 133    | Munich    | 23.02.2025 05:00:54 | <input checked="" type="checkbox"/> | 250    | 52           | Frank-Walter Gysi     |
| 139    | Munich    | 28.03.2025 08:51:54 | <input checked="" type="checkbox"/> | 250    | 85           | Horst Özdemir         |
| 158    | Chicago   | 10.03.2025 06:45:54 | <input checked="" type="checkbox"/> | 250    | 67           | Sigmar Steinmeyer     |
| 164    | Chicago   | 12.04.2025 10:36:54 | <input checked="" type="checkbox"/> | 250    | 100          | Martin Gabriel        |
| 173    | Chicago   | 28.01.2025 01:58:54 | <input checked="" type="checkbox"/> | 250    | 26           | Sigmar Wagenknecht    |
| 184    | Chicago   | 03.08.2025 23:47:54 | <input checked="" type="checkbox"/> | 250    | 213          | Sigmar Seehofer       |
| 190    | Munich    | 06.09.2025 03:38:54 | <input checked="" type="checkbox"/> | 250    | 246          | Angela Lafontaine     |
| 191    | Munich    | 09.05.2025 13:45:54 | <input checked="" type="checkbox"/> | 250    | 127          | Cem Lindner           |

1080 ItemsPage 1 of 108

Angepasste Tabellendarstellung mit QuickGrid (**Bild 7**)



- Für das Paging benötigt man eine *<Paginator>*-Komponente mit Verweis auf ein Field oder eine Property vom Typ *PaginationState*. Auf dieses Objekt verweist man in *<QuickGrid>* beziehungsweise *<Paginator>* mit dem Attribut *Pagination* beziehungsweise *State*. Hier setzt man die Anzahl der Elemente pro Seite per *ItemsPerPage*. Die QuickGrid-Komponente verwendet das *PaginationState*-

Objekt dann, um den aktuellen Zustand des Blätterns zu erfassen.

Listing 2 enthält den Quellcode für die erweiterte Tabellenansicht mit Filtern und Paging. Das verbesserte Ergebnis zeigt Bild 7. Für ein besseres Benutzererlebnis kommen nun auch Symbole statt Textlinks zum Einsatz.

## Listing 2: Angepasste Version der generierten Datei „/FlightPages/Index.razor“

```
@page "/flights"
@using Microsoft.AspNetCore.Components.QuickGrid
@using BO.WWWings
@using Microsoft.EntityFrameworkCore
@inject BO.WWWings.WwwingsV1DbContext DB
@rendermode InteractiveServer
...

<QuickGrid Class="table" Items="FilteredFlights"
    Pagination="@paginationState">
    <PropertyColumn Title="FlugNr"
        Property="flight =>
            flight.FlightNo" Sortable="true" />
    <PropertyColumn Title="Abflugort"
        Property="@flight => flight.Departure"
        Sortable="true" Class="country-name">
        <ColumnOptions>
            <div class="search-box">
                <input type="search" autofocus
                    @bind="nameFilter" @bind:event="oninput"
                    placeholder="Name der Stadt..." />
            </div>
        </ColumnOptions>
    </PropertyColumn>

    <PropertyColumn Title="Zielort"
        Property="flight => flight.Destination"
        Sortable="true" />

    <TemplateColumn Title="Nichtraucher-Flug">
        <input type="checkbox" autofocus
            @bind="@context.NonSmokingFlight" />
    </TemplateColumn>

    <PropertyColumn Title="Plätze"
        Property="flight => flight.Seats"
        Sortable="true" />
    <PropertyColumn Title="Freie Plätze"
        Property="flight => flight.FreeSeats"
        Sortable="true" />
    <TemplateColumn Title="Pilot">
        @context.PilotPerson.Person.Person.GivenName
        @context.PilotPerson.Person.Person.Surname
    </TemplateColumn>

    <TemplateColumn Context="flight">
        <a href="@($"flights/details?flightno={
            flight.FlightNo}")"><i class=
            "bi bi-arrow-right-square"></i></a>
        <a href="@($"flights/edit?flightno={
            flight.FlightNo}")"><i class="bi bi-pencil">
            </i></a>
        <a href="@($"flights/delete?flightno={
            flight.FlightNo}")"><i class="bi bi-trash">
            </i></a>
    </TemplateColumn>
</QuickGrid>

<Paginator State="@paginationState" />

@code {
    PaginationState paginationState =
        new PaginationState { ItemsPerPage = 10 };
    IQueryable<Flight>? BaseQuery;
    string nameFilter;

    IQueryable<Flight> FilteredFlights
    {
        get
        {
            IQueryable<Flight> result = BaseQuery;

            if (!string.IsNullOrEmpty(nameFilter))
            {
                result = result.Where(c =>
                    c.Departure.Contains(nameFilter));
            }

            return result;
        }
    }

    protected override void OnInitialized()
    {
        BaseQuery = DB.Flights.Include(x =>
            x.PilotPerson).ThenInclude(x =>
            x.Person).ThenInclude(x => x.Person);
    }
}
```

### Listing 3: Ein ItemsProvider für QuickGrid, der direkt einen EF-Core-Kontext nutzt

```
@code {
    string log = "";
    GridItemsProvider<BO.WWWings.Flight>?
        itemsProvider = null;

    [Parameter]
    [SupplyParameterFromQuery]
    public int overscanCount { get; set; } = 3;

    int numResults = 10000;

    protected override void OnInitialized()
    {
        if (overscanCount == 0) overscanCount = 3;
        itemsProvider = async request =>
        {
            DA.WWWings.WwwingsV1EnContext ctx = new();

            if (!request.Count.HasValue)
            {
                return GridItemsProviderResult.From(new List<
                    BO.WWWings.Flight>(), ctx.Flights.Count());
            }

            log += $"Anforderung: {(request.Count.HasValue ?
                request.Count.ToString() : "Unlimited")}"
                ({request.StartIndex} -> {request.StartIndex +
                request.Count}) <br> ";
            await this.InvokeAsync(this.StateHasChanged);

            var result = await ctx.Flights.Skip(
                request.StartIndex).Take(request.Count ??
                Int32.MaxValue).ToListAsync();
            log += $"<span style='color:green'>Geladen: {
                result.Count}</span><br> ";
            await this.InvokeAsync(this.StateHasChanged);

            return GridItemsProviderResult.From(result,
                numResults);
        };
    }
}
```

### Virtualisierung beim QuickGrid

Die QuickGrid-Komponente bietet auch eine Virtualisierung an, bei der nur der sichtbare Teil der Daten und darüber hinaus ein wenig Überhang geladen wird.

Seit .NET 9.0 darf man im Virtualisierungsmodus die Größe dieses Überhangs (Overscan) explizit angeben. Damit legt man fest, wie viele Zeilen vor oder nach dem aktuell sichtbaren Bereich das QuickGrid noch auf Vorrat rendern soll. Standard sind hier drei Zeilen.

Entwicklerinnen und Entwickler, die diesen Standardwert nun ändern wollen, erledigen das per Einstellung *OverscanCount* im Tag `<QuickGrid>`:

```
<QuickGrid ItemsProvider="@itemsProvider"
    TGridItem="BO.WWWings.Flight" Virtualize="true"
    OverscanCount="@overscanCount">
    <PropertyColumn Property="@p => p.FlightNo"
        Title="FlugNr" Sortable="true" />
    <PropertyColumn Property="@p => p.Departure"
        Title="Abflugort" Sortable="true" />
    <PropertyColumn Property="@p => p.Destination"
        Title="Zielort" Sortable="true" />
    <PropertyColumn Property="@p => p.FlightDate"
        Title="Datum" Format="dd.MM.yyyy" Sortable="true" />
</QuickGrid>
```

#### Tabelle mit 10000 Flights

OverscanCount:

Microsoft "The QuickGrid component now exposes an OverscanCount property that specifies how many additional rows are rendered **before and after** the visible region when virtualization is enabled."

|     |              |              |            |
|-----|--------------|--------------|------------|
| 115 | Seattle      | Paris        | 22.07.2024 |
| 116 | Milan        | Frankfurt    | 24.03.2024 |
| 117 | Berlin       | Frankfurt    | 02.08.2024 |
| 118 | Kapstadt     | Frankfurt    | 05.04.2024 |
| 119 | Berlin       | Dallas       | 14.08.2024 |
| 120 | Berlin       | London       | 16.04.2024 |
| 121 | London       | New York/JFC | 25.08.2024 |
| 122 | Berlin       | Madrid       | 27.04.2024 |
| 123 | Oslo         | New York/JFC | 06.09.2024 |
| 124 | Frankfurt    | Prague       | 08.05.2024 |
| 125 | New York/JFC | Dallas       | 08.01.2024 |
| 126 | London       | Madrid       | 19.05.2024 |
| 127 | Milan        | Paris        | 20.01.2024 |
| 128 | Seattle      | Prague       | 31.05.2024 |
| 129 | Oslo         | Munich       | 31.01.2024 |

Anforderung: 9 (0 -> 9)  
Geladen: 9  
Anforderung: 25 (0 -> 25)  
Geladen: 25  
Anforderung: 25 (4 -> 29)  
Geladen: 25  
Anforderung: 25 (8 -> 33)  
Geladen: 25

**Beim OverscanCount = 3 werden beim Einzeldatensatz-Scrolling immer vier weitere Datensätze geladen: der erste fehlende und drei weitere (Bild 8)**

Der *ItemsProvider* ist dann für das Liefern der gewünschten Datensätze zuständig. **Listing 3** zeigt die Implementierung eines *ItemsProvider* für das QuickGrid-Steuerelement, der direkt einen Entity-Framework-Core-Kontext nutzt – selbstverständlich sollte man dies in der Praxis in eine untere Schicht auslagern, hier wird dies jedoch der besseren Übersicht wegen nicht gemacht. Dabei kann die Größe des Overscans per Parameter an die Komponente übergeben werden (siehe auch **Bild 8**). Für .NET 10.0 plant Microsoft, dass der Codegenerator Programmcode erzeugt, der *ItemsProvider* mit Paging verwendet statt wie bisher die *Items*-Eigenschaft, siehe [4].

## Datenbasierte Formatierung mit der Eigenschaft RowClass

Im Tabellensteuerelement `<QuickGrid>` können Entwicklerinnen und Entwickler ab .NET 10.0 (zum Redaktionsschluss für diesen Beitrag auf Stand Preview 5) eine Methode beim neuen Attribut `RowClass` angeben, die bei jeder Zeile aufgerufen wird, um die CSS-Klasse der Zeile individuell auf Basis der Daten zu setzen. Die Eigenschaft `RowClass` erwartet als Rückgabewert von der Methode den Namen einer CSS-Klasse, die der Entwickler beziehungsweise die Entwicklerin

| FlugNr | ≡ Abflugort | Zielort | Datum      | Freie Plätze |
|--------|-------------|---------|------------|--------------|
| 100    | Hamburg     | Paris   | 02.09.2025 | 5211         |
| 101    | Berlin      | Seattle | 04.05.2025 | 87           |
| 102    | Hamburg     | Prague  | 04.01.2025 | 3            |
| 103    | Prague      | Chicago | 16.05.2025 | 134          |
| 104    | Chicago     | Oslo    | 16.01.2025 | 0            |
| 105    | Rome        | Berlin  | 27.05.2025 | 145          |
| 106    | London      | Berlin  | 27.01.2025 | 25           |

**Datenbasierte Formatierung** im QuickGrid mit Farbhervorhebung über die Eigenschaft „`RowClass`“ (**Bild 9**)

selbst definieren muss. Im **Listing 4** wird die Methode `ApplyRowClass()` aufgerufen, die Flüge mit weniger als zehn freien Plätzen orange und bei ausgebuchten Plätzen (0) oder Überbuchung (<0) rot markiert. Das Ergebnis sieht man in **Bild 9**.

## Automatisches Schließen des Filterkriterien-dialogs via `HideColumnOptionsAsync()`

Außerdem bietet Microsoft in .NET 10.0 im `<QuickGrid>`-Steuerelement eine neue Methode `HideColumnOptionsAsync()`, die Entwicklerinnen und Entwickler aufrufen kön-

### ● Listing 4: RowClass-Eigenschaft im Steuerelement QuickGrid in .NET 10.0

```
<style>
    .red {
        color: red;
    }

    .orange {
        color: orange;
    }

    .ok {
        color: black;
    }

    .scrollable {
        overflow-y: scroll;
        max-height: 400px; /* Höhe nach Bedarf anpassen */
    }
</style>

<div class="row">

    <div class="col scrollable">

        <QuickGrid @ref="flightGrid" RowClass=
            "ApplyRowClass" ItemsProvider="@itemsProvider"
            TGridItem="BO.WWWings.Flight" Virtualize="true"
            OverscanCount="@overscanCount">
            <PropertyColumn Property="@p => p.FlightNo"
                Title="FlugNr" Sortable="true" />

            <PropertyColumn Property="@p => p.Departure"
                Title="Abflugort" Sortable="true">
                <ColumnOptions>
                    <input type="search" @bind="departureFilter"
                        placeholder="Filter by Departure"
                        @bind:after="@(() =>
                            flightGrid.HideColumnOptionsAsync())" />
                </ColumnOptions>
            </PropertyColumn>
            <PropertyColumn Property="@p => p.Destination"
                Title="Zielort" Sortable="true" />
            <PropertyColumn Property="@p => p.FlightDate"
                Title="Datum" Format="dd.MM.yyyy"
                Sortable="true" />
            <PropertyColumn Property="@p => p.FreeSeats"
                Title="Freie Plätze" />
        </QuickGrid>

        @code {
            private string ApplyRowClass(BO.WWWings.Flight f)
            {
                if (f.FreeSeats <= 0) return "red";
                if (f.FreeSeats <= 10) return "orange";
                return "ok";
            }
        }
    </div>
</div>
```

### Listing 5: Einsatz von HideColumnOptionsAsync() im Steuerelement <QuickGrid>

```
<QuickGrid @ref="flightGrid" RowClass="ApplyRowClass"
    ItemsProvider="@itemsProvider"
    TGridItem="BO.WWWings.Flight" Virtualize="true"
    OverscanCount="@overscanCount">
    <PropertyColumn Property="(p => p.FlightNo)"
        Title="FlugNr" Sortable="true" />
    <PropertyColumn Property="(p => p.Departure)"
        Title="Abflugort" Sortable="true">
        <ColumnOptions>
            <input type="search" @bind="departureFilter"
                placeholder="Filter by Departure"
                @bind:after="@(() =>
                    flightGrid.HideColumnOptionsAsync())" />
        </ColumnOptions>
    </PropertyColumn>
    <PropertyColumn Property="(p => p.Destination)"
        Title="Zielort" Sortable="true" />
    <PropertyColumn Property="(p => p.FlightDate)"
        Title="Datum" Format="dd.MM.yyyy" Sortable="true" />
</QuickGrid>
```

nen, damit sich die Eingabe von Filterkriterien schließt, wenn der Filter aktiviert wird (siehe Listing 5 und Bild 10).

### Weitere generierte Dateien

Alle weiteren von dem Codegenerator erzeugten Dateien verwenden nicht das QuickGrid-Steuerelement:

- *Create.razor* und *Edit.razor* erstellen ein Eingabeformular mit *<EditForm>*.
- *Create.razor* enthält dabei Code für das Anfügen des Objekts per Entity-Framework-Core-Kontextklasse: *Add()* und *SaveChangesAsync()*.
- *Edit.razor* enthält hingegen Codes für das Aktualisieren des Objekts per Entity-Framework-Core-Kontextklasse: *DB.Attach(Flight!).State = EntityState.Modified* und *SaveChangesAsync()*.
- *Details.razor* ist eine reine Ansicht, die mit den HTML-Tags *<dl>*, *<dd>* und *<dt>* erzeugt wird.
- *Delete.razor* verwendet ebenfalls diese Tags zur Ansicht des Datensatzes, bietet aber noch eine Schaltfläche, die dann *Remove()* und *SaveChangesAsync()* in der Entity-Framework-Core-Kontextklasse aufruft.

Auf ein weiteres Problem stößt man in *Edit.razor*, wenn man Timestamp-Spalten (alias *RowVersions*) in der darzustellenden Tabelle verwendet. In der Regel kann (und will) man die Spalte nicht im Formular darstellen. Wenn aber die Timestamp-Spalte beim Blazor Static Server-Side Rendering fehlt, geht die Timestamp-Information beim Rundgang verloren, und ein Speichern der Aktualisierung wird von Entity Framework Core verweigert mit dem Hinweis, dass der Datensatz in der Datenbank schon geändert wurde, auch wenn das nicht stimmt.

Die einfache Lösung ist, dass man auch die Seite *Edit.razor* per Blazor Server rendert (*@rendermode InteractiveServer*), dann bleibt der Zustand des *Flight*-Objekts via Blazor Server-Circuit erhalten und der Timestamp somit auch. Wenn man komplett auf Blazor Static SSR setzen will oder muss, muss man sich die Timestamp-Spalte manuell merken, zum Beispiel in einem versteckten Feld, einem Cookie oder einer Session-Variablen, und sie vor dem Aufruf von *SaveChangesAsync()* wieder ins Objekt speichern. ■

- [1] Microsoft Devblog, *Erstankündigung des QuickGrid-Steuerelements*, <https://devblogs.microsoft.com/dotnet/asp-net-core-updates-in-dotnet-7-preview-6>
- [2] QuickGrid Live Demo, <https://aspnet.github.io/quickgridsamples>
- [3] StackOverflow, *Making a Cell Editable in Blazor QuickGrid*, <https://stackoverflow.com/questions/76847619/making-a-cell-editable-in-blazor-quick-grid>
- [4] Javier Calvarro Nelson, *[Blazor] Improve the experience with QuickGrid and EF Core #58716*, <https://github.com/dotnet/aspnetcore/issues/58716>

| FlugNr | Abflugort    | Zielort | Datum      |
|--------|--------------|---------|------------|
| 100    |              |         | 02.09.2025 |
| 101    | Paris        |         | 04.05.2025 |
| 102    | Hamburg      | Prague  | 04.01.2025 |
| 103    | Prague       | Chicago | 16.05.2025 |
| 104    | Chicago      | Oslo    | 16.01.2025 |
| 105    | Rome         | Berlin  | 27.05.2025 |
| 106    | London       | Berlin  | 27.01.2025 |
| 107    | Paris        | Madrid  | 07.06.2025 |
| 108    | Munich       | Milan   | 08.02.2025 |
| 109    | Milan        | Prague  | 18.06.2025 |
| 110    | Paris        | Madrid  | 19.02.2025 |
| 111    | New York/JFC | Chicago | 29.06.2025 |
| 112    | Frankfurt    | Milan   | 02.03.2025 |

Der Aufruf „*HideColumnOptionsAsync()*“ sorgt dafür, dass sich die geöffnete Filtereingabe automatisch wieder schließt, wenn der Nutzer beziehungsweise die Nutzerin eine Eingabetaste drückt (Bild 10)

### Dr. Holger Schwichtenberg

gehört zu den bekanntesten .NET- und Webexperten in Deutschland. Von Microsoft ist er seit 21 Jahren als MVP ausgezeichnet. Mit dem 53-köpfigen Expertenteam bei [www.IT-Visions.de](http://www.IT-Visions.de) bietet er Beratungen, Schulungen und Softwareentwicklung an.

[www.dotnet-doktor.de](http://www.dotnet-doktor.de)



## AUTOMATISIEREN MIT MICROSOFT .NET, DATAVERSE UND POWER AUTOMATE

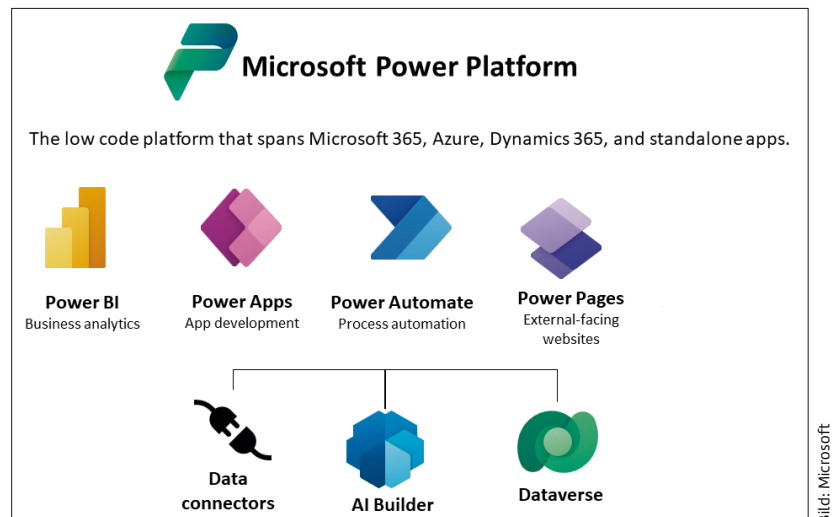
# Moderne Produktivität

Mit Microsoft Power Automate lassen sich schnell und einfach eigene Apps entwickeln, die bei der Prozessautomatisierung mit .NET unterstützen können.

Die Prozessautomatisierung spielt – neben generativer AI und KI-Agenten – auch heute eine zentrale Rolle in der Softwareentwicklung und ist ein Schlüsselansatz zur modernen Produktivität.

Im einfachsten Fall bezeichnet die Prozessautomatisierung die Umwandlung von manuell ausgeführten Prozessen oder Services in vollständig automatisierte Prozesse. Im Rahmen der heutigen Digitalisierung und Transformation bilden Technologien wie Robotic Process Automation und Workflow-Automatisierung die zentralen Ansätze für moderne Produktivität.

Das primäre Ziel ist somit die Optimierung von Arbeitsabläufen durch Software und die Automatisierung täglicher Routineaufgaben, um so die Effizienz zu steigern.



Bausteine der Microsoft Power Platform (Bild 1)

Bild: Microsoft

## Prozessautomatisierung

Hinter der Prozessautomatisierung verbirgt sich einfach die Möglichkeit, Arbeitsabläufe durch Software zu optimieren. Das soll aber nicht bedeuten, dass Mitarbeiter durch die Softwarelösung ersetzt werden sollen, sondern ist es bei der Automation vielmehr wichtig, tägliche Routineaufgaben zu automatisieren.

Die Prozessautomatisierung kann einerseits sehr einfach sein und nur wenige Kernaufgaben erfassen, oder sie kann Teil einer umfassenden IPA- (Intelligente Prozessautomatisierung) und Hyperautomatisierungsstrategie sein. Die wichtigsten Arten der Prozessautomatisierung sind somit:

- **Einfache Automatisierung:** Hierbei handelt es sich um feste Prozesse, die wenige Kernaufgaben erfassen.
- **Regelbasierte Automatisierung:** Diese Art umfasst mehrere mögliche Prozesswege, die von variablen Bedingungen und Entscheidungen abhängen.
- **Intelligente Automatisierung (IPA):** Diese Form ist für Prozesse konzipiert, bei denen Ausnahmen auftreten, die nicht mit vordefinierten Regeln verarbeitet werden können. Hier kommen fortgeschrittene Technologien wie Robotic Process Automation (kurz RPA), Workflow-Automatisierung oder künstliche Intelligenz zum Einsatz.

Daraus ergibt sich, dass eine Automatisierung besonders sinnvoll für Prozesse ist, die folgende Merkmale aufweisen:

- Wo Prozesse bislang manuell durchgeführt werden, aber digital effizienter wären,

- sich regelmäßig wiederholen,
- standardisiert oder regelbasiert ablaufen,
- ein hohes Datenverarbeitungs- oder Transaktionsvolumen besitzen und
- eine hohe Prozessqualität erfordern.

## Ablauf der Prozessautomatisierung

Der genaue Ablauf einer Prozessautomatisierung unterscheidet sich selbstverständlich von Fall zu Fall und hängt stark von der Art des Prozesses und der Vorgehensweise ab. Allgemein lassen sich folgende Schritte bei der Prozessautomatisierung ausmachen:

- **Analyse und Planung:** Prozesse verstehen, wiederkehrende Schritte identifizieren und Automatisierung evaluieren.
- **Auswahl** einer Automatisierungslösung.
- **Testen:** Funktionieren die automatisierten Prozesse und liefern sie die gewünschten Ergebnisse?
- **Implementierung:** Die Automatisierung wird in den laufenden Betrieb überführt.
- **Überwachung und Wartung.**
- **Optimierung** der Prozesse.

Was den zweiten Punkt, die Auswahl einer Automatisierungslösung, angeht, soll zum besseren Verständnis noch darauf eingegangen werden, was sich hinter den Begriffen Robotic Process Automation und Workflow-Automatisierung verbirgt.

## Robotic Process Automation

Robotic Process Automation, im Folgenden nur noch kurz RPA genannt, bezeichnet einen Oberbegriff für Technologien der robotergestützten Prozessautomatisierung. Hierbei bezieht man sich jedoch nicht auf physische Roboter, sondern auf eine softwarebasierte Automatisierung, die es ermöglicht, sich wiederholende und zeitaufwendige Prozesse zu automatisieren.

RPA stellt programmierbare Tools und Softwareroboter, sogenannte Bots, zur Verfügung, die in Geschäftssysteme integriert werden, um dann über Skriptanweisungen verschiedene Aufgaben und Interaktionen zu automatisieren und zu vereinfachen. Die Bots können im System navigieren, Daten lesen und eingeben sowie eine Vielzahl von regelbasierten Aufgaben ausführen.

## Workflow Automation

Der Begriff Workflow Automation wird sehr oft in einem Zug mit RPA verwendet. Im eigentlichen Sinn stellt Workflow Automation aber keine Robotic Process Automation dar. Sie spart vielmehr allgemein Zeit und steigert zugleich die Effizienz. Dazu konzentriert sie sich auf die Optimierung alltäglicher Prozesse durch eine Reihe von automatischen Aktionen. Die Workflow Automation kann als die unabhängige Ausführung von Aufgaben definiert werden, indem Arbeitsaufgaben identifiziert werden, die mithilfe von Technologien, digitalen Apps und anderen Tools automatisiert werden können.

Workflow Automation befasst sich also vorrangig mit der Identifikation von Arbeitsaufgaben, die mithilfe von Technologien, digitalen Apps und weiteren Tools automatisiert werden.

## RPA-Software

Sucht man nach Robotic Process Automation, findet man auf dem Markt ein weites Feld aus neuen, speziell entwickelten Tools, die mit den unterschiedlichsten Automatisierungsfunktionen auch für Spezialgebiete ausgestattet sind.

Neben Anbietern wie Airslate (Dokumentenverarbeitung) oder Appian (Client- und Compliance-Management) findet man auch die großen Player wie AWS Lambda (Amazon Cloud), IBM Cloud Pak for Business Automation und Microsoft Power Automate.

Die Tools ermöglichen die Unterstützung von Desktop-Anwendungen, Webseiten, Cloud-Plattformen und hybriden Lösungen. So gibt es für fast jeden Anwendungsfall auch eine spezielle RPA-Software, wie zum Beispiel für Web-Scraping- und Browser-Automatisierung.

## Microsoft Power Automate

Power Automate ist ein integrierter Bestandteil der Microsoft Power Platform (Bild 1). Es ist darauf ausgelegt, Entwicklerinnen und Entwickler beim Erstellen von Apps, virtuellen Agenten und Business-Intelligence-Reportagen zu unterstützen, und stellt eine leistungsstarke RPA-Lösung dar, die die Automatisierung von Aufgaben und Prozessen erheblich erleichtert.

Das Desktop-Tool (Desktop Flow) ist fokussiert auf die Automatisierung gängiger Prozesse in Windows (ab Windows 10), während Cloud Flows serverseitige Aufgaben übernehmen können.

Ein großer Vorteil von Power Automate ist seine benutzerfreundliche Oberfläche, die es jedem Benutzer ermöglicht, individuelle Arbeitsabläufe nachzuverfolgen und in eine automatisierte, editierbare Routine umzuwandeln. Da es sich um ein cloudbasiertes Toolset handelt, ist Power Automate für verschiedene Plattformen verfügbar, einschließlich Android- und iOS-Phones und Tablets.

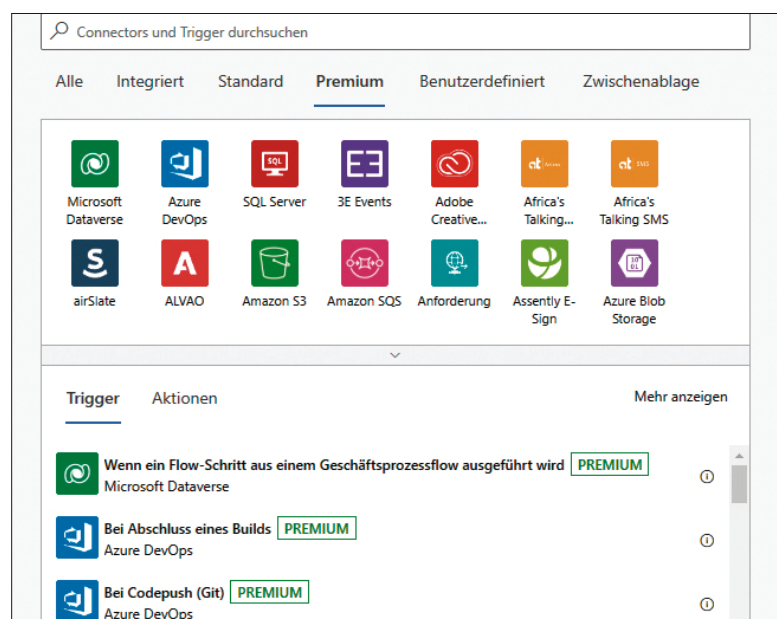
Power Automate stellt damit eine leistungsstarke RPA-Lösung dar, die darauf abzielt, die Automatisierung von Aufgaben und Prozessen zu erleichtern. Des Weiteren unterstützt Power Automate die assistierende Automatisierung (Attended RPA), darunter versteht man die Automatisierung sich wiederholender manueller Aktivitäten. Hierbei werden Aktionen imitiert, die auf dem Desktop oder Browser ausgeführt werden, wie etwa ein Mausklick, indem diese Aktion in Echtzeit aufgezeichnet und wiedergegeben wird.

Der größte Vorteil von Power Automate ist, dass Sie als Entwickler in einem Workflow die unterschiedlichsten Anwendungen wie Excel, Outlook, OneDrive, OneNote, SharePoint und Dynamics 365 integrieren können.

Von Vorteil ist auch die Vielzahl an verfügbaren Konnektoren (Bild 2), über die weitere Anwendungen eingebunden werden können. Darunter fallen Adobe-Programme, Dropbox, Facebook, Gmail, Instagram, YouTube und eine Reihe von Datenbanken wie MySQL oder auch Oracle. Positiv anzumerken ist außerdem die Verwendung von Microsoft Dataverse.

Hier sind die wichtigsten Aspekte von Power Automate:

- **Prozessautomation:** Power Automate ermöglicht es, mehrstufige Geschäftsprozesse zu automatisieren.
- **Integration mit anderen Anwendungen:** Es unterstützt die Integration unterschiedlicher Anwendungen wie Excel, ►



Mögliche Konnektoren in Power Automate Flow (Bild 2)

Outlook und SharePoint sowie die Anbindung an externe Dienste über Konnektoren.

- **Vorgefertigte Vorlagen:** Power Automate bietet eine ganze Reihe an vorgefertigten Prozessvorlagen.
- **Verschiedene Flow-Typen:** Power Automate erlaubt die Erstellung unterschiedlicher Flow-Typen wie automatisierte, geplante oder Geschäftsprozess-Flows.

## Microsoft Dataverse

Dataverse ermöglicht das rollenbasierte Speichern und Verwalten von Daten in Microsoft-Geschäftsanwendungen. Die Standardentitäten und benutzerdefinierten Tabellen in Dataverse bieten eine sichere und cloudbasierte Speicheroption für Daten. Dataverse bietet bei der Verwendung folgende Vorteile:

- **Einfache Verwaltung:** Daten und Metadaten werden in der Cloud über ein umfassendes Rechte- und Rollenkonzept gespeichert.
- **Präzise Zugriffsrechte:** Benutzer haben nur Zugriff oder Sichtbarkeiten auf Daten, die ihnen über die rollenbasierte Sicherheit zugewiesen sind.
- **Nahtloser Zugriff auf Dynamics 365:** Alle Daten aus Dynamics-365-Anwendungen wie Sales oder Customer Service werden in Dataverse gespeichert. Dies ermöglicht Entwicklern den schnellen und effektiven Zugriff auf diese Daten in Prozess-Automationen über Power Apps, Power Automate und Flows, ohne dass spezielle Schnittstellenintegrationen erforderlich wären, sofern die entsprechenden Zugriffsrechte vorhanden sind.

Das bedeutet für Sie als Entwicklerin oder Entwickler, dass sämtliche Daten aus Dynamics-365-Anwendungen in Dataverse verfügbar sind. Auch der Zugriff über Power Apps, Power Automate und Flows kann hier mit entsprechenden Zugriffrechten ohne eine spezielle Integration von Schnittstellen, die sonst erforderlich wären, erfolgen (Bild 3). Dies vereinfacht und beschleunigt das Erstellen von Apps und Flows ganz erheblich.

## Integration von .NET Framework mit Power Automate und Dataverse

Durch den Einsatz von Power Automate und Dataverse werden Ihre .NET-Anwendungen jedoch nicht obsolet. Microsoft bietet hier über entsprechende Web-APIs und über ein SDK für .NET [1] die Möglichkeit an, Daten mithilfe von Code in Dataverse zu bearbeiten (Bild 4).

Dataverse bietet somit zwei Möglichkeiten an, um Daten abzurufen, hinzuzufügen und zu ändern: einmal sehr einfach über das SDK für .NET, oder durch ein entsprechendes Web-API.

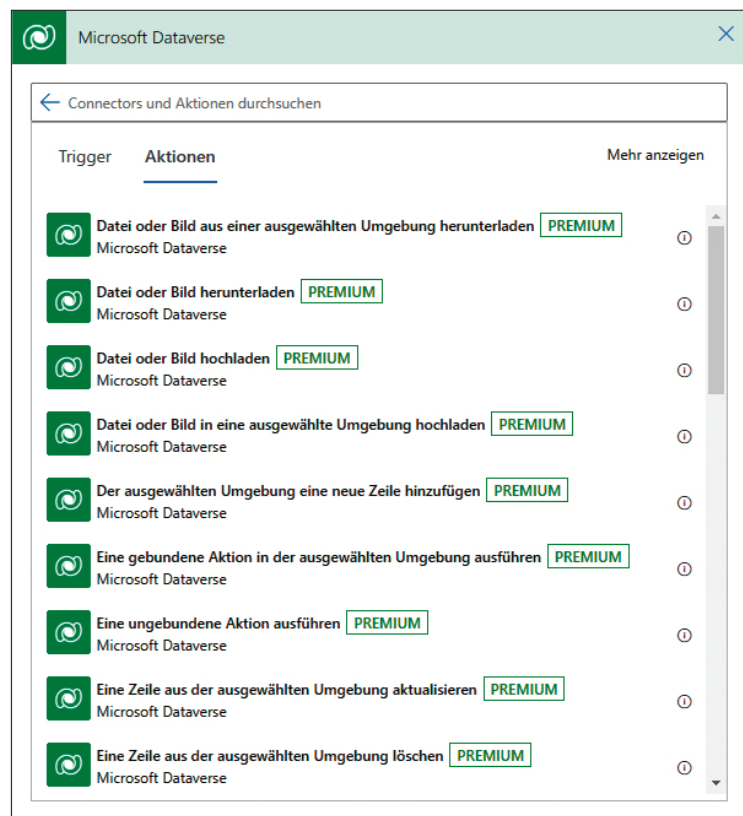
Beim SDK verwenden Sie die *ServiceClient*-Klasse in Dataverse Service Client für entsprechende Clientanwendungen oder die *Microsoft.CrmSdk.CoreAssemblies* für ein Plug-in oder eine Workflow-Aktivität im Dynamics-365-Umfeld.

Das Dataverse-Web-API ist ein OData-v4.-RESTful-Endpunkt. Dieses Web-API kann mit jeder Programmiersprache verwendet werden, die HTTP-Anforderungen und die Authentifizierung mit OAuth 2.0 unterstützt.

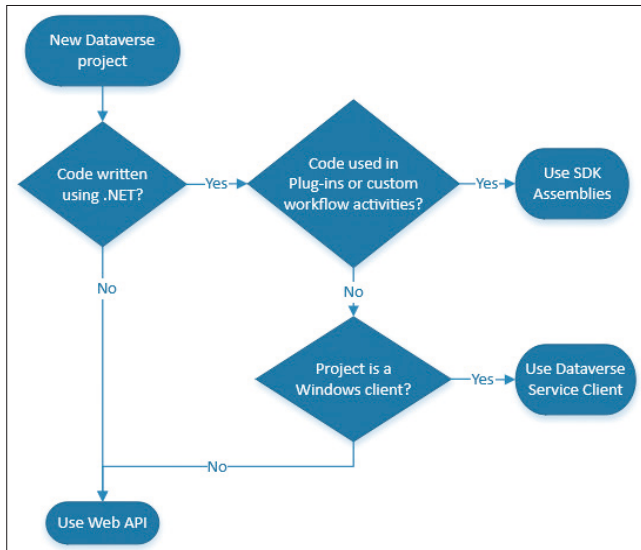
Durch den Low-Code- beziehungsweise No-Code-Ansatz können traditionelle Entwicklerinnen und Entwickler diese Plattform sehr einfach für ihre Projektaufgaben nutzen. Dies ermöglicht eine schnelle Nutzung der Prozessautomatisierung in bestehenden oder neuen Applikationen oder Web-Apps, ohne großen Zusatzentwicklungsaufwand für Schnittstellen und Zugriffsrechte.

Die Kombination dieser Technologien eröffnet vielfältige praktische Möglichkeiten für Entwickler. Hier sind drei kurz skizzierte Anwendungen:

- Ein Entwickler kann beispielsweise über eine ASP.NET-Core-Blazor-App einen in Power Automate geschriebenen Flow auslösen, wenn ein Anwender in der Blazor-App eine bestimmte Funktion nutzt. Dies kann zum Beispiel der automatische Versand einer E-Mail nach der Erfassung eines Listeneintrags in der App sein. Der Entwickler spart sich dabei die komplette Implementierung einer Schnittstelle für den E-Mail-Versand.
- Alternativ können Daten automatisch in Excel-Tabellen verschoben oder in Salesforce beziehungsweise Microsoft Dynamics eingetragen werden, indem einfach ein HTTP-Endpunkt in einem Flow getriggert und somit der automatische Prozess ausgelöst wird.
- Bild 5 illustriert ein Beispiel, bei dem ein Power-Automate-Workflow einen HTTP-Endpunkt registriert, Daten von ei-



Aktionen für Daten in Microsoft Dataverse (Bild 3)



Prozess für ein neues Dataverse-Projekt (Bild 4)

ner Webseite empfängt und diese dann automatisch in eine neu geschaffene Entität im Dynamics-365-CRM einträgt.

Bereits diese wenigen Beispiele zeigen, dass Power Automate in Verbindung mit geeigneten Low-Code-Ansätzen digitale Workflows einfach und effektiv verschlanken kann.

Der Einsatz von Power Automate und .NET kann somit sehr effektiv sein, wenn man die Stärken beider Technologien gezielt einsetzt. Sie ergänzen sich in den nachfolgenden Anwendungsbereichen ideal:

- **Low-Code trifft Full-Code:** Power Automate für einfache Workflows, .NET für komplexe Geschäftslogik und Datenverarbeitung.
- **Erweiterbarkeit:** Mit .NET können eigene APIs oder Azure Functions als benutzerdefinierte Konnektoren für Power Automate entwickelt werden.
- **Integration in Microsoft Cloud:** Power Automate und .NET sind nahtlos in das Microsoft-Power-Plattform-Ökosystem, Azure und Microsoft 365 integriert, was End-to-End-Lösungen ermöglicht.
- **Arbeitsteilung:** Power Automate triggert Prozesse, während .NET die komplexe Backend-Logik übernimmt.
- **Zusammenarbeit:** Business-Anwender können einfache Prozesse mit Power Automate gestalten, während Entwickler durch .NET die nötige Stabilität und Komplexität beisteuern.

Die aufgeführten Anwendungsbereiche können damit auch die eigene Softwareentwicklung erleichtern, da es ohne großen Aufwand möglich ist, eine beliebige Logik aus .NET über einen Custom Connector in Power Automate zu integrieren. Hierfür reichen die nachfolgenden drei Schritte in Power Automate:

- .NET-Web-API bereitstellen.
- Einen Custom Connector in Power Automate erstellen, den URL und die OpenAPI- beziehungsweise

Swagger-Dokumentation angeben und die Authentifizierung konfigurieren.

- Im Flow diesen Connector wie in jeden anderen Schritt im Low-Code-Bereich verwenden.

Nutzen Sie als Entwickler Power-Automate-Desktop-Flows, so können diese auch ganz einfach über Scripting oder benutzerdefinierte Aktionen .NET-DLLs ansprechen:

- über PowerShell-Skripte, die .NET-DLLs laden,
- über eigene COM-kompatible .NET-Klassen.

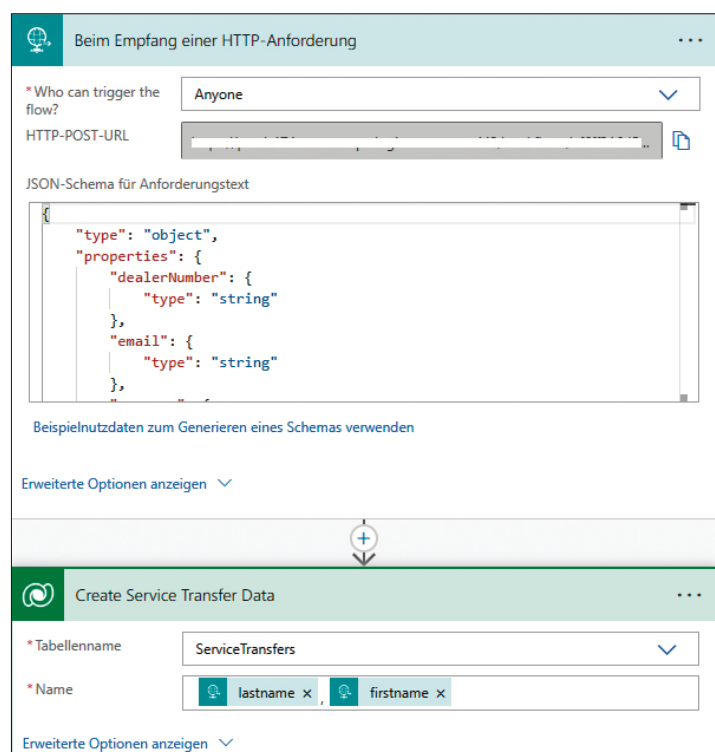
Des Weiteren ermöglicht Power Automate auch die Nutzung in modellgetriebenen Systemen für klassische Geschäftsprozesse.

## Model-Driven Apps und .NET

Das Zusammenspiel von Model-Driven Apps (MDA), modellgesteuerter Datenverarbeitung (MDV) und .NET bringt ebenfalls zahlreiche Vorteile mit sich, insbesondere in Bezug auf Effizienz, Wartbarkeit und Skalierbarkeit von Unternehmensanwendungen.

Model-Driven Apps sind eine Komponente der Microsoft Power Apps, die auf dem Common Data Service (Dataverse) basiert. Sie ermöglichen es, komplexe Geschäftsprozesse in einer benutzerfreundlichen, komponentenbasierten Oberfläche abzubilden. Das UI wird automatisch anhand der Datenstruktur und der Geschäftslogik generiert.

Die modellgesteuerte Datenverarbeitung bezeichnet innerhalb dieses Kontextes die strukturierte, regelbasierte Verarbeitung von Daten in einem modellierten Datenmodell, ►



Power-Automate-Aktionen beim Empfang einer HTTP-Anforderung (Bild 5)



### ● Listing 1: Das JSON-Datenformat

```
{
  "type": "object",
  "properties": {
    "body": {
      "type": "string"
    },
    "email": {
      "type": "string"
    }
  }
}
```

in dem Entitäten, Beziehungen und Geschäftsregeln zentral definiert sind.

Beim modellgesteuerten Design in der App geht es darum, mithilfe eines App-Designer-Tools Komponenten wie Formulare, Ansichten, Diagramme und Dashboards zu Tabellen hinzuzufügen. Und es werden Beziehungen in sogenannten Verbindungstabellen so zusammengefügt, dass die Navigation zwischen ihnen möglich und sichergestellt ist, sodass Daten nicht unnötig wiederholt werden. Mit dem Microsoft App-Designer (Bild 6) können Sie mit wenig oder gar keinem Code einfache bis hin zu sehr komplexen Anwendungen erstellen.

Auch hier ermöglicht die Integration von .NET das Erweitern von Model-Driven Apps durch komplexe Logik, wie sie in Plug-ins und benutzerdefinierten Services vorkommt, sowie den Zugriff auf .NET-Bibliotheken für Sicherheit, Datenverarbeitung und API-Integration. Darüber hinaus erhalten Sie die vollständige Unterstützung für objektorientierte Programmierung und Testbarkeit.

Ein weiteres Plus stellt die Skalierbarkeit und Wiederverwendbarkeit dar. Modelle und .NET-Komponenten können modular aufgebaut werden. Der Code lässt sich in Bibliotheken kapseln und in mehrere Anwendungen wiederverwenden. Die Datenmodelle und das UI können standardisiert werden und bilden eine gute Grundlage für große, unternehmensweite Lösungen.

Ziel des Einsatzes von MDA, MDV, Power Automate und .NET ist es, eine robuste Plattform zu schaffen, mit der

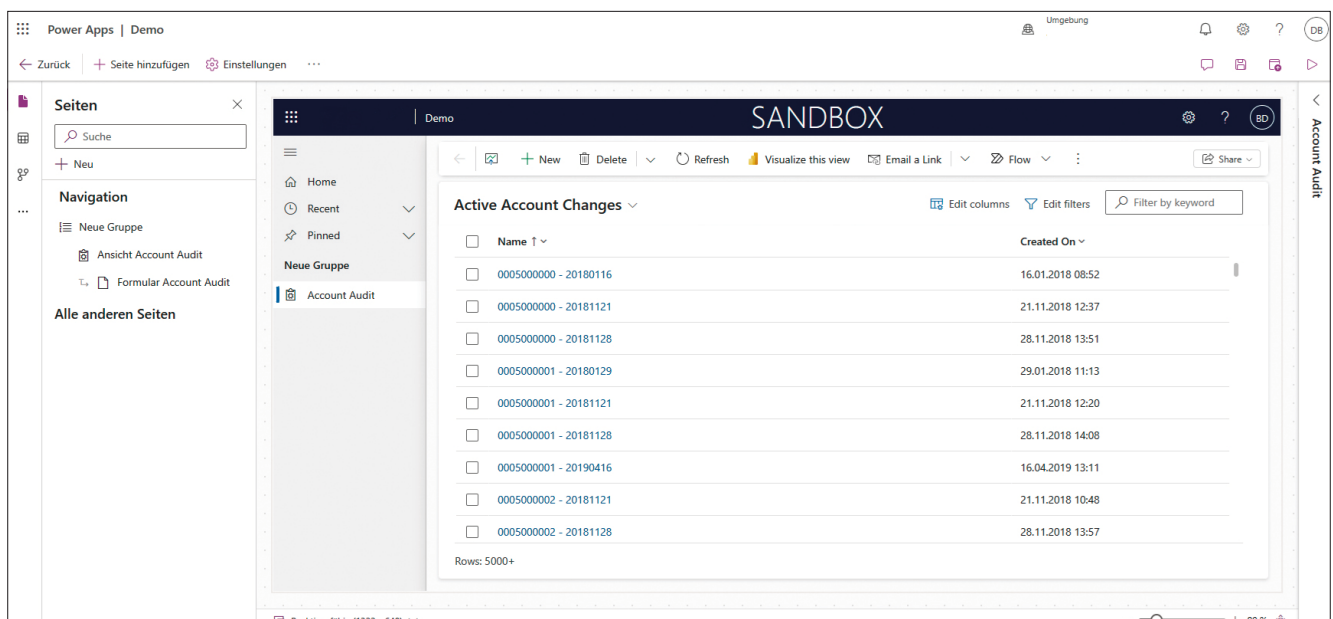
- Geschäftsprozesse konsistent abgebildet und ausgeführt werden können,
- Benutzer durch standardisierte Eingabemasken geführt werden,
- Daten zentral verwaltet und in Echtzeit verarbeitet werden und
- manuelle, repetitive Aufgaben durch automatisierte Workflows ersetzt werden.

Das Zusammenspiel von Model-Driven Apps, modellgesteuerter Datenverarbeitung und .NET kombiniert das Beste aus beiden Welten: schnelle, modellbasierte Entwicklung und mächtige Erweiterbarkeit durch klassischen Code. Ideal für komplexe, skalierbare Anwendungen mit hohem Anpassungsbedarf.

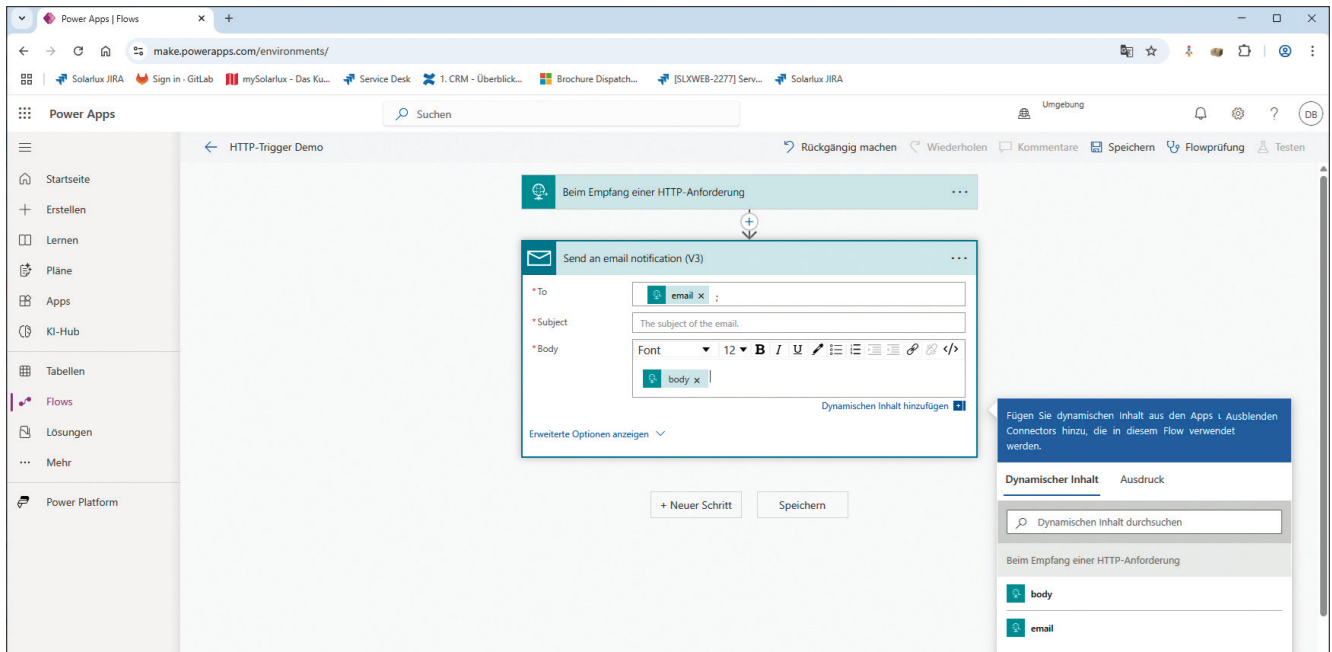
Auch die Kombination von MDA, MDV, Power Automate und .NET stellt ein sehr leistungsfähiges kombiniertes Framework zur Verfügung, mit dem Geschäftsprozesse effizient digitalisiert und automatisiert werden können.

Die klare Trennung zwischen Datenmodell, Benutzeroberfläche und Prozesslogik erleichtert die Wartung und Erweiterung. Neben den vielen Vorteilen gibt es jedoch auch typische Herausforderungen und Stolperfallen, auf die man achten sollte. Hier die wichtigsten im Überblick:

- Komplexe oder langsame Flows: Hierbei geht es um zu viele Schritte, Schleifen oder unnötige Aktionen im Flow.



Arbeiten mit dem App-Designer von Power Apps (Bild 6)



HTTP-Request an einen Power Automate Flow (Bild 7)

Diese können zu Performance-Problemen oder Zeitüberschreitungen führen.

- Fehlende oder falsche Trigger-Konfiguration: Trigger reagieren zu früh oder zu spät. Der Flow wird zu oft ausgelöst, oder wichtige Änderungen werden übersehen.
- Berechtigungen und Sicherheitsrollen: Der erstellte Flow läuft unter einem Benutzer, der nicht die nötigen Rechte hat, oder der Flow schlägt fehl beim Zugriff auf Dataverse oder andere Systeme.
- Asynchrone Verarbeitung und Verzögerung: Einige Flows oder Aktivitäten benötigen Zeit, wie zum Beispiel für einen HTTP-Request. Das muss bei der Entwicklung und Ausführung beachtet werden.
- Fehlende Fehlerbehandlung.
- Limitierungen von Power Automate.

Am häufigsten entstehen Probleme durch Fehlkonfigurationen, Rechteprobleme oder fehlende Transparenz für den Benutzer.

## HTTP-Request an einen Power Automate Flow

Das nachfolgende Beispiel soll veranschaulichen, wie einfach das Zusammenspiel von Power Automate Flow und einer C#-Konsolenanwendung sein kann. Mithilfe einer C#-Konsolenanwendung soll ein HTTP-Request an einen Power Automate Flow mit einem HTTP-Trigger gesendet werden. Ziel ist, dass das .NET-Programm einen Power Automate Flow auslöst, der über einen HTTP-Endpunkt getriggert wird.

Hierfür erstellen Sie in Power Automate einen Flow mit dem Trigger *Beim Empfang einer HTTP-Anforderung*. Fügen Sie das JSON-Beispiel aus Listing 1 für den Body des HTTP-Anforderungstextes ein.

Im nächsten Schritt fügen Sie die gewünschte Aktion hinzu, zum Beispiel *E-Mail-Benachrichtigung senden*. Tragen

Sie bei *To* den Dynamischen Inhalt *email* ein und bei *Body* den Inhalt von *body* (Bild 7). Als *Subject* vergeben Sie einfach den Begriff *Demo*.

Speichern Sie den Flow, so erhalten Sie einen URL als Endpunkt-Adresse, an den Sie den *HTTP-Request* senden können. Fertig ist der Power Automate Flow zum Versenden von E-Mails.

Listing 2 zeigt den strukturierten C#-Code als Konsolenanwendung zum Aufrufen des Power Automate Flows. Dabei ist der *powerAutomateURL* der vom Flow automatisch generierte URL, der den HTTP-Endpunkt repräsentiert. Das Versenden des HTTP-Requests erfolgt mithilfe der *HttpClient*-Klasse aus dem .NET Framework und der *client.PostAsync*-Methode.

Beachten Sie bitte, dass der sehr einfache HTTP-Endpunkt, der den Trigger auslöst, öffentlich zugänglich ist. Verwenden Sie für produktive Einsätze zusätzliche Sicherheitsprüfungen wie Token oder noch besser Azure API Management oder Power Automate Premium-Konnektoren für gesicherte APIs.

Das Beispiel sollte aufzeigen, wie Sie alternative Automatisierungen im Umfeld von Power Automate mit C# implementieren können.

## Agentic AI

Agentic AI stellt eine autonom und zielorientiert arbeitende Form einer künstlichen Intelligenz dar. Mit KI ausgestattete Agenten sollen in der Lage sein, selbstständig zu lernen, Entscheidungen zu treffen, Maßnahmen zu ergreifen, Aktivitäten auszuführen, sich dynamisch anzupassen und Ziele zu verfolgen.

Die Agenten sollen somit auch komplexe oder aus mehreren Einzelschritte bestehende Problemstellungen lösen. Agentic AI wird als die nächste Evolutionsstufe in der Prozessautomatisierung gesehen. ►

## Listing 2: HTTP-Request senden

```
using System.Text;

class Program
{
    static async Task Main(string[] args)
    {
        string powerAutomateUrl = "https://
        prod-XX.west-europe.logic.azure.com:443/
        workflows...";

        var jsonBody = @"{
            ""name"": ""Max Mustermann"",
            ""email"": ""max@example.com""
        }";

        using (HttpClient client = new HttpClient())
        {
            var content = new StringContent(jsonBody,
                Encoding.UTF8, "application/json");

            HttpResponseMessage response = await client.
                PostAsync(powerAutomateUrl, content);

            if (response.IsSuccessStatusCode)
            {
                Console.WriteLine(
                    "Flow erfolgreich ausgelöst!");
                string result = await
                    response.Content.ReadAsStringAsync();
                Console.WriteLine("Antwort vom Flow: " +
                    result);
            }
            else
            {
                Console.WriteLine("Fehler: " +
                    response.StatusCode);
                string error = await
                    response.Content.ReadAsStringAsync();
                Console.WriteLine("Details: " + error);
            }
        }
    }
}
```

Die heute bekannten KI-Agenten können mittlerweile sehr viel, doch gerade bei der Übergabe zwischen den IT-Systemen sind die Herausforderungen heute noch groß, weil Daten zunächst harmonisiert und verfügbar gemacht werden müssen. Viele Umgebungen sind heute noch stark fragmentiert oder bestehen aus Monolithen, bei denen noch zwingend APIs und Robotic Process Automation benötigt werden.

Im Vergleich zu Power Automate, Model-Driven Apps und .NET zeigt Agentic AI ein weniger vorhersehbares beziehungsweise unerwartetes Verhalten, was sie für sicherheitskritische IT-Systeme und Anwendungen derzeit weniger geeignet erscheinen lässt. Dennoch machen die KI-Agenten sehr große Fortschritte im Bereich des autonomen Troubleshooting und führen sicher in den nächsten Jahren zu einer intelligenten Automatisierung.

Das Risiko von Fehlverhalten ist beim Einsatz von Power Automate, MDA und .NET demgegenüber im Moment deutlich geringer, da klar definiert ist, was passiert. Die Skalierung erfolgt durch bewährte architektonische Mechanismen wie Lastverteilung und APIs. Auch Schulung und Wartung sind einfacher, da weniger Spezialwissen in KI oder Prompt Engineering erforderlich ist. Diese Kombination ist ideal für Stabilität, Transparenz und klassische Geschäftsprozesse.

In der Power Platform von Microsoft steht auch schon der AI Builder zur Verfügung [2]. Hiermit können KI-Modelle in Power Automate verwendet und mit Azure AI Studio auch entsprechende KI-Agenten entwickelt werden. Daher sollte man diese Entwicklung unbedingt im Auge behalten. Auch hybride Lösungen können entsprechend über alle Plattfor-

men kombiniert werden, um das Beste aus der jeweiligen Welt zu nutzen.

## Fazit

Dieser kurze Überblick hat aufgezeigt, dass automatische Prozessabläufe kein Hexenwerk sind und sich sehr gut mit eigenen Anwendungen kombinieren lassen. Die heutigen RPA-Plattformen bieten eine ideale Basis für eine zentrale visualisierte Prozessmodellierung, die man sich in den eigenen Apps zunutze machen kann.

Die interessanten Funktionen in Power Automate zeigen auch, wie gut Prozesse sich elektronisch unterstützen lassen und wie so zudem leichter und schneller auf Änderungen in einem agilen Umfeld reagiert werden kann. ■

[1] *Microsoft.PowerPlatform.Dataaverse.Client*,  
[https://www.nuget.org/packages/](https://www.nuget.org/packages/Microsoft.PowerPlatform.Dataaverse.Client/)

[2] *AI Builder*, <https://learn.microsoft.com/de-de/ai-builder/>



**Daniel Basler**

arbeitet als Lead Developer und Softwarearchitekt. Seine Schwerpunkte liegen auf Cross-Plattform-Apps, Android, JavaScript und Microsoft-Technologien.

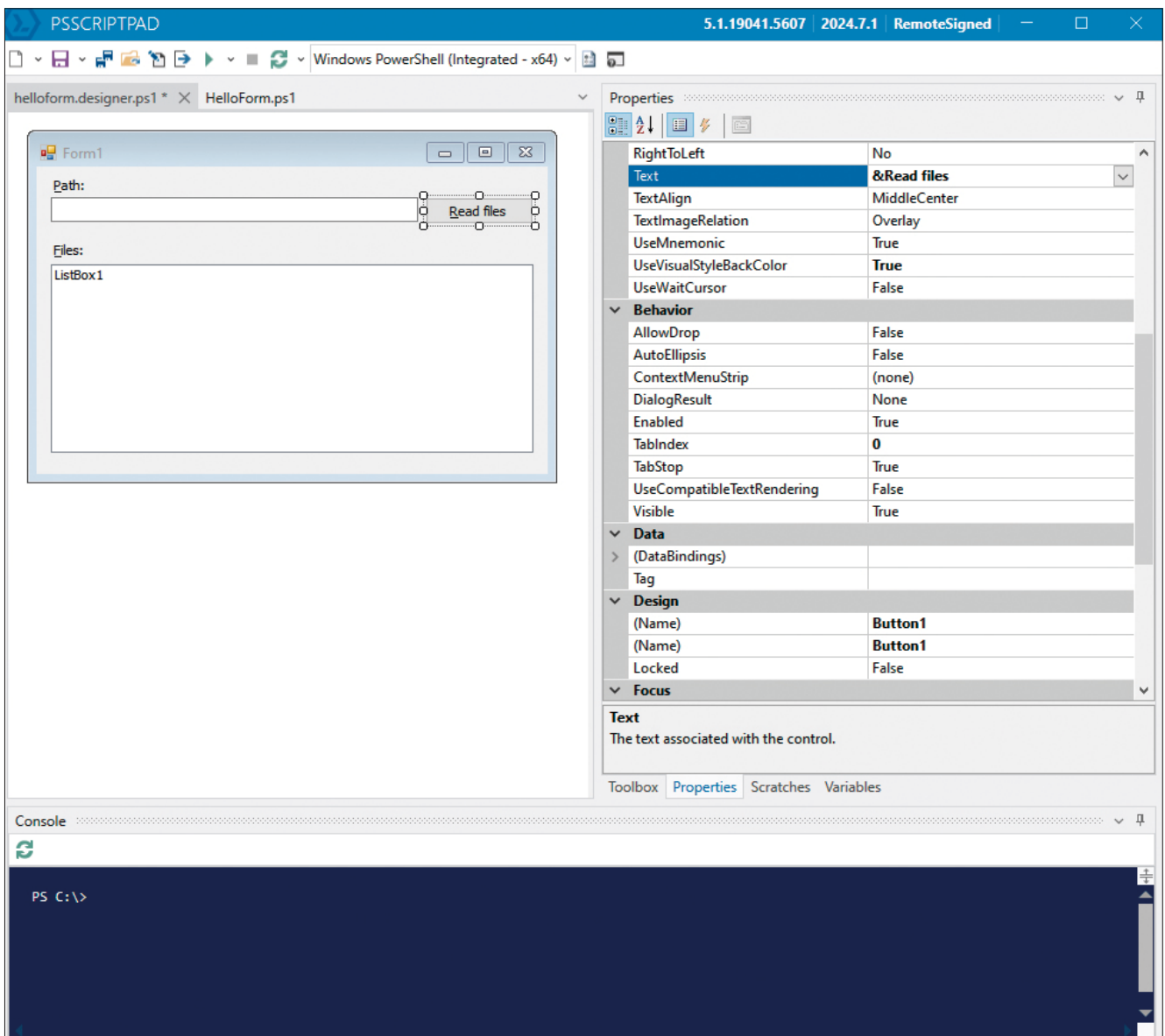
## MESSAGEBOXEN UND DIALOGS MIT POWERSHELL DARSTELLEN

# PowerShell im Dialog

Wenn Interaktion mit dem Anwender gefragt ist, kommt man mit dem Skriptfenster schnell an Grenzen. Dank .NET-Integration klappt die Anzeige von Dialogen dennoch.

Das „Shell“ in PowerShell macht es deutlich: Von Haus aus beschränkt sich die Textausgabe von PowerShell-Skripten auf ein schlichtes Konsolenfenster. Durch unterschiedliche Schriftfarbe und farbige Hintergründe ist das Setzen von Akzenten möglich. Will man aber mit der Person vor dem Bildschirm interagieren, so gehen Eingabeaufforderungen

schnell in der Textwüste unter. .NET-Integration kann hier Abhilfe schaffen. Mit WinForms und WPF bietet das .NET Framework bewährte GUI-Bibliotheken, deren Klassen und Methoden auch aus PowerShell-Skripten heraus angesprochen werden können. Hier reichen die Möglichkeiten von der Anzeige von simplen Nachrichten und Abfragen in einer ►



Das kostenlose Tool **PsScriptPad** ermöglicht das Anlegen von WinForms-GUIs in PowerShell. Auch die Eigenschaften von Komponenten und Ereignisbehandlungsroutinen lassen sich bequem bearbeiten (Bild 1)



### ● Listing 1: Beispieldialog mit MessageBox

```
#REQUIRES -version 2.0

Set-StrictMode -Version Latest

if ($Host.Name -ne "ConsoleHost")
{
    # Führt sonst zu Skript-Abbrüchen, wenn
    # Umgebungsvariablen nicht sauber
    # definiert sind (z.B. "LIB contains invalid path")
    Set-StrictMode -Version 2.0
} #if

Clear

# Load the Winforms assembly
[void] [reflection.assembly]::LoadWithPartialName(
    "System.Windows.Forms")

$Title = "Text Edit"
$Text = "The program will terminate now.

Save changes?"
$Icon =
[System.Windows.Forms.MessageBoxIcon]::Question
$Buttons =
[System.Windows.Forms.MessageBoxButtons]::YesNoCancel

# Show the MessageBox by using the parameters
# specified above
$Message = [System.Windows.Forms.MessageBox]::Show(
    $Text, $Title, $Buttons, $Icon)

# Use Switch/Case to evaluate the return value
switch($Message)
{
    "Yes" {Write-Host "Answer = YES"}
    "No" {Write-Host "Answer = NO"}
    "OK" {Write-Host "Answer = OK"}
    "Cancel" {Write-Host "Answer = CANCEL"}
    Default {Write-Host "Unknown return value"}
}
```

*MessageBox*-Instanz bis zur Darstellung selbst gestalteter Dialoge mit beliebigen Kontrollelementen.

Die folgenden Beispiele verwenden Klassen und Methoden aus der WinForms-Bibliothek, eine Internet-Suche nach *PowerShell WPF* wird aber auch sehr viele Beispiele zur Interaktion von PowerShell-Skripten mit der WPF-Bibliothek aufzeigen, siehe etwa [1].

### MessageBox für einfache Abfragen

Die naheliegendste Klasse zur Interaktion mit dem Anwender ist sicherlich *MessageBox* aus der Assembly *System.Windows.Forms*. Die Gestaltungsmöglichkeiten sind hier eher bescheiden; im Fenster wird ein Dialogtext neben einer oder mehreren vordefinierten Schaltflächen angezeigt. Darüber hinaus kann man als Akzent eines von vier vorgegebenen Symbolen neben dem Text zeigen: Information, Warnung, Stopp sowie Fragezeichen. Die möglichen Werte zur Definition der angezeigten Schaltfläche sind in der Aufzählung

*MessageBoxButtons* zu finden, die für die Icons in *MessageBoxIcon*. Die möglichen Rückgabewerte von *MessageBox.Show* sind vom Typ *DialogResult*, die im Skript wie Zeichenketten mit doppelten Anführungszeichen dargestellt werden, wie es Listing 1 zeigt.

Von der Methode *MessageBox.Show* existieren rund ein Dutzend Varianten. So ist es auch möglich, nur eine Zeichenkette zu übergeben und sich die weiteren Parameter zu sparen. In diesem Fall wird nur der Text mit einer Schaltfläche OK angezeigt, Dialog-Überschrift und Symbol fehlen. Um im dargestellten Text einen Zeilenumbruch zu erzwingen, genügt es, die Zeichenfolge `\n` einzufügen – man beachte den dem „n“ vorangestellten Akzent. Diese Zeichenfolge wird vergleichbar `\n` in C und verwandten Sprachen ausgewertet.

### Auf Benutzereingaben warten

Auch die Eingabe von Text durch den Anwender ist möglich. Hierzu kann die Funktion *InputBox* aus dem Namensraum

### ● Listing 2: Anwendungsbeispiel für die Funktion InputBox

```
# Show input box popup.

Clear

Add-Type -AssemblyName Microsoft.VisualBasic

$inputText = [Microsoft.VisualBasic.Interaction]::

InputBox("Enter some value:", `
    "Window Title", `
    "Default value")

if ($inputText.Length -eq 0)
{ "No text received" }
else { "Text = $inputText" }
```

## ● PowerShell und High DPI

Auf hochauflösenden Monitoren mit Skalierungsfaktoren größer 100 Prozent passt Windows in PowerShell erzeugte Fenster selbsttätig an diese Skalierung an, was zu unscharf wirkender Schrift und generell „pixelig“ wirkendem Aussehen führt. Um diese Effekte zu umgehen, müssen im Skript Vorkehrungen getroffen werden.

Ein Aufruf der Funktion `SetProcessDPIAware` aus dem Windows-API teilt dem System mit, dass das Skript mit hochauflösenden Monitoren umgehen kann. Die Funktion ist seit Windows Vista/Server 2008 verfügbar und muss aufgerufen werden, bevor das Programm irgendwelche GUI-Elemente anzeigt.

Nutzt das Skript nur die `MessageBox`-Funktion, genügt das.

**Bild 3** verdeutlicht die Unterschiede in der Darstellung. Zeigt das Skript hingegen ein WinForms-Fenster mit eingebetteten Kontrollelementen an, so ist ein deutlich größerer Aufwand erforderlich. Das folgende Skript zeigt, welche Funktionsaufrufe notwendig sind, um eine angepasste Anzeige zu erreichen:

```
#REQUIRES -version 2.0
```

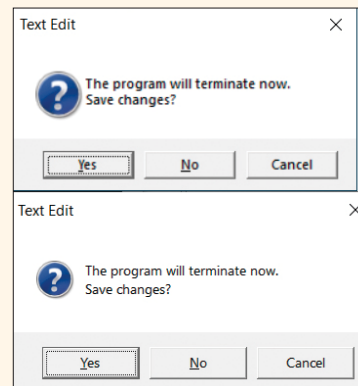
```
Set-StrictMode -Version Latest
```

```
Add-Type -Name User32 -Namespace WinApi
-MemberDefinition @"
    [DllImport("user32.dll", SetLastError=true)]
    public static extern bool SetProcessDPIAware();
"@
```

```
Clear
```

```
$null = [WinApi.User32]::SetProcessDPIAware()
```

```
# Load the Winforms assembly
[void] [reflection.assembly]::LoadWithPartialName(
    "System.Windows.Forms")
```



**Die obere MessageBox** zeigt das Aussehen bei einer Skalierung von 150 Prozent, wenn kein voriger Aufruf von `SetProcessDPIAware` erfolgt ist. Die untere MessageBox wird nach diesem Aufruf hingegen deutlich schärfer dargestellt (**Bild 3**)

```
$Title = "Text Edit"
$Text =
    "The program will terminate now.`nSave changes?"
$Icon =
    [System.Windows.Forms.MessageBoxIcon]::Question
$Buttons =
    [System.Windows.Forms.MessageBoxButtons]::YesNoCancel

# Show the MessageBox by using the parameters
# specified above
$Message = [System.Windows.Forms.MessageBox]::Show(
    $Text, $Title, $Buttons, $Icon)

# Use Switch/Case to evaluate the return value
switch($Message)
{
    "Yes" {Write-Host "Answer = YES"}
    "No" {Write-Host "Answer = NO"}
    "OK" {Write-Host "Answer = OK"}
    "Cancel" {Write-Host "Answer = CANCEL"}
    Default {Write-Host "Unknown return value"}
}
```

*Microsoft.VisualBasic.Interaction* verwendet werden. *InputBox* erwartet als Parameter mindestens einen Fragetext. Optional kann man auch noch das Eingabefeld mit einem Text eigener Wahl vorbelegen.

Nach Klick auf OK gibt die Funktion die Benutzereingabe zurück. War im Feld *DefaultResponse* ein vorgegebener Wert definiert, der nicht geändert wurde, so ist das dann auch das Ergebnis. Es ist auch möglich, den Dialog mit OK zu schließen, wenn das Eingabefeld leer ist. Um diesen Fall abzufangen, sollte man die Länge des zurückgegebenen Textes überprüfen. Ist sie 0, dann war entweder das Eingabefeld leer oder der Dialog wurde durch Drücken des *Cancel*-Buttons geschlossen.

**Listing 2** zeigt ein Beispiel zur Nutzung der Funktion. Im Netz finden Sie auch Beispiele, die demonstrieren, wie man etwa aus einem PowerShell-Skript heraus Systemdia-

log zur Auswahl einer Datei oder eines Verzeichnisses aufruft, so unter [2]. In **Listing 3** sehen Sie einige Beispiele.

### Alle Möglichkeiten mit WinForms

Die hohe Schule beim Gestalten eigener Dialoge stellt die Anzeige von Fenstern mithilfe der verschiedenen WinForms-Kontrollelemente dar. Aus einem PowerShell-Skript heraus hat man hier praktisch die gleichen Möglichkeiten wie aus C# oder einer anderen .NET-Sprache. Allerdings muss alles „zu Fuß“, also im Code, erledigt werden. Von Haus aus hat PowerShell keinen visuellen Formdesigner vergleichbar dem in Visual Studio, doch gibt es von verschiedenen Anbietern Designer für WinForms- und WPF-Formulare, die den notwendigen PowerShell-Code generieren, siehe [3] bis [5].

Seit vergangenem Jahr ist das vormals kostenpflichtige Produkt PowerShell Pro Tools [6] als kostenloses Add-in ►

### ● Listing 3: Systemdialoge aufrufen

```
# https://www.mariotti.de/powershell-dialog/

# Show an Open File Dialog and return the file
# selected by the user.
function Read-OpenFileDialog(
    [string]$InitialDirectory,
    [switch]$AllowMultiSelect)
{
    Add-Type -AssemblyName System.Windows.Forms
    $openFileDialog =
        New-Object System.Windows.Forms.OpenFileDialog
    $openFileDialog.initialDirectory = $InitialDirectory
    $openFileDialog.filter = "All files (*.*)| *.*"
    if ($AllowMultiSelect) { $openFileDialog.MultiSelect
        = $true }
    $openFileDialog.ShowDialog() > $null
    if ($allowMultiSelect)
        { return $openFileDialog.FileNames }

    else { return $openFileDialog.FileName }
}
Read-OpenFileDialog C:\Temp -AllowMultiSelect

# Show an Open Folder Dialog and return the directory
# selected by the user.
function Read-FolderBrowseDialog(
    [string]$InitialDirectory)
{
    Add-Type -AssemblyName System.Windows.Forms
    $openFolderDialog =
        New-Object System.Windows.Forms.FolderBrowserDialog
    $openFolderDialog.ShowNewFolderButton = $true
    $openFolderDialog.RootFolder = $InitialDirectory
    $openFolderDialog.ShowDialog()
    return $openFolderDialog.SelectedPath
}
Read-FolderBrowseDialog Desktop
```

### ● Listing 4: WinForms-Kontrollelemente nutzen

```
Set-StrictMode -Version Latest

Clear-Host

Add-Type -Name Kernel32 -Namespace WinApi
    -MemberDefinition '
    [DllImport("kernel32.dll")]
    public static extern IntPtr GetConsoleWindow();
    '

Add-Type -Name User32 -Namespace WinApi
    -MemberDefinition '
    [DllImport("user32.dll")]
    public static extern bool ShowWindow(
        IntPtr hWnd, Int32 nCmdShow);
    '

function PS_Hide-Show-Console
{ # Function to hide and restore the PowerShell
    console window if it exists
    if (Test-Path variable:global:_hConsoleWnd)
        # Variable exists
    {
        if ($_hConsoleWnd -ne 0) # Handle saved ->
            show the console window
        { [Void] ([WinApi.User32]::ShowWindow(
            $_hConsoleWnd, 1)) }
    } #if
    else

        { # Variable does not exist -> Hide the console
            # window
            $PSConsole = [WinApi.Kernel32]::GetConsoleWindow()

            if ($PSConsole -ne 0) # Valid window handle
                retrieved
            { [Void] ([WinApi.User32]::ShowWindow(
                $PSConsole, 0)) }

            # Save the console window handle in a global,
            # readonly variable
            New-Variable -Name _hConsoleWnd -Scope Script
                -Option ReadOnly -force `
                -Value $PSConsole
            } #else
        } #PS_Hide-Show-Console

    # Load the Winforms assembly
    [void][reflection.assembly]::LoadWithPartialName(
        "System.Windows.Forms")

    # Create the form
    $MsgWindowForm = New-Object Windows.Forms.Form

    #Set the dialog title
    $MsgWindowForm.Text = "Message Window"
    $MsgWindowForm.StartPosition = 'CenterScreen'

    # Create the listbox control and set text, size and
```

für Visual Studio Code erhältlich, das auch einen Formular-designer für WinForms enthält. Dessen Anwendung und Integration in Visual Studio Code ist in den Augen des Autors allerdings eher gewöhnungsbedürftig.

Dieselbe Firma bietet aber auch das kostenfreie Produkt PsScriptPad [7] zum Download an. Dabei handelt es sich um eine Entwicklungsumgebung für PowerShell-Skripte, die einen Editor für WinForms- wie auch WPF-Formulare enthält. Das Programm bietet eine wesentlich bessere Integration der Formulareditoren im Vergleich zu der in Visual Studio Code, wie in **Bild 1** zu sehen ist.

Eine Alternative ist, das Formular im Dialogdesigner von Visual Studio anzulegen, die Dialogelemente darin zu platzieren, deren Eigenschaften soweit möglich zu setzen und die von der IDE generierten Instruktionen händisch nach PowerShell zu übertragen. Welche Syntax PowerShell erwartet, um speziell die Eigenschaften eines Kontrollelements zu modifizieren, ist dabei aber nicht immer offensichtlich.

Bei Nutzung von WPF statt WinForms wird eine XAML-Datei benötigt, in der die visuellen Elemente definiert sind. Diese Datei kann man mithilfe eines Dialogeditors generieren, wie er in Visual Studio integriert ist.

## ● Das Konsolenfenster verbergen

PowerShell-Skripte laufen normalerweise in einem Konsolenfenster. Zeigt ein Skript aber ein Fenster an, ist das Konsolenfenster oft störend und sollte nicht sichtbar sein.

Mithilfe einiger Funktionen aus dem Windows-API lässt sich das Fenster jedoch recht einfach verbergen. Maßgeblich sind hier die Funktion *GetConsoleWindow*, um das Handle der PowerShell-Konsolenfensters zu erhalten, und *ShowWindow*, um das Fenster anschließend zu verbergen.

Die Funktion *PS\_Hide-Show-Console* in **Listing 4** zeigt die Anwendung der Funktionen. Beim ersten Aufruf der Funktion wird das Konsolenfenster verborgen, beim zweiten wieder angezeigt.

**Listing 4** zeigt ein einfaches Beispiel, wie man ein Fenster mit einem Kontrollelement anlegt, einen *Timer* startet und auf Ereignisse reagiert. Das Vorgehen folgt dabei in der Regel dem folgenden Muster: ►

```
location
$Message_LB = New-Object Windows.Forms.ListBox

$Message_LB.Anchor = 'Top,Left,Bottom,Right'
$Message_LB.Width = $MsgWindowForm.Width
$Message_LB.Height = $MsgWindowForm.Height
$Message_LB.ScrollAlwaysVisible = $true

# Add the controls to the Form
$MsgWindowForm.Controls.Add($Message_LB)

$MsgWindowForm_OnFormClosing = { param($sender,$e)
    $result = [System.Windows.Forms.MessageBox]::Show(
        "Are you sure you want to exit?", `
        "Close", [System.Windows.Forms.
        MessageBoxButtons]::YesNo)
    if ($result -ne
        [System.Windows.Forms.DialogResult]::Yes)
    { $e.Cancel= $true }
} # $MsgWindowForm_OnFormClosing

# Assign the event handler
$MsgWindowForm.Add_FormClosing(
    $MsgWindowForm_OnFormClosing)

$timer = New-Object Windows.Forms.Timer

$timer.Tag = 0
$timer.Interval = 1000

$timer_OnTick = {
    $Message_LB.SelectedIndex =
        $Message_LB.Items.Add($timer.Tag)
    $Message_LB.TopIndex = $Message_LB.SelectedIndex
    $timer.Tag = $timer.Tag + 1

    if ($timer.Tag -eq 10)
    { $MsgWindowForm.Remove_FormClosing(
        $MsgWindowForm_OnFormClosing) }
} # $timer_OnTick

# Assign the event handler
$timer.add_tick($timer_OnTick)

PS_Hide-Show-Console # Hide the console window if it
                    # exists

$timer.Start()

# Display the dialog
[Void] $MsgWindowForm.ShowDialog()

$timer.Enabled = $false
$timer.Dispose()
$MsgWindowForm.Dispose()

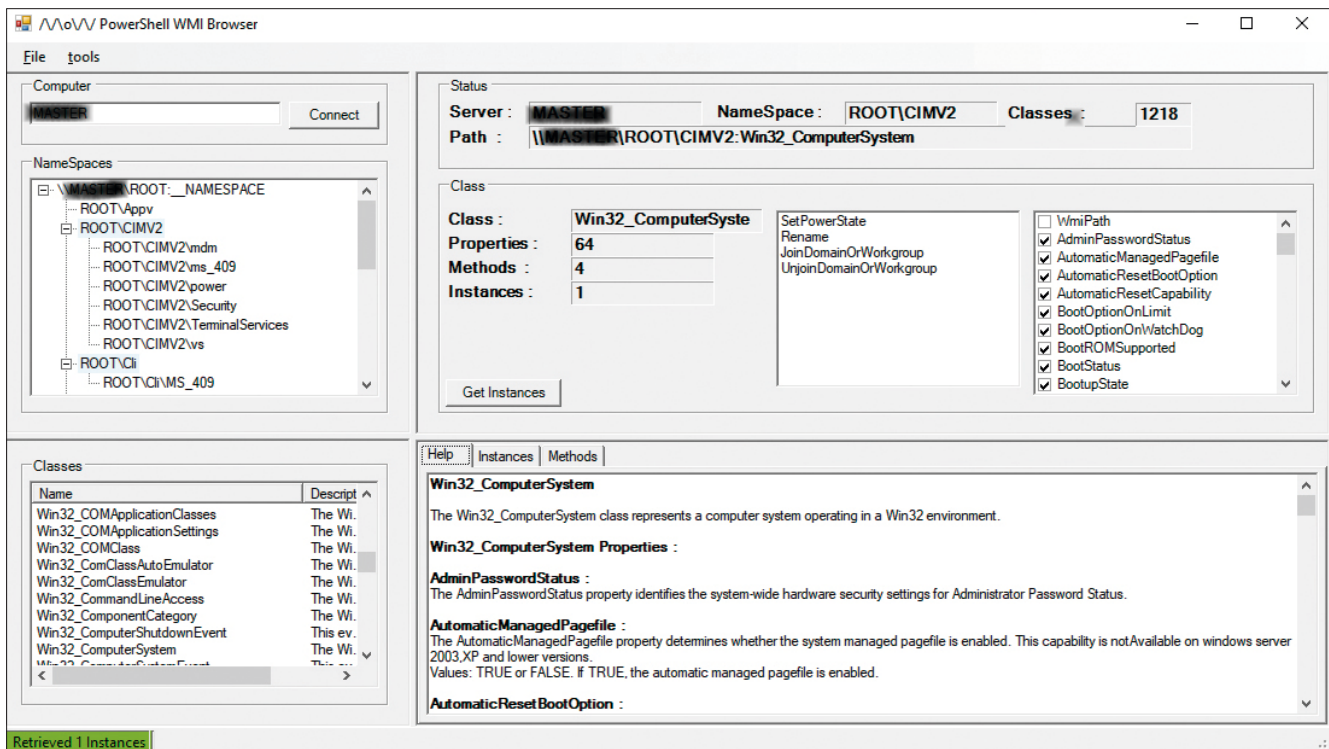
PS_Hide-Show-Console # Restore a hidden console
                    # window
```



- Anlegen des Hauptfensters
- Hinzufügen von visuellen und nicht visuellen Komponenten zum Fenster
- Setzen der Eigenschaften des Fensters und der darin platzierten Komponenten
- Definieren von Ereignisbehandlungsroutinen und Verknüpfen dieser mit dem Fenster und den sonstigen Komponenten
- Anzeigen des Fensters oder Dialogs

```
$Message_LB = New-Object Windows.Forms.ListBox
$Message_LB.Anchor = 'Top,Left,Bottom,Right'
$Message_LB.Width = $MsgWindowForm.Width
$Message_LB.Height = $MsgWindowForm.Height
$Message_LB.ScrollAlwaysVisible = $true
$MsgWindowForm.Controls.Add($Message_LB)
```

Um das Fenster schließlich anzuzeigen, genügt folgender Aufruf:



Der WMI-Explorer ist ein Beispiel für eine in PowerShell geschriebene Anwendung mit einem GUI, die von einer „normalen“ Anwendung nicht zu unterscheiden ist (Bild 2)

## Die Schritte im Einzelnen

Vor dem Zugriff auf Objekte muss die jeweilige Assembly geladen werden. Werden nur Windows-Controls verwendet, genügt dieser Aufruf:

```
[void][reflection.assembly]::LoadWithPartialName(
    "System.Windows.Forms")
```

Die folgenden drei Zeilen legen anschließend das Fenster-Objekt an, setzen den Fenstertitel und positionieren das Fenster in der Mitte des Bildschirms. Das Fenster ist zu diesem Zeitpunkt allerdings noch nicht sichtbar.

```
$MsgWindowForm = New-Object Windows.Forms.Form
$MsgWindowForm.Text = "Message Window"
$MsgWindowForm.StartPosition = 'CenterScreen'
```

Im Hauptformular der Anwendung soll eine Listbox zu sehen sein, die das Fenster komplett ausfüllt:

```
[Void] $MsgWindowForm.ShowDialog()
```

Die Zuweisung von Werten an Eigenschaften hat eine in Teilen etwas eigentümliche Syntax. Zahlen und boolesche Variablen können direkt zugewiesen werden, Zeichenketten werden in doppelte Anführungszeichen gesetzt. Um Werte an Aufzählungen sowie Datensätze zuzuweisen, müssen diese hingegen mit einfachen Anführungszeichen umfasst werden.

## Auf Ereignisse reagieren

Mittels Ereignisbehandlungsmethoden kann das Programm sowohl auf Aktionen des Anwenders, wie das Drücken einer Taste, als auch auf Ereignisse im Programm reagieren, zum Beispiel wenn das Programm den Fokus erhält oder verliert. Die Methode wird als normale PowerShell-Funktion definiert. Wichtig ist, dass ihre Parameter denen entsprechen, die im .NET Framework dafür definiert wurden. So muss die Methode zur Behandlung des Ereignisses *FormClosing* eines WinForms-Fensters in C# wie folgt aufgebaut sein:

```
public delegate void FormClosingEventHandler
(Object sender, FormClosingEventArgs e)
```

Nach PowerShell übertragen sieht die Methode wie folgt aus:

```
$MsgWindowForm_OnFormClosing = { param($sender,$e)
...
}
```

Die Verknüpfung der PowerShell-Funktion mit dem Ereignis erfolgt schließlich über den Aufruf

```
$MsgWindowForm.Add_FormClosing
($MsgWindowForm_OnFormClosing)
```

Die für die Verknüpfung der verschiedenen Ereignisse mit den jeweiligen Behandlungsmethoden zuständigen Funktionen haben immer den Namen *Add\_xxx*, wobei *xxx* durch den Namen des Ereignisses zu ersetzen ist. Da es hier auf die exakte Schreibung des Namens ankommt, sollte man diesen entweder aus einer vom Visual-Studio-Forms-Designer generierten Datei herauskopieren oder aus der Online-Hilfe zu einer Komponente.

Um die Verknüpfung zwischen einem Ereignis und seiner Behandlungsroutine aufzuheben, genügt ein Aufruf der Methode *Remove\_xxx*, wobei *xxx* wiederum durch den Namen des Ereignisses zu ersetzen ist. Auch hier muss wieder der exakte Name angegeben werden. Als Parameter übernehmen die *Remove\_-*Methoden den Namen der Ereignisbehandlungsfunktion.

Um die weiter oben zugewiesene Ereignisbehandlungsmethode *\$MsgWindowForm\_OnFormClosing* vom Ereignis zu trennen, ist daher folgender Aufruf notwendig:

```
$MsgWindowForm.Remove_FormClosing(
$MsgWindowForm_OnFormClosing)
```

## Den Dialog anzeigen

Nachdem die benötigten Objekte angelegt und befüllt und die Ereignisbehandlungsroutinen definiert wurden, kann das Fenster schließlich angezeigt werden. Der Aufruf ist eher unspektakulär:

```
[Void] $MsgWindowForm.ShowDialog()
```

Das vorangestellte *[Void]* unterdrückt Ausgaben in der PowerShell-Konsole. Die Nutzung von *ShowDialog* führt dazu, dass die darauffolgenden Befehle im Skript erst ausgeführt werden, nachdem der Dialog geschlossen wurde; *ShowDialog* blockiert also den Programmfluss.

Wenn das Skript auf Eingaben des Anwenders warten soll, die einen Einfluss auf die weiteren Operationen des Skripts haben, ist das auch so in Ordnung. Anders sieht es aber aus, wenn im Hintergrund Operationen laufen sollen, deren Ergebnis im Dialog dargestellt werden soll; zum Beispiel das Einlesen des Inhalts eines Ordners mit Ausgabe der Dateinamen in einer Listbox. Wenn das zu lange dauert, friert das GUI

ein und in der Titelzeile ist eine Meldung zu sehen, dass das Programm nicht mehr reagiert. Um das zu verhindern, sollte das Skript Nebenläufigkeit in Form von Runspaces nutzen. Dabei handelt es sich um einen Mechanismus, der die Ausführung von Skriptbefehlen in unterschiedlichen Ausführungsfäden (Threads) erlaubt. Das gehört allerdings zu den komplexeren Themen in PowerShell und würde den Rahmen dieses Artikels sprengen.

## Fazit

Solange es in der Hauptsache um das Abfragen beziehungsweise die Anzeige von Informationen geht, sollten Sie mit den Handreichungen aus diesem Artikel gut klarkommen. Im Netz finden sich auch komplexere Anwendungen mit einer komplett in PowerShell realisierten Oberfläche. Ein Beispiel dafür ist der in PowerShell geschriebene WMI-Explorer, ein sehr nützliches Tool zum Umgang mit WMI, wie **Bild 2** zeigt [8].

Aus Sicht des Autors wird der Code einer solchen Anwendung aber schnell unübersichtlich, und man ist unter Umständen besser mit einem in C# geschriebenen Programm unterwegs, das punktuell PowerShell-Routinen aufruft. Einen Artikel, der näher darauf eingeht, finden Sie unter [9].

Der Beispielcode zum aktuellen Artikel steht vollständig unter [10] zum Download bereit. ■

- [1] PowerShell WPF GUI, [www.script-example.com/powershell-wpf-gui](http://www.script-example.com/powershell-wpf-gui)
- [2] Dialoge aus dem .NET-Framework in PowerShell aufrufen, [www.mariotti.de/powershell-dialog/](http://www.mariotti.de/powershell-dialog/)
- [3] Formdesigner für PowerShell, [poshgui.com/](http://poshgui.com/)
- [4] powershell-designer 2.7.7, [www.powershellgallery.com/packages/powershell-designer](http://www.powershellgallery.com/packages/powershell-designer)
- [5] Sapien Technologies, PowerShell Studio 2025, [www.sapien.com/software/powershell\\_studio](http://www.sapien.com/software/powershell_studio)
- [6] PowerShell Pro Tools von Ironman Software, <https://marketplace.visualstudio.com/items?itemName=ironmansoftware.powershellprotools>
- [7] PSScriptPad on Ironman Software, [ironmansoftware.com/release/psscriptpad](http://ironmansoftware.com/release/psscriptpad)
- [8] In PowerShell geschriebener WMI-Explorer, <https://jdhitsolutions.com/blog/powershell/2848/wmi-explorer-from-the-powershell-guy/>
- [9] Olaf Heß, Skripten mit Klasse, *dotnetpro* 1/2016, Seite 122 ff.
- [10] Beispielcode zum Artikel, <https://eu-assets.contentstack.com/v3/assets/blt22276bff283176bd/bltcf3d113d34fe85af/PowerShellGUI.zip>



**Olaf Heß**

arbeitet als IT-Spezialist für das Industrieunternehmen Knorr-Bremse. Er ist dort für die Installation und Konfiguration des Windows-Clients mitverantwortlich. Sie erreichen ihn unter [ohess@gmx.de](mailto:ohess@gmx.de).

## DEN PROJEKTZOO ORGANISIEREN

# Ordnung statt Chaos

Eigenschaften und Packages zentral verwalten.

Nahzu jedes Projekt beginnt als kleiner Prototyp. Doch schon nach einigen wenigen Iterationen wächst es zu einer beachtlichen Visual Studio-Solution mit Dutzenden oder gar Hunderten Projekten. Ab diesem Punkt wird es beschwerlich, bestimmte Eigenschaften des Projekts zu verwalten. Benötigt zum Beispiel ein NuGet-Paket ein Update, sind drei oder vier Projektdateien anzupassen. Erscheint sogar eine neue .NET-Version und das Projekt soll dafür aktualisiert werden, gilt es dies in allen Projekten zu hinterlegen. Für alle Entwickler, denen diese Probleme bekannt vorkommen, ist dieser Artikel genau richtig.

### Die Quelle des Übels heißt .csproj

Werfen wir gemeinsam zunächst einen Blick auf die mehr oder weniger typische Projektdatei (.csproj) aus [Listing 1](#). Diese Projektdatei gehört zu einer auf .NET 8 basierenden ASP.NET-Core-Anwendung und zeigt drei typische Problemstellen:

- Im Abschnitt *PropertyGroup* sind „harte“ Vorgaben für den Compiler enthalten, wie beispielsweise die zu verwendenden .NET- und C#-Versionen, aber auch eine Regel zum Umgang mit Null-Referenz-Typen.
- Ebenfalls in der *PropertyGroup* zu finden sind Metadaten, wie der Produkt- und der Firmenname. Gibt es eine größere Anzahl an Projekten, entsteht insbesondere bei diesen optionalen Angaben ein zunehmender Wildwuchs.
- Im darunter stehenden Abschnitt *ItemGroup* sind die referenzierten NuGet-Pakete inklusive Versionsnummer aufgelistet. Hat man mehrere Projekte, muss man alle Projekte aktualisieren.

Die gute Nachricht ist, dass MSBuild, genauer das .NET Build System sowie der Paketmanager NuGet, elegante Lösungen für all diese Probleme anbietet, die hier vorgestellt werden.

### Directory.Build.props

Der erste Schritt führt zur Datei *Directory.Build.props*. MSBuild sucht diese Datei, noch bevor ein Projekt aufgebaut wird, und interpretiert die darin definierten Werte als Standardwerte (*Defaults*). Innerhalb der jeweiligen .csproj-Datei können diese Standards später überschrieben werden. Der Aufbau von *Directory.Build.props* entspricht dem einer normalen Projektdatei.

Möchte man beispielsweise die Metainformationen, die Sprachversion und weitere Eigenschaften für alle Projekte vordefinieren, legt man zum Beispiel im Solution-Verzeichnis eine *Directory.Build.props*-Datei mit folgendem Inhalt an:

```
<Project>
  <PropertyGroup>
```

```
    <LangVersion>latest</LangVersion>
    <TreatWarningsAsErrors>true</TreatWarningsAsErrors>
    <Nullable>enable</Nullable>
    <AssemblyVersion>1.0.0.0</AssemblyVersion>
    <FileVersion>1.0.0.0</FileVersion>
    <InformationalVersion>1.0.0.0</InformationalVersion>
    <Company>Awesome Company</Company>
    <Product>Awesome Product</Product>
    <Copyright>...</Copyright>
  </PropertyGroup>
</Project>
```

Grundsätzlich kann man in der Datei *Directory.Build.props* auch auf andere MSBuild-Eigenschaften zugreifen, allerdings sollte man bedenken, dass die Datei sehr früh in der Verarbeitung geladen wird. Es ist aber wie folgt möglich, das Ausgabeverzeichnis (*OutDir*) zu setzen:

```
<Project>
  <PropertyGroup>
    <OutDir>C:\output\$(MSBuildProjectName)</OutDir>
  </PropertyGroup>
</Project>
```

### MSBuild-Suchlogik für Directory.Build.props

Die Datei *Directory.Build.props* wird von MSBuild immer dann gesucht, wenn ein Projekt gebaut wird. Dabei durchläuft MSBuild den Verzeichnispfad vom Projektordner aus rekursiv nach oben, bis zum Wurzelverzeichnis. Sobald eine *Directory.Build.props*-Datei gefunden wird, wird die Suche gestoppt.

Es gibt jedoch Möglichkeiten, mehrere geschachtelte *Directory.Build.props*-Dateien zu verwenden. Hier eine beispielhafte Struktur dafür:

```
\
MySolution.sln
Directory.Build.props    (1)
\src
  Directory.Build.props  (2-src)
  \Project1
  \Project2
\test
  Directory.Build.props  (2-test)
  \Project1Tests
  \Project2Tests
```

In diesem Beispiel gibt es eine globale *Directory.Build.props* im Wurzelverzeichnis (1), eine eigene Datei im *src*-Verzeich-

## Listing 1: Eine typische Projektdatei

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <LangVersion>13.0</LangVersion>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <AssemblyVersion>1.0.0.0</AssemblyVersion>
    <FileVersion>1.0.0.0</FileVersion>
    <InformationalVersion>1.0.0.0
    </InformationalVersion>
    <Company>Awesome Company</Company>
    <Product>Awesome Product</Product>
    <Copyright>...</Copyright>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="
      Swashbuckle.AspNetCore" Version="6.6.2" />
    <PackageReference Include="
      Microsoft.Extensions.Logging.Abstractions"
      Version="8.0.3" />
  </ItemGroup>
</Project>
```

nis (2-src) sowie eine weitere im *test*-Verzeichnis (2-test). Dieser Aufbau eignet sich zum Beispiel dafür, in allen produktiven Projekten (unter *src*) die Option *TreatWarningsAsErrors* zu aktivieren, während sie in den Testprojekten (unter *test*) deaktiviert bleibt.

Damit MSBuild die Suche nach *props*-Dateien nicht nach dem ersten Treffer stoppt, sondern auch die übergeordnete *Directory.Build.props* lädt, muss in den untergeordneten Dateien (2-src und 2-test) folgender Import-Befehl eingefügt werden:

```
<Import Project=
  "$([MSBuild]::GetPathOfFileAbove(
    'Directory.Build.props',
    '$(MSBuildThisFileDirectory)../'))"
  Condition="'$(MSBuild)' != '$([MSBuild]::GetPathOfFileAbove(
    'Directory.Build.props',
    '$(MSBuildThisFileDirectory)../'))'"
/>
```

## Directory.Build.targets

Das Gegenstück zur Datei *Directory.Build.props* heißt *Directory.Build.targets*. Sie wird von MSBuild nach dem Laden des Projekts eingebunden. Grundsätzlich ist es hier auch möglich, die Eigenschaften zu setzen, allerdings nutzt man die Datei in der Regel eher, um MSBuild-Targets zu erweitern oder anzupassen.

MSBuild hat eine Reihe von Targets wie etwa *Build*, *Clean*, *Pack* und *Test* definiert. Diese Aktionen kann man überschrei-

ben oder über die Einstiegspunkte *After...* und *Before...* andere Aktionen ausführen lassen, wie zum Beispiel ein schlichtes Logging:

```
<Target Name="BeforePack">
  <Message Text="Bereite NuGet-Packaging für
    $(MSBuildProjectName) vor" />
</Target>
```

Dies ist kein Alleinstellungsmerkmal der Datei *Directory.Build.targets*, sondern das Standardverhalten von MSBuild, das man auch in jeder einzelnen Projektdatei definieren kann. Die Datei *Directory.Build.targets* hilft allerdings, generelle Tasks zu automatisieren oder Eigenschaften „hart zu überschreiben“.

## Fehler und Warnungen deaktivieren

Da die Datei *Directory.Build.targets* am Ende des Build-Prozesses geladen wird, bietet sie eine gute Möglichkeit, bestimmte Eigenschaften gezielt zu überschreiben.

Hier ein typisches Szenario: Wir arbeiten mit einer großen Solution, die viele Projekte und entsprechend viele NuGet-Pakete enthält. Gelegentlich müssen wir auf einen älteren Branch zurückgreifen, um das Verhalten einer früheren Version zu reproduzieren – etwa zur Fehlersuche oder Analyse.

In solchen Fällen kann es passieren, dass beim Build plötzlich Sicherheitswarnungen von NuGet auftreten (etwa NU1901 bis NU1904), weil bekannte Schwachstellen in den referenzierten Paketen erkannt wurden. Ein Update dieser Abhängigkeiten im alten Branch wäre jedoch nicht sinnvoll, da der Zustand bewusst eingefroren ist.

Problematisch wird es, wenn Projekte so konfiguriert sind, dass Warnungen als Fehler behandelt werden (*<TreatWarningsAsErrors>true</TreatWarningsAsErrors>*). In der Folge schlägt der Build fehl, obwohl das Verhalten erwartbar und akzeptabel ist.

Die Lösung: Wir hinterlegen in der Datei *Directory.Build.targets* eine deaktivierbare Konfiguration, um Warnungen temporär zu unterdrücken:

```
<Project>
  <PropertyGroup>
    <!-- Mute all vulnerabilities and don't
      escalate warnings -->
    <!-- Uncomment these lines if you need to build
      an older version (with open vulnerabilities!):
    <TreatWarningsAsErrors>>false
    </TreatWarningsAsErrors>
    <NoWarn>NU1901, NU1902, NU1903, NU1904</NoWarn>
    -->
  </PropertyGroup>
</Project>
```

## NuGet-Paketmanagement

Um NuGet-Pakete zentral zu verwalten, gibt es die Datei *Directory.Packages.props*. Ihr Aufbau ähnelt dem der zuvor besprochenen Dateien: ▶



```
<Project>
  <PropertyGroup>
    <ManagePackageVersionsCentrally>true
  </ManagePackageVersionsCentrally>
</PropertyGroup>
<ItemGroup>
  <PackageVersion Include=
    "Newtonsoft.Json" Version="13.0.1" />
</ItemGroup>
</Project>
```

In dieser Datei pflegt man alle NuGet-Pakete mit Versionsnummern, welche in der Solution benötigt werden. In der eigentlichen Projektdatei ist dann nur noch die Referenz enthalten, allerdings ohne die Versionsnummer:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup> ...
</PropertyGroup>
<ItemGroup>
  <PackageReference Include="Newtonsoft.Json">
  </PackageReference>
</ItemGroup>
</Project>
```

In der Datei *Directory.Packages.props* kann man alle NuGet-Pakete angeben, die referenziert werden und als *PackageReference* im Projekt hinterlegt sind.

### Mehrstufige Nutzung:

Ähnlich wie zuvor anhand der Datei *Directory.Build.props* gezeigt, können Sie auch hier mit mehreren Ebenen der Datei *Directory.Packages.props* arbeiten, wobei die nächst höhere Ebene mithilfe des folgenden Befehls inkludiert wird:

```
<Import Project="$(
  [MSBuild]::GetPathOfFileAbove(
    Directory.Packages.props,
    $(MSBuildThisFileDirectory)...)"
/>
```

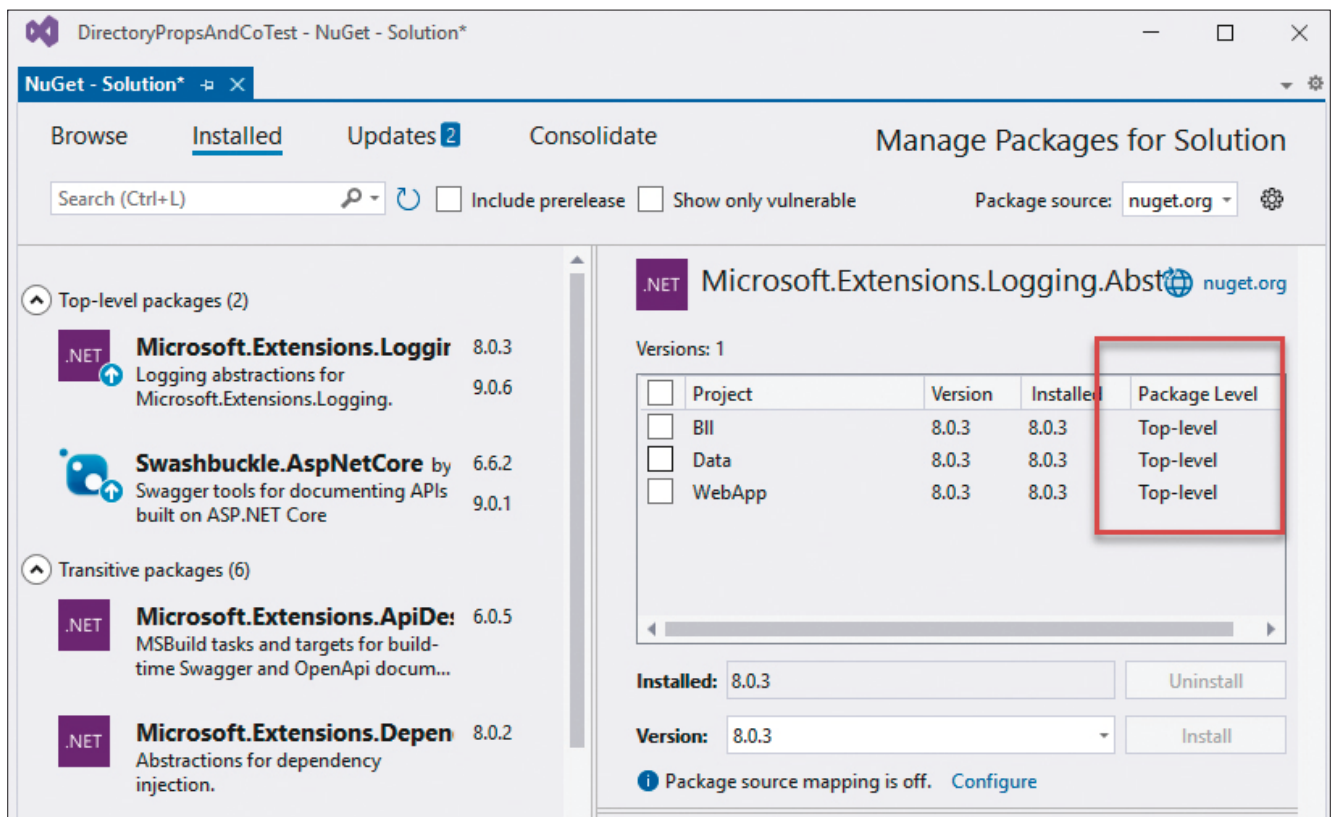
### Ausnahmen und Erweiterungen:

Es besteht zudem die Möglichkeit, in einzelnen Projekten das zentrale Paketmanagement gänzlich zu deaktivieren oder explizit andere Paketversionen zu erzwingen:

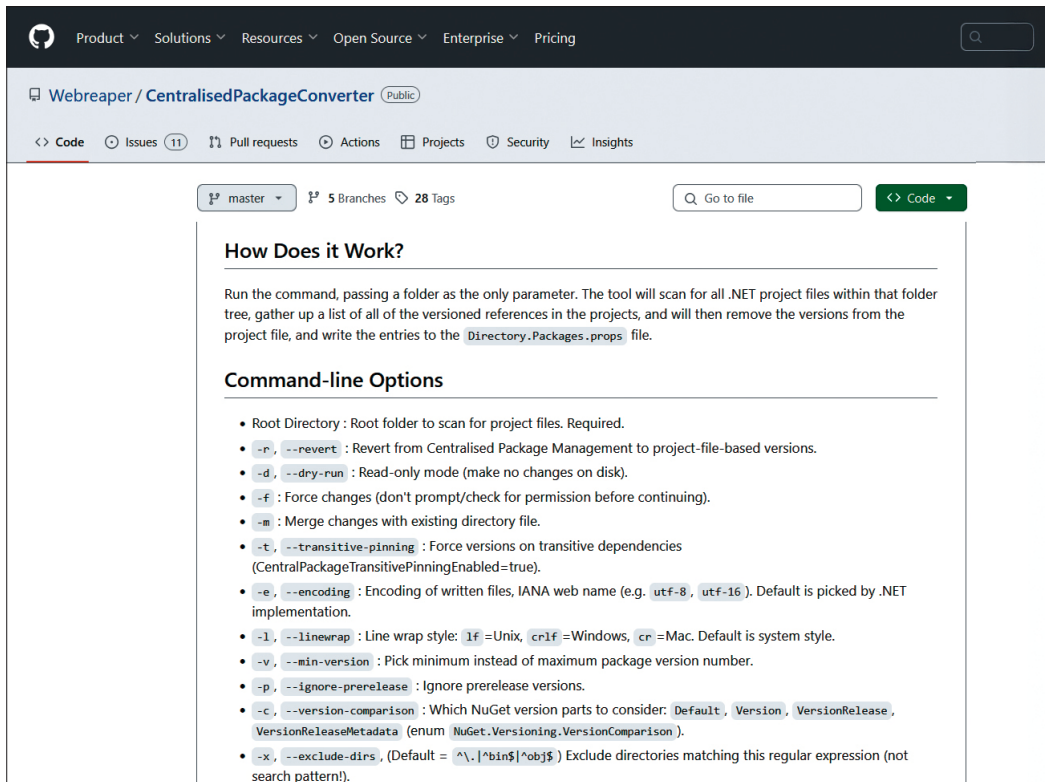
```
<PackageReference
  Include="Microsoft.Extensions.Logging"
  VersionOverride="9.0.6"
/>
```

Über das zentrale Paketmanagement lassen sich darüber hinaus noch weitere Einstellungen tätigen, sodass man beispielsweise globale Referenzen erzwingen kann:

```
<GlobalPackageReference
  Include="Microsoft.Extensions.Logging"
```



Der NuGet-Paketmanager zeigt beim Vorhandensein einer „Directory.Packages.props“-Datei die NuGet-Pakete mit dem Vermerk des „Package Level“ an (Bild 1)



**Das Open-Source-Tool**  
Centralised Package  
Converter auf GitHub  
(Bild 2)

```
Version="9.0.6"
/>
```

### Integration in Visual Studio:

Der NuGet Paketmanager in der Entwicklungsumgebung kennt die Datei ebenfalls und aktualisiert die zentrale Datei, wenn erforderlich. Außerdem wird dort der Paket-Level aller NuGet-Pakete angezeigt, wie in Bild 1 zu sehen ist.

### Schneller per CLI

Auch über das Command Line Interface (.NET CLI) gelingt der Einstieg flott. Mit den folgenden Befehlen lassen sich die zentralen Build- und Paketdateien erstellen:

```
dotnet new buildprops
dotnet new buildtargets
dotnet new packagesprops
```

Wer ein Werkzeug für das zentrale Paketmanagement sucht, kann sich den Centralised Package Converter [1] anschauen (Bild 2). Das Open-Source-Tool scannt die bestehenden Projektdateien, extrahiert die erforderlichen Infos und erstellt die Datei *Directory.Packages.props* automatisch. Installieren können Sie es mit folgenden Befehlen:

```
dotnet tool install CentralisedPackageConverter -global
central-pkg-converter /SomeAwesomeProject
```

### Fazit

Mit nur drei kleinen Dateien lässt sich auch ein umfangreicher Projektzoo deutlich besser strukturieren:

- *Directory.Build.props* erlaubt das zentrale Festlegen von Standardeigenschaften – etwa für Firmenname, Sprachversion oder Build-Konfigurationen.
- *Directory.Build.targets* gestattet den gezielten Eingriff in die Build-Pipeline – sei es zur Automatisierung oder um Warnungen beziehungsweise Fehler global zu steuern.
- *Directory.Packages.props* zentralisiert die Verwaltung von NuGet-Paketversionen und verhindert Versionswildwuchs. In [2] und [3] finden Sie weiterführende Infos zum Thema.

Auch der Aufwand für das Umwandeln einer bestehenden Lösung hält sich dank der drei Hilfsdateien in Grenzen, und man hat das gute Gefühl, aufgeräumt zu haben. ■

- [1] Centralised Package Converter auf GitHub, <https://github.com/Webreaper/CentralisedPackageConverter>  
 [2] Customize the build by folder, <https://learn.microsoft.com/en-us/visualstudio/msbuild/customize-by-directory>  
 [3] Zentrales Paketmanagement, <https://learn.microsoft.com/en-us/nuget/consume-packages/central-package-management>



**Robert Mühsig**

ist Softwareentwickler und arbeitet vornehmlich mit dem Microsoft-Technologie-Stack. Durch sein Community-Engagement – von User Groups bis hin zu Blogs – ist er seit mehr als 15 Jahren Microsoft MVP.  
[robert.muehsig@codeinside.eu](mailto:robert.muehsig@codeinside.eu)

## DEFAULT INTERFACE METHODS IN C#

# Eine Spracherweiterung auf dem Prüfstand

Implementierungen in C#-Interfaces lösten eine intensive Debatte aus: Pragmatische Nutzenargumente treffen auf Sorgen über die Verwässerung bewährter Designprinzipien.

**W**ir stehen in der Welt der Softwareentwicklung ständig vor der Herausforderung, Code so zu gestalten, dass er nicht nur seine Aufgabe erfüllt, sondern auch modular, wartbar und flexibel bleibt. Eine wiederkehrende Frage dabei ist, wie Funktionalitäten oder Dienste an verschiedenen Stellen in einer Anwendung leicht zugänglich gemacht werden können, ohne dabei unerwünschte, starre Abhängigkeiten zu schaffen.

Diese Überlegung führte in der Vergangenheit zur Entwicklung von Konzepten wie dem Method Aggregator, der in einem früheren Artikel vorgestellt wurde [1]. Die Idee war, eine zentrale Einheit zu schaffen, über die Methoden an verschiedenen Stellen im Code auf einfache Weise aufgerufen

werden können. Die Suche nach immer besseren Wegen zur Reduktion von Abhängigkeiten führte den Blick auf das Prinzip der Dependency Injection (DI) und den Einsatz von IoC-Containern. Um die Nutzung zu vereinfachen, wurde die Aufmerksamkeit auf eine Neuerung aus C# 8.0 gelenkt.

Die Default Interface Methods (DIMs) [2] sorgten für erheblichen Wirbel in der Entwicklergemeinschaft. Denn plötzlich war es möglich, Implementierungen direkt innerhalb von Interfaces zu schreiben – eine Funktionalität, die in C# bisher undenkbar schien und das traditionelle Bild des reinen „Vertrags“ auf den Kopf stellte. Einige Reaktionen darauf waren polarisierend [3]: Während die einen darin einen pragmatischen Schritt zur Vereinfachung der API-Evolution sahen und

### ● Listing 1: Einfaches Interface mit DIM und zwei Implementierungen, eine nutzt Standard, eine überschreibt

```
public interface ILogger
{
    void Log(string message);

    // Eine Default Interface Method
    void LogWarning(string message)
    {
        // Standardimplementierung
        Log($"WARNING: {message}");
    }
}

public class ConsoleLogger : ILogger
{
    public void Log(string message)
    {
        Console.WriteLine(message);
    }
    // LogWarning muss hier nicht implementiert werden,
    // da eine Standardimplementierung im Interface
    // existiert.
}

public class FileLogger : ILogger
{
    private readonly string _filePath;

    public FileLogger(string filePath)
    {
        _filePath = filePath;
    }

    public void Log(string message)
    {
        File.AppendAllText(_filePath, message +
            Environment.NewLine);
    }

    // Die Default-Implementierung kann überschrieben
    // werden
    public void LogWarning(string message)
    {
        File.AppendAllText(_filePath,
            $"[W] {message}{Environment.NewLine}");
    }
}
```

es als nützlichen „Syntactic Sugar“ begrüßten, beäugten andere die Neuerung mit Skepsis. Sie sahen darin eine Verwässerung des Interface-Konzepts, ein potenzielles „Design-Antipattern“, das die klare Trennung zu abstrakten Klassen aufhob und Entwickler dazu verleiten könnte, unsauberen Code zu schreiben.

## DIMs als sinnvolle Ergänzung des Sprachumfangs oder Risiko für das Softwaredesign?

Aus dem Wunsch, eine Alternative für den Method Aggregator zu schaffen sowie die Potenziale des neuen Sprachfeatures im Kontext robuster Architekturmuster auszuloten, entstand die Idee für das Beispielpunkt, das in diesem Artikel vorgestellt wird. Dieses zeigt auf, wann und wie DIMs sinnvoll einsetzbar sind. Zunächst beleuchtet dieser Artikel die offiziellen Gründe für die Einführung der DIMs und stellt die Hauptkritikpunkte aus der Community dar. Danach wendet sich der Text dem Beispiel zu. Es wird untersucht, wie das gewählte Designmuster funktioniert und inwiefern es einen strukturierten Rahmen für den Einsatz von DIMs bieten kann. Ziel ist es, zu analysieren, wie das Projekt dazu beitragen kann, die kritischen Sorgen zu adressieren, und zu zeigen, dass der Wert eines Sprachfeatures maßgeblich davon abhängt, wie es innerhalb einer Architektur und mithilfe geeigneter Muster eingesetzt wird.

Formal erlauben DIMs es, einer Methode innerhalb eines Interfaces eine Standardimplementierung zu geben. Vor C# 8.0 konnten Interfaces lediglich Methodensignaturen, Properties, Events und Indexers deklarieren – reine „Verträge“, die von implementierenden Klassen vollständig erfüllt werden mussten. Mit DIMs ändert sich das grundlegend (Listing 1).

Wichtig zu verstehen ist, dass Interfaces mit DIMs keinen Zustand speichern können. Sie können keine Instanzfelder oder auto-implementierte Properties besitzen. Implementierungen von DIMs können auf andere Member des Interfaces zugreifen, auch auf abstrakte Methoden, jedoch nicht auf den Zustand der implementierenden Klasse, es sei denn, dieser Zustand wird über Interface-Member exponiert. Interfaces bleiben also konzeptionell zustandslos, auch wenn sie Verhalten enthalten können. Dies ist ein entscheidender Unterschied zu abstrakten Klassen, die Zustände halten dürfen.

Die Hauptmotivation für die Einführung von DIMs aus Sicht des C#-Sprachteams war die Erleichterung der API-Evolution. Stellen Sie sich vor, Sie sind der Autor einer weit verbreiteten Bibliothek, die zentrale Funktionalitäten über Interfaces anbietet. Fügen Sie einer bestehenden Schnittstelle eine neue Methode hinzu, würden alle existierenden Klas-

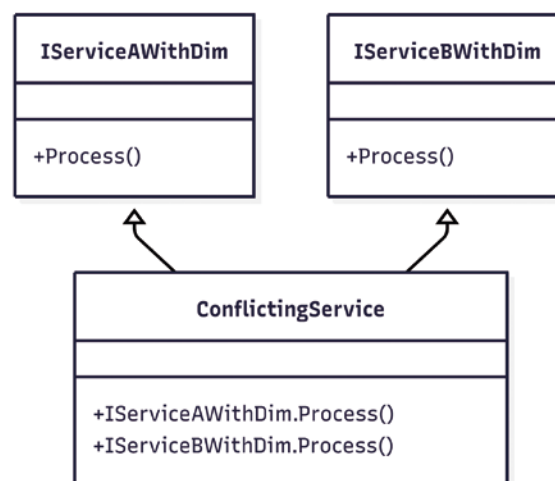
sen, die dieses Interface implementieren, plötzlich nicht mehr kompilieren. Dies stellt einen Breaking Change dar und erfordert oft umfangreiche Anpassungen im Code der Nutzer der Bibliothek. Default Interface Methods lösen dieses Problem: Eine neue Methode kann mit einer Standardimplementierung zum Interface hinzugefügt werden, und bestehende Implementierungen, die dieses Standardverhalten nutzen möchten, funktionieren ohne Code-Änderungen weiter.

Ein weiterer genannter Grund für die Einführung der DIMs ist die Unterstützung von Konzepten, die in anderen Sprachen als Traits oder Mixins bekannt sind. Dabei handelt es sich um Mechanismen zur Code-Wiederverwendung, die es erlauben, Verhaltensweisen, also Methoden mit Implementierung, zu definieren und diese verschiedenen Klassen „einzumischen“, unabhängig von deren Platz in einer traditionellen Vererbungshierarchie. Stellen Sie sich ein Interface *ILoggable* vor, das eine DIM *LogActivity(string activity)* hat, die standardmäßig in eine Datei schreibt. Jede Klasse, die *ILoggable* implementiert, erhält dieses Verhalten „gratis“,

ohne selbst eine Vererbungshierarchie eingehen zu müssen. Dies fördert die Komposition über Vererbung und erlaubt die Wiederverwendung von Code auf eine neue Art und Weise in C#. Nicht zuletzt wird auch die Interoperabilität mit anderen Sprachen und Plattformen genannt, die ähnliche Konzepte bereits länger unterstützen, wie zum Beispiel Default Methods in Java oder Protocol Extensions in Swift.

Trotz der offensichtlichen Vorteile, insbesondere für Bibliotheksautoren und die Ermöglichung von Traits, stießen DIMs in der C#-Community teils auf heftige Kritik, die sich grob in drei Hauptpunkte unterteilen lässt:

- **Verwässerung des Interface-Konzepts:** Der wohl am häufigsten genannte Kritikpunkt. Traditionell waren Interfaces reine Verträge, Versprechen ohne jegliche Implementierung. Kritiker sehen in DIMs eine unzulässige Vermischung der Rollen von Interfaces und abstrakten Klassen. Sie befürchten, dass die klare Trennung aufgeweicht wird und Entwickler die eigentliche Bestimmung von Interfaces aus den Augen verlieren könnten. Ein Interface, das Implementierung enthält, fühlt sich für Puristen „falsch“ an.
- **Erhöhte Komplexität und das „Diamond Problem“:** Die Möglichkeit, dass eine Klasse mehrere Interfaces implementiert, die dieselbe Methode als DIM anbieten, führt zum bekannten „Diamond Problem“ aus der Mehrfachvererbung, das C# bei Klassenvererbung bewusst vermeidet. Einige sehen hier ein erhebliches Potenzial für Verwirrung und schwer durchschaubaren Code, der die Wartung erschwert. Welcher Code wird wann ausgeführt? Das kann bei komplexen Hierarchien unübersichtlich werden. ►



Interfaces mit DIMs und Implementierungen (Bild 1)



## Listing 2: Das Diamond-Problem und die C#-Auflösung durch explizite Implementierung

```
public interface IServiceAWithDim
{ void Process() { Console.WriteLine("A"); } }

public interface IServiceBWithDim
{ void Process() { Console.WriteLine("B"); } }

public class ConflictingService : IServiceAWithDim,
    IServiceBWithDim
{
    // Der Compiler erzwingt hier eine explizite
    // Implementierung, um den Konflikt aufzulösen,
    // da beide DIMs gleich spezifisch sind.
    void IServiceAWithDim.Process()
    {
        Console.WriteLine(
            "Calling A's Process via explicit
            implementation");
    }

    // Man kann die Default-Implementierung immer noch
    // explizit aufrufen, wenn nötig:
    // IServiceAWithDim.Process();

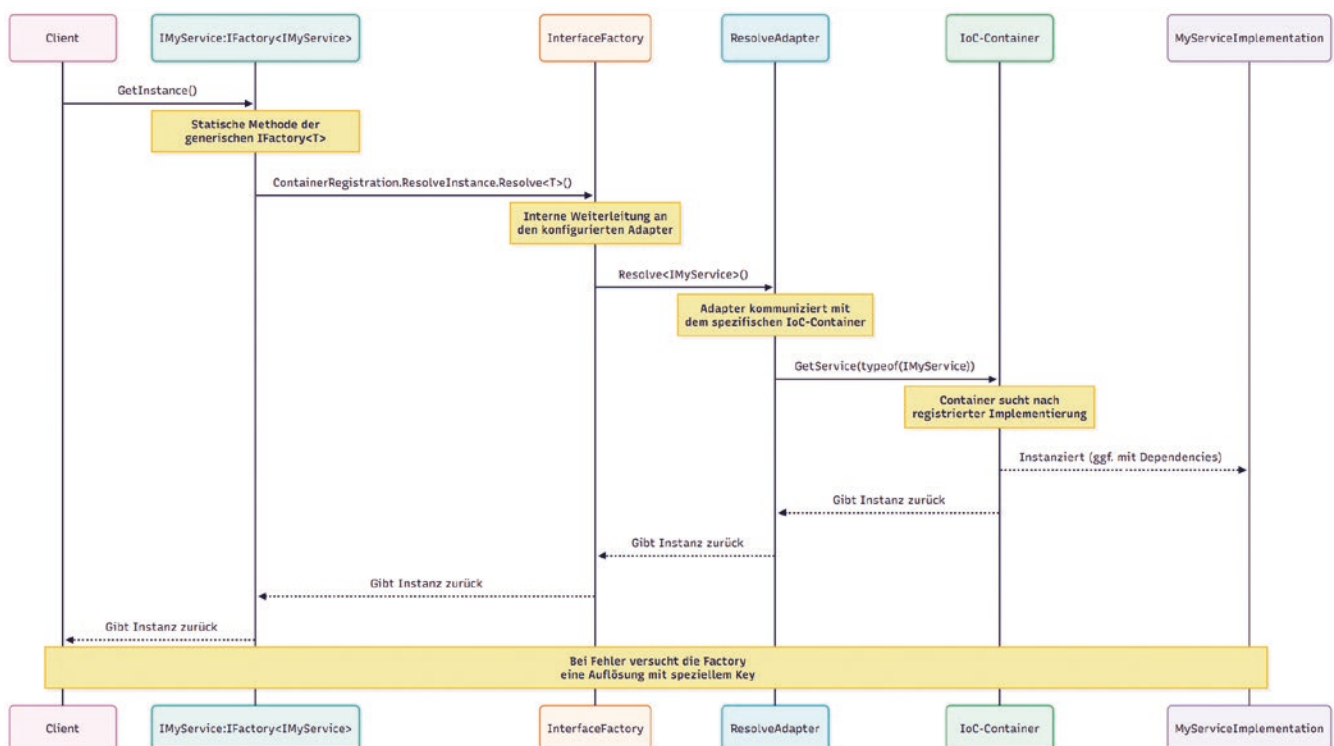
    void IServiceBWithDim.Process()
    {
        Console.WriteLine("Calling B's Process via
            explicit implementation");
    }

    // Ein einfacher Aufruf von myService.Process() ist
    // bei dieser Konstellation nicht möglich.
    // Man muss zum spezifischen Interface casten:
    // ((IServiceAWithDim)myService).Process();
}
```

- Potenzial für Missbrauch:** Entwickler könnten DIMs nutzen, um „Hilfsfunktionen“ oder generische Logik in Interfaces zu platzieren, die dort konzeptionell nicht hingehören. Interfaces könnten als eine Art Ersatz für Utility-Klassen oder Basisfunktionalität missbraucht werden, was zu unsauberem, schwer lesbarem und schlecht strukturiertem Code führen kann. Die Versuchung ist groß, wenn „es einfach geht“.

Diese Bedenken sind partiell valide und sollten deshalb keinesfalls ignoriert werden. Sie weisen auf Herausforderungen hin, die bei unsachgemäßem Gebrauch von DIMs entstehen können.

Allerdings liegt das Problem hier oft weniger im Sprachfeature selbst, sondern hauptsächlich in seiner Anwendung und in dem zugrunde liegenden Design.



**Client ruft IFactory.GetInstance() -> An Resolve-Adapter weitergeleitet -> Adapter nutzt IServiceProvider des IoC-Containers -> Container liefert Instanz -> Instanz wird über Adapter/Factory an Client zurückgegeben (Bild 2)**

Die behauptete „Verwässerung“ des Interface-Konzepts ist technisch betrachtet nicht zutreffend. Wie bereits erwähnt, bleiben Interfaces mit DIMs zustandslos. Dies ist ein fundamentaler Unterschied zu abstrakten Klassen, die Zustand speichern und nichtöffentliche Member haben können. Interfaces mit DIMs ermöglichen im Kern die Mehrfachvererbung von Verhalten, nicht von Zustand oder vollständigen Implementierungen mit internen Details. Die Unterscheidung bleibt für ein sauberes Design entscheidend und ist sprachlich weiterhin klar getrennt.

Die Komplexität beim „Diamond Problem“ wird bei C# durch klar definierte Regeln zur Konfliktlösung gelöst. Wenn eine Klasse zwei Interfaces implementiert, die beide eine DIM mit identischer Signatur bereitstellen, und die Implementierungen nicht über eine gemeinsame Basis im Interface-Baum zusammenlaufen, verlangt der Compiler eine explizite Implementierung in der Klasse selbst (*InterfaceName.MethodName()*). Dies zwingt den Entwickler, den Konflikt bewusst aufzulösen, und dokumentiert klar, welche Implementierung von welchem Interface angesprochen wird (Bild 1 und Listing 2). Dieses Verhalten mag anfangs ungewohnt sein, ist aber deterministisch und zwingt zu Klarheit an den potenziellen Bruchstellen.

Auch die Sorge vor Missbrauch ist berechtigt, aber wiederum ein Problem der Entwicklerdisziplin und des Designs,

nicht des Features. Jedes mächtige Sprachfeature kann suboptimal eingesetzt werden. Niemand würde behaupten, Schleifen seien schlecht, nur weil man Endlosschleifen schreiben kann. Der Schlüssel liegt in der Schulung, dem Verständnis für den Zweck von DIMs und der Einhaltung etablierter Designprinzipien wie SOLID. DIMs sind kein Ersatz für eine gut strukturierte Klassenhierarchie oder Utility-Klassen, wenn diese die passendere Lösung sind.

Es stellt sich nun die Frage, wie dieses Sprachfeature im praktischen Einsatz zu sauberem Code beitragen könnte. Anstatt DIMs pauschal als Design-Antipattern abzutun, wird am Beispiel eines konkreten Designmusters untersucht, wie sie in einem strukturierten Kontext vorteilhaft genutzt werden können. Hier bietet das Beispielprojekt einen Ansatz.

Die *InterfaceFactory* [4] ist eine Bibliothek, die das bewährte Factory-Pattern mit den Prinzipien der Dependency Injection (DI) verbindet. Ihr Hauptzweck ist es, eine entkoppelte Möglichkeit zu schaffen, um Instanzen, die durch Schnittstellen definiert sind, zu beziehen. Der Client-Code soll eine benötigte *IMyService*-Instanz erhalten können, ohne eine konkrete Implementierung instanzieren oder den IoC-Container direkt bedienen zu müssen.

Ein Element der *InterfaceFactory* ist die generische Schnittstelle *IFactory<T>*. Diese Schnittstelle definiert statische Methoden wie *GetInstance()* und *GetRequiredInstance()*, die ►

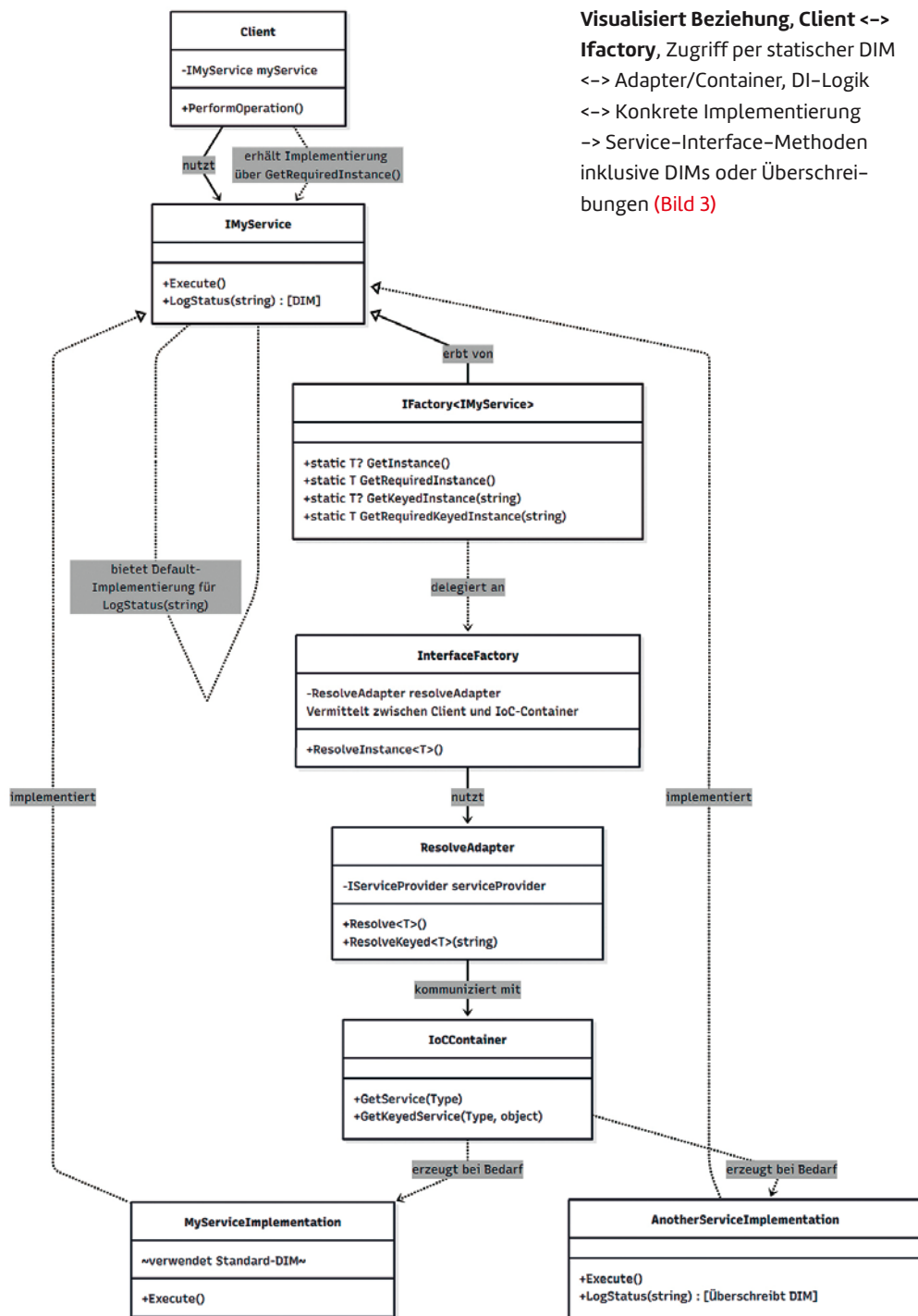
### Listing 3: Vereinfachte Darstellung der *IFactory<T>* und eines abgeleiteten Service-Interfaces

```
public interface IFactory<T>
{
    // Wird vom Service-Interface implementiert (z.B.
    // IMyService : IFactory<IMyService>)
    // Statische Methode, die über das ableitende
    // Interface T aufgerufen werden kann (z.B.
    // IMyService.GetInstance())
    public static T? GetInstance()
    {
        // Delegate Aufruf an den internen Resolve-Adapter,
        // der mit dem zugrunde liegenden IoC-Container
        // kommuniziert.
        // ... interne Logik zur Auflösung der T-Instanz ...
        try
        {
            return ContainerRegistration.ResolveInstance.
                Resolve<T>();
        }
        catch (InvalidOperationException)
        {
            // Fallback-Logik für spezielle Registrierungen,
            // Details siehe Code.
            return ContainerRegistration.ResolveInstance.
                ResolveKeyed<T>(ContainerRegistration.
                    TransientKey);
        }
    }
}

// Weitere statische Methoden, die über T aufgerufen
// werden:
public static T GetRequiredInstance() ...
public static T? GetKeyedInstance(string key) ...
public static T GetRequiredKeyedInstance(string key)
...

// Beispiel für die Definition eines Service-
// Interfaces, das über die Factory verwaltet wird:
public interface IMyService : IFactory<IMyService>
{
    // Leitet von IFactory<T> ab
    void Execute();

    // Optional: Eine DIM, die von der Factory
    // verwaltenden Implementierungen genutzt werden
    // kann
    void LogStatus(string status)
    {
        Console.WriteLine($"Status: {status}");
        // Standardverhalten
    }
}
```



**Visualisiert Beziehung, Client <--> IFactory, Zugriff per statischer DIM**  
 <--> Adapter/Container, DI-Logik  
 <--> Konkrete Implementierung  
 -> Service-Interface-Methoden inklusive DIMs oder Überschreibungen (Bild 3)

dem Client, benötigte Instanzen auf direkte Weise anzufordern, ohne explizit auf einen separaten Factory-Typ oder den IoC-Container referenzieren zu müssen.

Im Hintergrund agiert die InterfaceFactory als Vermittler zu einem zugrunde liegenden IoC-Container, wobei sie durch Adapter-Interfaces (*IContainerRegisterAdapter*, *IContainerResolveAdapter*) containerunabhängig bleibt.

Die Integration in ein bestehendes DI-System erfolgt üblicherweise in zwei Schritten während der Anwendungsinitialisierung: Zuerst werden über eine Erweiterungsmethode auf der Service Collection (zum Beispiel *serviceCollection.RegisterInterfaceFactories()*) die Service-Interfaces und ihre konkreten Implementierungen, die über die Factory bezogen werden sollen, im tatsächlichen IoC-Container registriert. Hierbei werden gegebenenfalls Metadaten wie *ContainerRegistrationAttribute* berücksichtigt. Anschließend wird, nachdem der IoC-Container gebaut wurde, dieser der InterfaceFactory zugänglich gemacht, zum Beispiel über *serviceProvider.UseInterfaceFactory()*. Dieser IService-Provider wird dann intern für die Auflösung von Abhängigkeiten verwendet, wenn die statischen *GetInstance*-Methoden der *IFactory<T>* aufgerufen werden (Bild 2).

So zentralisiert die Interface-

Factory die Logik der Objekt-Erzeugung und der Auswahl der korrekten Implementierung aus dem IoC-Container. Wenn ein Service-Interface DIMs verwendet und komplexe Beziehungen aufweist, wird die Auflösung der Instanz, die dieses Verhalten umsetzt, durch den IoC-Container unter Steuerung der Factory-Infrastruktur gehandhabt. Der Client-Code, der eine Instanz über *IMyService.GetInstance()* erhält und dann Methoden aufruft, ist von dieser internen Auflösungslogik entkoppelt. Er interagiert nur mit dem Interface, das ihm die benötigte Funktionalität über DIMs anbietet.

Die Factory ist explizit dafür ausgelegt, Schnittstelleninstanzen bereitzustellen, die typischerweise Geschäftslogik in

den Mechanismus zur Beschaffung von Instanzen des Typs T bereitstellen. Das Besondere am Design ist, dass das Service-Interface selbst von *IFactory<T>* ableitet, zum Beispiel *IMyService : IFactory<IMyService>*. Dies ermöglicht, dass die Factory-Methoden direkt über das Service-Interface aufgerufen werden können (Listing 3).

Die Fähigkeit, statische Methoden wie *GetInstance()* direkt in der *IFactory<T>*-Schnittstelle zu definieren und diese dann über das ableitende Service-Interface (wie *IMyService.GetInstance()*) aufrufen zu können, ist eine direkte Anwendung der statischen Methoden in Interfaces, einer Funktionalität, die zusammen mit den DIMs eingeführt wurde. Dies erlaubt

#### ● Listing 4: Client-Code, der Factory nutzt, und ein Service-Interface mit optionaler DIM

```

public class MyServiceConsumer
{
    // Keine direkte Abhängigkeit von der konkreten
    // Implementierung oder dem IoC-Container
    private readonly IMyService _myService;
    // IMyService ist ein Interface, das von
    // IFactory<IMyService> ableitet

    public MyServiceConsumer()
    {
        // Instanz über die Factory beziehen -
        // einfacher Aufruf direkt über das Service-
        // Interface dank statischer Methode in IFactory<T>
        myService = IMyService.GetRequiredInstance();
    }

    public void PerformOperation()
    {
        myService.Execute();
        // Aufruf einer Methode auf dem Interface
        // Wenn IMyService eine DIM LogStatus() hat, kann
        // diese hier genutzt werden, unabhängig davon,
        // ob die konkrete Implementierung sie
        // überschreibt oder den Standard nutzt.
        myService.LogStatus("Operation complete");
    }
}

// Beispiel für die Definition eines Service-
// Interfaces, das über die Factory verwaltet wird
// und optional eine DIM nutzt (wie im Teil 1
// besprochen).
// Dieses Interface leitet von IFactory<IMyService>
// ab, um die statischen Factory-Methoden zu erhalten.
public interface IMyService : IFactory<IMyService>
{
    void Execute();

    // Optional: Eine DIM, die von der Factory
    // verwaltenden Implementierungen genutzt werden
    // kann
    void LogStatus(string status)
    {
        Console.WriteLine($"Status: {status}");
        // Standardverhalten
    }
}

// Eine Implementierung, die die DIM nutzt (würde
// über die Factory registriert werden)
public class MyServiceImplementation :
    IMyService { ... }

// Eine Implementierung, die die DIM überschreibt
// (würde über die Factory registriert werden)
public class AnotherServiceImplementation :
    IMyService { ... }

// Anwendungsinitialisierung (vereinfachtes Konzept)
var services = new ServiceCollection();
services.AddScoped<IMyService,
    MyServiceImplementation>();
// Registrierung im IoC-Container (hier Standard)
services.AddKeyedScoped<IMyService,
    AnotherServiceImplementation>("Another");
// Beispiel Keyed Registrierung

// Die RegisterInterfaceFactories() Erweiterung
// scannt Assemblies nach Typen, die IFactory<T>
// implementieren.
// Da IMyService von IFactory<IMyService> ableitet,
// wird IMyService gescannt.
// RegisterInterfaceFactories() findet dann konkrete
// Klassen, die IMyService implementieren
// (MyServiceImplementation,
// AnotherServiceImplementation)
// und registriert diese im IoC-Container, basierend
// auf Attributen oder Standardregeln.
services.RegisterInterfaceFactories();

var serviceProvider = services.BuildServiceProvider();
serviceProvider.UseInterfaceFactory();
// Macht den ServiceProvider für die Factory
// zugänglich

// Nutzung durch Client (in einer anderen
// Klasse/Methode):
var consumer = new MyServiceConsumer();
// Holt IMyService über
// IMyService.GetRequiredInstance()
consumer.PerformOperation();
// Man könnte auch direkt über die statischen Methoden
// Instanzen holen:
var myServiceInstance = IMyService.GetInstance();
var anotherServiceInstance =
    IMyService.GetKeyedInstance("Another");

```

ihren implementierenden Klassen kapseln. DIMs in den Service-Interfaces, wie zum Beispiel eine *LogStatus*-Methode in *IMyService*, erhalten im Kontext dieses Musters eine klare

Rolle: Sie definieren ein sinnvolles Standardverhalten, das von allen Implementierungen genutzt oder überschrieben werden kann. Sie steuern, welche Implementierung – und ►



damit welches Verhalten zur Laufzeit – geliefert wird. Dies lenkt den Entwickler weg vom Gedanken, „irgendwelche“ Logik in Interfaces zu schreiben, hin zu einer klaren Trennung von Verantwortlichkeiten:

- **Factory:** Beschaffung,
- **Interface:** Vertrag und optionales Standardverhalten,
- **Implementierung:** spezifische Geschäftslogik.

Auch die Befürchtung einer Verwässerung des Interface-Konzepts kann relativiert werden. Der spezifische Einsatz von DIMs in der *IFactory<T>*-Schnittstelle dient einem klaren architektonischen Zweck: der Ermöglichung eines typbasierten, statischen Factory-Zugriffs. Es erweitert die Möglichkeiten von Interfaces um ein spezifisches Muster, anstatt das Konzept des zustandslosen Vertrags grundsätzlich aufzuweichen. Wenn Service-Interfaces DIMs enthalten, dienen diese weiterhin primär der API-Evolution oder als Traits-ähnliches Standardverhalten, und die Factory sorgt für die konsistente Bereitstellung von Instanzen, die dieses Verhalten umsetzen.

Die Nutzung von DIMs in diesem Kontext bietet somit mehrere Vorteile für das Softwaredesign: Sie ermöglicht eine starke Entkopplung des Client-Codes von der Implementierung und dem IoC-Container. Der Zugriff auf benötigte Dienste wird durch die statischen Factory-Methoden vereinfacht. Die Struktur fördert eine klare Trennung der Verantwortlichkeiten zwischen Factory, Interface und Implementierung. Dies verbessert die Wartbarkeit und die Testbarkeit, da Implementierungen leicht ausgetauscht und Mocks über die Factory konfiguriert werden können. DIMs tragen hier bei, indem sie Boilerplate-Code in Implementierungen reduzieren und die API-Evolution erleichtern, während die Factory den Rahmen für ihren disziplinierten Einsatz bildet (Bild 3 und Listing 4).

### Fazit: DIM ist ein Werkzeug, dessen Wert im Kontext liegt

Dieser Artikel betrachtet die Einführung der DIMs, ein Sprachfeature, das von Beginn an eine Debatte in der Entwicklergemeinde ausgelöst hat. Die Skepsis verdient zum Teil tatsächlich Beachtung. Die entscheidende Erkenntnis, die sich aus der Betrachtung ergibt, ist jedoch, dass der Wert und die „Gefährlichkeit“ eines Sprachfeatures oft nicht in ihm selbst liegen, sondern maßgeblich davon abhängen, wie es im architektonischen Kontext und mithilfe etablierter Designmuster eingesetzt wird. Am Beispiel der InterfaceFactory lässt sich erkennen, wie DIMs in einen strukturierten Rahmen eingebettet werden können.

Das abschließende Fazit muss daher lauten: Default Interface Methods sind keine intrinsisch „guten“ oder „schlechten“ Features. Sie sind ein mächtiges Werkzeug, das mit Sorgfalt und Verständnis eingesetzt werden muss. Wie bei jedem neuen Sprachkonstrukt liegt es letztlich in der Verantwortung des Entwicklers, die Vor- und Nachteile im spezifischen Kontext abzuwägen und die Funktion im Einklang mit etablierten Designprinzipien und passenden Architekturmustern zu nutzen.

Die Kritik an DIMs ist daher eher als eine Mahnung zur Vorsicht zu verstehen, aber sie sollte uns nicht davon abhal-

ten, die Potenziale dieses Features dort zu erkennen und zu nutzen, wo sie einen echten Mehrwert für die Qualität und Wartbarkeit unserer Software bieten können – insbesondere, wenn sie in einen strukturierten Rahmen, wie ihn die InterfaceFactory bietet, eingebettet sind.

Nicht alles Neue ist per se schlecht, und Offenheit für Veränderungen in Verbindung mit architektonischer Disziplin ist oft der Schlüssel zu besseren Lösungen. ■

[1] Stefan Bechtel, *Rufzentrale für Methoden*, dotnetpro 8/2023, Seite 78 ff.

[2] Default Interface Methods, Proposal für C# 8.0: <https://github.com/dotnet/csharp-lang/blob/main/proposals/csharp-8.0/default-interface-methods.md>; Microsoft .NET Foundation: C# Language Design Meeting Notes for C# 8.0, March 8, 2017, <https://github.com/dotnet/csharp-lang/blob/main/meetings/2017/LDM-2017-03-08.md>; March 21, 2017: <https://github.com/dotnet/csharp-lang/blob/main/meetings/2017/LDM-2017-03-21.md>; April 18, 2017: <https://github.com/dotnet/csharp-lang/blob/main/meetings/2017/LDM-2017-04-18.md>; May 17, 2017: <https://github.com/dotnet/csharp-lang/blob/main/meetings/2017/LDM-2017-05-17.md>; May 31, 2017: <https://github.com/dotnet/csharp-lang/blob/main/meetings/2017/LDM-2017-05-31.md>; June 14, 2017: <https://github.com/dotnet/csharp-lang/blob/main/meetings/2017/LDM-2017-06-14.md>

[3] Jeremy Clark, *Jeremy Bytes Blog: Interfaces in C# 8 are a Bit of a Mess*, 31. August 2019, <https://jeremybytes.blogspot.com/2019/09/interfaces-in-c-8-are-bit-of-mess.html>; Reddit r/dotnet: The Most Controversial C# 8.0 Feature. Diskussion im Thread, 20. Februar 2019, [www.reddit.com/r/dotnet/comments/asq3jl/the\\_most\\_controversial\\_c\\_8\\_0\\_feature\\_default/](http://www.reddit.com/r/dotnet/comments/asq3jl/the_most_controversial_c_8_0_feature_default/); Stack Overflow: When Should We Use Default Interface Method in C#?, Diskussion zu „When should we use default interface method in C#?“, 9. Juli 2020, <https://stackoverflow.com/questions/62832992/when-should-we-use-default-interface-method-in-c>; GitHub Discussion: Default Interface Methods Improvements, Discussion #6171, 26. Mai 2022, <https://github.com/dotnet/csharp-lang/discussions/6171>

[4] Die Bibliothek ist Open Source (MIT Licence) auf GitHub und Nuget.org, <https://github.com/chr0mcom/InterfaceFactory>



**Stefan Bechtel**

leitet die Softwareabteilung bei der Werbas KSR GmbH in Holzgerlingen und ist seit 20 Jahren Softwareentwickler und IT-Manager mit umfangreicher Expertise in der Entwicklung von Applikationen und KI-Lösungen.  
**stef@nbech.tel**

# DWX ACADEMY

Hands-on Workshops  
und Weiterbildung für  
Software-Entwickler  
und -Architekten

REMOTE  
ODER  
INHOUSE

## Hands-on Workshops .NET / C#

|                                                                            |          |
|----------------------------------------------------------------------------|----------|
| C#-Grundlagen                                                              | (3 Tage) |
| C# für Fortgeschrittene                                                    | (3 Tage) |
| ASP.NET Core und Web API                                                   | (3 Tage) |
| Datenbankentwicklung mit dem Entity Framework Core                         | (3 Tage) |
| Clean Code und Software Design                                             | (3 Tage) |
| Async & Await in der Praxis                                                | (2 Tage) |
| C#, .NET & SQL-Server-Update                                               | (1 Tag)  |
| Fix von .NET Framework zu .NET (CORE)                                      | (1 Tag)  |
| Einstieg in test- und behaviour-driven Development (TDD/BDD) mit .NET Core | (2 Tage) |
| Testen von .NET-Anwendungen                                                | (3 Tage) |
| Einstieg in MVVM mit WPF                                                   | (2 Tage) |
| Modulare WPF-Anwendungen mit PRISM                                         | (3 Tage) |
| Einstieg in NoSQL mit MongoDB für .NET-Entwickler                          | (3 Tage) |
| ASP.NET Blazor – SPA-Anwendungen mit C# und .NET                           | (3 Tage) |



Weitere Informationen auf [developer-world.de/dwx-academy](https://developer-world.de/dwx-academy)  
Preise auf Anfrage!

Deine Ansprechpartnerin: Susanne Herl • +49 731 88005-8835 • [susanne.herl@ebnermedia.de](mailto:susanne.herl@ebnermedia.de)



## BLAZOR

# Razor Components



Wie Razor-Komponenten und Blazor zusammenarbeiten.

**B**lazor hat sich mittlerweile etabliert, und eine Vielzahl an Projekten wird bereits mit dieser UI-Technologie umgesetzt. Microsoft selbst nutzt die hauseigene Technologie für sein Lern- und Entwicklungsprogramm Aspire und das dazugehörige Dashboard, was darauf hindeutet, dass Microsoft weiter an ihr festhält und sich eine Investition in Blazor auszahlt.

Zentraler Punkt von Blazor sind die Razor Components, mit denen man am Ende alles aufbauen kann. Doch wie funktioniert das Rendering einer solchen Razor-Komponente eigentlich genau?

## Rendering Instructions

Wie immer hilft ein kleines Beispiel, dies technisch weiter zu zerlegen und im Detail zu betrachten. Etwa Microsofts Standardbeispiel: Ein Counter, der einen Button bereitstellt, welcher wiederum bei einem Klick eine Zahl erhöht.

Die wenigen Codezeilen in **Bild 2** formulieren den grundsätzlichen Aufbau der Komponente. Mit der Razor-Syntax lassen sich HTML und C#-Code mischen, und wir können unsere Oberfläche somit deklarativ in HTML beschreiben. Dabei wird im Paragraph-Tag (`<p ... </p>`) auf den `_currentCount` zugegriffen, womit bereits etwas Dynamik in den HTML-Code kommt.

## Der Compiler spielt mit

Beim Kompilieren einer solchen Razor Component trickst Microsoft etwas und konvertiert die eigentliche Razor-Komponente in einfaches C#. Etwas, das ich mir übrigens auch für WPF wünschen würde: Statt des kryptischen und alten BAML-Weiges – das in Binär-Code (Binary Application Markup Language) gewandelte XAML – liefern Razor-Komponenten unterstützt durch den Compiler sofort den finalen Code.



Das Grundkonzept von Razor-Komponenten (Bild 1)

## Razor Component

Eine Razor-Komponente ist eine wiederverwendbare Benutzeroberflächenkomponente, die HTML und C#-Code kombiniert. Sie wird typischerweise in eine `.razor`-Datei geschrieben und kann interaktiv sein – also zum Beispiel auf Benutzeraktionen reagieren (**Bild 1**). Außerdem kann sie andere Komponenten referenzieren, und es entsteht so ein Baum mit einer Wurzelkomponente, der je nach Bedarf gerendert wird. Das Entscheidende dabei ist, dass eine Razor Component immer für sich selbst zuständig ist; selbst eine Integration einer anderen Komponente ist nur die Weiterleitung der Anweisung „Bitte rendere auch noch diese Komponente“.

Eine Razor-Datei beschreibt die Komponente, und man muss sich nicht weiter um das Rendering kümmern – das übernimmt dann Blazor. Am Ende entstehen dabei Rendering Instructions, die beschreiben, wie sich das finale HTML am Ende zusammensetzt.

War es früher (mit .NET Core 3) noch so, dass in der Build Chain ein externes Tool aufgerufen wurde, so ist in der neueren Roslyn Compiler Platform ein Source Generator dafür zuständig, den Code zu generieren (**Bild 3**).

Der Source Generator erzeugt beim Kompilieren den fertigen C#-Code, etwa um das Counter-Beispiel umzusetzen, und generiert dabei die Rendering Instructions.

```

1 {
2     @page "/counter"
3     <p role="status">Current count: @_currentCount</p>
4     <button class="btn btn-primary" @onclick="@(() => _currentCount++)">
5         Click me
6     </button>
7
8
9     @code {
10         private int _currentCount = 0;
11     }

```

Microsofts Counter-Beispiel (Bild 2)



## BuildRenderTree

Beim Kompilieren erzeugt der Source Generator eine Klasse, deren Name vom Dateinamen abgeleitet wird. Die Basisklasse ist dabei – im Standardfall – *ComponentBase*, die am Ende die Grundinfrastruktur für Blazor darstellt (Bild 4). Diese Basisklasse ist – anders als andere UI-Basisklassen von Microsoft – extrem schlank. Sie hat nicht nur gerade mal knapp 400 Zeilen Code, sondern nutzt auch keine weitere Basisklasse.

Innerhalb der Klasse erwartet Blazor nun lediglich eine einzige Methode namens *BuildRenderTree*. Diese Rendering-Instructions-Methode (Bild 5) beschreibt, wie Blazor das HTML final zusammenbauen soll.

Dabei wird ein *RenderTreeBuilder* übergeben, der über eine überschaubare Anzahl an Methoden verfügt, welche die Struktur des HTML beschreiben. Das Spannende an der Methode *BuildRenderTree* ist, dass sie kein HTML zusammenbaut und zurückgibt, sondern lediglich Instruktionen angibt, wie das zu geschehen hat:

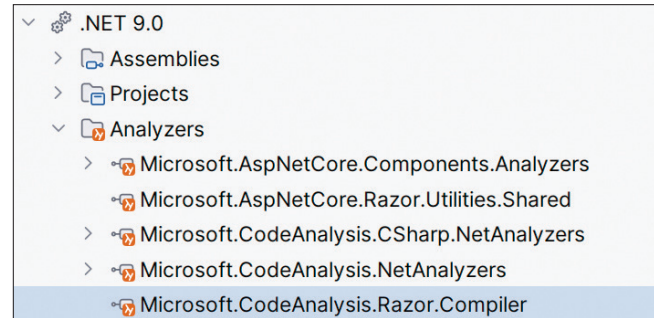
- Öffne ein *p*-Tag,
- platziere ein Attribut darauf,
- füge Content hinzu,
- gib dann nochmal einen Content dazu,
- schließe das Element.

*BuildRenderTree* produziert also kein fertiges HTML, liefert aber genug Informationen, um es daraus ableiten zu können.

## Diff: Veränderungen erkennen

Blazor ist recht ausgeklügelt. Und natürlich könnte man diese Instruktionen nun einmalig durchlaufen und daraus HTML-Code berechnen. Aber viel spannender ist, dass eine Razor Component „lebt“ und ihren Status eben ändert, entweder durch den Klick auf einen Button oder durch eine sonstige Aktion. Allerdings wirft das die Frage auf: Wie erkennt ein Webframework die Änderungen am HTML-Code? Das ist durchaus knifflig, denn jedes Mal einfach alle Knoten zu ersetzen wäre zäh und langsam und würde unweigerlich zu Problemen führen. Das ist auch der Grund, weshalb viele Frameworks ein virtuelles Document Object Model (Virtual DOM) aufbauen, um darin die Änderungen zu erkennen und diese später im richtigen DOM im HTML anzuwenden.

Blazor geht hier einen anderen Weg, der simpel und genial zugleich ist. Dadurch, dass Microsoft einen Compiler verwendet, werden für alle Instruktionen sogenannte Sequence Numbers erzeugt. Das sind Ganzzahlen, die jedem Element, Attribut oder Inhalt im *RenderTreeBuilder* zugewiesen werden. Sie geben an, welche Zeile im Quellcode ein bestimmtes UI-Element erzeugt hat, und helfen Blazor dabei, schnell zu erkennen, ob sich etwas im UI geändert hat. Diese Zahl muss dabei gar nicht eindeutig sein und ist es oft auch nicht. Allerdings kann man beim Schreiben eines Differenz-Berechnungs-Algorithmus



Der Source Generator (Bild 3)

```
[Route("/counter")]
public class Counter : ComponentBase
{
```

Verweis auf die Basisklasse ComponentBase (Bild 4)

mus sehr gut Änderungen erkennen, und das in linearer Geschwindigkeit. Man stelle sich vor, man hat zwei Baumstrukturen und muss nun herausfinden, in welchen Ästen des Baumes sich etwas geändert hat – das ist recht komplex! Durch die Sequence Numbers kann der Abgleich komplett linear abgebildet werden, allerdings mit einem kleinen Nachteil, von dem weiter unten noch die Rede sein wird.

## Abgleich von Änderungen

Blazor geht nun folgendermaßen vor: Beim erstmaligen Rendern muss Blazor die Komponente und all die Instruktionen durchlaufen und das HTML einmalig erzeugen – so weit nichts Unerwartetes. Doch dabei wird – unabhängig vom Rendering Mode (Server, WebAssembly, Hybrid) – nicht nur das HTML zurückgegeben, sondern der Renderer für die Komponente merkt sich auch die Liste der Instruktionen, ►

```
private int _currentCount = 0;

protected override void BuildRenderTree(RenderTreeBuilder __builder)
{
    __builder.OpenElement(0, "p");
    __builder.AddAttribute(1, "role", "status");
    __builder.AddContent(2, "Current count: ");
    __builder.AddContent(3, (object) this._currentCount);
    __builder.CloseElement();
    __builder.AddMarkupContent(4, "\r\n");
    __builder.OpenElement(5, "button");
    __builder.AddAttribute(6, "class", "btn btn-primary");
    __builder.AddAttribute<MouseEventArgs>(7, "onClick",
        EventCallback.Factory.Create<MouseEventArgs>((object) this,
            (Action) (() => ++this._currentCount)));
    __builder.AddMarkupContent(8, "\r\n    Click me\r\n");
    __builder.CloseElement();
}
```

Die Methode BuildRenderTree (Bild 5)



also 0 war *p*, 1 war das Attribut, 2 war *Current Count*: und 3 war 0.

Gibt es nun aus Sicht von Blazor den An-  
lass, genau diese Komponente noch ein-  
mal zu rendern, wird einfach dasselbe er-  
neut gemacht. Statt aber das HTML direkt  
zu erzeugen, werden die Rendering In-  
structions gesammelt. Also erneut: 0 ist *p*,  
1 ist das Attribut, 2 ist *Current Count*: und  
3 ist nun zum Beispiel 1. Oh, erkannt? Man  
kann sofort sehen, dass sich beim zweiten  
Rendern die Instruktion 3 verändert hat.

Und genau das ist am Ende für Blazor  
der springende Punkt: Unabhängig vom  
Render Mode wird einfach die Informati-  
on weitergegeben: „Nur die Instruktion 3  
hat sich geändert“. Und egal in welchem  
Modus man die eigene Blazor-Anwen-  
dung auch immer laufen lässt – an den ei-  
gentlichen Renderer wird ausschließlich  
die Instruktion gesendet, die sich verändert hat, und wie de-  
ren neuer Inhalt (hier 1 statt 0) lautet.

In meinen Augen ist das ein sensationell simpler und  
gleichzeitig genialer Schachzug.

## Der Nachteil

Doch wie überall im Leben: Es gibt kein Licht ohne Schatten.  
Razor Components können Änderungen zweifellos äußerst  
schnell erkennen und auch die Unterschiede zwischen zwei  
Versionen (das Diff) in kürzester Zeit bereitstellen, sogar mit  
sehr wenig CPU- und Memory-Aufwand. Das Verfahren stößt  
aber auch an Grenzen. Nehmen wir zum Vergleich eine klei-  
ne Abwandlung unseres Beispiels vor, siehe **Bild 6**.

Wir iterieren mit einer *foreach*-Schleife über eine Menge  
von Zahlen. In diesem Fall kann man gut erkennen, dass der  
Source Generator für den Schleifendurchlauf immer diesel-  
ben Sequence Numbers verwendet – in unserem Fall 1 und 2.

Das tut der Algorithmus weil er erkennt, dass die Nummern  
1 und 2 im Beispiel aus **Bild 6** viermal verwendet werden: 1,2  
– 1,2 – 1,2 – 1,2.

Würde man das Array *numbers* erweitern und ein fünftes  
Element anhängen, würde der Algorithmus erkennen, dass  
es statt vier nunmehr fünf Zahlenpaare gibt, und daraus ab-  
leiten können, dass er ein Element anfügen muss. Ändert sich  
nun eine Zahl innerhalb der Menge, kann der Algorithmus  
problemlos erkennen, dass beispielsweise das zweite Paar ei-  
nen anderen Content hat, und daraus ein Update bestimmen.

So weit ist das perfekt. Was passiert allerdings, wenn wir  
aus der Menge den ersten Eintrag entfernen. Da der Algorith-  
mus darauf baut, dass die Zahlen in einer bestimmten Rei-  
henfolge ausgegeben werden, erkennt er nun zwar, dass sich  
ein Eintrag verabschiedet hat, aber auch, dass aus der ehe-  
maligen 1 nun eine 2 wurde, aus der 2 die 3 und so weiter.

In solch einem Fall muss der Algorithmus einen Eintrag lö-  
schen und danach alle anderen Einträge aktualisieren. Das  
ist aufwendig, teuer und die größte Schwachstelle des Algo-  
rithmus an diesem Punkt.

```
<ul>
    @foreach (var number:int in numbers)
    {
        <li>@number</li>
    }
</ul>

@code {
    private List<int> numbers = [1, 2, 3, 4];
}
```

```
protected override void BuildRenderTree(
    #nullable disable
    RenderTreeBuilder __builder)
{
    __builder.OpenElement(0, "ul");
    foreach (int number in this.numbers)
    {
        __builder.OpenElement(1, "li");
        __builder.AddContent(2, (object) number);
        __builder.CloseElement();
    }
    __builder.CloseElement();
}
```

Der Nachteil bei Listen und Schleifen (Bild 6)

```
<li @key="number">@number</li>

__builder.OpenElement(1, "li");
__builder.SetKey((object) number);
__builder.AddContent(2, (object) number);
__builder.CloseElement();
```

@key ist die Lösung (Bild 7)

## Abhilfe

Wie Sie in **Bild 7** sehen, hat Microsoft einen Lösungsvorschlag:  
Verwendet man in der Razor-Komponente nämlich das  
Schlüsselwort *@key* und verbindet dieses mit einem Daten-  
wert wie der Zahl in obigem Beispiel, fügt der Source Gene-  
rator eine weitere Instruktion ein und verbindet auf diese  
Weise die Sequence Numbers mit dem Datenwert. Das hat  
den Vorteil, dass Blazor nun erkennt, dass die Sequence  
Number 1 dem Wert 1 entspricht und die 2 dem Wert 2. Löscht  
man nun den ersten Eintrag aus der Liste, erkennt Blazor  
nicht nur, dass ein Eintrag fehlt, sondern auch, dass die an-  
deren Werte gleich geblieben sind, und optimiert die Ände-  
rung dahingehend, dass nur ein Knoten entfernt wird. Pro-  
blem gelöst!

Das Ganze funktioniert natürlich auch mit komplexen und  
sogar eigenen Datentypen und ergibt (meistens) wirklich  
Sinn. Allerdings sollte man stets wissen, was genau im Hin-  
tergrund passiert, um sich vor Überraschungen zu schützen.

## Fazit

Ich mag das Konzept, das Microsoft bei Blazor mit den Razor  
Components eingeführt hat, da es mit relativ wenig Aufwand  
entzaubert werden kann. Der Riesenvorteil ist, dass Microsoft  
für C# einen Compiler hat und Sequence Numbers statisch  
kompiliert definieren kann. Dadurch fällt es sehr leicht, Än-  
derungen differenziell zu erkennen. Damit hat man eine ech-  
te „Performancewaffe“ zur Verfügung. ■



### Christian Giesswein

studierte Wirtschaftsinformatik in Wien und ent-  
wickelt von klein auf Software mit .NET und C#.  
In Tirol hat er das Unternehmen Giesswein Soft-  
ware-Solutions gegründet, das sich auf Individu-  
alsoftware und Consulting spezialisiert hat.

christian@software.tirol

# .NET DEVELOPER CONFERENCE

Die Konferenz für .NET-Entwickler:innen

**24. BIS 27.  
NOVEMBER '25**  
Pullman Cologne Hotel

**10%  
Rabatt**  
mit  
DNPAboSpecial

Software Design | Software Architecture | Software Quality  
Software Testing | Frontend Development | GenAI and Agentic  
Data & Analytics | .NET | Approaches | Backend Technologies  
Cloud & Infrastructure



## WORKSHOPS – MONTAG, 24. NOVEMBER 2025

| Uhrzeit             | Workshop 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Workshop 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Workshop 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Workshop 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 09.00<br>–<br>18.00 |  <p><b>Moderne Softwarearchitekturen</b><br/><i>David Tielke</i></p> <p>In Zeiten, in denen Anwendungen mit Cloud, künstlicher Intelligenz und ähnlichen Technologien erstellt werden, ist ein Qualitätsmerkmal ihrer Architektur wichtiger denn je: Flexibilität. Eine moderne Architektur sollte daher nicht nur die gegenwärtige Anforderungen abdecken, sondern auch künftige Einsatzszenarien berücksichtigen. Im Workshop befassen wir uns mit der Frage, wie einer Architektur die notwendige Flexibilität verliehen werden kann, um alle zukünftigen Anforderungen zu erfüllen und wirtschaftlich optimal betrieben zu werden, und lernen Wege und Methoden kennen, mit denen sich dies sicherstellen lässt.</p> |  <p><b>Autonom und Intelligent: AI-Agentensysteme mit .NET entwickeln</b><br/><i>Jörg Neumann</i></p> <p>AI-Agenten bilden die Basis einer völlig neuen Anwendungsgeneration – Apps, die mit Hilfe von AI nicht nur Fragen beantworten, sondern völlig autonom Aufgaben erledigen können. Hierbei generieren sie nicht nur Text, sondern haben auch Zugriff auf Tools und Daten. Dies lässt sie mit anderen Systemen kommunizieren und so ganze Geschäftsprozesse abarbeiten. Und das Ganze allein oder zusammen in cross-funktionalen Agenten-Teams! Jörg Neumann führt in das Thema ein und zeigt euch, wie ihr intelligente Agentensysteme mit .NET und Semantic Kernel entwickelt.</p>                                                           |  <p><b>SPAs mit Blazor 10.0 und C# 14.0</b><br/><i>Dr. Holger Schwichtenberg</i></p> <p>Microsoft hat die Einsatzszenarien von Blazor durch Blazor Static-Server-Rendering und Auto-Modus mit Umschaltung zwischen Blazor Server und Blazor WebAssembly erheblich erweitert. Im Workshop präsentiert DOTNET-DOKTOR Holger Schwichtenberg den Stand von Blazor in Version 10.0 und stellt die verschiedenen Blazor-Architekturmodelle mit ihren Vor- und Nachteilen gegenüber. Danach wird er die Teilnehmer:innen anleiten, selbst eine Blazor 10.0-Anwendung zu entwickeln, die serverseitig vorrendert, dann Blazor Server nutzt und im Hintergrund automatisch die Dateien für Blazor WebAssembly herunterlädt (Progressive Enhancements).</p> |  <p><b>EF Core 10 – Das Datenmodell im Griff</b><br/><i>Christian Giesswein</i></p> <p>EF Core 10 bringt wieder einige spannende Neuerungen mit – und dieser Workshop zeigt dir, wie du sie in der Praxis sinnvoll einsetzt. Statt oberflächlichem Feature-Hopping gehen wir tief rein: Modellierung, Migrationen, Performance, komplexe Abfragen, Multi-Tenancy, Interceptors, JSON-Spalten, Testing – der Workshop bietet nahezu alles, was du brauchst, um EF Core produktionsreif zu beherrschen.</p>                                                                                                                                                                                     |
|                     |  <p><b>C# als Fremdsprache</b><br/><i>Thorsten Kansy</i></p> <p>Der Workshop richtet sich an all diejenigen, die modernes C# schnell und effizient lernen wollen und bereits in anderen Programmiersprachen zu Hause sind. Also Entwickler:innen, denen die Konzepte der Programmierung bekannt sind und nun die Erklärungen von for-Schleifen, if-Statements und vieles andere mehr kennenlernen wollen.</p> <p>Thorsten Kansy stellt bei dieser kompakten Druckbetankung alles wichtige vor, ohne sich in Details zu verlieren und ohne die Praxis aus den Augen zu verlieren.</p>                                                                                                                                   |  <p><b>CI/CD-Pipelines mit Azure DevOps meistern</b><br/><i>Marc Müller</i></p> <p>Beginnend mit modularisierten YAML-Pipelines und Configuration-as-Code zur Optimierung von Workflows lernst du, Docker-Container für effiziente Build- und Laufzeitumgebungen zu integrieren, um Portabilität und Konsistenz zu gewährleisten. Automatisierte Sicherheitsprüfungen, bei denen Code und Abhängigkeiten analysiert werden, um Schwachstellen zu beheben, verbessern den CI-Prozess. Weitere Themen: Automatisierte Bereitstellung von Artefakten, Einbettung von Sicherheitspraktiken für die Agentenkonfiguration, Verwaltung von Anmeldeinformationen, Blue-Green- und Canary-Deployments für nahtlose Updates mit minimalen Ausfallzeiten.</p> |  <p><b>Von Doing Agile zu Being Agile</b><br/><i>Ina Einemann</i></p> <p>In vielen Teams läuft Scrum, aber niemand redet offen. Jira ist voll, aber keiner fühlt sich verantwortlich. Dailys passieren, aber bewegen nichts. Kommt dir bekannt vor? Dann ist dieser Workshop für dich. Denn Agilität ist mehr als Sprints und Dailys. Sie zeigt sich vor allem in unserer Haltung. Denn machen wir uns nichts vor: Jira ersetzt keine Haltung. Und „Scrum machen“ reicht nicht, wenn Mut, Offenheit oder echte Zusammenarbeit fehlen. Wir schauen uns an, was echte Agilität bedeutet und du als Developer dafür tun kannst. Ohne auf den großen Kulturwandel zu warten. Sondern einfach bei dir anzufangen.</p>                                |  <p><b>Raus aus .NET-Brownfield. Migrationsstrategien für Entwickler:innen und Architekt:innen</b><br/><i>Oliver Sturm</i></p> <p>Alte Software vollständig zu ersetzen ist selten realistisch. Doch am Status quo festzuhalten bringt uns auch nicht weiter. Oliver Sturm zeigt, wie eine neue, moderne Architektur und ein zukunftsfähiger Entwicklungsansatz entstehen können – trotz bestehender Abhängigkeiten, eingeschränkter Ressourcen und zahlreicher Altlasten. Es geht nicht um kosmetische Reparaturen am Bestandscode, sondern um fundierte Neuentwicklung – mit realistischer Rücksicht darauf, was vorhanden ist und mit dem auch eine neue Lösung umgehen können muss.</p> |



Die Programmdetails sind zu finden unter:

[developer-world.de/ddc/ddc-programm](https://developer-world.de/ddc/ddc-programm)

# .NET DEVELOPER CONFERENCE

KONFERENZ – DIENSTAG, 25. NOVEMBER 2025

| UHRZEIT       | FRONTEND DEVELOPMENT                                                                                                                                                               | GENAI & AGENTIC                                                                                                                                     | DATA & ANALYTICS                                                                                                                                                               | SOFTWARE ARCHITECTURE, TESTING, AND QUALITY                                                                                                                                 |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 08.00 – 09.00 | Registrierung, Kaffee, Networking                                                                                                                                                  |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 09.00 – 10.00 | Keynote 1: Agentic AI Development with GitHub Copilot, Daniel Meixner, GitHub                                                                                                      |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 10.00 – 10.30 | Kaffeepause und Besuch der Expo                                                                                                                                                    |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 10:30 – 11:30 | Moderne Cross-Plattform-Applikationen mit .NET und Avalonia UI<br><i>Roland König</i>                                                                                              | In geheimer Mission: AI Agents mit .NET und OpenAI entwickeln<br><i>Jörg Neumann</i>                                                                | SQL Server 2025 – eine kritische Würdigung<br><i>Thorsten Kansy</i>                                                                                                            | Architektur per Knopfdruck<br><i>David Tielke</i>                                                                                                                           |
| Raumwechsel   |                                                                                                                                                                                    |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 11.40 – 12.40 | SaaS-Startup mit .NET, ASP.NET Core & Blazor – Der Traum, der Stack und die Realität<br><i>Christian Giesswein</i>                                                                 | Building an Agentic RAG Pipeline<br><i>Francesco Esposito</i><br> | Vektordatenbanken verstehen und optimal einsetzen: Was hinter dem Hype der semantischen Suche steckt<br><i>Gregor Biswanger</i>                                                | Webanwendungen testen mit Microsoft Playwright<br><i>Dr. Holger Schwichtenberg</i>                                                                                          |
| 12.40 – 14.00 | Mittagspause und Besuch der Expo                                                                                                                                                   |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 14.00 – 15.00 | Auf welche Web-UI Technologie setzt Microsoft? Ist das React?<br><i>Harald Binkle</i>                                                                                              | Getting Started with MCP in .NET<br><i>Michael Staib</i>                                                                                            | EF Core migrieren<br><i>angefragt</i>                                                                                                                                          | Die richtige Frontend-Architektur<br><i>angefragt</i>                                                                                                                       |
| Raumwechsel   |                                                                                                                                                                                    |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 15.10 – 16.10 | GUI-Frameworks für .NET: Die Qual der Wahl<br><i>Dr. Holger Schwichtenberg</i>                                                                                                     | LLMs demystified<br><i>angefragt</i>                                                                                                                | Unleashing the Power of Vector Search with Azure SQL Database<br><i>Taswar Bhatti</i><br> | Telemetrie und Distributed Tracing mit OpenTelemetry für ASP.NET Core Microservices<br><i>Marc Müller</i>                                                                   |
| 16.10 – 16.40 | Kaffeepause und Besuch der Expo                                                                                                                                                    |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 16.40 – 17.40 | Infusing local AI in your WPF and WinUI apps using Windows AI Foundry<br><i>Niels Laute</i><br> | RAG-tastisch: Der neue Standard in der intelligenten Informationssuche<br><i>Thomas Tomow</i>                                                       | Next-Gen AI Insights: Native Vektorsuche mit dem SQL Server 2025<br><i>Torsten Strauß</i>                                                                                      | Build event-driven systems with Azure<br><i>Sofía Platas</i><br><i>Andrea Díez</i><br> |
| Raumwechsel   |                                                                                                                                                                                    |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 17.50 – 18.50 | Creating Data-Driven .NET MAUI Apps<br><i>Mark Allibone</i>                                                                                                                        | Build AI with Azure AI Foundry<br><i>Codrina Merigo</i><br>      | KI-gestützte Datenanalysen unter .NET<br><i>angefragt</i>                                                                                                                      | Taktisches Refactoring von .NET Legacy Code<br><i>Gregor Biswanger</i>                                                                                                      |
| Raumwechsel   |                                                                                                                                                                                    |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 19.00 – 20.00 | Night Session: SignalRC – Wenn WebSockets auf Allrad treffen                                                                                                                       |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |
| 20.00 – 22.00 | Networking, Fingerfood & Drinks                                                                                                                                                    |                                                                                                                                                     |                                                                                                                                                                                |                                                                                                                                                                             |



## KONFERENZ – MITTWOCH, 26. NOVEMBER 2025

| UHRZEIT       | .NET FUNDAMENTALS & BEST PRACTICES                                                                                                                                           | APPROACHES                                                                                                                                                                                       | BACKEND TECHNOLOGIES                                                                                                                                                                                           | CLOUD DEVELOPMENT                                                                                                                                                                     |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 08.00 – 09.00 | Registrierung, Kaffee, Networking                                                                                                                                            |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| 09.00 – 10.00 | Keynote 2: tba                                                                                                                                                               |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| 10.00 – 10.30 | Kaffeepause und Besuch der Expo                                                                                                                                              |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| 10.30 – 11.30 | <b>.NET 10 – Zwei Leute lieben's, der Rest wartet auf .NET 11</b><br><i>Christian Giesswein</i>                                                                              | <b>Meet Your New Teammates: Wie GitHub Copilot Agents die Softwareentwicklung neu definiert</b><br><i>Michael Kaufmann</i>                                                                       | <b>.NET MAUI and . NET Aspire: a beautiful combination</b><br><i>Johan Smarius</i><br>                                      | <b>.NET 10 für verteilte Cloud- &amp; Enterprise-Anwendungen</b><br><i>Marc Müller</i>                                                                                                |
| Raumwechsel   |                                                                                                                                                                              |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| 11.40 – 12.40 | <b>AI Driven Development (ADD) in .NET</b><br><i>David Tielke</i>                                                                                                            | <b>How to become a software hero? – Tips and tricks from the trenches</b><br><i>Ivo Verburgh</i><br>          | <b>Minimal API 2.0: Advanced Patterns and Best Practices for Lightweight Web Services</b><br><i>Kajetan Duszyński</i><br> | <b>Next-Gen Cloud Native .NET Apps: Aspire and Dapr Unleashed</b><br><i>Florian Lenz</i>                                                                                              |
| 12.40 – 14.00 | Mittagspause und Besuch der Expo                                                                                                                                             |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| 14.00 – 14.45 | DDC Panel: Developer Productivity                                                                                                                                            |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| Raumwechsel   |                                                                                                                                                                              |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| 15.00 – 16.00 | <b>How to build well-designed and reusable NuGet packages</b><br><i>Dennis Doomen</i><br> | <b>Scrum Master 2.0: Ist KI der bessere Facilitator?</b><br><i>Miriam Hunger</i>                                                                                                                 | <b>Web-Security, up-to-date: Die OWASP Top Ten 2025</b><br><i>Christian Wenz</i>                                                                                                                               | <b>Building next-gen applications with event-driven architectures</b><br><i>Teena Idnani</i><br> |
| 16.00 – 16.30 | Kaffeepause und Besuch der Expo                                                                                                                                              |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| 16.30 – 17.30 | <b>Mono Repo in .NET: Should You Use It or Avoid It?</b><br><i>Emanuele Bartolesi</i><br> | <b>Migrating from a monolith to a new service</b><br><i>Connell Sharp</i><br>                                 | <b>Building Ultra-Fast, Globally Distributed Apps with Serverless at the Edge</b><br><i>Florian Lenz</i>                                                                                                       | <b>Cosmos DB and Azure Functions</b><br><i>Johan Smarius</i><br>                                 |
|               | Raumwechsel                                                                                                                                                                  |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |
| 17.40 – 18.40 | <b>Writing a .NET profiler in C# with Silhouette</b><br><i>Kevin Gosse</i><br>            | <b>Beyond the Code: Can GitHub Copilot Be Your Co-Architect? Yuliya Khadasevich</b><br><i>Liuba Gonta</i><br> | <b>Taking Entity Framework Core data seeding to the next level with Bogus</b><br><i>Sander ten Brinke</i><br>             | <b>The 7 Deadly Cloud Sins (and How to Avoid Them)</b><br><i>Ruby Jane Cabagnot</i><br>          |
| 18.40 – 20.00 | Feierabendbier                                                                                                                                                               |                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                       |



Die Programmdetails sind zu finden unter:

[developer-world.de/ddc/ddc-programm](https://developer-world.de/ddc/ddc-programm)

# .NET DEVELOPER CONFERENCE

DEVSESSIONS – DONNERSTAG, 27. NOVEMBER 2025

| Uhrzeit             | DevSession 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | DevSession 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | DevSession 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | DevSession 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 09.00<br>–<br>13.00 | <b>Agentic AI explained</b><br><i>angefragt</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | <br><b>Automatisiertes Testen – die schwierigen Teile</b><br><i>Florian Winkelbauer</i><br>Einführende Tutorials für automatisierte Testframeworks lassen es einfach aussehen: Man schreibt etwas Code, erstellt ein paar Tests, und fertig. Wenn man sich allerdings mit Produktionscode befasst, stößt man nicht selten auf größere Herausforderungen und lernt, dass die Realität anders aussieht. Wir lernen solche Herausforderungen kennen und erarbeiten Lösungen für die Erstellung einer Reihe automatisierter Tests in C#. Lerne auf diese Weise neue Aspekte der Denkweise in Bezug auf Code kennen, probiere verschiedene Teststrategien aus und entdecke Techniken, um Tests lesbar und unterhaltsam zu gestalten. | <br><b>Moderne Datenzugriffsschicht mit GraphQL – ein pragmatischer Einstieg</b><br><i>Oliver Sturm</i><br>Die DevSession knüpft inhaltlich an den Workshop „Raus aus dem .NET-Brownfield“ an. Ihr Fokus liegt auf der konkreten Umsetzung: Oliver zeigt, wie sich mit GraphQL eine flexible, moderne Datenschnittstelle über bestehende .NET-Datenmodelle legen lässt – als strategisches Bindeglied zwischen Alt und Neu. Wir entwickeln ein schlankes und robustes Setup, das sich für erste Modernisierungsschritte im Brownfield eignet und Grundlage für größere Architekturvorhaben ist. Keine Framework-Schlachten, keine Komplettlösungen – einfach nur praktikable Techniken, die sich in echten Projekten bewährt haben. | <br><b>Mein eigener Kubernetes-Cluster mit ASP.NET Core &amp; Blazor</b><br><i>Christian Giesswein</i><br>Christian zeigt dir, wie du deinen eigenen Kubernetes-Cluster ohne teure Cloud-Abos aufsetzt, darauf produktionsreif eine moderne Webanwendung mit ASP.NET Core & Blazor hostest und warum du das überhaupt willst. Themen sind unter anderem: Kubernetes selbst hosten   Load Balancing, TLS & Ingress   CI/CD für .NET & Blazor   ASP.NET Core & Blazor Server / WebAssembly im Cluster   Logging, Monitoring, Updates<br>Die DevSession richtet sich an all diejenigen, die keine Lust auf Black-boxes in der Cloud haben und verstehen wollen, was unter der Haube passiert.                                  |
| 14.00<br>–<br>18.00 | <br><b>Azure meets Containers: Maximiere deine Cloud-Performance</b><br><i>Florian Bader</i><br>Möchtest du erfahren, wie du deine Anwendungen in Azure mit Containern betreiben kannst? In dieser DevSession lernst du verschiedene Möglichkeiten kennen, Container in Azure zu deployen und zu skalieren. Es werden die Vor- und Nachteile von Containern in der Cloud erläutert und die Frage beantwortet, welcher Azure Service sich besonders gut für welche Anforderungen eignet. Außerdem stelle ich dir verschiedene Anwendungsszenarien vor, in denen der Einsatz von Containern sinnvoll sein kann. Nimm teil und erfahre so mehr über die Möglichkeiten, die Container in Azure bieten. | <br><b>.NET Aspire in Action – Jenseits von „Hello World“</b><br><i>Florian Lenz</i><br>Florian Lenz zeigt dir, wie du mit .NET Aspire moderne, cloud-native .NET-Anwendungen entwickelst, die weit über einfache Demo-Beispiele hinausgehen. Statt „Hello World“ warten echte Architekturmuster, praktische Tipps aus dem Projektalltag und Insights, die dir helfen, Aspire produktiv einzusetzen. Dabei nimmt dir .NET Aspire komplexe Infrastrukturthemen ab, bringt Observability, Cloud-Readiness und modulare Architektur out of the box mit und sorgt so dafür, dass du dich auf die Entwicklung von Business-Logik und schnelle Ergebnisse konzentrieren kannst.                                                     | <br><b>Von WPF zum Cross-Plattform UI Framework Avalonia</b><br><i>Roland König</i><br>WPF galt lange Zeit als Standard, wenn es um die Entwicklung von (Windows) Desktop-Applikationen mit C# ging. Selbst heute ist das Framework trotz der geringen Weiterentwicklung durch Microsoft und der Plattformbindung an Windows noch sehr beliebt. Dabei ist es höchste Zeit, über mögliche Alternative Frameworks nachzudenken. Eine dieser Alternativen ist Avalonia. Es weist viele Ähnlichkeiten zu WPF auf, setzt auf XAML, bevorzugt MVVM, bringt einen visuellen Designer mit, läuft stabil auf allen Plattformen und bietet unter anderem ein CompiledBinding.                                                               | <br><b>Architecting and Developing Custom Agents and Copilots</b><br><i>Dr. Damir Dobric</i><br>Als Software Architect oder Developer bist du gefordert, Anwendungen zu entwickeln, die Large Language Modelle (LLM) nutzen. Ich zeige dir, wie LLMs und GenAI unter der Haube funktionieren. Wir beginnen dazu mit wesentlichen Grundlagen und tauchen anschließend tief in Token-Generierung, Vector Similarity, Embeddings, Modelltypen, Plugins, Agents & Co. ein. Du erfährst, wie sich AI-Agents erstellen lassen, die intelligenten Code in jeder Anwendung ausführen, lernst Semantic Kernel kennen und integrierst AI in die Azure-Plattform, um leistungsfähige Anwendungen und mächtige Backends zu erstellen. |

## LOW CODE UND KÜNSTLICHE INTELLIGENZ

# Das Dreamteam

Warum die Kombination aus Low Code und künstlicher Intelligenz enorme Vorteile für die Softwareentwicklung bietet. Ein Interview mit Mendix-CEO Raymond Kok.

**D**ass Low-Code-Ansätze die Softwareentwicklung verändert haben, ist fast schon ein alter Hut, denn: Wo klassischer Code komplexe Hürden schafft, bieten modellbasierte Ansätze wie Low Code Raum für Vereinfachung. Einen enormen Schub erfuhr Low Code durch den rasant wachsenden Einfluss künstlicher Intelligenz. Deren Integration schafft gleich in mehrfacher Hinsicht Mehrwerte: Nicht nur werden Entwicklungsprozesse beschleunigt und entsteht dank KI eine neue Form intelligenter Softwareanwendungen. Vielmehr schafft sie die Grundlage für mehr Effizienz, engere Zusammenarbeit und innovativere Produkte und sorgt außerdem für eine gewisse Demokratisierung der Softwareentwicklung.

Raymond Kok, langjähriger CTO der Siemens-Sparten Cloud & Edge und Digital Industries und seit 2024 CEO von Mendix, Marktführer im Bereich Low Code [1], sprach mit dotnetpro über die Verzahnung von Low Code und KI. Mit der Mendix Artificial Intelligence Assistance, kurz Maia [2], bietet die Siemens-Tochter einen Pair-Developer an, der unter anderem Kontext-Empfehlungen für Logik und UI-Modelle oder Unterstützung beim Erstellen von Domain-Modellen, Microflows oder Seiten bereitstellt [3] und so die Softwarequalität steigert und die Entwicklerproduktivität erhöht.

**dotnetpro:** Wie transformiert KI die Rolle der Softwareentwicklung im Allgemeinen und die Rolle von Low Code im Besonderen?

**Raymond Kok:** Künstliche Intelligenz ist das Beste, was dem Low-Code-Business je passiert ist. Sie ist ein weiterer Beleg dafür, dass klassischer Code nicht die richtige Darstellungsform ist, um Anwendungen der nächsten Generation zu entwickeln. Low Code und KI werden sich zunehmend verbinden und die nächste Evolutionsstufe der Developer Experience hervorbringen. Gleichzeitig gilt: Auch wenn KI die Art und Weise, wie wir Software entwickeln, zweifellos verändert, gibt es nach wie vor grundlegende Herausforderungen in der Unternehmenssoftwareentwicklung. Als Mendix vor zwanzig Jahren an den Start ging, war unser Ziel, mit Low Code die enorme Kluft zwischen dem Business – also den Nutzenden der Software – und der IT – den Entwicklern – zu überbrücken. Heute ist eine dritte Partei hinzugekommen –

die Menschen, die sich um die Daten kümmern, auf denen moderne Software basiert, und die sie organisieren.

In dieser dynamischen neuen Welt ist es entscheidend, herauszufinden, wie Business, IT und Data Scientists effektiv zusammenarbeiten können. Andernfalls laufen wir Gefahr, in ein altbekanntes Wasserfallmodell zurückzufallen: Das Business formuliert seine Anforderungen und übergibt sie an ein KI- oder Data-Science-Team. Dieses entwickelt daraufhin eine Lösung – etwa einen Agenten –, der auf einem API basiert, das sich letztlich anders verhält als erwartet. Das führt zu endlosen Iterationen, bis der Agent endlich so funktioniert, wie es das Business braucht. Genau an diesem Punkt befinden wir uns heute – und es braucht jemanden, der diese Barrieren erneut überwindet.

**„Künstliche Intelligenz ist das Beste, was dem Low-Code-Business passieren konnte.“**

*Glauben Sie, dass sich der traditionelle Software Development Lifecycle aufgrund von Low Code und KI grundlegend verändert?*

**Kok:** KI ist eine disruptive Kraft, die traditionelle Workflows im Software Development Lifecycle

(SDLC) verändert. Ein Beispiel dafür ist der wachsende Markt KI-getriebener App-Generatoren. Die Idee dahinter: Solche Tools erzeugen mit erstaunlicher Geschwindigkeit funktionsfähige Anwendungen – und Entwickler müssen diese nur noch feinjustieren, bevor sie live gehen. Das Problem liegt weniger in der Verfügbarkeit von Daten, sondern vielmehr darin, dass solche Generatoren zwar gut für den Einstieg und für einfache Anwendungen mit klaren Anforderungen geeignet sind, aber bei komplexeren Szenarien oder unklaren Zielbildern an ihre Grenzen stoßen.

Wir nähern uns KI im SDLC auf ganzheitlichere Weise. Statt „AI First“ heißt es bei uns: „AI Always“. Wir integrieren KI strategisch an allen kritischen Punkten des SDLC – von der Entwicklung bis zum Betrieb –, um den gesamten Prozess zu beschleunigen. Mendix-Nutzer profitieren dabei doppelt: zum einen durch KI-Funktionalitäten im Entwicklungsprozess selbst, zum anderen durch die Möglichkeit, intelligente Anwendungen zu erstellen, die ihrerseits KI-Funktionen für die Endanwender bereitstellen.

*Wie verändert sich die Rolle von Softwareentwicklern in einer Welt, die zunehmend von KI bestimmt wird?*

**Kok:** Mit der zunehmenden Bedeutung von KI im SDLC werden Entwickler zu Komponisten: Sie orchestrieren die sich weiterentwickelnden Komponenten des SDLC zu einer kohärenten und funktional tragfähigen Architektur. Statt also der beste Programmierer beziehungsweise die beste Programmiererin der Welt zu sein, wird sich die Rolle grundlegend verändern: Gefragt ist nicht mehr das virtuose Spiel auf einem Instrument – sondern die Fähigkeit, das gesamte Orchester zu leiten.

In dieser Umgebung werden vor allem kritische und kreative Köpfe erfolgreich sein. Denn statt sich nur auf das Coden oder die Entwicklung von Anwendungen zu konzentrieren, um die Anforderungen zur Digitalisierung eines einzelnen Prozesses zu erfüllen, geht es künftig darum, ganze End-to-End-Geschäftsprozesse neu zu gestalten – mithilfe eines neuen Orchesters aus KI- und SaaS-gestützten Entwicklungstools, gemeinsam mit Endnutzenden und Dateningenieuren.

Wer mit Code arbeitet, schließt das Business automatisch aus. Modellbasierte Softwareentwicklung mit Low Code ist daher die deutlich bessere Lösung, weil sie den Zugang zur Anwendungsentwicklung demokratisiert – durch eine gemeinsame Sprache und eine gemeinsame Arbeitsumgebung für die, die Software bauen, und die, die sie nutzen. Und: Low Code passt auch viel besser zu dem, wie sich der SDLC, so

wie wir ihn heute kennen, künftig entwickeln wird – hin zu einem „Composer Lifecycle Management“.

*Wo sehen Sie echte Synergien zwischen Low Code und KI, und wie schaffen diese einen Mehrwert (für Entwickler)?*

**Kok:** Entwickler profitieren doppelt, wenn KI in eine Low-Code-Plattform integriert ist. Denn sie erhalten alle Vorteile visueller Entwicklung – kombiniert mit KI-Unterstützung genau dort, wo sie gebraucht wird. Ob bei Workflows, der Seitengestaltung oder der Logikentwicklung: KI-Assistenten stehen direkt zur Seite, schlagen nächste Schritte vor und automatisieren wiederkehrende Aufgaben. Da die KI-Funktionen tief in die Plattform eingebettet sind, ist kein Kontextwechsel nötig – man muss nicht erst woanders nach Hilfe suchen. Die passende Orientierungshilfe ist jederzeit griffbereit.

Das Ergebnis: Es muss weniger Zeit für Routineaufgaben aufgewendet werden, wodurch mehr Freiraum für das entsteht, was wirklich zählt: Wertschöpfung und die Lösung von Business-Problemen. Zusätzlich können Entwickler mit Mendix ganz einfach KI-Funktionen in ihre Anwendungen integrieren, sei es durch das Hinzufügen intelligenter Chatbots, die Erstellung von KI-Agenten oder die Entwicklung von GenAI-basierten Funktionen.

*Mit „Maia“ bietet Mendix KI-Funktionen. Aber was genau ist Maia, und welche Bedarfe erfüllt es?*

**Kok:** Maia, kurz für „Mendix AI Assistance“, ist eine Sammlung KI-gestützter Funktionen, die Entwicklungsteams während des gesamten Anwendungsentwicklungszyklus unterstützen. Maia ist tief in die Mendix-Plattform integriert und supportet alle, die an der Anwendungsentwicklung beteiligt sind: Business-Analysten hilft Maia dabei, Anforderungen ►



#### Raymond Kok

Der gebürtige Niederländer Raymond Kok ist seit Februar 2024 CEO von Mendix, einem Tochterunternehmen der Siemens AG. Er besitzt einen Master of Science sowie einen Bachelor of Science in Maschinenbau der Technischen Universität Eindhoven. Seine Laufbahn bei Siemens Digital Industries Software begann Kok im Jahr 2003. Dort war er in verschiedenen technischen Führungsrollen tätig, unter anderem in den Bereichen Cloud Application Solutions und Industrial Operations X. Als früher Anwender der Low-Code-Plattform von Mendix war Kok maßgeblich an den strategischen Gesprächen beteiligt, die 2018 zur Übernahme von Mendix durch Siemens Digital Industries Software führten. Er verfolgt das Ziel, die Position von Mendix als marktführende Low-Code-Plattform für die Umsetzung komplexer Unternehmenssoftware weiter zu stärken.

Foto: Mendix, a Siemens Company



in User Stories und Workflows zu übersetzen. Entwickler erhalten KI-basierte Unterstützung bei der Programmierung, bei Best Practices und der Automatisierung wiederkehrender Aufgaben. Product Owner profitieren von mehr Klarheit in der Anforderungsdefinition und besserer Projektübersicht, während UX-Designer intelligente Vorschläge für Seitenlayouts und Benutzeroberflächen erhalten.

Maia wirkt wie ein vielseitiger digitaler Assistent, der sich an die Bedürfnisse jedes Teammitglieds anpasst – und den gesamten Entwicklungsprozess kollaborativer und effizienter macht. Zudem stellen wir unseren Kunden „Maia for Smart Apps“ zur Verfügung – ein Funktionspaket, das die Entwicklung generativer KI-Anwendungen und intelligenter Lösungen vereinfacht und beschleunigt. Damit können Unternehmen die KI-Einführung vorantreiben und KI-basierte Funktionen nahtlos in ihre Anwendungen integrieren.

Visuelle Entwicklung bleibt das Herzstück unserer Plattform – doch Maia eröffnet Entwicklern eine neue Dimension des Arbeitens: Durch die Kombination aus klassischen Entwicklungsansätzen und Interaktionen in natürlicher Sprache können sie künftig selbst entscheiden, ob sie mit visuellen Tools oder per Conversational Prompts, also dialogartigen Eingaben in natürlicher Sprache, am effizientesten entwickeln und Anpassungen vornehmen.

*Wie funktioniert Maia technisch, und auf welchen Technologien basiert es?*

**Kok:** Maia ist in die Mendix-Entwicklungsumgebung integriert. Es gibt drei spezialisierte Empfehlungssysteme, sogenannte „Recommender“, die man sich als virtuelle Co-Entwickler-Bots vorstellen kann. Da wäre „Maia Logic Recommender“, der beim Erstellen von Modellen und Anwendungslogik hilft, der andere ist „Maia Best Practices Recommender“, der sicherstellt, dass eine App den eigenen Best

Spezialisierte Modelle wie Mistral 7B übernehmen Aufgaben wie das Umschreiben von Abfragen.

Unsere Architektur ist bewusst modellagnostisch aufgebaut – so bleiben wir flexibel und können die besten verfügbaren Technologien kontinuierlich integrieren, wenn sich der KI-Markt weiterentwickelt. Eine vollständige Aufschlüsselung aller eingesetzten Dienste und der damit verbundenen Datenübertragungen sind in unserer Dokumentation zu finden, die für vollständige Transparenz sorgt.

*In welchen Phasen eines Projekts kann Maia konkret hilfreich sein?*

**Kok:** Maia unterstützt alle Phasen eines Projekts. Schon in der Ideenfindung können Entwickler mit einem einfachen Prompt oder einem Anforderungsdokument starten – und innerhalb kürzester Zeit einen ersten Prototyp erstellen. Entscheidend ist jedoch, dass der KI-Support nicht an dieser Stelle aufhört, sondern immer verfügbar ist. Das bedeutet: Auch wenn ein Projekt wächst, bleibt Maia ein zuverlässiger Begleiter – bei der weiteren Anforderungsdokumentation ebenso wie in der Entwicklungsarbeit an komplexen, fortgeschrittenen Anwendungen. Und selbst nach dem Deployment hilft Maia weiter, indem Rückmeldungen smart kategorisiert und konkrete Verbesserungen leicht umsetzbar gemacht werden. Mit „Maia for Smart Apps“ erhalten Entwickler zudem alle Werkzeuge, die sie brauchen, um agentenbasierte Lösungen zu realisieren.

*Wie unterscheiden sich die KI-Funktionen von Mendix und Maia von anderen KI-gestützten Entwicklungstools?*

**Kok:** Maia unterscheidet sich durch ihre modulare Architektur und die tiefe Integration in den gesamten Anwendungsentwicklungszyklus. Während viele Plattformen generative KI-Funktionen mit Schwerpunkt auf Prompting in natürlicher Sprache und automatisierte Code-Empfehlungen haben, basiert Maia auf einem Set spezialisierter Recommender – wie zuvor beschrieben. Dazu zählen KI-gesteuerte Empfehlungen für Microflows, Best Practices und Workflow-Design, jeweils gezielt auf einen bestimmten Aspekt des Modellierungsprozesses ausgerichtet. Dadurch kann Maia stark kontextbezogene, präzise und umsetzbare Vorschläge machen – weit über die generische Codegenerierung hinaus. Ein weiterer entscheidender Unterschied liegt in der Art, wie Maia Governance auf Plattformentebene adressiert. Statt sich nur auf die Produktivität einzelner Entwickler zu konzentrieren, unterstützt Maia ganze Teams dabei, konsistente und qualitativ hochwertige Anwendungen zu liefern – einschließlich Unternehmensanforderungen wie Compliance oder Richtlinien für die Systemarchitektur. Das macht Maia besonders relevant für Organisationen in regulierten Branchen oder mit großen Entwicklungsportfolios.

Maia überzeugt durch ihre tiefe Integration in die Mendix-Plattform und den Fokus auf sichere, richtliniengestützte Entwicklung. Im Gegensatz zu generischen KI-Tools, die un-

## „Gefragt ist nicht mehr das virtuose Spiel auf einem Instrument, sondern die Fähigkeit, das gesamte Orchester zu leiten.“

Practices entspricht. Und schließlich gibt es noch den „Maia Workflow Recommender“, der passende Microflows und zulässige Aktionen innerhalb eines Workflows empfiehlt. Unsere Empfehlungssysteme helfen auch unerfahrenen Entwicklern, Low Code zu verstehen, indem sie ihnen die besten Vorgehensweisen in jeder Phase des Anwendungsentwicklungsprozesses vermitteln.

Aktuell nutzt Maia verschiedene KI-Technologien, die für spezifische Anwendungsfälle optimiert sind. Zum Beispiel kommt Llama 3.1 8B für Chat- und Erklärungshilfen zum Einsatz, Claude über Amazon Bedrock wird für die Domain-Modell-Generierung und Generierung von UI-Seiten verwendet.



Foto: Mendix, a Siemens Company

strukturierten Code generieren, arbeitet Maia innerhalb der visuellen Modellierungsumgebung von Mendix – und erzeugt dort klare, überprüfbare Modelle, die von Haus aus die Sicherheit von Applikationen unterstützen. Diese enge Verzahnung erlaubt es Maia außerdem, kontextgenaue und präzise Hilfestellungen zu geben – immer auf die jeweilige Anwendung zugeschnitten.

Über individuelle Produktivitätsgewinne hinaus hilft Maia Organisationen dabei, Qualität und Compliance im großen Maßstab sicherzustellen: durch die Einhaltung von Architekturstandards, die Unterstützung teamweiter Best Practices und die konsequente Ausrichtung von Anwendungen auf Unternehmensvorgaben. Das macht Maia zu einem echten KI-Assistenten auf Enterprise-Niveau.

*Wie stellt Mendix sicher, dass Maia vertrauenswürdige und sichere Ergebnisse liefert?*

**Kok:** Wie jede KI-Technologie kann auch Maia gelegentlich Halluzinationen erzeugen – also Antworten, die plausibel klingen, aber nicht vollständig korrekt sind. Wir arbeiten kontinuierlich daran, diese Fälle zu minimieren – durch verbessertes Kontextverständnis und gezieltes Modelltraining. Derzeit gibt es noch Einschränkungen hinsichtlich der Menge an Kontext, die Maia auf einmal verarbeiten kann, aber auch hier erweitern wir die Fähigkeiten laufend. Ein großer Vorteil von Maia ist ihre Einbettung in die visuelle Entwicklungsumgebung von Mendix. Dadurch können Entwickler sämtliche KI-generierten Vorschläge einfach prüfen und validieren, bevor sie implementiert werden. Dieser visuelle Ansatz schafft ein zusätzliches Maß an Sicherheit im KI-unterstützten Entwicklungsprozess.

Mendix verfolgt einen mehrschichtigen Ansatz, um die Zuverlässigkeit von Maia sicherzustellen. Erstens bewegt sich Maia innerhalb definierter Leitplanken der Plattform, die unangemessene oder unsichere Vorschläge verhindern. Zwei-

tens ist Maia auf Mendix-spezifisches Wissen abgestimmt – so stellen wir sicher, dass alle Empfehlungen den Best Practices der Plattform entsprechen. Und drittens – und besonders wichtig: Da Maia vollständig in die visuelle Entwicklungsumgebung eingebettet ist, können alle Vorschläge vor der Umsetzung leicht überprüft und nachvollzogen werden.

Diese Kombination aus Kontrolle und Transparenz ermöglicht es Teams, KI-gestützte Entwicklung vertrauensvoll zu nutzen – ohne Abstriche bei Qualität oder Sicherheit der Anwendung.

*Wie wird Maia weiterentwickelt – was sind die nächsten Schritte auf der Roadmap?*

**Kok:** Wir arbeiten daran, den kommenden Entwicklungen immer einen Schritt voraus zu sein und sicherzustellen, dass

wir plattformweite GenAI-Services bereitstellen, die für Mendix-Anwender von echtem Wert sind und ihnen KI-Funktionen zugänglicher machen und Werkzeuge an die Hand geben, die sie in der Praxis anwenden können. Wir entwickeln die Fähigkeiten von Maia stetig weiter, insbesondere im Hinblick auf agentenbasierte KI.

Die Entwicklung von KI-Agenten ist eine demokratisierte Praxis oder sollte es zumindest sein: Professionelle Entwickler auf der IT-Seite und Endanwender werden sich aktiv an der Entwicklung maßgeschneiderter Agenten beteiligen. Wir konzentrieren uns also darauf, diese Funktionalität so weiterzuentwickeln, dass sie für die verschiedenen Nutzergruppen (Rollen) im SDLC zugänglich ist.

Die Einführung neuer Technologien im Unternehmen ist herausfordernd – eine reibungslose, intuitive UX ist daher entscheidend für die Akzeptanz. Sie muss funktionieren – und so selbsterklärend sein, dass auch Fachbereiche sie problemlos nutzen können. In diesem Moment widmet sich ein großer Teil unseres F&E-Teams der Integration von KI-Beschleunigern in die Plattform, um sicherzustellen, dass unsere Kunden von beiden Welten profitieren: der Geschwindigkeit und Zugänglichkeit von Low Code in Kombination mit KI-gesteuerter Automatisierung. Unsere Philosophie ist klar: Wir nutzen Low Code, um die Zusammenarbeit zwischen Fachbereichen und IT zu demokratisieren – und KI, um so viele manuelle Aufgaben wie möglich zu automatisieren. ■

*Interview: Fernando Schneider*

- [1] Mendix-Pressemitteilung,  
[www.mendix.com/blog/gartner-mq-lcap-2025/](http://www.mendix.com/blog/gartner-mq-lcap-2025/)
- [2] Mendix AI-Assisted Development,  
[www.mendix.com/evaluation-guide/app-lifecycle/develop/ai/mendix-ai-assistance/](http://www.mendix.com/evaluation-guide/app-lifecycle/develop/ai/mendix-ai-assistance/)
- [3] Mendix AI Assistance (Maia),  
<https://docs.mendix.com/refguide/mendix-ai-assistance/>

## MICROSOFT FABRIC ALS DATENBACKBONE FÜR GENAI-ANWENDUNGEN

# Von Daten zu Antworten

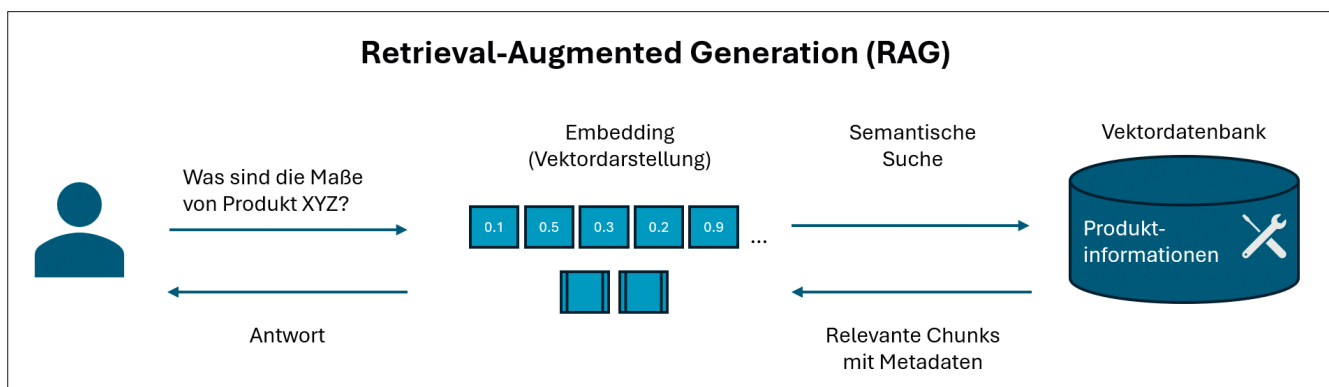
RAG, Vektorsuche und Agenten: Warum moderne GenAI-Projekte eine starke Datenplattform brauchen und wie Microsoft Fabric sie liefert.

Wer GenAI ernsthaft nutzen will, muss bei den Daten anfangen. Ohne strukturierte, kontextreiche und zugängliche Informationen bleibt selbst das leistungsfähigste Sprachmodell blind für das Wissen im Unternehmen. Dieser Artikel zeigt, wie Microsoft Fabric als zentrale Datenplattform die Verbindung zwischen moderner Datenstrategie und innovativen KI-Anwendungen schafft.

Die rasante Entwicklung generativer KI hat in den letzten Jahren neue Maßstäbe gesetzt. Sprachmodelle wie GPT-4

gezielt Inhalte aus einer externen Wissensbasis abrufen. Diese Inhalte werden dem Modell als zusätzlicher Kontext mitgegeben. So entsteht eine hybride Architektur, in der das Sprachmodell nicht mehr nur auf sein internes Wissen angewiesen ist, sondern gezielt auf unternehmensspezifische Informationen zugreifen kann.

Damit dieses Prinzip funktioniert, braucht es eine leistungsfähige Datenplattform, die sowohl strukturierte als auch unstrukturierte Daten effizient verwalten, transformieren und



Retrieval-Augmented Generation zur semantischen Suche (Bild 1)

oder Mistral erzeugen beeindruckend natürliche Texte, beantworten komplexe Fragen und unterstützen bei der Automatisierung von Geschäftsprozessen.

Doch trotz ihrer Leistungsfähigkeit bleiben diese Modelle in einem entscheidenden Punkt limitiert: Sie verfügen nicht über aktuelles, unternehmensspezifisches Wissen. Ihre Antworten basieren auf Trainingsdaten, die oft veraltet oder zu allgemein sind, um im Unternehmenskontext verlässlich zu funktionieren.

Damit generative KI in der Praxis echten Mehrwert liefert, braucht sie Zugriff auf relevante, strukturierte und kontextreiche Daten. Erst wenn ein Modell mit den richtigen Informationen angereichert wird, kann es präzise, nachvollziehbare und vertrauenswürdige Antworten geben. Die Qualität dieser Daten entscheidet über die Qualität der KI. Wer GenAI ernsthaft nutzen will, muss bei den Daten anfangen.

Ein vielversprechender Ansatz, um Sprachmodelle mit aktuellem Wissen zu versorgen, ist Retrieval-Augmented Generation. Dabei wird das Modell nicht allein gelassen, sondern durch eine Suchkomponente ergänzt, die bei jeder Anfrage

bereitstellen kann. Microsoft Fabric bietet genau das. Mit OneLake als zentralem Datenspeicher, Lakehouse und Eventhouse als Analyse-Engines sowie integrierten Tools für Datenpipelines, Transformationen und Machine Learning stellt Fabric eine durchgängige Umgebung bereit, um Daten für GenAI-Anwendungen nutzbar zu machen.

Dieser Artikel zeigt, wie sich mit Microsoft Fabric eine vollständige RAG-Pipeline aufbauen lässt. Von der Datenaufnahme über die Vektorisierung bis hin zur Integration in intelligente Agenten und Workflows wird Schritt für Schritt erläutert, wie Daten aufbereitet und in generative KI-Anwendungen eingebunden werden können.

## Warum GenAI ohne Kontext nicht gut funktioniert

Generative KI, insbesondere in Form von sogenannten Large Language Models (LLMs), ist in der Lage, Texte zu verfassen, Fragen zu beantworten, Code zu generieren oder komplexe Inhalte zusammenzufassen. Diese Modelle basieren auf Milliarden von Parametern und wurden mit gewaltigen Mengen

an Textdaten trainiert. Ihre Stärke liegt in der Fähigkeit, Sprache zu verstehen und kontextbezogen zu reagieren.

Doch genau hier liegt auch eine zentrale Schwäche. Denn ein LLM kennt nur das, was es beim Training gesehen hat. Es hat keinen Zugriff auf aktuelle Informationen, keine Verbindung zu unternehmensspezifischen Datenquellen und keine Möglichkeit, neues Wissen selbstständig zu erschließen. In der Praxis führt das dazu, dass Antworten oft veraltet, ungenau oder schlicht erfunden sind. Dieses Phänomen ist als Halluzination bekannt und stellt ein zentrales Problem beim produktiven Einsatz von generativer KI dar.

Die Idee liegt nahe, das Modell einfach mit neuen Daten nachzutrainieren. Doch in der Praxis ist dies selten sinnvoll. Das Nachtrainieren eines LLM ist technisch aufwendig, teuer und erfordert spezialisierte Infrastruktur. Zudem besteht das Risiko, dass das Modell dabei bestehendes Wissen verliert oder unerwünschte Verzerrungen übernimmt. Vor allem aber ist das Nachtrainieren nicht dynamisch. Neue Informationen lassen sich nicht kurzfristig einpflegen, was gerade in sich schnell ändernden Geschäftskontexten problematisch ist.

das Modell abdriftet oder irrelevante Inhalte verarbeitet. Der Prompt ist kein Speicherort für Wissen, sondern ein situativer Kontextgeber.

Um diese Einschränkungen zu überwinden, wurde das Konzept der Retrieval-Augmented Generation (RAG) entwickelt [1]. Dabei wird das Sprachmodell durch eine semantische Suchkomponente ergänzt (siehe Bild 1).

Wenn ein Nutzer eine Frage stellt, wird diese in einem ersten Schritt in eine Suchanfrage umgewandelt. Die Suche liefert dann relevante Inhalte aus einer externen Wissensquelle, etwa aus Dokumenten, Datenbanken oder Logs. Diese Inhalte werden dem Modell als zusätzlicher Kontext mitgegeben, bevor es eine Antwort generiert. Das Modell bleibt also generativ, aber es antwortet auf Basis von aktuellen, überprüfbaren Informationen.

Damit eine solche Architektur funktioniert, sind mehrere technische Schritte erforderlich. Sie reichen von der Zerlegung der Ausgangsdokumente über die semantische Aufbereitung bis hin zur Speicherung und Abfrage der Daten. Microsoft Fabric bietet für all diese Schritte eine integrierte

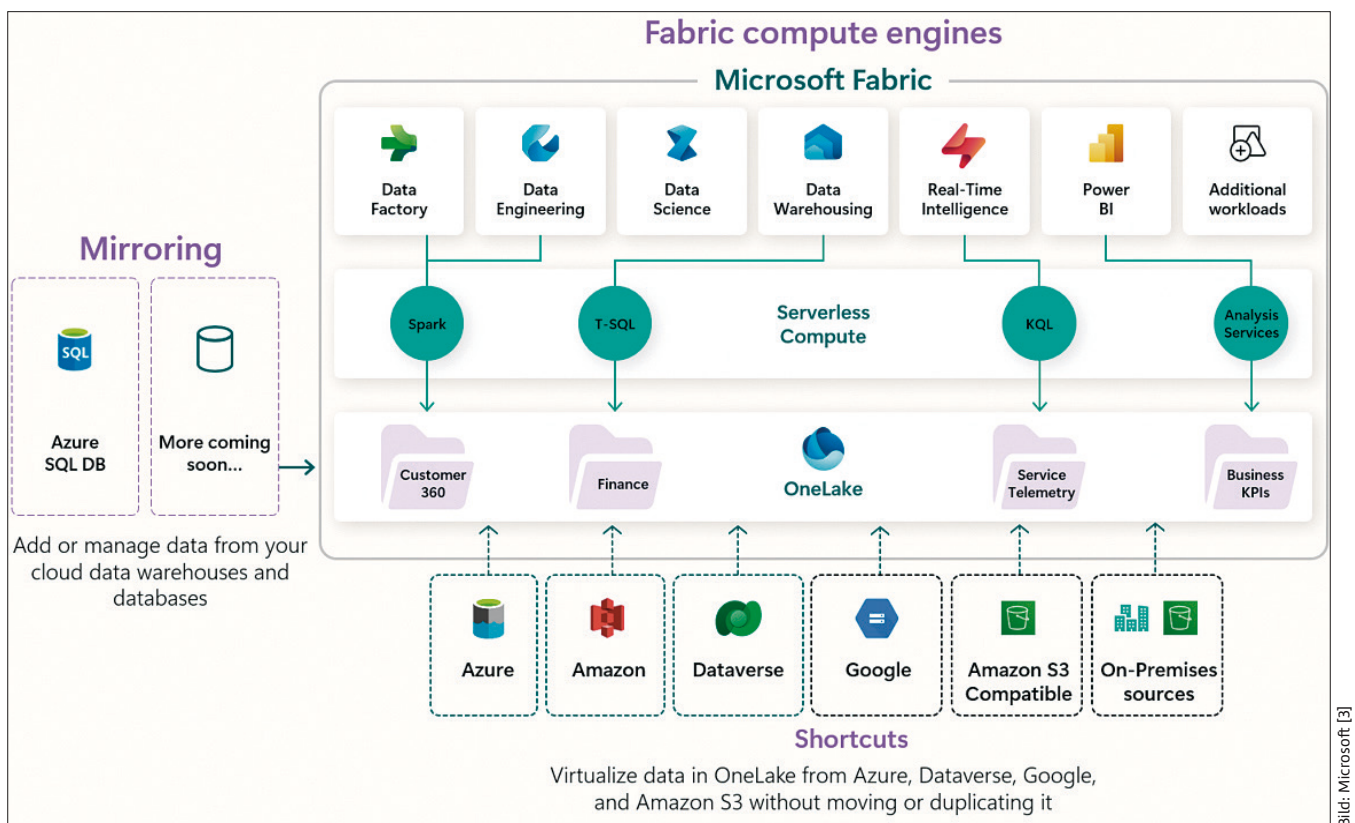


Bild: Microsoft [3]

**Microsoft Fabric Foundation** mit OneLake als zentralem Datenspeicher (Bild 2)

Ein anderer Ansatz besteht darin, möglichst viele Daten direkt in den Prompt des Modells zu geben. Doch auch das ist keine tragfähige Lösung. Prompts unterliegen technischen Begrenzungen hinsichtlich ihrer Länge, und je mehr Kontext man einfügt, desto schwieriger wird es für das Modell, die relevanten Informationen korrekt zu gewichten. Zudem steigt mit der Menge an Kontext auch die Wahrscheinlichkeit, dass

Plattform, die es ermöglicht, den gesamten Prozess konsistent und skalierbar umzusetzen. Von der Datenaufnahme über die Transformation bis hin zur Speicherung und Analyse stehen alle notwendigen Werkzeuge zur Verfügung. Besonders relevant sind dabei OneLake als zentraler Datenspeicher und Lakehouse sowie Eventhouse als Analyse-Engines. In Kombination mit Azure OpenAI, Azure AI Foundry oder anderen ►



Services zum Betreiben von eigenen Modellen lassen sich so leistungsfähige RAG-Pipelines aufbauen, die generative KI mit unternehmensspezifischem Wissen versorgen.

## Die Datenbasis für intelligente Systeme: Microsoft Fabric im Überblick

Microsoft Fabric [2] ist eine moderne Datenplattform, die darauf ausgelegt ist, Datenintegration, Datenspeicherung, Datenverarbeitung und Datenanalyse in einer einheitlichen Umgebung zusammenzuführen (siehe Bild 2 und [3]). Für Unternehmen, die generative KI produktiv einsetzen wollen, bietet Fabric eine zentrale Grundlage, um Daten effizient zu verwalten und für intelligente Systeme nutzbar zu machen. Im Kontext von Retrieval-Augmented Generation spielt Fabric eine Schlüsselrolle, da es alle notwendigen Bausteine für eine durchgängige Datenpipeline bereitstellt.

Im Zentrum von Microsoft Fabric steht OneLake [4], ein unternehmensweiter Data Lake, der als zentraler Speicher für

können sowohl über Apache Spark [7] als auch über automatisch generierte SQL-Endpunkte verarbeitet werden.

Das Eventhouse [8] ist auf zeitbasierte, semistrukturierte Daten wie Logs, Telemetrie oder Events spezialisiert. Es basiert auf dem Azure Data Explorer und eignet sich für Szenarien, in denen Daten kontinuierlich anfallen und schnell analysiert werden müssen. In der aktuellen Version unterstützt das Eventhouse auch Vektor-Datentypen. Dabei handelt es sich um Felder, die numerische Vektoren speichern, wie sie etwa bei Embeddings entstehen. Diese Vektoren können dann für eine semantische Suche genutzt werden, um die relevantesten Inhalte abzufragen [9].

Mit der aktuellen Public Preview von Azure Cosmos DB in Microsoft Fabric erweitert sich das Spektrum an KI-fähigen Speicherlösungen innerhalb der Plattform [10]. Cosmos DB in Fabric bringt Funktionen wie Vektorindizierung, Volltextsuche und hybride Suche direkt in die Fabric-Umgebung. Ein wesentlicher Vorteil liegt darin, dass sich Nutzer nicht um das

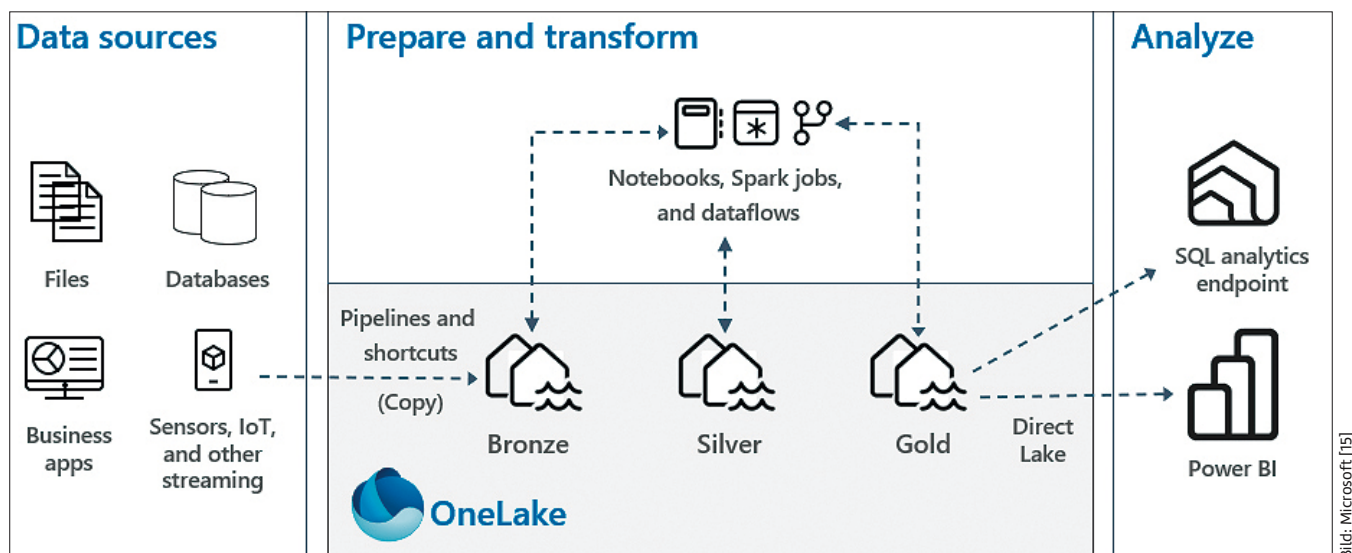


Bild: Microsoft [15]

Microsoft Fabric und die Medallion-Lakehouse-Architektur (Bild 3)

alle Komponenten dient. OneLake speichert strukturierte, semistrukturierte und unstrukturierte Daten gemeinsam und ermöglicht so eine konsistente Datenhaltung über verschiedene Anwendungsbereiche hinweg. Die Daten stehen zentral und zugriffsoptimiert zur Verfügung, indem Daten entweder gespiegelt oder direkt über sogenannte Shortcuts von ihren Quellsystemen abgerufen werden.

Fabric nutzt auf OneLake aufbauend verschiedene logische Speicherformen, die jeweils für bestimmte Datenarten und Analysezwecke optimiert sind. Das Lakehouse [5] eignet sich für strukturierte und unstrukturierte Daten, die analytisch verarbeitet werden sollen. Es basiert auf dem offenen Delta-Lake-Format [6], das die Skalierbarkeit eines Data Lake mit der Abfragefähigkeit eines Data Warehouse kombiniert. Anders als klassische relationale Datenbanken ist ein Lakehouse nicht auf transaktionale Workloads ausgelegt, sondern auf große, analytische Datenmengen. Die Daten

Management der Datenbank kümmern müssen. Durch die tiefe Integration mit OneLake ist Cosmos DB in Fabric vollständig in das Fabric-Ökosystem eingebunden und kann dort genutzt werden.

Für die Datenaufnahme unterstützt Fabric verschiedenste Formate. Neben strukturierten Tabellen können auch unstrukturierte Dokumente wie zum Beispiel PDF, Word oder PowerPoint verarbeitet werden. Wenn diese Dateien nicht direkt als Text vorliegen, kann Azure AI Search [11] genutzt werden, um Inhalte zu extrahieren, zu analysieren oder automatisch zu beschreiben. Der OneLake File Indexer [12] in Azure AI Search ermöglicht es, solche Dateien in einen Index zu überführen und mit Metadaten anzureichern. Die so gewonnenen Inhalte lassen sich über Azure AI Search abrufen. Aber auch über eigene Data Pipelines und Spark Notebooks können Dateien durch eigene Implementierung in strukturierte Daten umgewandelt werden.

Ein wesentlicher Aspekt bei der Nutzung von Microsoft Fabric ist das sogenannte Capacity-Modell [13]. Statt einzelne Dienste separat zu lizenzieren, basiert Fabric auf einem einheitlichen Ressourcenmodell, das über sogenannte Capacities bereitgestellt wird. Eine Capacity ist eine reservierte Menge an Rechenleistung, Speicher und IO-Ressourcen, die einem bestimmten Tenant oder Arbeitsbereich zugewiesen wird. Sie bildet die Grundlage für alle Workloads in Fabric, darunter Lakehouse, Data Pipelines, Notebooks, Eventhouse oder Power BI.

Capacities werden in sogenannten Capacity Units (CUs) gemessen. Die kleinste Einheit ist F2 mit zwei Capacity Units, größere CUs bieten entsprechend mehr Leistung. Die Capacity Unit gibt aber nicht nur die Kapazität vor, sondern bestimmt auch, welche Features vorhanden sind. Bei F64 beginnen diverse Features wie zum Beispiel die Möglichkeit, mit einem integrierten Copiloten die Daten zu befragen und zu durchsuchen.

und Speicherorte parallel verwaltet werden müssen. Auch die Integration mit Power BI erfolgt meist über separate Schritte.

Azure Databricks hingegen ist eine offene, stark auf Data Science und Machine Learning ausgerichtete Plattform. Sie bietet maximale Flexibilität für komplexe Datenpipelines, Modelltraining und fortgeschrittene Analysen. Databricks ist besonders geeignet für Teams mit tiefem technischem Know-how, die individuelle Lösungen auf Basis von Apache Spark, MLflow oder Delta Lake entwickeln wollen. Die Plattform ist leistungsfähig, aber in der Bedienung komplexer und weniger auf Endanwender ausgerichtet.

Microsoft Fabric positioniert sich zwischen diesen beiden Welten. Es kombiniert die Benutzerfreundlichkeit und Governance eines BI-Systems mit der Flexibilität und Skalierbarkeit einer modernen Datenplattform. Durch OneLake als gemeinsamem Speicher, einheitliche Workspaces, rollenbasierte Sicherheit und native Unterstützung für Spark, SQL und Power BI entsteht eine Umgebung, in der Daten nicht nur

### ● Listing 1: Erstellen von Markdown Pages aus PDF als PySpark Notebook

```
# Create markdown based on text from PDF files
%pip install pdfminer.six
from pdfminer.high_level import extract_pages

input_folder =
    "/lakehouse/default/Files/bronze/product-pdfs"
output_folder =
    "/lakehouse/default/Files/silver/product-mds"
chunk_output_folder = output_folder + "/chunks/"

def extract_markdown_pages(pdf_path):
    for index, page_layout in enumerate(
        extract_pages(pdf_path), start=1):
        text = ""
        for element in page_layout:
            if hasattr(element, "get_text"):
                text += element.get_text()
        yield { "page_number": index, "page_content":
            text, "filename": pdf_path }

pdf_files = get_files(input_folder, "*.pdf")
for pdf_file in pdf_files:
    markdown_pages = extract_markdown_pages(
        pdf_file.filename)
    write_files(output_folder, markdown_pages)
```

Das Preismodell ist flexibel gestaltet. Capacities können entweder stundenweise oder monatlich gebucht werden, was eine gute Planbarkeit und Skalierbarkeit ermöglicht. Unternehmen können ihre Kapazitäten jederzeit anpassen, um auf Lastspitzen oder neue Projekte zu reagieren. Microsoft stellt dafür ein SKU Estimator [14] zur Verfügung, mit dem sich der Ressourcenbedarf für verschiedene Workloads wie Spark, SQL oder Power BI im Voraus abschätzen lässt.

Im Vergleich zu anderen Plattformen wie Azure Synapse Analytics oder Azure Databricks verfolgt Microsoft Fabric einen konsequent integrierten Plattformansatz.

Azure Synapse Analytics ist besonders stark im Bereich strukturierter Daten und Data Warehousing. Es bietet dedizierte SQL-Pools, Spark-Integration und eine Vielzahl an Konnektoren. Die Plattform eignet sich gut für klassische BI-Szenarien, bei denen strukturierte Daten in hoher Qualität für Reporting und Analyse aufbereitet werden. Allerdings ist die Benutzerführung fragmentierter, da verschiedene Engines

verarbeitet, sondern auch direkt für Analyse- und KI-Anwendungen nutzbar gemacht werden können. Besonders für Organisationen, die sowohl strukturierte als auch unstrukturierte Datenquellen in GenAI-Szenarien integrieren wollen, bietet Fabric eine konsistente und zukunftssichere Lösung.

Für GenAI-Projekte bedeutet das: Mit Fabric steht eine Plattform zur Verfügung, die alle relevanten Schritte von der Datenaufnahme über die Transformation bis zur Bereitstellung für Agenten und Workflows in einem konsistenten System abbildet. Das reduziert Komplexität, erhöht die Wiederverwendbarkeit und schafft eine solide Grundlage für skalierbare KI-Anwendungen.

### Wie Daten in Fabric GenAI-fähig werden

Die Qualität generativer KI steht und fällt mit der Qualität der Daten, auf denen sie basiert. Damit ein Sprachmodell im Rahmen von Retrieval-Augmented Generation präzise, kontextbezogene Antworten liefern kann, müssen die zu- ►

grunde liegenden Informationen nicht nur vorhanden, sondern auch strukturiert, semantisch sinnvoll aufbereitet und technisch zugänglich sein. Microsoft Fabric bietet eine durchgängige Plattform, um diesen Prozess effizient und skalierbar umzusetzen.

Die Datenaufbereitung beginnt nicht direkt mit dem Zerlegen von Dokumenten in Vektoren, sondern idealerweise mit einer mehrstufigen Datenarchitektur, wie sie aus dem Data Warehousing bekannt ist: Bronze, Silber und Gold [15] (siehe Bild 3). In der Bronze-Stufe werden Rohdaten aus verschiedenen Quellen wie PDFs, Office-Dokumenten oder Webseiten gesammelt und in OneLake abgelegt. Um aus einem Dokument wie zum Beispiel einem PDF eine semantische Struktur zu bekommen, eignet sich oft die Konvertierung nach Markdown, welche deutlich mehr Strukturinformationen weitergeben kann (siehe Listing 1).

In der Silber-Stufe erfolgt eine erste semantische Strukturierung, das sogenannte Chunking [16]. Hier werden Inhalte beispielsweise nach Kapiteln, Abschnitten oder logischen Einheiten gruppiert (siehe Listing 2). Besonders wichtig ist da-

Formen und Strukturen gebracht, die für den jeweiligen Use Case effizient und sinnvoll sind. Bei einem GenAI-Fall stellt sich die Frage, ob die Texte direkt aus den Rohdaten übernommen oder erst einmal umgewandelt werden sollten. Oft ist es sinnvoll, die Inhalte sprachlich oder strukturell so anzupassen, dass sie besser zu typischen Nutzerfragen oder Suchanfragen passen. Beispielsweise können Tabellen in beschreibende Texte überführt oder redundante Informationen entfernt werden. Diese semantische Optimierung erhöht die Relevanz der späteren Antworten deutlich, da das Sprachmodell mit besser zugeschnittenem Kontext arbeitet. Die Transformation kann regelbasiert, modellgestützt oder manuell erfolgen und ist ein zentraler Hebel für die Qualität der gesamten RAG-Pipeline.

Im Anschluss daran werden die Chunks mit Metadaten angereichert. Dazu zählen Informationen wie zum Beispiel Titel, Quelle, Dokumententyp, Erstellungsdatum oder thematische Kategorien. Diese Metadaten sind essenziell für spätere Filterungen und Relevanzbewertungen. Sie ermöglichen es, Suchergebnisse gezielt einzugrenzen, etwa nach Abteilung,

## ● Listing 2: Erstellen von Chunks aus Markdown als PySpark Notebook

```
# Create chunks from markdown files
from langchain.text_splitter import
    MarkdownTextSplitter

def create_chunks(markdowns, filename, page_number=1):
    splitter = MarkdownTextSplitter(chunk_size = 100,
        chunk_overlap=0)
    documents = splitter.create_documents(markdowns)
    return [{
        "page_number": page_number,
        "page_content": doc.page_content,
        "filename": filename,
    } for doc in documents]

md_files = get_files(output_folder, "*.md")
for idx, md_file in enumerate(md_files):
    # create chunks overlapping with previous and next
    # page
    chunks = create_chunks([
        get_previous_page(md_files, md_file.filename),
        md_file.content,
        get_next_page(md_files, md_file.filename)
    ], md_file.filename, page_number=idx + 1)
    # save chunks with page as metadata
    write_files(chunk_output_folder, chunks)
```

bei, dass Chunks nicht willkürlich nach Seitenumbrüchen oder Zeichenlängen geschnitten werden, sondern entlang inhaltlicher Grenzen. Ein Abschnitt, der Text aus mehreren Kapiteln oder Themenbereichen vermischt, ist für semantische Suche meist wenig aussagekräftig. Ziel ist es, inhaltlich kohärente Einheiten zu schaffen, die auch über Seiten hinweg zusammengehören können. Chunks sollten eine ähnliche Länge haben, die sich an der maximalen Kontextlänge des verwendeten Sprachmodells orientiert. Aber auch leichte Überlappungen zwischen Chunks sind sinnvoll, um den richtigen Kontext zwischen Chunks zu erhalten. Die Zerlegung kann über Spark Notebooks oder Data Flows in Fabric erfolgen. Letztere bieten eine visuelle Oberfläche, mit der sich auch komplexe Textoperationen ohne Code realisieren lassen.

In der Gold-Stufe werden diese vorbereiteten Inhalte weiter optimiert. Hier werden die Daten in eine oder mehrere

Aktualität oder Dokumentenart. Auch für Governance und Auditing sind konsistente Metadatenstrukturen von zentraler Bedeutung. Dabei greifen Konzepte in Fabric wie Row Level Security oder auch Object Level Security, um genau zu definieren, welche Nutzer auf welche Daten Zugriff haben [17].

Für jeden Chunk wird nun ein semantischer Vektor berechnet, ein sogenanntes Embedding (siehe Listing 3). Dieser Vektor bildet die Bedeutung des Textes in einem mehrdimensionalen Raum ab und ist die Grundlage für spätere Ähnlichkeitsvergleiche.

In Fabric können Embeddings über verschiedene Wege erzeugt werden. Neben dem manuellen Einbinden von Azure OpenAI Service stehen auch AI Services in Fabric in der Public Preview zur Verfügung [18]. Dieses erlauben den Zugriff auf Azure AI Services wie Embedding, Text Analytics, Translator oder OpenAI direkt innerhalb von Fabric. Der große Vor-

teil: Die Nutzung dieser Modelle fällt in die gebuchte Fabric Capacity und verursacht keine zusätzlichen Kosten.

Die erzeugten Embeddings müssen anschließend gespeichert werden. Fabric bietet dafür mehrere Optionen. Im Lakehouse können Embeddings als Vektor-Spalten in Delta-Tabellen abgelegt werden. Diese Lösung ist besonders geeignet, wenn die Daten zusätzlich analytisch ausgewertet oder mit anderen strukturierten Informationen verknüpft werden sollen. Für performante semantische Suchen empfiehlt sich jedoch die Speicherung im Eventhouse oder in einer SQL-Datenbank mit Vektorunterstützung.

einheitlichen Sicherheitsmodell, zentralem Monitoring und konsistenter Governance. Drittens lassen sich Spark-Jobs, Pipelines, AI Services und Power BI direkt auf denselben Datenbestand anwenden, ohne Medienbrüche oder Synchronisationsprobleme. Für viele Unternehmen ist das ein entscheidender Vorteil, insbesondere wenn Datenschutz, Compliance und Skalierbarkeit eine Rolle spielen.

Eine spezialisierte Vektordatenbank ist sinnvoll, wenn maximale Performance, Skalierbarkeit und unabhängige Funktionen für Vektorsuchen benötigt werden, etwa bei großen Datenmengen oder besonders komplexen Suchanfragen, die

### ● Listing 3: Erstellen und Speichern der Embeddings als PySpark Notebook ins Eventhouse

```
# Create embeddings for each chunk
%pip install openai azure-kusto-data

from pyspark.sql.functions import udf, col
from pyspark.sql.types import ArrayType, FloatType
from openai import AzureOpenAI
import os

def get_embedding(text):
    client = AzureOpenAI(
        api_key=os.environ.get("AZURE_OPENAI_API_KEY"),
        azure_endpoint=os.environ.get(
            "AZURE_OPENAI_ENDPOINT"),
        azure_deployment=os.environ.get(
            "AZURE_OPENAI_DEPLOYMENT_ID"),
        api_version = "2024-02-01")

    response = client.embeddings.create(
        input=text,
        model= os.environ.get(
            "AZURE_OPENAI_EMBEDDING_MODEL",
            "text-embedding-ada-002"))

    return response.data[0].embedding

data = get_chunks(chunk_output_folder)
df_chunks = spark.createDataFrame(data, ["filename",
    "chunk_index", "text", "page_number"])

embedding_udf = udf(get_embedding,
    ArrayType(FloatType()))
df_embeddings = df_chunks.withColumn("embedding",
    embedding_udf(col("text")))

# Save embeddings to Eventhouse
df_embeddings
    .select("filename", "chunk_index", "text",
        "embedding", "page")
    .write
    .format(
        "com.microsoft.kusto.spark.synapse.datasources")
    ...
    .save()
```

Beide Varianten unterstützen Vektor-Felder nativ und ermöglichen Ähnlichkeitsabfragen [19] direkt über Kusto oder SQL zur semantischen Suche. Die Entscheidung für Lakehouse oder Eventhouse hängt vom Anwendungsfall ab: Das Lakehouse eignet sich für analytische Szenarien mit hohem Datenvolumen, das Eventhouse für schnelle, interaktive Suchen auf semistrukturierten Daten.

Ein häufiger Diskussionspunkt ist die Frage, ob man nicht besser eine spezialisierte Vektordatenbank wie Pinecone, Weaviate oder Qdrant verwenden sollte. Diese Systeme sind leistungsfähig und bieten zahlreiche Features für semantische Suche. Dennoch spricht vieles dafür, die Embeddings direkt in Fabric zu speichern. Erstens entfällt die Notwendigkeit, Daten zwischen Systemen zu bewegen, was sowohl die Komplexität als auch das Sicherheitsrisiko reduziert. Zweitens profitieren alle Komponenten von Fabric von einem

über die Standardfunktionen von Fabric hinausgehen. In solchen Fällen bieten spezialisierte Systeme oft optimierte Algorithmen, dedizierte Indizes und schnellere Suchzeiten.

Ein weiterer entscheidender Vorteil von Fabric ist aber die strategische Perspektive. Daten, die für GenAI aufbereitet werden, sind oft auch für andere Analysezwecke relevant, etwa für Reporting, Business Intelligence, Data Science oder klassische Machine-Learning-Modelle. Wenn die Embeddings und ihre Quellinformationen bereits in Fabric liegen, können sie ohne zusätzliche Infrastruktur für verschiedenste Anwendungsfälle genutzt werden. Fabric wird damit nicht nur zur Plattform für GenAI, sondern zur zentralen Datenbasis für das gesamte Unternehmen. Diese Wiederverwendbarkeit und Offenheit machen Fabric zu einer zukunftssicheren Alternative gegenüber spezialisierten, isolierten Vektordatenbanken. ►



Eine typische Daten-Pipeline in Fabric sieht dabei wie folgt aus: Eine Datei wird in OneLake geladen, zum Beispiel durch eine Synchronisation mit einem On-Premises-Datenspeicher, triggert automatisch eine Data Pipeline, welche über ein Spark Notebook oder einen Data Flow die Datei in Chunks zerlegt und mit Metadaten versieht. Anschließend werden Embeddings berechnet und im Eventhouse gespeichert. Von dort aus können sie direkt über Kusto semantisch durchsucht werden (siehe Bild 4). Dieser Prozess lässt sich vollständig innerhalb von Fabric orchestrieren. Alternativ können Daten auch im Lakehouse gesammelt und über Azure AI Search vektorisiert und indiziert werden, wenn eine hybride Sucharchitektur gewünscht ist.

Ein oft unterschätzter Aspekt ist die Aktualisierung der Embeddings. Wenn sich Inhalte ändern, müssen auch die zugehörigen Vektoren neu berechnet werden. Fabric bietet dafür

ideale Voraussetzungen. Über Data Pipelines lassen sich Änderungen erkennen, betroffene Chunks neu generieren und die Embeddings gezielt aktualisieren. So bleibt der Vektorindex konsistent und aktuell, ohne dass das gesamte System neu aufgebaut werden muss.

Für eine robuste Datenbasis empfiehlt es sich, einige Best Practices zu beachten: Die Chunk-Größe sollte konsistent gewählt werden, um eine gleichmäßige semantische Dichte zu gewährleisten. Zu große Chunks enthalten oft irrelevante Informationen, zu kleine verlieren den Kontext. Eine einheitliche Metadatenstruktur erleichtert nicht nur die Suche, sondern auch die Wartung und Erweiterung der Pipeline. Die Trennung von Text- und Vektorlogik sorgt dafür, dass Änderungen in einem Bereich nicht unbeabsichtigt andere Teile beeinflussen. Und schließlich sollten Pipelines modular aufgebaut sein, sodass einzelne Schritte wie Chunking, Embed-

#### ● Listing 4: Einfacher Semantic Kernel Assistant mit RAG über Fabric Eventhouse

```
var openAIApiKey = Environment.GetEnvironmentVariable(
    "AZURE_OPENAI_API_KEY");
var openAIEndpoint =
    Environment.GetEnvironmentVariable(
        "AZURE_OPENAI_ENDPOINT");
var openAIDeployment =
    Environment.GetEnvironmentVariable(
        "AZURE_OPENAI_DEPLOYMENT");
var embeddingDeployment =
    Environment.GetEnvironmentVariable(
        "AZURE_OPENAI_EMBEDDING_DEPLOYMENT");

var kustoConnString =
    Environment.GetEnvironmentVariable(
        "KUSTO_CONNECTION_STRING");
var kustoDbName =
    Environment.GetEnvironmentVariable("KUSTO_DB_NAME");

var kernelBuilder = Kernel.CreateBuilder()
    .AddAzureOpenAIChatCompletion(
        deploymentName: openAIDeployment,
        apiKey: openAIApiKey,
        endpoint: openAIEndpoint
    )
    .AddAzureOpenAIEmbeddingGenerator(
        deploymentName: embeddingDeployment,
        apiKey: openAIApiKey,
        endpoint: openAIEndpoint);

var kernel = kernelBuilder.Build();
var embeddingGenerator =
    kernel.GetRequiredService<IEmbeddingGenerator>(
        string, Embedding<float>>>());
var chatCompletionService =
    kernel.GetRequiredService<IChatCompletionService>();

var memoryStore = new KustoMemoryStore(
    kustoConnString, kustoDbName);
var textMemory = new SemanticTextMemory(
    memoryStore, embeddingGenerator);

ChatHistory history = [];
history.AddSystemMessage("You are an expert in
    technical manuals. Only answer questions relevant
    to the technical manual context provided by the
    user.");

while (true)
{
    Console.WriteLine("Query: ");
    var prompt = Console.ReadLine();
    if (string.IsNullOrEmpty(prompt)) continue;

    // Retrieve top 5 most relevant manual sections
    var results = await textMemory.SearchAsync(
        "productInfoEmbeddings", prompt, limit: 5,
        minRelevanceScore: 0.79);
    var context = string.Join("\n--\n",
        results.Select(r => r.Metadata.Text));

    var userPrompt = prompt +
        (string.IsNullOrEmpty(context) ? "" :
        "\nContext:\n" + context);
    history.AddUserMessage(userPrompt);

    var response = await
        chatCompletionService.GetChatMessageContentAsync(
            history, kernel: kernel);
    Console.WriteLine($"Answer: {response.Content}\n");
}
```

ding oder Speicherung unabhängig voneinander angepasst oder wiederverwendet werden können. Diese Prinzipien erhöhen die Wartbarkeit, Skalierbarkeit und Transparenz der gesamten Architektur.

## Wenn Daten sprechen lernen: Agenten, Workflows und KI-Logik mit Microsoft Fabric

Strukturierte, semantisch aufbereitete Daten sind die Grundlage jeder leistungsfähigen GenAI-Anwendung. Doch erst durch Agenten, Workflows und orchestrierte Logik entsteht daraus ein intelligentes System, das auf Nutzeranfragen reagieren, Entscheidungen treffen oder Prozesse anstoßen kann.

Die eigentliche Interaktion mit den Daten erfolgt in der Regel über spezialisierte Dienste, die sich an Fabric anbinden. Für Fachanwender stehen mit Microsoft Copilot Studio und den Fabric Data Agents intuitive Werkzeuge zur Verfügung,

gen stellen, etwa zu Verkaufszahlen, Produktvergleichen oder operativen Kennzahlen, und erhalten datengestützte Antworten, ohne SQL schreiben oder die Datenstruktur kennen zu müssen. Das setzt natürlich voraus, dass das Datenmodell sowie Tabellen und Spalten sinnvoll benannt wurden. Denn auch der schlaueste Agent kann nichts mit „Tabelle A“ und „Feld X1“ anfangen.

Data Agents senken die Einstiegshürde für datengetriebene Entscheidungen erheblich und machen Business Intelligence konversationsfähig. Sie ergänzen klassische Agentenlogik um eine direkte Verbindung zu analytischen Datenquellen innerhalb von Fabric und lassen sich nahtlos in bestehende Workflows integrieren.

Während Fabric Data Agents vor allem für direkte Abfragen in der Fabric-Umgebung gedacht sind, bietet Microsoft Copilot Studio [21] eine Low-Code-Umgebung, um eigene Chat-

The screenshot displays the Microsoft Fabric Eventhouse interface. On the left, a sidebar shows navigation options like Home, Workspaces, OneLake, Monitor, Real-Time, Workloads, GenAI, SilverLakehouse, and Vectors. The main area is titled 'Vectors' and shows a KQL query being executed. The query is as follows:

```

1  -0.016585903, -0.0020781392, 0.014586184, -0.009273206, 0.009900568, -0.00030714634, -0.0041399407, 0.015480818, 0.0014793671, 0.00627036, -0.01832422,
2  -0.03419127, -0.005953411, -0.006881385, -0.030923756, -0.009090225, 0.0115147205, 0.0136451395, 0.00037780634, -0.02702888, 0.0034831704, 0.039471574,
3  -0.007907385, -0.0047607683, -0.016285291, -0.024793899, 0.04328803, -0.02266348, -0.020010257, -0.0040190425, -0.008593563]);
4  extend similarity = series_cosine_similarity(query, embedding)
5  project id, text, similarity*similarity*100
6  order by similarity desc
7  limit 10

```

Below the query, a table titled 'Table 1' shows the results of the query. The table has three columns: 'id', 'text', and 'similarity'. The results are as follows:

| id                               | text                              | similarity        |
|----------------------------------|-----------------------------------|-------------------|
| product_manual_page_15_chunk_471 | /RS485 3 x USB3.0 1 x USB...      | 79.69329093800668 |
| product_manual_page_15_chunk_181 | evice according to the data ...   | 79.23698777091084 |
| product_manual_page_15_chunk_483 | ions Wiring diagrams Digita...    | 79.15047625665628 |
| product_manual_page_15_chunk_452 | thernet port COM (X30) • 1 ...    | 78.68131060096717 |
| product_manual_page_15_chunk_348 | es and function M.2 modul...      | 78.5279676020704  |
| product_manual_page_15_chunk_550 | Flash, 76 Front view, 10 G G...   | 78.06053455173362 |
| product_manual_page_15_chunk_464 | dule High current Per slot ...    | 78.0562441209742  |
| product_manual_page_15_chunk_65  | ..... 8...                        | 78.04031479937014 |
| product_manual_page_15_chunk_472 | escriptions 8.8.2 External Int... | 78.0107310420925  |
| product_manual_page_15_chunk_349 | s 2242, 2260, 3042, 3052, 3...    | 77.55958494114398 |

Embedding und Ähnlichkeit zum User Prompt im Fabric Eventhouse (Bild 4)

um auf Daten in natürlicher Sprache zuzugreifen oder einfache Automatisierungen zu erstellen. Entwickler hingegen greifen auf Services und Frameworks wie Azure AI Foundry, Semantic Kernel oder das Model Context Protocol zurück, um komplexe Agenten mit modularer Logik, semantischer Suche und Tool-Integration zu realisieren. Allen Ansätzen gemeinsam ist, dass sie auf dieselbe Datenbasis in Fabric zugreifen, sei es über SQL-Endpunkte, Kusto-Abfragen, CosmosDB, GraphQL oder auch direkt auf Dateien über das Azure Blob Filesystem. Damit wird Fabric zur verbindenden Schicht zwischen Daten und Intelligenz.

Für Fachanwender ohne tiefes technisches Know-how bieten die Fabric Data Agents [20] einen vielversprechenden Einstieg. Diese Funktion befindet sich zum Zeitpunkt der Abfassung dieses Artikels in der Public Preview. Mit einem Data Agent lassen sich kleine Chatbots erstellen, die direkt auf Datenquellen wie Lakehouse oder Eventhouse zugreifen. Nutzerinnen und Nutzer können in natürlicher Sprache Fra-

bots und Automatisierungen zu erstellen. Diese ist zwar nicht Teil von Fabric, befindet sich aber in der Microsoft-365-Welt und kommt entweder über Microsoft-Copilot-Lizenzen mit oder kann via Pay-as-you-go genutzt werden. In Copilot Studio lassen sich Agents definieren, die auf natürliche Sprache reagieren, sowie Flows, die bestimmte Aktionen auslösen. Für den Zugriff auf Daten aus Microsoft Fabric stehen derzeit jedoch nur begrenzte Optionen zur Verfügung. Eine native Verbindung zu Lakehouses oder Eventhouse besteht aktuell nicht. Stattdessen gibt es zwei praktikable Wege, um Fabric-Daten dennoch in Copilot Studio nutzbar zu machen: Zum einen kann ein vorgelagerter Fabric Data Agent eingesetzt werden, der die Datenabfrage übernimmt und die Ergebnisse über einen Flow oder ein API an Copilot Studio zurückliefert. Zum anderen besteht die Möglichkeit, Fabric-Daten über Tools einzubinden, etwa über SQL-Endpunkte von Lakehouses oder als Knowledge Source über Azure AI Search, sofern die Daten aus Fabric indiziert und aufbereitet wurden. ►

Während Low-Code-Werkzeuge wie Copilot Studio und die Fabric Data Agents vor allem Fachanwendern den Zugang zu Unternehmensdaten erleichtern, benötigen komplexere GenAI-Szenarien oft deutlich mehr Flexibilität, Steuerbarkeit und technische Tiefe. Für Entwicklerteams, die maßgeschneiderte Agentenlogik, semantische Suche oder Modellorchestrierung umsetzen möchten, bietet das Microsoft-Ökosystem eine Reihe leistungsfähiger Tools und Frameworks, die sich direkt oder indirekt mit Fabric verbinden lassen. Den Anfang macht Azure AI Foundry [22], eine Plattform speziell für die Entwicklung und den Betrieb von GenAI-An-

Preview verfügbar ist. Zusätzlich gibt es in Azure AI Foundry dieselben Anbindungen wie in Copilot Studio, weil Letzteres auf Azure AI Foundry und seiner Funktionalität aufbaut.

Für Agenten und Workflows, die auf strukturierte Daten zugreifen sollen, kann zusätzlich noch das Model Context Protocol (MCP) eingesetzt werden [23]. MCP ermöglicht es, einem großen Sprachmodell strukturierte Informationen, Fähigkeiten und Werkzeuge bereitzustellen. Ein zentrales Merkmal von MCP ist die Definition sogenannter Tools, die von KI-Agenten genutzt werden können, um Aufgaben auszuführen. Diese Tools beschreiben beispielsweise, wie auf

● **Tabelle 1: Technologieübersicht für GenAI im Microsoft-Kosmos**

| Technologie / Dienst          | Funktion im GenAI-Kontext                                                                                 | Besonderheiten / Vorteile                                                                                                                                                                                    |
|-------------------------------|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Microsoft Fabric              | Zentrale Plattform für Datenintegration, -speicherung, -verarbeitung und -analyse                         | Einheitliche Umgebung mit OneLake, Lakehouse, Eventhouse, Pipelines und Notebooks                                                                                                                            |
| OneLake                       | Tenant-weiter Datenspeicher für alle Fabric-Komponenten                                                   | Einheitliches Sicherheitsmodell, Data Sharing, Versionierung                                                                                                                                                 |
| Lakehouse                     | Speicherung und Analyse strukturierter und unstrukturierter Daten                                         | Delta-Lake-Format, SQL- und Spark-Zugriff, ideal für ETL und Embedding-Vorbereitung                                                                                                                          |
| Eventhouse                    | Speicherung zeitbasierter, semistrukturierter Daten mit Vektorunterstützung                               | Native Vektorsuche mit KQL, ideal für semantische Suche in RAG-Workflows                                                                                                                                     |
| Data Pipelines und Data Flows | Visuelle und codebasierte Datenintegration und -transformation                                            | Automatisierung von Chunking, Metadatenanreicherung, Embedding-Erzeugung                                                                                                                                     |
| Spark Notebook                | Entwicklung und Ausführung von Datenvorverarbeitung, Chunking, Embedding-Erzeugung und semantischer Suche | Vollständig anpassbar, unterstützt Python und PySpark, ideal für komplexe Textverarbeitung, Tokenisierung und Vektorisierung. Besonders geeignet, wenn Data Flows oder Low-Code-Tools an ihre Grenzen stoßen |
| AI Services in Fabric         | Zugriff auf Azure-AI-Modelle (OpenAI, Embedding, Translator, Text Analytics)                              | Nutzung innerhalb der Fabric Capacity, keine Zusatzkosten, direkt in Pipelines integrierbar                                                                                                                  |
| Azure AI Foundry              | Plattform zur Entwicklung und Verwaltung von KI-Anwendungen                                               | Modellorchestrierung, Agentenentwicklung, Teamkollaboration                                                                                                                                                  |
| Microsoft Copilot Studio      | Low-Code-Umgebung für Chatbots und Workflows                                                              | Integration mit Microsoft 365, Verbindung von Agenten mit Daten und Aktionen                                                                                                                                 |
| Semantic Kernel               | Open-Source-Framework für Agentenentwicklung                                                              | Modularer Aufbau mit Skills, Prompts und Tool-Integration, ideal für wiederverwendbare Logik                                                                                                                 |
| CosmosDB in Microsoft Fabric  | KI-fähige Datenbank mit Vektorindizierung, Volltext- und hybrider Suche                                   | In Public Preview, tief in Fabric integriert, ideal für semantische Suche und KI-Anwendungen                                                                                                                 |

wendungen. Sie richtet sich an Teams, die Sprachmodelle, Retrieval-Komponenten und Agenten modular kombinieren und orchestrieren wollen. Azure AI Foundry bietet eine Umgebung, in der sich Modelle von OpenAI, Anthropic und Co. integrieren, orchestrieren und überwachen lassen. Besonders für agentenbasierte Architekturen ist Azure AI Foundry interessant, da es die Verbindung zwischen Daten, Modellen und Geschäftslogik unterstützt. Entwickler können hier nicht nur Modelle feinjustieren oder einbinden, sondern auch Workflows definieren, die auf bestimmte Datenereignisse oder Nutzeranfragen reagieren. Fabric kann in Azure AI Foundry bereits als Knowledge Source eingebunden werden, auch wenn diese Funktion aktuell noch im Rahmen einer Public

Datenquellen zugegriffen wird oder wie Aktionen ausgelöst werden können. Ziel des Protokolls ist es, eine standardisierte Schnittstelle zu schaffen, über die Sprachmodelle mit externen Systemen interagieren können, etwa mit Datenbanken, APIs oder andere Anwendungen. Bislang existiert noch kein offizieller Fabric-MCP-Server, aber es gibt bereits Open-Source-Implementierungen, die sich in Eigenregie hosten lassen und Daten aus Fabric über MCP bereitstellen. So werden Fabric-Daten einfach durchsuchbar.

Für die Implementierung und Orchestrierung komplexer Agentenlogik bietet sich der Microsoft Semantic Kernel [24] an, ein Open-Source-Framework von Microsoft, das speziell für die Entwicklung von KI-Agenten mit modularer Logik,

Gedächtnis und Tool-Integration entwickelt wurde. Semantic Kernel ermöglicht es, sogenannte Skills zu definieren, die bestimmte Aufgaben übernehmen, etwa das Ausführen einer semantischen Suche im Eventhouse, das Extrahieren von Informationen oder das Auslösen eines Workflows. Diese Skills können mit Prompts, Funktionen oder externen APIs kombiniert werden. Ein Semantic Kernel Agent kann beispielsweise so konfiguriert werden, dass er bei einer Nutzeranfrage automatisch relevante Daten aus dem Eventhouse abrufen, diese mit einem Sprachmodell verarbeiten und anschließend eine strukturierte Antwort generiert oder eine Aktion auslöst (siehe [Listing 4](#)).

In unserem Beispiel beginnt die Entwicklung einer GenAI-Anwendung auf Basis von Microsoft Fabric mit einem Agenten, der mit dem Semantic Kernel entwickelt wurde. Dieser Agent wird als Chatbot in einer Webanwendung oder einem internen Portal bereitgestellt. Nutzerinnen und Nutzer können dort in natürlicher Sprache Fragen stellen, etwa zu Produkten, Prozessen oder internen Richtlinien.

Die Anfrage wird vom Agenten entgegengenommen und in eine semantische Suchanfrage umgewandelt. Diese wird an das Eventhouse übergeben, wo eine Vektorsuche nach den relevantesten Textabschnitten durchgeführt wird. Die gefundenen Inhalte werden als zusätzlicher Kontext in den Prompt des Sprachmodells eingebettet. Das Sprachmodell generiert auf dieser Grundlage eine Antwort, die nicht nur auf seinem Trainingswissen basiert, sondern auch auf aktuellen und unternehmensspezifischen Informationen. Die Antwort wird dem Nutzer direkt im Chat angezeigt.

Alle beteiligten Komponenten, von der Datenabfrage über die semantische Suche bis zur Antwortgenerierung, greifen auf dieselbe Datenbasis in Microsoft Fabric zu. Dadurch entsteht eine konsistente, sichere und nachvollziehbare Datenarchitektur, die sich flexibel erweitern und an unterschiedliche Anwendungsfälle anpassen lässt (vergleiche [Tabelle 1](#)).

Für den produktiven Einsatz solcher Agenten empfiehlt es sich, einige Prinzipien direkt in die Architektur zu integrieren. Die Trennung von Benutzeroberfläche, Modell und Datenzugriff sorgt für bessere Wartbarkeit und Flexibilität. Agentenlogik sollte modular aufgebaut sein, sodass sie in verschiedenen Kontexten wiederverwendet werden kann. Semantic Kernel unterstützt dies durch seine Plug-in-Architektur, bei der einzelne Skills unabhängig voneinander entwickelt und kombiniert werden können. Logging und Monitoring sind essenziell, um das Verhalten der Agenten zu analysieren und kontinuierlich zu verbessern. Feedbackschleifen, etwa durch Nutzerbewertungen oder automatische Auswertungen, helfen dabei, die Qualität der Antworten und die Relevanz der Datenbasis stetig zu optimieren. Auch die Wiederverwendbarkeit von Prompt-Vorlagen, Skill-Komponenten und Datenzugriffsmodulen trägt dazu bei, dass Agenten nicht nur schneller entwickelt, sondern auch langfristig gepflegt und erweitert werden können.

## Fazit

Die Idee eines unternehmensweiten Assistenten, der Mitarbeiterinnen und Mitarbeitern auf natürliche Spracheingaben

hin präzise und kontextbezogene Antworten liefert, ist längst keine Zukunftsvision mehr. Mit Microsoft Fabric steht eine Plattform zur Verfügung, die alle notwendigen Bausteine vereint, um diese Vision Realität werden zu lassen. Dazu gehören die Datenspeicherung und -integration, die semantische Aufbereitung sowie die Integration in intelligente Agenten und Workflows.

Im Verlauf dieses Artikels wurde gezeigt, wie strukturierte, semistrukturierte und unstrukturierte Daten aus unterschiedlichen Quellen wie SQL-Tabellen, PDFs, HTML-Seiten oder Office-Dokumenten in OneLake zentralisiert und entlang einer Bronze-Silber-Gold-Architektur aufbereitet werden. Aus den Rohdaten entstehen durch semantisches Chunking, Metadatenanreicherung und Vektorisierung hochwertige Embeddings, die im Eventhouse gespeichert und für semantische Suchen genutzt werden können. Die Suche erfolgt über Agenten, die mit Semantic Kernel entwickelt wurden. Diese sind in der Lage, relevante Inhalte aus dem Eventhouse zu extrahieren, mit Sprachmodellen zu kombinieren und daraus fundierte Antworten zu generieren.

Trotz dieser Stärken ist Fabric keine Lösung, die ohne Weiteres sofort einsatzbereit ist. Die Plattform bietet viele Wege zum Ziel, aber nicht alle sind bereits vollständig ausgereift. Zwar lassen sich PDFs in Data Flows in Abschnitte zerlegen, doch die semantische Erkennung von inhaltlichen Einheiten ist dort noch begrenzt. In der Praxis greifen viele Teams daher auf Spark Notebooks mit Python zurück, um das Chunking präzise zu steuern, oder indizieren Dokumente, die in OneLake landen, direkt und einfacher mit Azure AI Search. Auch die Erzeugung von Embeddings erfolgt häufig über eigene Implementierungen mit Azure OpenAI oder Open-Source-Modellen, da die Integration in Fabric zwar möglich, aber noch nicht durchgängig verfügbar ist.

Ein weiteres Beispiel ist die Speicherung der Embeddings. Diese können im Lakehouse, im Eventhouse oder in Cosmos DB abgelegt werden. Die Anbindung an Frontends wie Semantic Kernel oder Copilot Studio erfordert jedoch häufig zusätzliche Logik. Es fehlen standardisierte Schnittstellen, um Vektorindizes direkt in Agenten zu nutzen oder semantische Suchen ohne Zwischenschritte zu orchestrieren. Viele dieser Komponenten befinden sich noch in der Public Preview. Das bedeutet, dass sich Funktionen, APIs und Stabilität noch verändern können und zudem meist kein Support für den Produktivbetrieb gegeben ist.

Auch das Thema Governance ist nicht vollständig gelöst. Wer Dokumente im Spark Notebook in Chunks zerlegt und Embeddings erzeugt, muss selbst sicherstellen, dass Berechtigungen aus den Quelldaten korrekt übernommen werden. Fabric bietet zwar Konzepte wie Row- und Object-Level-Security, doch diese müssen auch in den entsprechenden Zieldateien und -strukturen mit übernommen werden. Damit ein Nutzer nur die Embeddings sieht, auf deren Ursprungsdaten er Zugriff hat, muss die gesamte Pipeline im Nutzerkontext ausgeführt werden. Das ist technisch möglich, aber nicht trivial.

Ein weiterer zentraler Erfolgsfaktor liegt in der Qualität der Daten selbst und in der Art und Weise, wie sie aufbereitet werden. In diesem Artikel wurden viele Schritte im Code ►



vereinfacht dargestellt. In der Praxis entscheiden jedoch gerade die semantisch sinnvolle Zerlegung von Texten, die Auswahl geeigneter Chunk-Größen und die gezielte Anreicherung mit Metadaten über die Qualität der Antworten. Ein schlecht geschnittener Abschnitt oder fehlende Kontextinformationen können dazu führen, dass ein Sprachmodell irrelevante oder ungenaue Antworten liefert. Umgekehrt kann eine durchdachte Datenstruktur mit klaren Metadaten, konsistenten Formaten und inhaltlich kohärenten Abschnitten die Relevanz und Präzision der Ergebnisse erheblich steigern. Dieser Prozess ist selten beim ersten Versuch perfekt. Vielmehr bedarf es oft mehrerer Iterationen, Tests und Anpassungen, um die optimale Balance zwischen Kontexttiefe, Antwortqualität und Performance zu finden.

Trotz dieser Herausforderungen bietet Fabric einen entscheidenden strategischen Vorteil. Es schafft eine gemeinsame Dateninfrastruktur, die nicht nur für generative KI, sondern auch für klassische Business Intelligence, Data Science und Automatisierung sowohl von Fachanwendern als auch von Entwicklern genutzt werden kann. Daten müssen nicht zwischen Systemen bewegt werden, sondern stehen zentral zur Verfügung. Die Wiederverwendbarkeit von Daten, Pipelines und Sicherheitskonzepten reduziert Komplexität und erhöht die Konsistenz über alle Anwendungsfälle hinweg.

Fabric ist noch nicht perfekt, aber ein mächtiges Fundament. Wer bereit ist, in Architektur, Datenqualität und Governance zu investieren, erhält eine Plattform, die nicht nur einzelne Anwendungsfälle ermöglicht, sondern eine skalierbare und zukunftssichere Umgebung für datengetriebene Intelligenz schafft. Die Kombination aus technischer Tiefe, strategischer Offenheit und enger Integration in das Microsoft-Ökosystem macht Fabric zum zentralen Baustein einer unternehmensweiten Datenplattform, die nicht nur AI-, sondern jegliche Data-Analytics- und Reporting-Szenarien unterstützt. ■

- [1] Build advanced retrieval-augmented generation systems, <https://learn.microsoft.com/en-us/azure/developer/ai/advanced-retrieval-augmented-generation>
- [2] Microsoft Fabric, [www.microsoft.com/de-de/microsoft-fabric](https://www.microsoft.com/de-de/microsoft-fabric)
- [3] What is Microsoft Fabric?, <https://learn.microsoft.com/en-us/fabric/fundamentals/microsoft-fabric-overview>
- [4] OneLake, the OneDrive for data, <https://learn.microsoft.com/en-us/fabric/onelake/onelake-overview>
- [5] What is a lakehouse in Microsoft Fabric?, <https://learn.microsoft.com/en-us/fabric/data-engineering/lakehouse-overview>
- [6] Lakehouse and Delta Lake tables, <https://learn.microsoft.com/en-us/fabric/data-engineering/lakehouse-and-delta-tables>
- [7] What is Apache Spark compute in Microsoft Fabric?, <https://learn.microsoft.com/en-us/fabric/data-engineering/spark-compute>
- [8] Eventhouse overview, <https://learn.microsoft.com/en-us/fabric/real-time-intelligence/eventhouse>
- [9] Kusto: series\_cosine\_similarity(), <https://learn.microsoft.com/en-us/kusto/query/series-cosine-similarity-function>

- [10] Announcing Cosmos DB in Microsoft Fabric (Preview), <https://blog.fabric.microsoft.com/en-US/blog/22987/>
- [11] What's Azure AI Search?, <https://learn.microsoft.com/en-us/azure/search/search-what-is-azure-search>
- [12] Index data from OneLake files and shortcuts, <https://learn.microsoft.com/en-us/azure/search/search-how-to-index-onelake-files>
- [13] Plan your capacity size, <https://learn.microsoft.com/en-us/fabric/enterprise/plan-capacity>
- [14] Estimate your Fabric capacity needs, [www.microsoft.com/en-us/microsoft-fabric/capacity-estimator](https://www.microsoft.com/en-us/microsoft-fabric/capacity-estimator)
- [15] Understand medallion lakehouse architecture for Microsoft Fabric with OneLake, <https://learn.microsoft.com/en-us/fabric/onelake/onelake-medallion-lakehouse-architecture>
- [16] RAG chunking phase, <https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/rag/rag-chunking-phase>
- [17] Security in Microsoft Fabric, <https://learn.microsoft.com/en-us/fabric/security/security-overview>
- [18] AI services in Fabric (preview), <https://learn.microsoft.com/en-us/fabric/data-science/ai-services/ai-services-overview>
- [19] Azure Data Explorer for Vector Similarity Search, <https://techcommunity.microsoft.com/blog/azuredataexplorer/azure-data-explorer-for-vector-similarity-search/3819626>
- [20] Create a Fabric data agent (preview), <https://learn.microsoft.com/en-us/fabric/data-science/how-to-create-data-agent>
- [21] Copilot Studio overview, <https://learn.microsoft.com/en-us/microsoft-copilot-studio/fundamentals-what-is-copilot-studio>
- [22] What is Azure AI Foundry?, <https://learn.microsoft.com/en-us/azure/ai-foundry/what-is-azure-ai-foundry>
- [23] Build Agents using Model Context Protocol on Azure, <https://learn.microsoft.com/en-us/azure/developer/ai/intro-agents-mcp>
- [24] Introduction to Semantic Kernel, <https://learn.microsoft.com/en-us/semantic-kernel/overview/>



**Florian Bader**

ist Solution Architect bei Lunarix Digital Solutions und Microsoft MVP für Azure IoT. Mit langjähriger Erfahrung im Bereich DevOps, Cloud und Digitalisierung unterstützt er Unternehmen bei ihren digitalen Projekten.

**florian.bader@lunarix.digital**



**Lars Roith**

ist Solution Consultant bei Lunarix Digital Solutions und berät Unternehmen und Entwicklungsteams ganzheitlich bei der Umsetzung ihrer Softwareentwicklungsprojekte.

**lars.roith@lunarix.digital**

## FALLSTUDIE: KI-INTEGRATION IN DER KUNDENKOMMUNIKATION

# KI in Outlook

Bei [www.IT-Visions.de](http://www.IT-Visions.de) hilft eine KI beim Auswerten und Schreiben von E-Mails.

Nachdem sich der Einsatz künstlicher Intelligenz in der Softwareentwicklung bei uns etabliert hat und man sich manchmal beim Entwickeln fragt, wie man denn früher ohne Copilot gearbeitet hat, stellte sich in unserer Firma [www.IT-Visions.de](http://www.IT-Visions.de) die Frage, wie wir unsere Kaufleute durch KI unterstützen können. Der größte Teil der Kunden- und Lieferantkontakte läuft bei uns über E-Mail mit einem lokal auf Windows installierten Outlook-Client. Daher war es nur konsequent, Outlook um KI-Funktionen zu erweitern.

Erweiterungen für Outlook sind bei uns seit Jahren im Einsatz. Zu eingehenden E-Mails stellen wir dem kaufmännischen Team verschiedene Funktionen mit direkter Anbindung an unser Customer Relationship Management (CRM) bereit. Unser CRM ist kein Standardprodukt, sondern selbstentwickelt. Es ist inzwischen auch viel mehr als ein CRM: Es verwaltet nicht nur Kundendaten und Aufträge, sondern ist auch eine weitestgehend automatisierte Buchhaltung sowie ein Managementsystem für die von uns durchgeführten Schulungen und Beratungen inklusive Teilnehmer:innenverwaltung und allem, was dazugehört (etwa An- und Abmeldungen oder Teilnahmebestätigungen). Aus Outlook heraus lassen sich Kontakte anlegen und pflegen, Angebote erstellen, Belege in die Buchhaltung einstellen und vieles mehr.

## Die KI antwortet selbstständig auf eine E-Mail

Bild 1 zeigt eine eingehende E-Mail eines potenziellen neuen Kunden im Posteingang des Kundenteams. Im Screenshot

sieht man aus Datenschutzgründen einen künstlich erzeugten Beispielfall; alle anderen E-Mails und Ordner sind mit weißen Flächen verdeckt. Die zu Outlook hinzugefügten Funktionen sind wahlweise über das Kontextmenü einer E-Mail oder die Registerkarte *Add-Ins* im Ribbon aufrufbar.

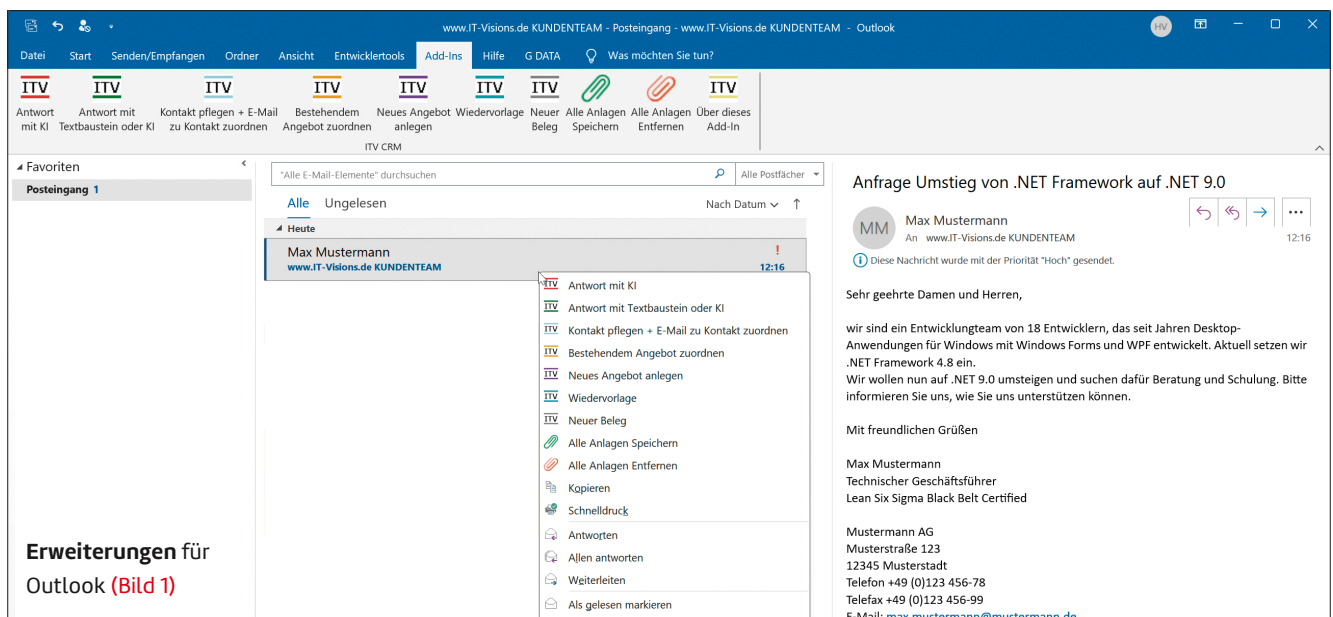
Der erste Punkt im Menü heißt: *Antwort mit KI*. Diese Funktion ist so implementiert, dass sie ohne weitere Eingabe des Benutzers oder der Benutzerin eine Antwort auf Basis unserer Standards implementiert. Diese Standards liegen bei uns in Form von Dokumenten und einer Datenbank vor:

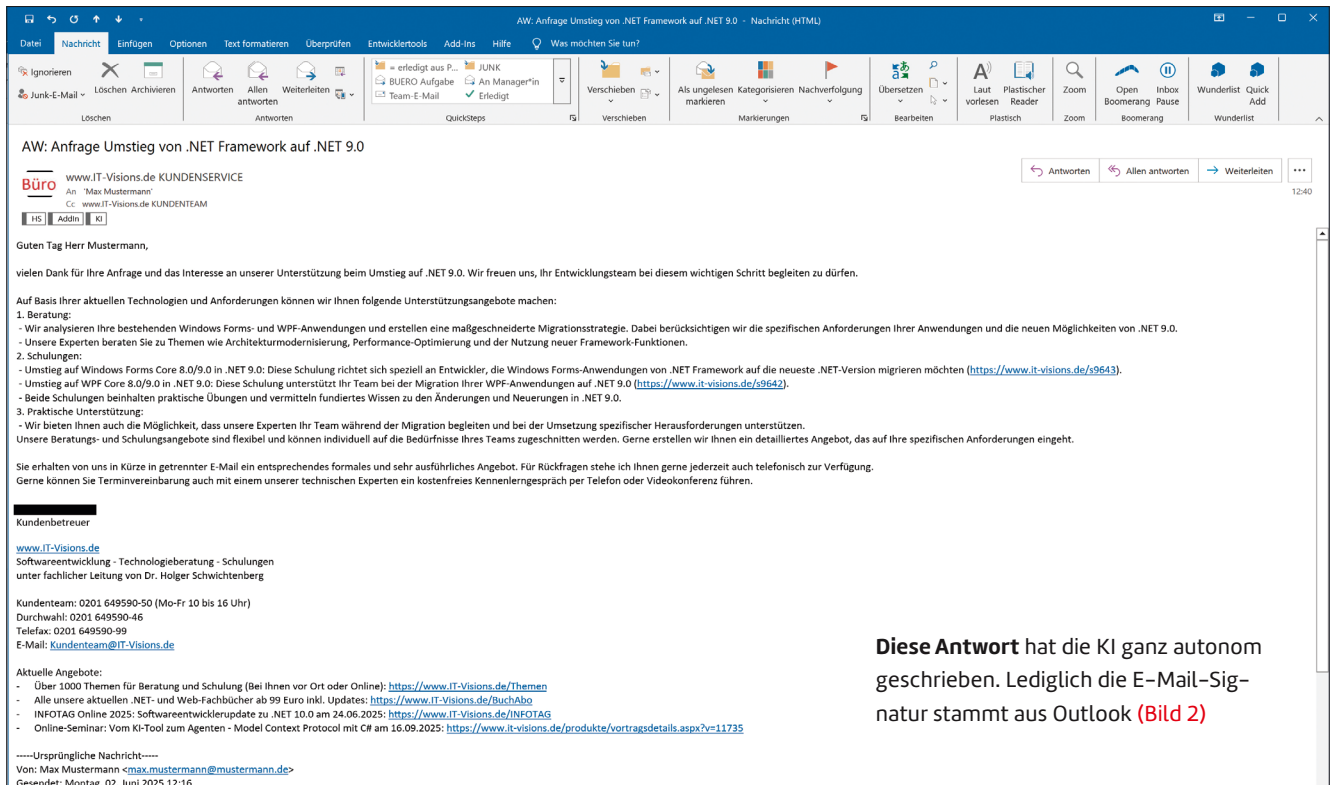
- Das Dokument *Häufige Fragen und Antworten (FAQ)* ist für unsere Kunden als Word-Dokument und PDF im Umfang von 41 Seiten verfügbar.
- Unsere Firmenbroschüre hat als Indesign-Datei oder PDF einen Umfang von 16 Seiten.
- In einer Microsoft-SQL-Server-Datenbank sind über 1000 Themen gespeichert, zu denen wir unterstützen können.

Bild 2 zeigt, wie die Antwort unserer KI auf die Anfrage aus Bild 1 aussieht. Die Nachricht wird nicht direkt versendet, sondern dem Kundenbetreuer oder der Kundenbetreuerin als Entwurf präsentiert. Eine Nachbearbeitung ist also möglich.

## Manchmal braucht die KI eine zusätzliche Anweisung

Natürlich ist die dem Kunden zu gebendene Antwort nicht immer klar und eindeutig aus unseren Standards ableitbar. ►

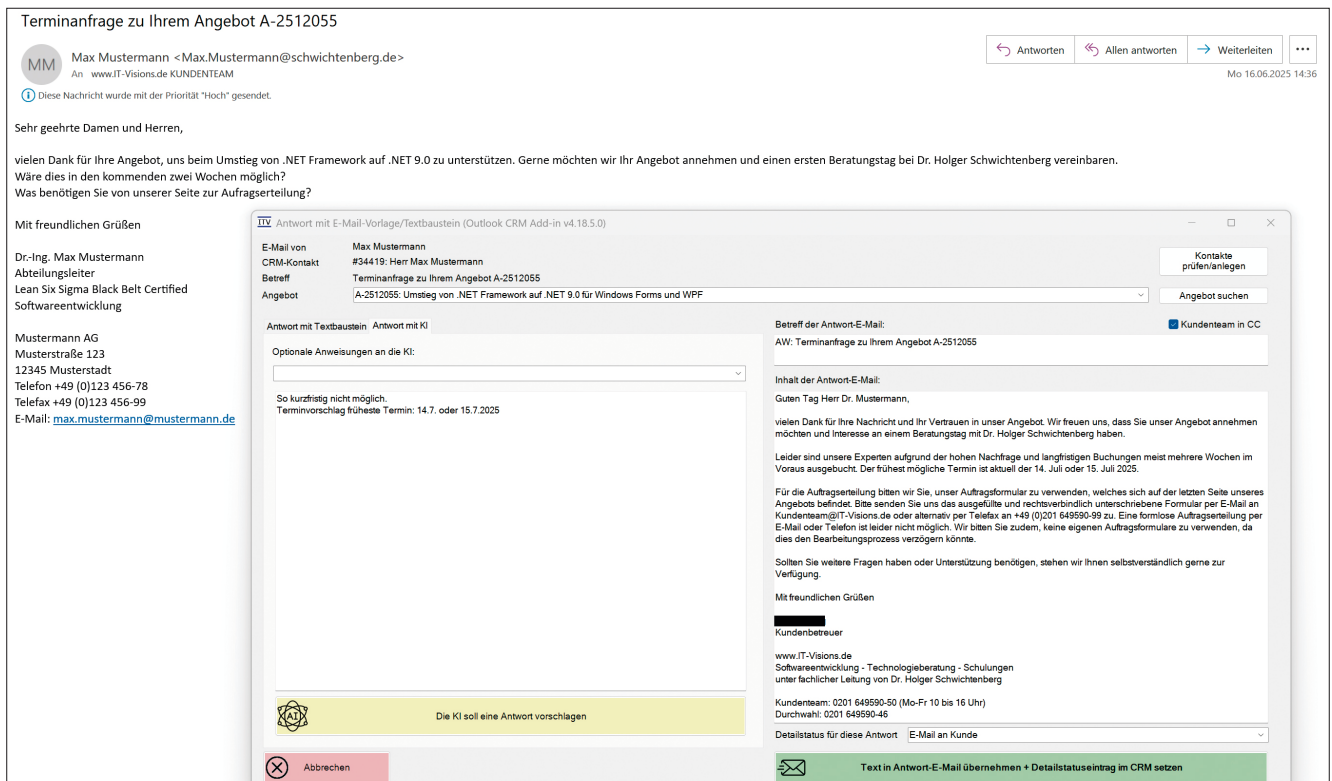




Diese Antwort hat die KI ganz autonom geschrieben. Lediglich die E-Mail-Signatur stammt aus Outlook (Bild 2)

Nehmen wir als Beispiel an, dass der Kunde die Frage stellt, ob kurzfristig in den kommenden zwei Wochen bei einem bestimmten unserer technischen Experten noch ein Termin frei

ist. Die KI könnte theoretisch eine Antwort ermitteln, wenn wir ihr Zugriff auf die Kalender aller Expertinnen und Experten geben würden. Da allerdings in unseren Kalendern oft



Bei dieser Kundenfrage brauchte die KI einen Hinweis, wie die Terminfrage zu beantworten ist. Die Form der Auftragserteilung stammt hingegen wieder aus den FAQ (Bild 3)

nur feste Kundentermine, nicht aber Projektarbeitszeiten und Vorbereitungszeiten erfasst sind, würde die KI hier viele unpassende Termine vorschlagen. Das Kundenteam hält daher oft persönliche Rücksprache mit dem Experten beziehungsweise der Expertin und gibt der KI einen knappen Hinweis für die Antwort wie: „So kurzfristig nicht möglich. Terminvorschlag frühestmögliche Termine: 14.7. oder 15.7.2025.“ Was die KI daraus macht, sehen Sie in **Bild 3**.

### Kontaktdaten aus E-Mails extrahieren

Der dritte Menüpunkt *Kontakt pflegen* führt zu einer Bildschirmmaske, die einen Kontakt in unserem CRM anlegt oder einen bestehenden Kontakt aktualisiert. Wir suchen den Kontakt anhand der E-Mail-Adresse in unserer Datenbank und präsentieren dem Kundenteammitglied dann eine umfangreiche Maske, aus der **Bild 4** einen Ausschnitt darstellt. Hier schlägt die KI auf Basis der E-Mail von Herrn Mustermann vor, dass im CRM der akademische Titel, die Funktion im Unternehmen, die Abteilung, die Straße, die Postleitzahl und der Ort sowie Telefon- und Telefax-Nummer zu ergänzen sind. Diese Felder sind daher farbig hervorgehoben.

### E-Mail-Inhalte auswerten

Wir speichern im CRM nicht nur die Kontakte, sondern zu jedem Angebot auch den Kommunikationsverlauf ab. Vor dem Einsatz der KI mussten die Kundenbetreuer:innen selbst die E-Mails kurz zusammenfassen. Nun erledigt KI auch dies, siehe **Bild 5**. Auf ähnliche Weise funktioniert der hier nicht dargestellte Menüpunkt *Neues Angebot anlegen*. Dieser extrahiert die Wünsche des Kunden und macht dem Kundenteam Vorschläge, wie die Felder in der Angebotsmaske unseres CRM zu befüllen sind.

### Rechnungen auswerten

Seit dem 1. Januar 2025 gibt es in Deutschland die Pflicht zur strukturierten elektronischen Rechnung (E-Rechnung) zwischen deutschen Unternehmen (B2B), die dotnetpro ging darauf bereits näher ein [1]. Während wir bei www.IT-Visions.de schon seit Jahren E-Rechnungen an alle Kunden senden und empfangene E-Rechnungen auch automatisiert auswerten können, zeigt die Praxis, dass sehr viele Unternehmen die großzügigen Übergangsfristen nutzen. Wirklich verpflichtend ist die E-Rechnung für alle Unternehmen nämlich erst ab 2028.

Während der Getränkemarkt in unserem Dorf, der uns einmal im Monat Getränke liefert, seit dem 1. Januar eine ZUG-

**Bestehender CRM-Kontakt #34419**

Organisation: **Mustermann AG (#13607)** NEU

Kontakt-ID: **34419** Typ: **K** Kürzel: **Mustermann**

Anrede: **Herr** Titel: **Dr.-Ing.**

Vorname: **Max**

Name: **Mustermann**

E-Mail: **max@mustermann.de**

Abteilung: **Softwareentwicklung**

Funktion: **Abteilungsleiter**

Strasse: **Musterstraße 123**

PLZ: **12345** Ort: **Musterstadt**

Land:

Telefon: **0123 456-78**

Telefax: **0123 456-99**

Mobil-Tel:

Memo: **Erstkontakt zu Dr. Holger Schwichtenberg im Rahmen der DWX-Keynote Juli 2024, Thema .NET 9.0**

 **Kontakt speichern und diese E-Mail dem Kontakt zuordnen**

**Die KI macht Vorschläge**, wie der bestehende Kundendatensatz erweitert werden kann (**Bild 4**)

FerD-Rechnung (PDF mit X-Rechnung im Anhang) sendet, bekommen wir von Telekommunikationsunternehmen, Providern, Energieversorgern und Online-Shops weiterhin ►

Terminanfrage zu Ihrem Angebot A-2512055

 Max Mustermann  
An: www.IT-Visions.de KUNDENTEAM  
Diese Nachricht wurde mit der Priorität "Hoch" gesendet.

Sehr geehrte Damen und Herren,

vielen Dank für Ihre Angebot, uns beim Umstieg von .NET Framework auf .NET 9.0 zu unterstützen. Gerne möchten wir Ihr Angebot annehmen und einen ersten Beratungstag bei Dr. Holger Schwichtenberg vereinbaren.  
Wäre dies in den kommenden zwei Wochen möglich?  
Was benötigen Sie von unserer Seite zur Auftragserteilung?

Mit freundlichen Grüßen

Dr.-Ing. Max Mustermann  
Abteilungsleiter  
Lean Six Sigma Black Belt Certified  
Softwareentwicklung

Mustermann AG  
Musterstraße 123  
12345 Musterstadt  
Telefon +49 (0)123 456-78  
Telefax +49 (0)123 456-99  
E-Mail: [max.mustermann@mustermann.de](mailto:max.mustermann@mustermann.de)

**Die KI fasst in Stichpunkten zusammen, was der Kunde in seiner E-Mail schrieb (Bild 5)**

 Antworten  Allen antworten  Weiterleiten Mo 16.06.2025 14:36

**E-Mail zu bestehendem Angebot im CRM speichern (Outlook CRM Add-in v4.18.5.0)**

Angebot: **A-2512055: Umstieg von .NET Framework auf .NET 9.0 für Windows Forms und WPF** Angebot suchen

Detaillstatus für diese E-Mail: **E-Mail von Kunde**

Hauptstatus für A-2512055: **Gesendet**

Zeitpunkt: **16.06.2025 14:35:49**

Dateianhänge: **...**  Anhang speichern  Auch Grafiken speichern

Max Mustermann: Terminanfrage zu Ihrem Angebot A-2512055  
- Annahme des Angebotes zur Unterstützung beim Umstieg von .NET Framework auf .NET 9.0  
- Anfrage zur Terminvereinbarung eines ersten Beratungstages bei Dr. Holger Schwichtenberg  
- Wunsch nach Termin in den kommenden zwei Wochen.  
- Frage nach benötigten Unterlagen oder Informationen zur Auftragserteilung

----- Originaltext -----  
Von: Max Mustermann (Max.Mustermann@schwichtenberg.de)  
An: www.IT-Visions.de KUNDENTEAM  
Betreff: Terminanfrage zu Ihrem Angebot A-2512055

Sehr geehrte Damen und Herren,

vielen Dank für Ihre Angebot, uns beim Umstieg von .NET Framework auf .NET 9.0 zu unterstützen. Gerne möchten wir Ihr Angebot annehmen und einen ersten Beratungstag bei Dr. Holger Schwichtenberg vereinbaren.  
Wäre dies in den kommenden zwei Wochen möglich?  
Was benötigen Sie von unserer Seite zur Auftragserteilung?

Mit freundlichen Grüßen

Dr.-Ing. Max Mustermann  
Abteilungsleiter  
Lean Six Sigma Black Belt Certified  
Softwareentwicklung

Mustermann AG  
Musterstraße 123  
12345 Musterstadt

 Schließen  Im CRM Speichern

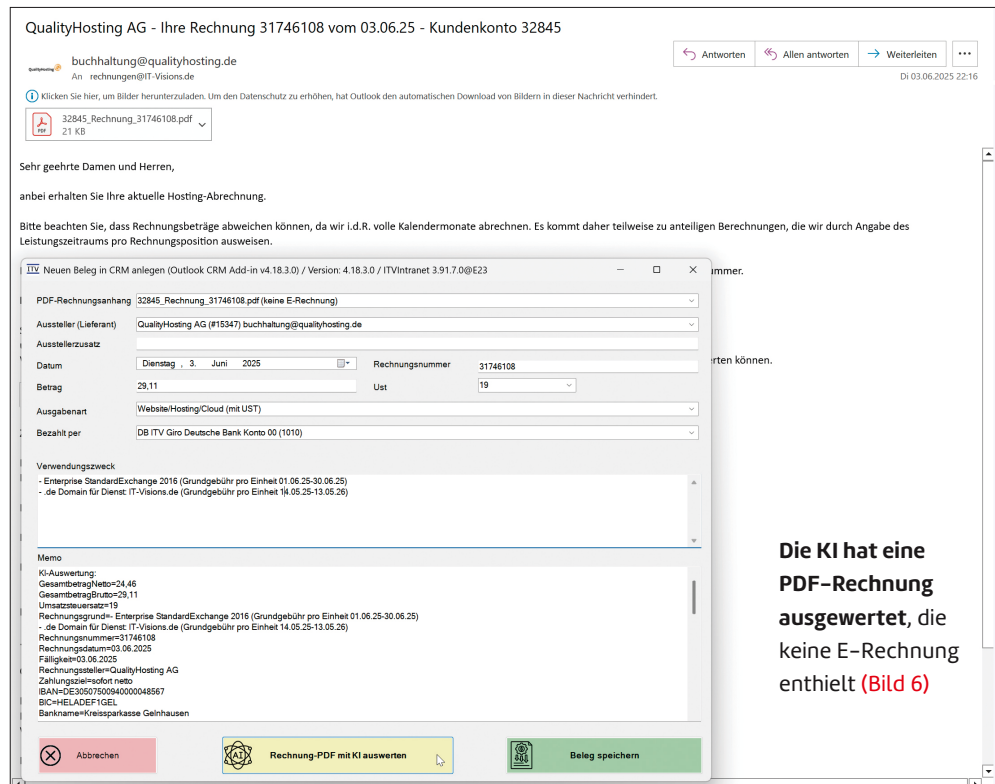


nur ein PDF ohne solchen XML-Anhang. Zum Glück hilft auch hier die KI.

Der Menüpunkt *Neuer Beleg* führt zu der in **Bild 6** dargestellten Bildschirmmaske, über die man eine Buchung mit zugehörigem Buchungsbeleg in unserer Buchhaltung erstellt. Zunächst prüft der Programmcode, ob sich in der Anlage eine E-Rechnung in Form einer XML-Datei (reine X-Rechnung) oder ein PDF mit X-Rechnungsanhang befindet. Das war im Fall der Rechnung des Providers QualityHosting AG vom 3. Juni 2025 leider nicht der Fall. Also musste die KI die Rechnung auswerten, und das hat sie auch getan, sodass die Felder für Betrag, Umsatzsteuersatz und Verwendungszweck gefüllt werden konnten. Intern wurden auch die IBAN und die BIC ausgelesen und mit dem Datenbestand im Lieferantenstamm abgeglichen. Das spart viel Arbeit!

## Azure AI Foundry

Es soll an dieser Stelle nicht dabei bleiben, dass ich Ihnen Ideen gebe, wie Sie mithilfe von KI Ihre eigenen Prozesse im



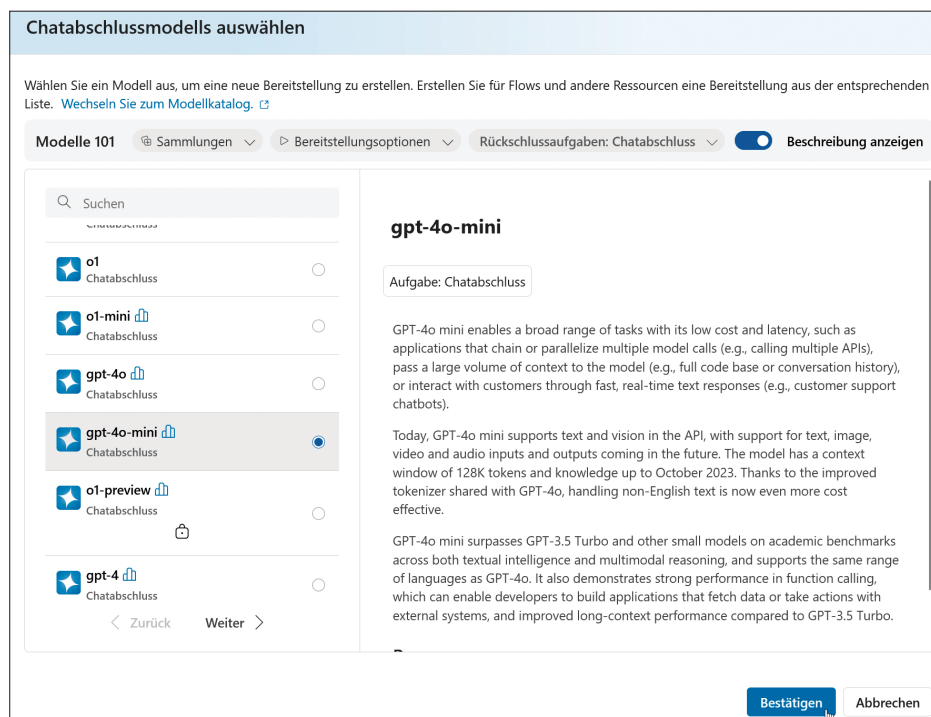
Die KI hat eine PDF-Rechnung ausgewertet, die keine E-Rechnung enthielt (**Bild 6**)

Unternehmen digitalisieren, sondern Sie sollen natürlich auch Hinweise zur technischen Umsetzung der gezeigten Outlook-Erweiterungen erhalten.

Es gibt viele Wege für die KI-Umsetzung, und hier sollen keineswegs mehrere oder alle davon beschrieben werden. Der Beitrag beschränkt sich auf den Weg, den wir (bislang) ge-

wählt haben. Ziel war es, zu einer schnellen Lösung zu kommen. Diese schnelle Lösung bestand für uns in einer Cloud-Lösung mit der Azure AI Foundry von Microsoft [2]. Azure AI Foundry bietet ein webbasiertes Portal, das Azure AI Foundry Portal [3]. Dort erstellt man in einem Assistenten eine sogenannte Bereitstellung eines KI-Modells (siehe **Bild 7**). Wir haben uns für GPT-4o entschieden, das anders als von ChatGPT angegeben (siehe **Tabelle 1** und **Bild 8**) für unsere Fälle schneller antwortete als GPT-4o-mini.

Einer Bereitstellung lassen sich dann Datenquellen hinzufügen (siehe **Bild 9** und **10**). Dabei kann man einen vorhandenen Index in Azure AI Search oder MongoDB Atlas nutzen oder aus dem Assistenten heraus einen neuen Index erstellen und dabei einen vorhandenen Speicherplatz angeben



**Einstellung** für die KI-Modell-Bereitstellung im Azure AI Foundry Portal (**Bild 7**)

oder einfach Dateien hochladen. Dabei gibt man als Ziel einen Azure-AI-Suchdienst an, den man entweder zuvor im Azure Portal angelegt hat oder, falls Letzteres nicht der Fall ist, hier aus dem Assistenten heraus erstellt.

Im letzten Schritt wählt man dann noch, ob man die Vektorsuche hinzufügen möchte. Das haben wir aktiviert. Das Hinzufügen der Vektorsuche unterstützt die Suchmodi Hybrid (Vektor + Schlüsselwort), Hybrid + Semantik (genaueste Suchergebnisse für generative KI-Anwendungen), Vektor-, Semantik- und Schlüsselwortabruf. Die Indexerstellung dauert einige Minuten, man sieht aber im Azure AI Foundry Portal den Fortschritt (siehe **Bild 11**).

Wer beide Assistenten erfolgreich durchlaufen hat, kann im Playground innerhalb des Azure AI Foundry Portals die Bereitstellung (Deployment) und die Datenquelle wählen und dann testweise mit den hochgeladenen Daten hier schon interagieren. Links darf man bei *Modellanweisungen und Kontext angeben* einen Systemprompt eingeben, und rechts darf man mit der eigenen KI chatten (siehe **Bild 12**).

## Client-Bibliothek für Azure OpenAI

Die Outlook-Add-in-Erweiterungen gab es bei uns wie erwähnt schon vor der KI-Integration. Sie basieren auf den Visual Studio Tools for Office (VSTO) [4], einer klassischen .NET-Framework-Technik. Ein Pendant im modernen .NET existiert leider nicht. Heute wäre ein Outlook-Web-Add-in wünschenswert, das nicht nur im klassischen Outlook auf Windows, sondern auch im neuen Outlook auf Windows, Outlook auf macOS und Outlook im Webbrowser [9] funktioniert. Aber noch fehlen diese Funktionen aus VSTO in den Outlook-Web-Add-ins, sodass eine Migration für uns noch nicht infrage kommt.

Bei der Auswahl einer Clientbibliothek für den Zugriff auf die in Azure AI Foundry erstellte KI-Modellbereitstellung war also darauf zu achten, eine Clientbibliothek zu wählen, die auch im klassischen .NET Framework läuft. Dies trifft auf die Azure OpenAI Client Library for .NET zu. Laut der Dokumentationsseite [5] ist „die Azure OpenAI-Clientbibliothek für

## gpt-4o-mini bereitstellen

Bereitstellungsname \*

ITVKI\_gpt-4o-mini

Bereitstellungstyp

Globaler Standard

Globaler Standard: Bezahlung pro API-Aufruf mit den höchsten Ratenlimits. Weitere Informationen [Globale Bereitstellungstypen](#).

Daten werden möglicherweise global außerhalb der Azure-Geografie der Ressource verarbeitet, aber der Datenspeicher verbleibt in der Azure-Geografie der KI-Ressource. Weitere Informationen [Datenresidenz](#).

^ Bereitstellungsdetails

Reduzieren

Richtlinie für Modellversionsupgrade

Upgrade durchführen, sobald eine neue Standardversion verfügbar ist

Modellversion

2024-07-18 (Standard)

Verbundene KI-Ressource

ITVAzureOpenAI

150M Token pro Minute Kontingent für Ihre Bereitstellung verfügbar

Ratenlimit: Token pro Minute

64.46M

Entsprechende Anforderungen pro Minute (RPM) = 644.62K

Inhaltsfilter

DefaultV2

Bereitstellen

Abbrechen

### Erstellen einer Bereitstellung im Azure AI Foundry Portal (Bild 8)

.NET eine Ergänzung zur offiziellen OpenAI-Clientbibliothek für .NET. Die Azure OpenAI-Bibliothek konfiguriert einen Client für die Verwendung mit Azure OpenAI und bietet zusätzliche stark typisierte Erweiterungsunterstützung für Anforderungs- und Antwortmodelle, die speziell auf Azure-OpenAI-Szenarien zugeschnitten sind. Azure OpenAI ist ein verwalteter Dienst, der es Entwicklerinnen und Entwicklern ermöglicht, OpenAI-Modelle auf Azure-Ressourcen bereitzustellen, anzupassen und Inhalte daraus zu generieren.“

---

**Feder von ChatGPT**

Das zugehörige NuGet-Paket ist `Azure.AI.OpenAI`. Vor der Integration in das Outlook-Add-in wurde erst ein Proof of Concept in einer .NET-Konsolenanwendung implementiert, siehe [Listing 1](#). Notwendig sind zwei Zugänge:

1. Ein Zugang zu der Azure-OpenAI-Bereitstellung: Dafür braucht man einen Endpunkt, einen API-Schlüssel und den selbst gewählten Namen der Bereitstellung. Den Endpunkt und den API-Schlüssel findet man auf der Startseite der Azure AI Foundry. ►

● **Tabelle 1: Vergleich GPT-4o versus GPT-4o-mini aus der Feder von ChatGPT**

| Merkmal                        | GPT-4o                                   | GPT-4o-mini (o4-mini)                   |
|--------------------------------|------------------------------------------|-----------------------------------------|
| Modelltyp                      | Multimodales Flaggschiffmodell           | Kleineres, effizienteres Modell         |
| Verfügbarkeit                  | ChatGPT (Plus), API (GPT-4o)             | ChatGPT (o4-mini), API                  |
| Leistung (Qualität)            | Sehr hoch (nahe GPT-4-turbo)             | Mittel (über GPT-3.5, unter GPT-4o)     |
| Multimodal (Text, Bild, Audio) | Ja (inklusive Sprache, Kamera et cetera) | Eingeschränkt (meist nur Text und Bild) |
| Latenz / Geschwindigkeit       | Höher (aufwendig)                        | Sehr schnell, niedrige Latenz           |
| Kosten (API)                   | Höher als mini                           | Günstiger                               |
| Empfohlen für                  | Komplexe Aufgaben, Multimodalität        | Schnelle, kostengünstige Anwendungen    |
| Kontextlänge (API)             | Bis zu 128k Tokens                       | Je nach Version deutlich weniger        |
| Training                       | Neueste Architektur (Mai 2024)           | Auch neue Architektur, aber kleiner     |

**Daten hinzufügen** VORSCHAU

1 Quelldaten  
2 Indexkonfiguration  
3 Sucheinstellungen  
4 Überprüfen und fertig stellen

**Daten auswählen**  
Wählen Sie die Daten aus, die die generative KI als Referenz verwenden soll, damit ihre Antworten auf Ihren spezifischen Daten basieren können.

Ihre Daten werden in einem Index erfasst, sodass das generative KI-Modell schnell und präzise Informationen für Ihren spezifischen Anwendungsfall finden kann.

Derzeit werden nur die Dateitypen TXT, MD, HTML, HTM, PY, PDF, PPT(X) und XLS(X) unterstützt. Die maximale Dateigröße beträgt 16 MB.

Datenquelle \* Dateien hochladen

Hochladen

☐ Überschreiben, wenn bereits vorhanden

**Liste hochladen**

|                                       |                     |     |
|---------------------------------------|---------------------|-----|
| Themen.txt                            | 88.38 KB/88.38 KB   | ... |
| www.IT-Visions.de_FAQ.pdf             | 627.66 KB/627.66 KB | ... |
| www.IT-Visions.de_Firmenbroschüre.pdf | 1.17 MB/1.17 MB     | ... |

Zum Indizieren Ihrer Daten sind eine Azure KI-Suche-Ressource und eine Azure OpenAI-Verbindung erforderlich. [Neue Azure KI-Suchressource erstellen](#) und erstellen Sie eine Verbindung mit ihr, um sie beim Erstellen eines Index auszuwählen.

**Erstellung einer Datenquelle (Bild 9)**

Weiter Vektorindex erstellen Abbrechen

**Daten hinzufügen** VORSCHAU

1 Quelldaten  
2 Indexkonfiguration  
3 Sucheinstellungen  
4 Überprüfen und fertig stellen

**Indexeinstellungen**  
Ihren Index konfigurieren

Indekspeicher \* Azure KI-Suche

Azure KI-Suchdienst auswählen \*   
AzureAISearch

[Neue Azure KI-Suchressource erstellen](#)

Vektorindex \*   
itvki-broschuere-faq-themen-juni2025

VM \*   
☒ Automatisch auswählen ☐ Aus empfohlenen Optionen auswählen ☐ Aus allen Optionen auswählen

Bei der Auswahl eines virtuellen Computers fallen zusätzliche Kosten an.

**Erstellung eines Azure-AI-Search-Index (Bild 10)**

Zurück Weiter Vektorindex erstellen Abbrechen

2. Für den Azure AI Search Service braucht man den Endpunkt und einen API-Schlüssel. Beides findet man im Azure Portal unter *AI Search*. Den URL findet man auf der Seite *Overview* und den API-Schlüssel auf der Seite *Settings/Keys*.

In [Listing 1](#) sind die sicherheitsrelevanten Zugangsdaten durch xxx ersetzt. In der Praxis wird man diese Daten aus einer geschützten Quelle holen. Der Aufbau von [Listing 1](#) ist recht einfach: Es wird eine Instanz der Klasse *AzureOpenAIClient* unter Angabe von Endpunkt und API-Key erzeugt. Von der

*AzureOpenAIClient*-Instanz holt man sich via Methodenaufruf von *GetChatClient()* unter Angabe des Namens der Bereitstellung eine Instanz von *OpenAI.Chat.ChatClient*.

Die Instanz von *ChatClient* wird auf zwei Weisen konfiguriert:

1. Die Temperatur der KI wird auf 0 gesetzt. Temperatur ist ein Parameter, der das Maß an Zufälligkeit oder Kreativität in den Antworten steuert. Der Wert 0 bedeutet, dass der *ChatClient* nicht kreativ sein soll, sondern deterministisch immer dieselbe Antwort liefern soll. Eine eindeutige Antwort ist, was wir wollen. Temperaturwerte über 1,0 machen die KI übrigens sehr chaotisch und unzuverlässig.

2. Mit dem Aufruf der Methode *AddDataSource()* wird der Azure-AI-Search-Index als Datenquelle hinzugefügt. Vor *AddDataSource()* sieht man in [Listing 1](#) ein `#pragma warning disable AOAI001`. Das ist nötig, weil Microsoft sich hier noch Änderungen am API-Design vorbehält. Ohne dies erscheint die Warnung: „Extension methods to use data sources with options are subject to SDK surface changes.“

An den Client sendet man dann Nachrichten via *CompleteChatAsync()*, wobei man eine Liste von *ChatMessage*-Instanzen übergibt. Konkret übergibt man dabei den Systemprompt

in einer Instanz von *SystemChatMessage* und die eigentliche Frage/Aufgabe in einer Instanz von *UserChatMessage*.

Als Ergebnis erhält man eine Instanz von *OpenAI.Chat.ChatCompletion*. Hier fragt man den erhaltenen Text via Unterobjekt *Content* ab.

In dem Programmcode in [Listing 1](#) wird bewusst bei jeder Frage eine neue Instanz von *AzureOpenAIClient* und *ChatClient* erzeugt, denn der spätere Anwendungsfall ist ja kein kontinuierlicher Chat mit einem Kunden, sondern die isolierte Beantwortung einzelner E-Mails. Die KI soll dabei na-

türlich nicht vorangehende Fragen von anderen Kunden und zugehörige Antworten im Hinterkopf haben!

Der Programmcode im Outlook-Add-in, der hier nicht abgedruckt ist, weil er über verschiedene Klassen verteilt ist, sieht dem Programmcode in Listing 1 sehr ähnlich. Bild 13 zeigt das Ergebnis.

## Verschiedene Aufgabenarten

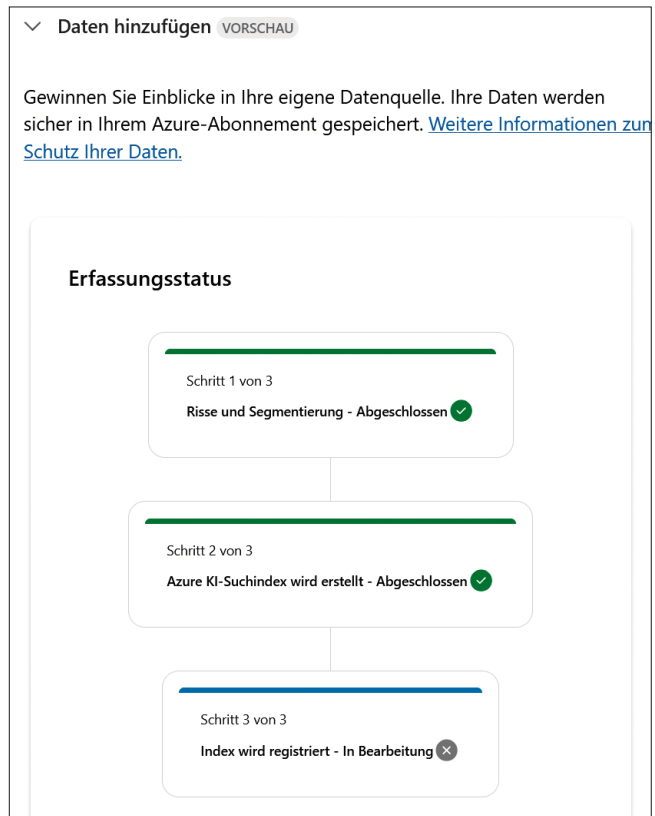
In der Realisierung des Outlook-Add-ins gibt es drei Arten von Aufträgen an die KI:

1. Auswertung vorhandener Texte mit Rückgabe von unstrukturiertem Text (zum Beispiel der Funktion *Bestehendem Angebot zuordnen* und *Neues Angebot anlegen*)
2. Auswertung vorhandener Texte mit Rückgabe strukturierter Daten (etwa bei *Neuer Beleg* und *Kontakt pflegen*)
3. Generierung von Texten (zum Beispiel bei *Antwort mit KI*)

## Text analysieren

Für den Fall 1 und 2 kann man sich die Verwendung der Datenquelle im Azure AI Search Service sparen, denn in diesen Fällen muss die KI nichts beziehungsweise nicht viel über unser Unternehmen wissen. Alles, was sie wissen muss, können wir via Systemprompt übergeben. Der Aufruf *AddDataSource()* entfällt hier also. Das spart Kosten.

Für die Extraktion der Texte aus der PDF-Rechnung kommt das NuGet-Paket PdfPig [7] zum Einsatz. Während PdfPig zwar bei *PdfPig.PdfDocument.Open()* einen Dateisystempfad, ein Stream-Objekt oder eine Bytefolge verarbeiten kann, ist das *Attachment*-Objekt von Outlook leider nicht in der Lage, einen Stream oder eine Bytefolge zu liefern. Daher muss man das *Attachment*-Objekt erst im Dateisystem persistieren, um dann per PdfPig die Texte zu bekommen, siehe Listing 2. Da



Das Azure AI Foundry Portal zeigt den Fortschritt der Indexerstellung an (Bild 11)

die *Attachments*-Menge des *MailItem*-Objekts von Outlook auch kein *IEnumerable* unterstützt, kann man auch nicht LINQ einsetzen, um den gewünschten Anhang zu finden. ►

Auf Basis von Bereitstellung, eigenen Daten und Systemprompt chatten wir mit unserer KI (Bild 12)



Der Systemprompt für die Extraktion von Daten aus dem Rechnungstext sieht so aus:

Extrahiere aus der Rechnung die folgenden Felder:  
 GesamtbetragNetto, GesamtbetragBrutto, Umsatzsteuersatz,  
 Rechnungsgrund, Rechnungsnummer, Rechnungsdatum,  
 Fälligkeit, Rechnungssteller, Zahlungsziel, IBAN, BIC,  
 Bankname.  
 Liefere nur diese Daten jeweils mit Name=Wert  
 Beträge ohne Eurozeichen.  
 Beträge in deutscher Schreibweise.  
 Prozentsätze ohne Prozentzeichen.

Bei Rechnungsgrund liste auch alle Rechnungspositionen mit Aufzählungszeichen auf.

Kein Markdown.

Wenn es keine Rechnung ist, liefere nur eine leere Zeichenkette.

Die KI liefert dann brav eine solche Zeichenkette, die sich leicht in .NET in ein Dictionary umwandeln und verwerten lässt:

GesamtbetragNetto=24,46  
 GesamtbetragBrutto=29,11

### ● Listing 1: Fragen an die KI via .NET-Konsole (Teil 1)

```
using System.ClientModel;
using System.Diagnostics;
using Azure.AI.OpenAI;
using Azure.AI.OpenAI.Chat;
using ITVisions;
using OpenAI.Chat;

namespace KI;

public static class AzureAIOpenAI
{
    #region Zugangsdaten --> bitte in appsettings.json
    #oder Umgebungsvariablen speichern
    static string azureOpenAIEndpoint =
        "https://itvazureopenai.openai.azure.com/";
    static string azureOpenAIKey = "xxx";
    static string deploymentName = "gpt-4o";
    static string searchEndpoint =
        "https://xxx.search.windows.net";
    static string searchKey = "xxx";
    static string searchIndex =
        "itvki-broschuere-faq-themen-juni2025";
    #endregion

    static string SystemPrompt =
        """
        Du bist ein KI-Assistent, der Fragen der Kunden von
        www.IT-Visions.de beantworten soll. Antworte höflich
        und aus der Sicht eines Kundenbetreuers. Gib eine
        ausführliche Antwort. Wenn Du nach einer Agenda
        gefragt wirst, dann ohne Zeitangaben.
        www.IT-Visions.de bietet keine anonymen/namenlosen
        Trainer und Berater an. Zusammen mit dem Angebot
        erhalten die Kunden eine konkrete Information,
        welcher Experte die Maßnahme bei Ihnen durchführen
        würde. Dem Angebot liegt auch ein Expertenprofil bei.

        Wenn jemand nach einer Schulungsagenda, einem
        Programmvorschlag oder einem Lehrplan fragt, suche
```

die Antwort aus Themen.txt oder die passenden  
 Themen heraus und gib dem Kunden den Link (URL).  
 Versuche nicht, eine Agenda selbst zu erstellen.

Wenn jemand nach einer Schulung oder Beratung  
 fragt, bitte die Person am Ende des Textes, eine  
 Preis- und Terminanfrage zu stellen:  
[www.IT-Visions.de/anfrage](http://www.IT-Visions.de/anfrage)

Themen, die nicht in der Themenliste genannt sind,  
 können auch angefragt werden:  
[www.IT-Visions.de/anfrage](http://www.IT-Visions.de/anfrage)

\* WICHTIG:

- Verwende nicht das Wort vermutlich!
- Keine Quellenangaben!
- Kein Markdown! Entferne alle \*\* für Hervorhebungen/Überschriften!
- Wenn Du eine Frage nicht beantworten kannst oder dazu keine Informationen verfügbar sind, bitte den Kunden um einen Anruf beim Kundenteam: +49 (0) 201 649590-50!
- Wenn jemand nach Beratung fragt, dann antworte auch mit Beratung! Erwähne dann nicht Schulung!

```
// Codequelle: https://learn.microsoft.com/en-us/
// azure/ai-services/openai/use-your-data-quickstart
// ?tabs=keyless%2Cjavascript-keyless%2C
// typescript-keyless%2Cpython-new&pivots=
// programming-language-csharp
public static async Task
    AzureOpenAICustomData(string Frage)
{
    CUI.LineYellow(Frage);

    var sw = new Stopwatch();
    sw.Start();
```

## ● Listing 1: Fragen an die KI via .NET-Konsole (Teil 2)

```
// ChatClient erstellen
AzureOpenAIClient azureClient = new(
    new Uri(azureOpenAIEndpoint),
    new ApiKeyCredential(azureOpenAIKey));
ChatClient chatClient =
    azureClient.GetChatClient(deploymentName);

// Optionen einstellen
ChatCompletionOptions options = new();
options.Temperature = 0.0f;

// Extension methods to use data sources with
// options are subject to SDK surface changes.
// Suppress the warning to acknowledge and this
// and use the subject-to-change AddDataSource
// method.
#pragma warning disable AOAI001
options.AddDataSource(new
    AzureSearchChatDataSource()
{
    Endpoint = new Uri(searchEndpoint),
    IndexName = searchIndex,
    Authentication = DataSourceAuthentication.
        FromApiKey(searchKey),
});

// Nachricht an den ChatClient erstellen
OpenAI.Chat.ChatCompletion completion = await
    chatClient.CompleteChatAsync([
        new SystemChatMessage(SystemPrompt),
        new UserChatMessage(Frage),
    ], options);

sw.Stop();

//Console.WriteLine(completion.Content[0].Text);
foreach (var c in completion.Content)
{
    Console.WriteLine(c.Text);
}

CUI.Cyan("Dauer: " + Math.Round((decimal)sw.
    ElapsedMilliseconds / 1000, 2) + " sek");

CUI.BR();
}
```

```
Umsatzsteuersatz=19
Rechnungsgrund=- Enterprise StandardExchange 2016 |
    - .de Domain
Rechnungsnummer=31746108
Rechnungsdatum=03.06.2025
Fälligkeit=03.06.2025
Rechnungssteller=QualityHosting AG
Zahlungsziel=sofort netto
IBAN=DE30507500940000048567
BIC=HELADEF1GEL
Bankname=Kreissparkasse Gelnhausen
```

Der Systemprompt für die Extraktion von Kontaktdaten aus einer E-Mail sieht ähnlich aus:

```
Extrahiere aus der EMail die folgenden Felder: Anrede,
AkademischeTitel, Vorname, Nachname, Telefon, Telefax,
EMail, Abteilung, Funktion, Straße, PLZ, Stadt, Land.
Liefere nur diese Daten jeweils mit Name=Wert
Die Anrede nur Herr oder Frau. Wenn unbekannt, dann
Leerlassen!
Verwende nur die letzte Nachricht im E-Mail-Thread.
Verwende auf gar keinen Fall Daten aus zitierten
Nachrichten!
Verwende auf gar keinen Fall Daten von www.IT-Visions.de
Wenn das Land Deutschland ist, lasse das Land leer!
Kein Markdown!
```

Natürlich war der Systemprompt im ersten Entwurf schlanker. Beim Testen kommt man darauf, welche zusätzlichen Hinweise die KI benötigt, also zum Beispiel, dass sie wirklich nur die letzte Nachricht in einer E-Mail-Konversation auswerten und die eigenen Firmendaten ignorieren soll.

Das gilt auch für den knappen Systemprompt bei *Bestehendem Angebot zuordnen*. Zuerst gab es nur den ersten Satz. Die KI neigte aber dazu, dabei zu schreiben, dass der/die E-Mail-Sender(in) freundlich sei oder herzlich begrüßt und/oder sich verabschiedet habe, was uns im Schnellüberblick über den E-Mail-Inhalt gar nicht interessiert.

Fasse die Kernaussagen von E-Mails sehr prägnant in bis zu 5 Stichpunkten zusammen.

Schreibe nichts darüber, wie der Autor sich verabschiedet hat oder wie er den Empfänger angesprochen hat.

## Texte generieren

Der Systemprompt für die Funktion *Antwort mit KI* ist hingegen sehr lang, denn die KI musste hier viel lernen. Zum Beispiel neigte die KI dazu, Kunden zu duzen. Wir mussten der KI sehr explizit beibringen, wie wir uns Anrede und Verabschiedungsformel vorstellen.

Du bist ein Assistent für die Erstellung von E-Mail-Antworten an Kunden. Antworte sehr höflich und aus der Sicht eines Kundenbetreuers. Antworte auf Basis aller ►

verfügbaren Informationen. Bitte formuliere eine freundliche, ausführliche Antwort an den Absender der aktuellsten E-Mail in der nachstehenden E-Mail-Konversation.

#### # Wichtige Regeln

- Beantworte die Frage primär mit den Texten aus den FAQ!
- Schreibe nicht den Betreff!
- Siezen ist der Standard!
- Duze den Empfänger nur, wenn er dich auch geduzt hat oder wenn er nur mit seinem Vornamen unterschrieben hat.
- Wenn du siezt, beantworte mit der Anrede Guten Tag und dann Herr bzw. Frau und dem Nachnamen.
- Wenn du duzt, beantworte mit der Anrede Hallo und dann den Vornamen.
- Wenn du siezt, beende die E-Mail mit Mit freundlichen Grüßen sowie unbedingt dem Vornamen UND dem Nachnamen \${GanzerName}
- Wenn du duzt, beende die E-Mail mit Viele Grüße und nur dem Vornamen \${Vorname}
- Wenn der Kunde ein Angebot wünscht, antworte mit ,Sie erhalten von uns in Kürze in getrennter E-Mail ein entsprechendes formales und sehr ausführliches Angebot. Für Rückfragen stehe ich Ihnen gerne jederzeit auch telefonisch zur Verfügung. Sehr gerne können Sie auch vor einer Auftragserteilung einen unserer Experten in einem kostenlosen Telefonat oder Videogespräch kennenlernen und fachliche Fragen zum Angebot erörtern, um sich von der inhaltlichen und didaktischen Kompetenz zu überzeugen. ‘

#### # Ausgabeformat

- Kein Markdown!

- Füge nach jedem Absatz eine Leerzeile in den Text ein!
- Keine Quellenangaben!
- Wenn du auf Themen aus Themen.txt verweist, dann mit Hyperlink!
- Prüfe nochmals, ob die Antwort zu den FAQ passt!

## Kosten und Nutzen

Unsere kaufmännischen Mitarbeiterinnen und Mitarbeiter sparen durch den KI-Einsatz viel Arbeitszeit. Der Kostenpunkt für die Cloud-KI bedrückt uns aber dennoch etwas: Zusammen mit einer Chat-Anwendung zahlen wir für die KI in Outlook an Microsofts Azure-Dienst rund 100 Dollar pro Monat und Nutzer.

## Ausblick

Für die Zukunft prüfen wir, aufgrund der Kosten zumindest einen Teil der KI ins eigene Haus zu holen, unsere Texte in einer eigenen Vektordatenbank zu speichern und diese mithilfe von Semantic Kernel [6] mit einem LLM zu verbinden. Unsere Themendatenbank exportieren wir bisher als Textdatei und geben diese Datei in Azure AI Search hinein. Mit der Umstellung auf Semantic Kernel könnten wir dort ein Plugin schreiben und unsere Live-Datenbank anzapfen. Einen Prototyp dieses Plug-ins haben wir schon.

Seit März 2025 macht Microsoft den Einstieg in Semantic Kernel mit dem .NET AI Template einfach. Dieses installiert man im modernen .NET mit dem .NET-SDK-Werkzeug:

```
dotnet new install Microsoft.Extensions.AI.Templates
```

Danach steht in Visual Studio oder per Kommandozeile die Projektvorlage *AI Chat Web App* (Kurzname: *aichatweb*) zur Verfügung. Es entsteht eine Blazor-basierte Chat-Anwen-

```
Microsoft Visual Studio Debug Console
www.IT-Visions.de KI Demo
.NET 9.0.6 @ Microsoft Windows 10.0.26100

--- Ich suche eine Schulung zu C# 13.0 ---
Wir bieten Ihnen eine Schulung zu C# 13.0 an, inklusive der Neuerungen gegenüber C# 12.0. Alle wichtigen Themen und Updates werden umfangreich behandelt. Weitere Informationen finden Sie hier :
C# 13.0 - Neuerungen gegenüber C# 12.0: https://www.it-visions.de/s11544
Bitte stellen Sie eine Preis- und Terminanfrage, damit wir Ihnen ein individuelles Angebot unterbreiten können: www.IT-Visions.de/anfrage
Dauer: 3,76 sek

--- Ich suche eine Schulung für Umsteiger von Java nach C# ---
Für Umsteiger von Java nach C# bietet IT-Visions.de maßgeschneiderte Schulungen an. Eine mögliche Schulung zu diesem Thema finden Sie unter folgendem Link:
C# für Umsteiger von Java: https://www.it-visions.de/s6441
Diese Schulung könnte genau auf Ihre individuellen Anforderungen zugeschnitten werden.
Bitte stellen Sie eine Preis- und Terminanfrage unter: www.IT-Visions.de/anfrage
Dauer: 2,78 sek

--- Wie groß ist die maximale Teilnehmeranzahl? ---
Die maximale Teilnehmeranzahl bei Schulungen mit Praxisübungen ("Hands-on") wird aus pädagogischen Gründen in der Regel auf 10 Personen beschränkt. Dies stellt sicher, dass die Zielerreichung , der Lernerfolg sowie die Zufriedenheit der Teilnehmenden optimal gewährleistet sind. Sollte die Teilnehmeranzahl die empfohlene Grenze von 10 Personen übersteigen, ist eine Rücksprache mit dem Experten notwendig, ob dies im Einzelfall möglich ist. Eine vorherige Abstimmung mit allen Teilnehmenden wird ebenfalls empfohlen, um mögliche Diskussionen oder Verstimmungen während der Veranstaltung zu vermeiden.
Für Schulungen ohne Praxisübungen kann die Gruppenstärke größer sein. Allerdings wird bei Beratungsmaßnahmen eine Gruppengröße von ein bis vier Personen empfohlen, um effektiv diskutieren zu können. Abschließend kann eine Zusammenfassung der Beratung vor einer größeren Gruppe erfolgen.
Bitte stellen Sie eine Anfrage für Preise und Termine: [www.IT-Visions.de/anfrage](www.IT-Visions.de/anfrage). Wenn Sie weitere Fragen haben, kontaktieren Sie unser Kundenteam unter +49 (0) 2 01 649590-50!
Dauer: 3,41 sek

FERTIG!

H:\git\ITVDemos\KI\KI\bin\Debug\net9.0\KI.exe (process 58676) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Die Konsolenanwendung stellt der KI drei Fragen (Bild 13)

## Listing 2: Extraktion der Texte aus einem PDF

```

var invoicePath = "";

string selectedFileName =
    (this.C_Attachments.SelectedItem as
    SelectListItemWithInvoiceDescriptor)?.Value?.
    ToString();
// Speichere das im UI gewählte Attachment im
// Dateisystem
for (int i = 1; i <= item.Attachments.Count; i++)
{
    Microsoft.Office.Interop.Outlook.Attachment a =
        item.Attachments[i];
    if (string.Equals(a.FileName, selectedFileName,
        StringComparison.OrdinalIgnoreCase))
    {
        invoicePath = Path.Combine(
            IcommingInvoiceBaseDir, a.FileName);
        a.SaveAsFile(invoicePath);
        break;
    }
}

if (invoicePath.IsNullOrEmpty())
{
    GUI.Warning("Anhang " + selectedFileName +
        " nicht gefunden!");
    return;
}

// Textextraktion aus PDF mit PdfPig
string rechnungsText = "";
using (var document =
    UglyToad.PdfPig.PdfDocument.Open(invoicePath))
{
    foreach (Page page in document.GetPages())
    {
        foreach (Word word in page.GetWords()) {
            rechnungsText =
                rechnungsText.Add(" ", word.Text);
        }
    }
}

// KI-Auswertung des Rechnungstextes
var invoiceData = await
    new KI().GetRechnungsdaten(rechnungsText);
if (invoiceData.IsNullOrEmpty())
{
    GUI.Warning("Die KI konnte keine Rechnungsdaten
        finden in dieser Datei:\n" + fileName);
    return;
}

// Aufspalten der KI-Antwort in Dictionary
var invoiceDataDic = DictionaryExtensions.
    FromNameValuePair(invoiceData);
...

```

derung, die mithilfe von Semantic Kernel eigene Daten verwendet, wobei die Daten wahlweise in Azure AI Search oder Local on-disk liegen. Bei Letzterem schreibt Microsoft dabei „for prototyping“. Im Test ist die Lösung, die einen lokalen Vektorspeicher im JSON-Format verwendet, aber gar nicht so langsam. Man kann dort auch PDF-Dokumente bereitstellen, deren Texte mithilfe des NuGet-Pakets PdfPig [7] extrahiert werden.

Microsoft plant, künftig auch eine KI-Vorlage für Konsolenanwendungen und ein ASP.NET Core Web API anzubieten.

Ein noch weiter gehender Schritt wäre, ein lokales LLM zu verwenden, zum Beispiel Ollama [8], das verschiedene freie LLMs via Gemma von Google und Llama von Meta oder das chinesische DeepSeek-R1 in einem HTTP-Server verfügbar macht. Dann brauchen wir lokale Rechenkraft, haben aber keine Cloud-Kosten mehr. ■

[1] Holger Schwichtenberg, *Pflichterfüllung*, *dotnetpro* 1/2025, Seite 88 ff.

[2] What is Azure AI Foundry?, <https://learn.microsoft.com/en-us/azure/ai-foundry/what-is-azure-ai-foundry>

[3] Azure AI Foundry Portal, <https://ai.azure.com/>

[4] Office solutions development overview (VSTO), <https://learn.microsoft.com/en-us/visualstudio/vsto/office-solutions-development-overview-vsto>

[5] Azure OpenAI client library for .NET, <https://learn.microsoft.com/en-us/dotnet/api/overview/azure/ai.openai-readme?view=azure-dotnet>

[6] Semantic Kernel, <https://learn.microsoft.com/en-us/semantic-kernel/overview/>

[7] PdfPig, <https://github.com/UglyToad/PdfPig>

[8] Ollama, <https://ollama.com>

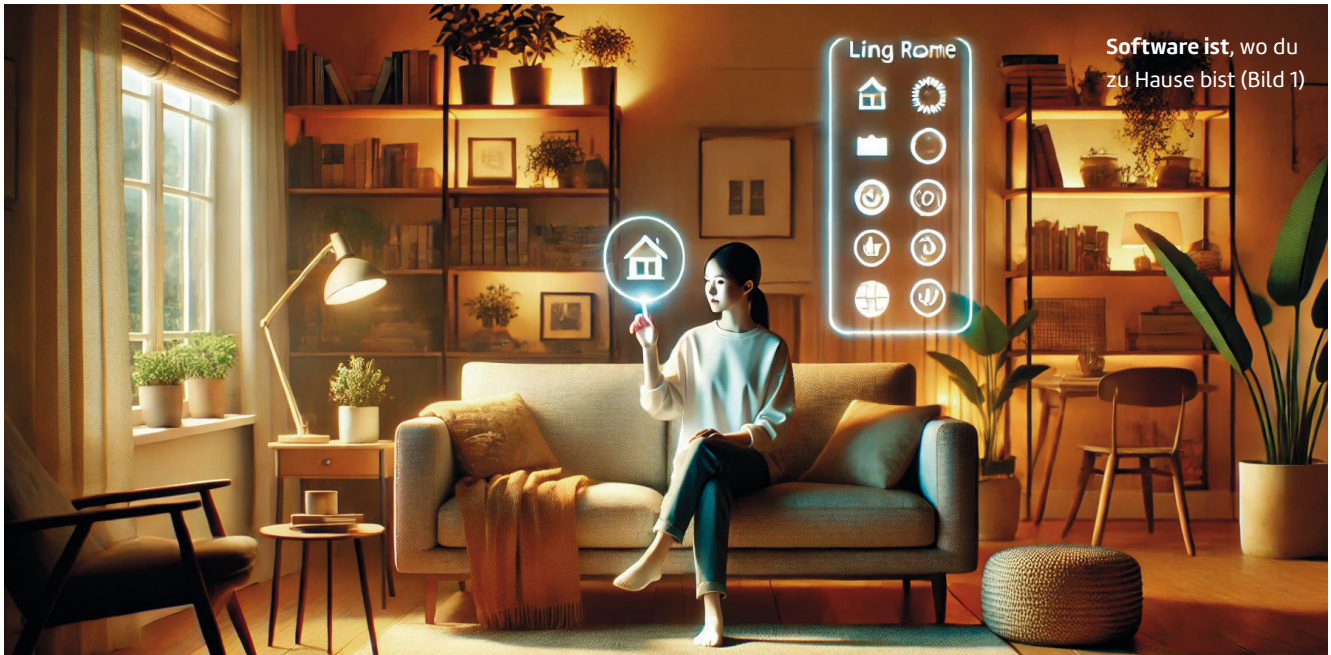
[9] Entwickeln von Outlook-Add-Ins für das neue Outlook unter Windows, <https://learn.microsoft.com/de-de/office/dev/add-ins/outlook/one-outlook>



**Dr. Holger Schwichtenberg**

zählt zu den bekanntesten .NET- und Webexperten in Deutschland. Microsoft zeichnet ihn seit 21 Jahren als MVP aus. Mit dem 53-köpfigen Expertenteam bei [www.IT-Visions.de](http://www.IT-Visions.de) bietet er Schulungen und Softwareentwicklung an. [www.dotnetdokter.de](http://www.dotnetdokter.de)





## WIE KI DIE SOFTWAREENTWICKLUNG VERÄNDERT

# Mit Denkmustern brechen

Womit sollten wir uns als Entwickler beschäftigen, um in einer sich ändernden Welt innovative und nützliche Antworten zu geben? Und wie sollten wir in Zukunft arbeiten?

Letzte Woche besuchte ich einen technischen Kongress, und Thema Nummer eins war – wen wundert es – die KI. Und es erstaunt auch nicht, dass viele Features als KI verkauft wurden, die es eigentlich nicht sind. Frei nach dem Motto: Was die Konkurrenz macht, dürfen wir schon lange. Das Buzzword „Cloud“ ist also abgelöst worden durch das Buzzword „KI“. So weit, so bekannt.

Das Spannende für mich war aber, dass, wenn an KI gedacht wurde und daran, was sie für uns in der Softwareentwicklung und als Anwender bedeuten könnte, immer noch von Fragestellungen ausgegangen wird, die uns derzeit bewegen. Also: Wie kann KI mir helfen, bestehenden Code besser zu kommentieren oder Testdaten zu schreiben, damit ich meine Unit-Tests optimiere. Wie kann eine KI mir bei der Bedienung der bestehenden Software besser helfen. Kurz gesagt: Wie kann KI meine bisherige Welt ohne KI verbessern. Das ist nur wenig zielführend, denn mit KI kommt ein neuer Player ins System, und so müssen wir die Frage stellen: Was kann KI tun, um die neue Welt mit KI zu verbessern?

Neues zu denken ist schwer. Unser menschliches Gehirn funktioniert ähnlich wie die KI selbst. Es ist ein neuronales Netz, das immer wieder aus der Vergangenheit geformt wurde. Sowohl die Fragen, die wir uns stellen, als auch die Antworten, die eine KI gibt, sind immer rückwärtsgewandt. Die KI hat (wie unser Gehirn) existierende Daten, die statistisch

ausgewertet und auf deren Grundlage Entscheidungen getroffen werden. Der Unterschied bei einer KI ist die umfangreichere Datenlage, und hier ist sie dem Menschen klar überlegen. Konkretes Beispiel: Auf die Frage, was man jemandem zum 50. Geburtstag schenken kann, kann eine KI auf alle in den letzten Jahren weltweit veröffentlichten Geschenke und auf alle Bestellungen zurückgreifen und somit beschreiben, was zurzeit in ist. Rückwärtsgewandt. Im Unterschied zu uns sicher bereichernd, weil wir eben noch nicht auf so vielen 50. Geburtstagen waren. Ein neuer Trend kann aber so nicht entstehen. Ein bestehender („Lasst uns Zeit schenken statt Gegenstände“) höchstens verstärkt werden.

Doch genau hier liegt unser großer Hebel. Wir müssen uns vorbereiten auf eine Welt, in der KI nicht nur Unterstützer, sondern aktiver Mitgestalter unserer Arbeit und unseres Lebens wird. Dies erfordert einen radikalen Bruch mit traditionellen Denkmustern. Innovation entsteht immer dann, wenn es uns gelingt, die bisherige Denkweise komplett zu verlassen.

Die Ausgangslage wird sich ändern mit einer KI. Uns so ist die Frage: Wie können wir als Entwickler vorbereitet sein für die Probleme von morgen. Mit was sollten wir uns beschäftigen, um in einer sich ändernden Welt innovative und nützliche Antworten zu geben? Und wie sollten wir in Zukunft arbeiten?

Auf diese Frage möchte ich mit Ihnen gemeinsam Antworten suchen. Das ist zugegebenermaßen etwas Kaffeesatzle-

serei, und wenn es dann auch noch praktische Auswirkungen für Sie im Hier und Jetzt haben soll, die Quadratur des Kreises. Aber wir wollen es zumindest einmal versuchen – Menschen sind immer kühn und wagemutig. Entweder, weil sie jung sind und die Schwierigkeiten nicht sehen, oder weil sie so alt sind wie ich und Vertrauen haben, dass sich jeder Weg im Gehen öffnet. Menschen sind wagemutig, obwohl sie mehr zu verlieren haben als eine Sammlung Daten, die wir KI nennen. Also, raus aus unserer Box! Lasst uns die richtigen Fragen stellen und dabei echte Innovation erreichen. Lasst uns brechen mit Bestehendem.

Wirklich brechen? Es gibt ja die Sorge, dass man Menschen auf Innovationen langsam mitnehmen muss. Schrittweise. Meine Erfahrung ist, dass wir immer dann, wenn wir das tun, eine Bauchlandung erleiden. Konkretes Beispiel: Wir nutzen KI, um schneller Reports zu erstellen, und denken dabei an Reports im klassischen Sinne wie Listen und Ausdrucke. Menschen sind dann frustriert, weil die Interpretation dieser Listen immer noch Aufgabe des Menschen bleibt und vielleicht nicht zu 100 Prozent so ist, wie man sich das gewünscht hat. Wir sind zu nah am Ergebnis, das wir kennen, so entsteht kein Wow-Effekt, der eine gewisse Unschärfe überdecken kann.

Oder wir fragen die KI, wie wir schneller Eingabemasken generieren können, die individuell nur für diesen einen Benutzer und seine Bedürfnisse gemacht sind. Das wird nicht klappen, weil wir eben hier neue Antworten auf alte Bedürfnisse geben, und dann kommen Dinge raus wie die „adaptiven Menüs“ von Office XP (2002), bei dem selten benutzte Menüpunkte ausgeblendet wurden. Was war das für ein Aufschrei unter den Usern, wenn das UI jedes Mal anders ist und man nichts mehr findet. So kommen wir nicht weiter. Nicht kleckern, sondern klotzen! Keine Menüs mehr. Wozu auch!

Für mich persönlich ist Innovation immer ein radikaler Bruch: Ein Handy, das plötzlich ohne Tasten auskommt, weil es einen (Touch-)Bildschirm hat. Innovationen sind disruptiv, und deshalb nehmen sie uns als Benutzer mit. Sie sind durchdacht. Und sie haben die Fragen von morgen beantwortet. Und so müssen auch wir es bei der Frage, wo und wie eine KI helfen kann, machen.

Dazu ein Beispiel: Wir haben uns wohl alle gefragt, wie das funktionieren kann, wenn Kontonummer und BLZ plötzlich durch eine 22 Zeichen lange IBAN abgelöst werden. 22 Ziffern! Wer kann sich die merken? Hätten wir alte Fragen gestellt, dann wäre die Antwort gewesen, dass eine KI uns unterstützen kann, die Nummer sauber einzugeben. Oder zu validieren. Oder eine Nummer vorzuschlagen, die unserer Nummer sehr ähnlich ist (ein bisschen Humor muss sein).

Mit der Foto-Überweisung über das Handy kommt aber etwas radikal Neues. KI hat es ermöglicht, eine komplette neue User-Erfahrung zu machen, ein klassischer Bruch mit dem bisherigen Bedienparadigma. Am Ende ist es einfacher, und das nimmt die Benutzer mit. Es wurde die richtige Frage gestellt (wie würde eine ideale Überweisung aussehen) und die richtige Antwort darauf gegeben (so einfach wie möglich).

Unser Learning: Eine KI zu fragen ergibt immer Antworten auf Fragen von gestern. Innovation entsteht durch einen Bruch! KI wird zum einen die Art, wie wir arbeiten, verändern, aber

## ● Wortshuffle

Die dtonrepto ist Eauoprs betse Zecirithfst und enähltt für mcih nlzücthie lmaoitonfrnen, die mir in meenim Buref wertlftheien. Es bahurct zawr bei deseim Txet eawts, aebr bei eneir nueen Spahcre ist das ja iemmr so. Man msus scih esrt ein weing daarn gwöeehnn. Und ich furee mcih, wnen ich in den Aetkriln der dpenotrtu acuh mal was Nuees erarhfe, an das ich mcih esrt geewöohnn msus.

genauso auch die Inhalte, an denen wir arbeiten. Wir sollten also unterscheiden, welche Anforderungen eine KI-Assistenz an unsere eigene Arbeitsweise stellt, und gleichzeitig auf den Prüfstand stellen, an was wir gerade arbeiten.

Zunächst zum Spannenderen: der Frage nach dem Was.

## Wie sieht in Zukunft Software aus?

Schauen wir uns zuerst an, wie sie bisher aussieht: Ich habe einen Input (hier brauchen wir einen Menschen, der analoge Daten in eine digitale Form bringt), einen klaren Algorithmus und einen entsprechenden Output.

Beispiel: Ich suche einen Mietwagen. Ich gebe einige Daten in eine Oberfläche ein, bekomme eine Liste von verfügbaren Angeboten dieses Anbieters, gebe wieder Daten ein (Kreditkarte und Versicherungsoption) und bekomme eine E-Mail mit der Reservierungsbestätigung. Ein klarer Prozess. Die Rolle des Menschen ist dabei, der Software über klassische Eingabemasken zu sagen, was man möchte. Und durch Interaktion mit dem Screen zu einem Ergebnis zu kommen.

Was kann eine KI hier leisten? Ich will versuchen, die Aufgabe der KI in drei große Themenbereiche zu unterteilen: KI kann helfen

- Medienbrüche zu überwinden,
- lernend und vorausschauend zu unterstützen und
- einzelne Aktionen zu einem Szenario mit Kontext zusammenzufassen.

## Medienbrüche überwinden

Wenn wir noch mal zurück zu der Online-Überweisung gehen, war der Medienbruch, dass wir eine ausgedruckte Rechnung hatten, die wir in ein digitales Format für eine Online-Überweisung bringen müssen. Oder wir haben wie in unserem zweiten Szenario einen Wunsch („Ich brauche einen Mietwagen“) und müssen ihn in eine Oberfläche packen.

KI hilft hier, diese Eingabeformulare zu ersetzen – und damit Medienbrüche zu eliminieren. Es wäre altbacken, zu fragen, wie eine KI die Eingabeformulare moderner gestalten kann.

Eine Eingabe kann eine Sprache sein. Eine Eingabe kann durch die Beobachtung meines Verhaltens entstehen. Eine Eingabe kann aufgrund des Kontextes vielleicht schon bekannt sein. Die große Stärke der KI ist es, aus heterogenen und vagen Daten scharfe und strukturierte Daten zu machen.

Das ist ein Quantensprung! Denn bisher konnte nur der Mensch in digitalen Prozessen Medienbrüche überwinden. ►

Ich denke hier an das klassische Bild eines Büroangestellten, der vor einem Kopierer steht. Seine Aufgabe: Papier auf eine genau vorgegebene Glasfläche zu platzieren und Papier einzulegen in einen eigenwilligen Schacht, der nur dann „voll“ anzeigt, wenn alles richtig drinliegt. Der Mensch als Zulieferer und Hilfskraft an einem technischen Gerät, weil dieses eben nicht selbst in die reale Welt eingreifen und Papier einlegen kann. Hier wird sich die Rolle des Users verändern.

Wir erleben das schrittweise: Zuerst OCR, also die Umwandlung von Bildern in Text ohne Bedeutung. Der nächste Schritt war eine automatische Erkennung einer Rechnungsnummer, weil meist das Wort „Rechnungsnummer“ davorstand. In mehreren algorithmischen Stufen haben wir uns angenähert und versucht, diese Nummer anhand von Logiken zu erkennen. Mit KI kommt aber eine echte semantische (!) Erkennung und eine Interpretation des Dargestellten, und das ist etwas völlig Neues. Der richtige Umgang mit Unschärfe, also mit nicht hundertprozentig klaren Daten, ist der entscheidende Unterschied. In der klassischen Entwicklung mit deterministischer Algorithmik gibt es nur ein richtig oder falsch bei der Erkennung der Rechnungsnummer. Der Ansatz, hier mit Wahrscheinlichkeiten zu arbeiten (KI ist ja nichts anderes als Wahrscheinlichkeitsrechnung), ermöglicht völlig neue Szenarien. Die Rechtschreibung ist nicht mehr wichtig, nicht einmal mehr eine korrekte Grammatik.

Auch wir funktionieren so: Vielleicht kennen Sie das Beispiel eines längeren Textes, bei dem nur die Länge der Wörter und der erste und letzte Buchstabe stimmen, die Buchstaben in der Mitte aber vertauscht sind (siehe Kasten **Wort-shuffle**). Wir können den Text trotzdem lesen. Wir und KI können mit Unschärfe umgehen, aber versuchen Sie einmal, mit klassischem Programmiercode diese Aufgabe zu lösen ...

Aber nicht nur der Umgang mit geschriebenem Text verändert sich. KI verknüpft Sprache, Gestik und Blicksteuerung, sodass in Zukunft eine natürlichere und schnellere Interaktion mit Software möglich wird. Touchscreens und Maus-Interaktionen werden erweitert um eine kombinierte Sprache-, Gesten- und Blicksteuerung: Konkretes Beispiel: Beim Lesen eines Artikels könnte der Benutzer per Blick auf ein Wort direkt eine Definition abrufen oder per Sprachkommando Passagen zusammenfassen lassen. Das verbessert die Barrierefreiheit und steigert die Effizienz. Neue Benutzerschnittstellen schaffen neue Use Cases.

Auch die Barriere zwischen verschiedenen Sprachen kann wegfallen. Apps übersetzen nicht nur Texte, sondern passen idiomatische Ausdrücke, Tonfall und kulturelle Referenzen dynamisch an das eigene Muttersprache-Niveau an.

Die Eingabe wird anders, aber wie ist das mit der Ausgabe? Wollen wir eine Liste der Mietwagen oder die gewünschte IBAN in einem Formular darstellen? Eher nicht. Denn was nützt es mir, die IBAN noch einmal zu lesen. Ich kann damit nichts anfangen. Aber – und jetzt kommt der Kontext – man könnte mir anzeigen, wann und wie ich diese IBAN bereits verwendet habe. Versuchen, den Inhalten über andere Medien mehr Bedeutung zu geben. Dass Daten immer eine Bedeutung bekommen, wird durch den ganzheitlichen Ansatz der KI möglich.

Die Frage an uns Entwickler oder Produktmanager ist also: Wie will ich, dass der User mit meinem Mehrwert, den ich durch die Software bringe, interagieren soll. Wie kann ich es schaffen, mehrere Wege zur Interaktion zu ermöglichen? Welche neuen Use Cases ergeben sich daraus?

## Aus Vergangenheit Zukunft erahnen

Sobald eine KI Zugriff auf eine Vielzahl persönlicher heterogener Daten hat, können daraus Vorhersagen über ein zu erwartendes Verhalten oder einen zu erwartenden Wunsch gemacht werden.

Die technische Hürde dazu ist mit der KI gefallen. Organisatorische und kommerzielle Hindernisse gibt es allerdings noch. Wenn wir uns innerhalb einer App befinden und diese App alle Daten vorhält, ist es ein Leichtes, beispielsweise individualisierte Trainings- und Lernpläne zu erstellen. Die KI muss nur die bisherigen Ergebnisse auswerten und gezielt Übungsinhalte generieren. Die Fremdsprachen-App erkennt zum Beispiel die häufigsten Fehler und generiert gezielt Übungen dazu, ohne dass man selbst Inhalte auswählen muss. Auch Kalender-Apps können basierend auf den bisherigen Daten (Ereignisse, Standorte und Gewohnheiten) proaktiv Optionen vorschlagen (zum Beispiel Reisemöglichkeiten, Einkaufsliste), bevor man selbst daran denkt. Dazu ist nicht einmal eine Bildschirmoberfläche nötig. Die Smartwatch könnte, nachdem ihr Wecker geklingelt hat, dich per Sprache fragen: „Dein Blutdruck ist erhöht, möchtest du jetzt nach dem Aufstehen gleich eine Übung machen?“, und die Handbewegung des Benutzers gibt darauf die richtige Antwort. Man muss sich nicht merken, welche Geste was bedeutet, man muss sich nur entsprechend verhalten: Aufstehen oder liegen bleiben. Auch hier wird die Zahl der Medienbrüche geringer.

Das mag nach Science-Fiction klingen, und mich selbst befallen viele Zweifel, ob es jemals so weit kommen wird. Zwei Blocker stehen noch vor der Tür ins vermeintliche Paradies: Erstens: Will ich das überhaupt, diesen tiefen Eingriff in meine Wünsche und mein Verhalten? Wie kann Software also so gestaltet sein, dass sie subtil und nicht aufdringlich ist? Und das zweite: Wer hat denn alle Daten, die mich auszeichnen? Die Reise-App kennt nur meine bisherigen Hotelbuchungen, unwissend, ob privat oder beruflich. Mein Wecker kennt nur die Zeiten, zu denen ich aufstehe, nicht aber die Gründe dafür. Wer weiß, was ich bestellt habe? Eine KI kann erst dann etwas prognostizieren, wenn sie auf diese Daten Zugriff hat.

Und man sieht, warum die Big Player (Apple, Amazon, Google et cetera) ein so riesiges Bedürfnis haben, Daten zu sammeln. Mit diesen Daten und der KI wissen sie bereits, wann die eigene Frau schwanger ist und was sie braucht, bevor sie es selbst weiß. Und können auf unterschwellige und unbewusste (Kauf-)Bedürfnisse eingehen (**Bild 1**).

## Szenarien mit Kontext

Das Zusammenführen aller Daten und das Erstellen von Bedürfnissen im jeweiligen Kontext (Ich reise in den Urlaub, werde dort mit dem Segelboot unterwegs sein und mich interessieren Restaurants in der Nähe von Ankerplätzen) erlauben ganz andere Business Cases als bisher.



KI kann hier die Orchestrierung der Aufgaben übernehmen und die jeweilige Bedeutung einer Entscheidung (ich reise mit dem Boot) auf andere Entscheidungen (ich kann nicht mit dem Auto an Land, also fallen gewisse Restaurants aus) berücksichtigen. Flug- oder Zugverbindungen werden basierend auf Terminkalender, Wetter und Verkehrsprognosen automatisch optimiert und dann zur Buchung vorgeschlagen. Eine KI könnte das Verhalten (über eine Kamera?) in Echtzeit analysieren und so genau die Inhalte, Funktionen und Empfehlungen anbieten, die relevant sind, ohne dass man Wünsche über einen Screen eingeben muss.

Routineaufgaben wie Sortieren von E-Mails, Terminplanung oder Übersetzung laufen vollautomatisch im Hintergrund ab, und die Inhalte werden miteinander in Beziehung gebracht. Apps und Services passen sich automatisch an das eigene Nutzungsverhalten an (zum Beispiel Lesegewohnheiten, Gerätestandort und Tageszeit). Dadurch sieht man immer die relevantesten Inhalte, ohne manuell filtern zu müssen.

Das macht das Modellieren von Software aber extrem komplex, denn nicht jeder Use Case steht fest. Vielleicht ergeben sich aufgrund der KI ganz neue Use Cases, die den „Entwicklern“ der Software selbst vorher nicht bekannt waren. Wichtig ist, dass wir Schnittstellen herstellen, sodass sich KI der Werkzeuge bedienen kann und dann komplexe Zusammenhänge orchestriert. Da wir aber meist nur eine bestimmte Software für einen bestimmten Use Case machen, müssen wir umdenken und kleine Services bereitstellen, derer sich eine KI bedienen kann. Und dann? Wie wird das lizenziert, abgerechnet, wie taucht man dann als ISV auf? Ehrlich: Ich habe keine Ahnung. Aber diese Fragen sollten wir uns stellen.

## Die Rolle der Programmierer

Mit der Software verändert sich auch die Rolle der Programmierer fundamental. Wo wir früher klar definierte Prozesse programmatisch umgesetzt haben, wird unser Job in Zukunft sein, Bedingungen zu schaffen, in denen KI sinnvoll agieren kann. Wir werden also immer weniger explizit vorgeben, wie etwas zu tun ist, sondern stattdessen Szenarien und Zielzustände beschreiben, in denen eine KI optimal wirken kann.

Ein zentraler Aspekt dabei ist die Anbindung an die Interpretation durch KI. Wir entwickeln nicht mehr primär Benutzeroberflächen, sondern Interaktionsmöglichkeiten. Unser Job ist, Interfaces zu gestalten, die KI-getrieben sind.

Neu ist, dass wir nicht mehr alles im Detail beherrschen müssen. Wer kennt schon das Drag-and-Drop Framework mit allen Interfaces? Aber wir wissen, wie cooles UI-Design geht, und können unsere Wünsche nach Drag-and-Drop-Funktionalitäten, dynamischen Animationen, intuitiven automatischen Kalkulationen und so fort an eine KI übergeben, die uns hilft, das alles in ein paar Minuten umzusetzen – und wieder komplett über den Haufen zu schmeißen und ganz neu anzugehen, wenn es sich nicht gut anfühlt. Entwicklung als Trial and Error von kompletten Komponenten. Wie oft haben wir uns nicht an eine Aufgabe getraut, weil sie ein komplettes Refactoring erfordert hätte, das uns stundenlang beschäftigt gehalten hätte. KI kann hier die Aufgaben eines Junior-Entwicklers übernehmen, man muss sie nur fordern.

Als Entwickler werden wir vielleicht mehr experimentieren. Wir haben eine Ahnung, wie es funktionieren könnte, es reicht, diese zu beschreiben. Ach ja, das leidige Thema Deutschkenntnisse. Ich denke, sprachliche Präzision ist wichtig. Wer gut kommunizieren kann, tut sich mit dem Sprachinterface einer KI leichter, formuliert nicht missverständlich.

Um eine Oberfläche zu testen (ich denke, wir werden viel stärker in Richtung Testing gehen) gewinnt das Beherrschen von Edge Cases an Bedeutung. Funktioniert die Idee auch noch mit 1000 Datensätzen oder wenn gar kein Inhalt da ist? Denn KI-Modelle arbeiten typischerweise mit Durchschnittsdaten. Es ist die Aufgabe von Programmierern, diese „Randfälle“ vorherzusehen, bewusst herbeizuführen und die KI entsprechend zu trainieren.

Das erfordert tiefes Domain-Wissen, denn nur wer fachlich versiert ist, erkennt, welche Fälle überhaupt relevant sein könnten. KI kann nur so gut arbeiten wie das Wissen, das ihr zugrunde liegt. Unser Job ist es daher, Fachwissen kontinuierlich aufzunehmen, zu strukturieren und in die KI einfließen zu lassen. Das bedeutet auch, dass lebenslanges Lernen essenziell wird – wir müssen nicht nur neue Programmiersprachen beherrschen, sondern auch die Fachbereiche, für die wir Lösungen schaffen. Aber Lernen macht auch immer mehr Spaß, weil man gezielt nachfragen kann und so deutlich schnellere Ergebnisse erzielt.

Der Fokus verschiebt sich zudem in Richtung Architektur und Prozesse. Wir definieren nicht mehr exakt jeden Schritt, sondern legen Regeln und Rahmenbedingungen fest, die KI helfen, die richtigen Entscheidungen zu treffen. User-Stories müssen neu gedacht werden, sie müssen stärker als zuvor Szenarien beschreiben und weniger konkrete Ablaufwege.

Last, but not least: Guidelines werden an Wert gewinnen. Wir müssen der KI präzise Vorgaben machen, wie sie Szenarien interpretieren und ausführen soll. Das erfordert nicht nur technisches, sondern vor allem sprachliches Können. Die Qualität von Guidelines entscheidet maßgeblich darüber, ob eine KI stabile Ergebnisse liefert oder nur Blabla-Code.

Unsere Rolle wird anspruchsvoller, spannender und definitiv zukunftssträchtiger. Wenn wir bereit sind, alte Denkmuster abzulegen und KI als Partner auf Augenhöhe zu betrachten, entstehen Lösungen, die wirklich innovativ sind und echte Mehrwerte schaffen.

Wenn wir diese Herausforderungen annehmen und traditionelle Grenzen überwinden, haben wir die Chance, bahnbrechende Innovationen zu schaffen, die nicht nur unser Arbeitsleben, sondern unsere gesamte Gesellschaft nachhaltig verändern. ■



**Dipl.-Theol. Bernhard Pichler**

war 20 Jahre Geschäftsführer der Softwarefirma informare, bevor er 2020 zum Leiter Entwicklung und Produktmanagement der DPS BS ernannt wurde. Die DPS BS ist Deutschlands größter Softwarepartner für betriebswirtschaftliche Software für den Mittelstand.



## UNWICHTIG GIBT'S NICHT

# Das Rollenproblem

In der Softwareentwicklung sind zahlreiche Rollen definiert. David Tielke zeigt, warum es so wichtig ist, diese konsequent zu besetzen – auch diejenigen, die nicht unmittelbar zum Umsatz beitragen.



Unterschiedliche Sprachen, Plattformen, Branchen, Anwendungsarten – ich habe in meiner Karriere schon viele Projekte gesehen. Neben der Architektur und der Qualität fällt in meinen Verantwortungsbereich meistens auch die Aufgabe, das Team, dessen Struktur sowie die verschiedenen Rollen zu hinterfragen und zu besetzen. Dabei gibt es immer zwei Arten dieser Rollen: Rollen, die direkt für Umsatz sorgen, und Rollen, die dies nicht tun.

Nun ist es natürlich auch in der Softwarebranche so, dass man so günstig wie möglich produzieren möchte. Und Rollen zu besetzen kostet Geld. Da könnte man leicht auf den Gedanken kommen, auf einzelne Rollen zu verzichten. Solche Gedankenspiele betreffen besonders diejenigen Rollen, die nicht direkt zum Umsatz beitragen; diese werden von Entscheidern dann auch nicht selten weggelassen. Zumal die Konsequenzen oft nicht direkt zu Projektbeginn ersichtlich sind, sondern erst mittel- oder langfristig – und dann allerdings mit oft enormen Kosten verbunden sind.

In diesem Beitrag nehmen wir uns dieser Fragestellung an und zeigen, was eine Rolle ausmacht und warum sie wichtig ist und besetzt werden sollte. Klingt nach einem langweiligen Thema? Ja, zugegeben – es mag spannendere geben. Aber glauben Sie mir: Kaum ein Fehler sorgt in der Softwareentwicklung für so gravierende Probleme wie die falsche oder gar unterlassene Rollenbesetzung.

## Was ist eine Rolle?

Vorab müssen wir mit einem häufigen Missverständnis aufräumen, denn viele verstehen den Begriff einer Rolle falsch. Eine Rolle ist immer ein Paket aus Verantwortlichkeiten, Pflichten und Kompetenzen. Die Verantwortlichkeiten besagen, für welchen Bereich die Rolle die Verantwortung trägt ... und später bei Erfolg gelobt beziehungsweise bei Misserfolg getadelt wird. Die Pflichten beschreiben, welche Tätigkeiten innerhalb der Softwareentwicklung durch diese Rolle zu erfüllen sind. Die Kompetenzen besagen schließlich, welche Befugnis die Rolle bekommt, um diese Verantwortlichkeiten und Pflichten zu erfüllen.

Ein Softwareentwickler etwa hat die Pflicht, die an ihn gestellten Tickets zu implementieren, also funktionale Mehrwerte zu schaffen. Seine Verantwortlichkeit ist es, diese Implementierung in der geforderten Qualität durchzuführen, und seine Kompetenz dazu ist es (unter anderem), den dazu erforderlichen Aufwand zu bestimmen.

Auch wenn diese Beschreibung für Sie „normal“ klingt, sollten Sie das für Ihre Entwickler einmal hinterfragen: Wird den Entwicklern vorgegeben, welche Qualität gefordert wird? Hat der Entwickler wirklich die Kompetenz, die Aufwände zu schätzen? Übernimmt der Entwickler mehr Pflichten als nur das Lösen von Tickets? Auch wenn es merkwürdig klingen mag: Allein mit solch simpel anmutenden Fragen kann man oft ernst zu nehmende Probleme innerhalb der Entwicklung identifizieren.

Neben diesem Paket aus Verantwortlichkeiten, Pflichten und Kompetenzen gibt es aber eine weitere wichtige Eigenschaft einer Rolle: Sie muss nicht zwingend eins zu eins an eine Person gekoppelt sein. Eine der häufigen Antworten, die ich von Entscheidern in Beratungsterminen höre, lautet: „Ja, aber wir stellen dafür doch nicht eigens jemanden ein oder opfern einen Entwickler.“ Das muss aber oft auch gar nicht sein, denn eine Person kann immer mehrere Rollen bekleiden, ganz besonders in kleineren Projekten, wo die erforderlichen Ressourcen für dedizierte Stellen nicht vorhanden sind und auch die Menge an Aufgaben eine Rolle gar nicht auslasten würde.

## Die Rollengruppen

Wie eingangs schon angesprochen, sorgen manche Rollen im Unternehmen direkt für Umsatz, andere wiederum nur indirekt. Das hängt damit zusammen, für welche Art von Anforderungen sie zuständig sind. Innerhalb der Softwareentwicklung gibt es viele verschiedene Arten von Anforderungen; hauptsächlich unterscheiden wir die funktionalen und die nicht funktionalen. Die funktionalen Anforderungen bestimmen das „Was?“ in der Entwicklung, also welche Funktion wünscht der Kunde in der Software.

Nehmen wir als Beispiel ein Unternehmen, das Webshops entwickelt. Stellen Sie sich vor, ein Kunde kontaktiert den Product Owner und wünscht eine Funktion, um den Webshop sperren zu können, sobald irgendwelche böse Aktivität am Werk ist. Wäre ich in der Rolle des Product Owners, könnte ich mit den Entwicklern sprechen, sie um eine Aufwandschätzung bitten und dem Kunden einen Preis nennen, zum Beispiel 5000 Euro. Würde die Funktion in Auftrag gegeben, implementiert und ausgeliefert, könnte ich sie in Rechnung stellen. Funktionale Anforderungen erzeugen somit immer direkt Umsatz in der Softwareentwicklung.

Die nicht funktionalen Anforderungen hingegen beschreiben das „Wie?“, also wie schnell soll die Funktion sein; wie bedienbar, effizient, erweiterbar oder testbar ist sie. Diese Art von Anforderungen nennt unser Kunde nur selten direkt, oft interessieren sie ihn gar nicht, denn es ist seine Erwartungshaltung, dass alles schnell, effizient und bedienbar ist. Den nicht funktionalen Anforderungen kann ich also kein Preisschild umhängen und mit ihnen demnach keinen direkten Umsatz generieren.

Das wirkt sich auch auf die Rollen aus. Ist eine Rolle an der Lösung oder Aufnahme von funktionalen Anforderungen beteiligt, erzeugt sie direkten Umsatz. Ist sie an der Ermittlung und Umsetzung nicht funktionaler Anforderungen beteiligt, erzeugt sie eben keinen direkten Umsatz. Zu den umsatzgenerierenden Rollen zählen Product Owner, die Anforderungen aufnehmen, und Entwickler, die diese Anforderungen umsetzen. Deshalb sind diese Rollen auch in jedem Projekt vertreten. Nicht umsatzgenerierende Rollen sind Tester, Architekten, Scrum Master oder Feel-Good-Manager.

### Wenn „unwichtige“ Rollen fehlen

Wenn die letztgenannte Gruppe nun also nicht direkt zum Umsatz beiträgt: Was ist dann ihr Sinn, und worin liegt die Gefahr, wenn solche Rollen nicht besetzt werden? Die Antwort liegt eigentlich auf der Hand. Rollen wie Scrum Master, Architekten oder Tester generieren zwar nicht direkt Umsatz, haben aber eine ebenso wichtige Aufgabe: Zu den nicht funktionalen Anforderungen, für die sie zuständig sind, zählen auch Aspekte wie Erweiterbarkeit und Wartbarkeit. Die nicht funktionalen Anforderungen garantieren die Langlebigkeit von Software. Vernachlässigt man sie, lässt sich das Projekt im Laufe der Zeit immer schwieriger warten und weiterentwickeln, bis hin zu einer Situation, in der keine neue Funktion mehr in die Software implementiert und auch kein Bug mehr gefixt werden kann.

Verzichtet man im Projekt also auf die Rollen von Testern, Scrum Master oder Architekten, kümmert sich niemand um die nicht funktionalen Anforderungen. Dass diese vernachlässigt werden, merkt man oft zu Projektbeginn gar nicht, sondern erst, wenn nach Jahren oder Jahrzehnten Probleme auftreten, die das Projekt zum Stillstand bringen.

### Die Schulden

Wir haben in den letzten Episoden dieser Kolumne einiges über Schulden in der Softwareentwicklung erfahren. Kurz zusammengefasst: Schulden sind Versäumnisse in der Soft-

wareentwicklung, also Fehler, die sich im Laufe der Zeit negativ auf das Projekt auswirken. Je mehr von diesen Schulden ich mache, umso größer werden die Probleme in den Projekten. Sie ahnen bereits, dass die Rollenbesetzung damit in direkter Verbindung steht.

Denn besetzt man Rollen nicht (richtig), werden Schulden erzeugt beziehungsweise gibt es niemanden, der sich um die Erkennung und Beseitigung dieser Schulden kümmert. Der Scrum Master beispielsweise sorgt dafür, dass keine Prozessschulden existieren. Entstehen sie dennoch, ist es seine Verantwortlichkeit, Schulden schnellstmöglich abzubauen, um Prozesse zu vermeiden, die langsam und ineffizient sind. Oder nehmen wir die Softwarearchitektin: Sie ist dafür zuständig, dass keine Strukturschulden entstehen. Existieren sie doch, ist es ihre Verantwortlichkeit, diese schnellstmöglich abzubauen.

### Fazit

Das Besetzen von Rollen in der Softwareentwicklung kostet Geld. Was läge also näher, als einzelne Rollen, die nicht direkt zum Betriebsergebnis beitragen, wegzulassen? Wir haben gesehen, dass dies keine gute Idee ist, denn jede unbesetzte Rolle erzeugt oder begünstigt verschiedene Schulden, die sich zu enormen Mengen anhäufen und das Projekt früher oder später negativ beeinflussen. Ist dieser Zeitpunkt erreicht, ist der Schaden meist gewaltig und nur sehr kostenintensiv zu beheben.

Wird beispielsweise erst nach fünf Jahren erkannt, dass ein Softwarearchitekt fehlt, liegt der dann oft riesige Berg struktureller Schulden schon vor. Oder stellt man erst nach Jahren das Fehlen des Scrum Masters fest, gibt es nicht nur erhebliche Prozessschulden, sondern der bis dahin nicht effizient gemachte Prozess hat das Unternehmen vermutlich schon viel Geld gekostet.

Die Vernachlässigung von Rollen ist auf Prozessebene das mit Abstand häufigste Problem, dem ich begegne, und sorgt zudem oft für den größten Schaden in Projekten. Sie sollten daher nicht nur bei sich selbst, sondern auch in Ihrem Team und ganz besonders bei Ihren Entscheidungsträgern ein Bewusstsein für die Bedeutung und die Arten von Rollen sowie deren korrekte Besetzung schaffen, denn unabhängig von der Größe Ihres Teams, der Art Ihrer Anwendung oder Ihrer Branche muss dieses Problem zwingend gelöst werden – je früher, desto besser.

Und wenn Sie sich jetzt fragen, wie man denn die verschiedenen Rollen richtig besetzt: Diesem Thema wird der nächste Beitrag dieser Kolumne gewidmet sein. ■



**David Tielke**

ist freiberuflicher Berater und von Microsoft zertifizierter Trainer für die Anwendungsentwicklung auf der .NET-Plattform. Darüber hinaus hat er sich auf die Bereiche Softwarearchitektur, Softwarequalität und ALM spezialisiert.  
[mail@david-tielke.de](mailto:mail@david-tielke.de)

## DENKANSTÖSSE ZUR FINDUNG GEEIGNETER STRATEGIEN

# Problemlösen leicht gemacht

Vom Kamel zum Löwen zum Kind.

Sicher nicht nur in der Softwareentwicklung gehört es zum Tagesgeschäft, mehr oder weniger abstrakte Sachverhalte zu analysieren und einen Weg zu finden, die sich daraus ergebenden Probleme in eine praktikable und zufriedenstellende Lösung zu überführen. Oft weiß man nicht, wo man anfangen soll. Man sieht das Bit vor lauter Bytes nicht. Als erfahrener Entwickler hat man sicher ein Gefühl dafür, wo was an der Anwendung klemmt. Als „Rookie“ aber vielleicht nicht, sei es, weil man sich als neuer Mitarbeiter in die Weiterentwicklung einer umfangreichen und komplexen Anwendung einarbeiten muss, oder weil man schlicht und ergreifend Berufseinsteiger ist.

## Gibt es überhaupt ein Problem?

Das ist gewissermaßen die Mutter aller Fragen zum hier eröffneten Thema. Im Entwicklerteam wird eine User-Story diskutiert, deren Umsetzung überlegt, bis jemand die User-Story selbst infrage stellt. Ist die Anforderung überhaupt sinnvoll? Ergeben sich Widersprüche zu anderen User-Stories? Gibt die Anwendung vielleicht die Lösung an anderer Stelle schon her? Oder steht bereits eine Programmbibliothek zur Verfügung, die man einbinden kann? Ist das Rad schon erfunden, dass man gerade erfinden will? Niemand wird beispielsweise einen O/R-Mapper selbst schreiben, sondern auf EF Core, NHibernate oder Ähnliches setzen. Eine zielstrebige Recherche fördert womöglich auch Lösungen für speziellere Probleme als das O/R-Mapping zutage.

## Divide et impera

„Teile und herrsche“ – damit ist hier natürlich die Zerlegung eines großen Problems in mehrere kleine Teilprobleme gemeint, eines der Grundprinzipien der Softwareentwicklung. Als Verdeutlichung ein einfaches Beispiel aus der Mathematik:

$$x = (1+2) * (3+4)$$

Hier gibt es ein großes Problem, nämlich den gesamten Ausdruck, der sich in drei Teilprobleme zerlegen lässt: zwei Additionen und die Multiplikation der beiden Summen.

Nichts anderes passiert im Prinzip beim Parsen von JSON-Dateien: Hier wird so lange zerlegt, bis es nur noch Schlüssel-Wert-Paare gibt, die zu größeren Einheiten gehören, die aus Schlüssel-Objekt-Paaren bestehen.

## „Natürlich führe ich Selbstgespräche – manchmal brauche selbst ich Expertenrat“

Sprache ist wichtig. Hiermit ist jetzt nicht irgendeine Programmiersprache gemeint, sondern Ihre Muttersprache. Übersetzen Sie Fachbegriffe, die ja üblicherweise dem Englischen entstammen, in die Ihnen am besten geläufige. Der Begriff „string“ wird doch viel griffiger und anschaulicher in der deutschen Übersetzung „Zeichenkette“, oder nicht?

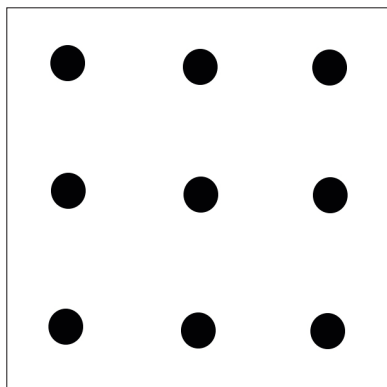
Formulieren Sie eine Frage, um ein Problem greifbar zu machen. Und zwar so, dass jemand, der nicht mit den Fachlichkeiten vertraut ist, es verstehen würde. Formulieren Sie präzise. Sprache beinhaltet ein immenses Instrumentarium, Sachverhalte äußerst genau zu beschreiben. Nicht umsonst gibt es in vielen Sprachen unzählige Wörter für Schnee [1][2].

Oft finden sich bereits in der Frage die gesuchten Antworten.

## Der freie Wille

Es sei der freie Wille, der den Menschen vom Tier unterscheidet, so war die Aussage eines Philosophielehrers, dessen Unterricht der Autor während der Schulzeit aufmerksam folgen durfte. Demzufolge würde ein Esel verhungern, dem rechts und links im Sichtfeld in exakt gleichem Abstand eine Möhre hingehalten wird, weil er sich nicht entscheiden könne, welche er zuerst fressen solle. Der Mensch hingegen habe die Fähigkeit, eine der beiden Möhren zu priorisieren und somit beide genüsslich nacheinander verspeisen zu können.

Ähnlich wird es Ihnen ergehen, wenn Sie auf der Suche nach einer Lösung die Internetsuchmaschine Ihres Misstrauens bemühen und Ihnen unter den ersten drei Ergebnissen fünf verschiedene Lösungsmöglichkeiten präsentiert werden. Sie können nun Stunden darauf verwenden, zu eruieren, welche Lösung denn am besten zu Ihrem Problem passen würde. Sie können aber auch die Ärmel hoch-



**Bilderrätsel:** Alle Punkte mit vier geraden Linien verbinden, ohne den Stift abzusetzen (Bild 1)

krempeln und loslegen und probieren. Dann aber nach dem Motto: Fail fast! Verwenden Sie eine durchaus überschaubare Zeitspanne für einen Versuch einer Implementierung, aber verschwenden Sie keine Zeit mit „Frickeleien“ und haben Sie keine Angst, Lösungsansätze wieder zu verwerfen. Eine suffiziente Versionskontrolle wird Ihnen helfen, für gescheitert erklärte Ansätze wieder aufgreifen zu können. Zur Auflockerung der Lektüre sei an dieser Stelle [3] empfohlen.

### Über das Muster hinausdenken

Nehmen Sie das kleine Bilderrätsel aus Bild 1. Die Aufgabe lautet: Verbinden Sie alle neun Punkte mit vier Geraden, die Sie zeichnen können, ohne den Stift abzusetzen. Die Lösung finden Sie – Spoileralarm – in Bild 3.

Dieses Rätsel zeigt ganz anschaulich, wie das Ignorieren von Grenzen (wie die des durch die neun Punkte beschriebenen Quadrats) eine Lösung ermöglicht.

### Jeder hat gedacht, das geht nicht ...

... bis jemand kam, der das nicht wusste – und der hat es dann gemacht. Dieser Punkt knüpft nahtlos an den vorigen Absatz an. Was so locker klingt, ist möglicherweise eine verständliche und prägnante Zusammenfassung der drei Verwandlungen des Geistes – vom Geist zum Kamel, vom Kamel zum Löwen und vom Löwen zum Kind –, die Friedrich Nietzsche in „Also sprach Zarathustra“ nur unwesentlich ausführlicher beschreibt:

Drei Verwandlungen nenne ich euch des Geistes:  
wie der Geist zum Kameele wird,  
und zum Löwen das  
Kameel, und zum Kinde zuletzt der Löwe.

...

Was ist schwer? so fragt der tragsame Geist,  
so kniet er nieder, dem Kameele gleich, und will gut beladen sein.

...

Neue Werthe schaffen – das vermag auch der  
Löwe noch nicht: aber Freiheit sich schaffen zu neuem  
Schaffen – das vermag die Macht des Löwen.

...

Unschuld ist das Kind und Vergessen, ein Neu-  
beginnen, ein Spiel, ein aus sich rollendes Rad, eine  
erste Bewegung, ein heiliges Ja-sagen.

Ja, zum Spiele des Schaffens, meine Brüder,  
bedarf es eines heiligen Ja-sagens...

Die Geschichte ist so natürlich noch nicht vollständig, aber falls Sie die erwähnte Literatur zufällig gerade nicht zur Hand haben, können sie das unter [4] nachlesen.

Was bedeutet das nun im übertragenden Sinne? Einfach gesagt: Machen Sie sich frei von alten Routinen und Gebräuchen, mit denen Sie sich mühsam beladen haben und die Sie ebenso mühsam erlernt haben. Bewerten Sie die Dinge neu. Machen Sie sich ein eigenes Bild von der Lage. Seien Sie unvoreingenommen. Wagen Sie einen Neuanfang.

### Intervallschachtelung

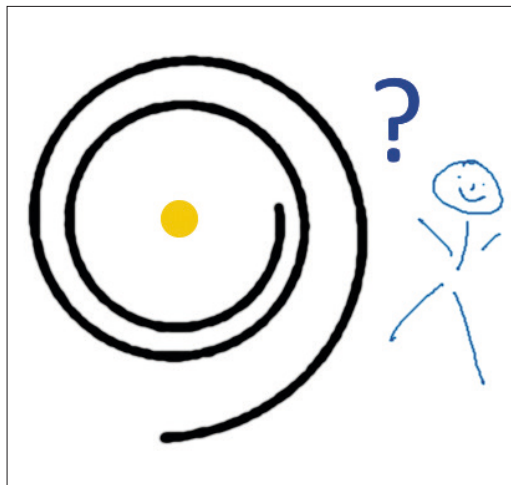
Jetzt wird es wieder mathematisch: Die Mathematiker haben ja im Gegensatz zu den Philosophen, die wahrscheinlich immer noch suchen, ein unbestrittenes Axiom, eine plausible Annahme der Wirklichkeit als Grundlage ihrer Wissenschaft, die keines Beweises bedarf.

Intervallschachtelung ist ein Verfahren, mit dem sich sehr schnell durch Ausprobieren eine richtige oder annähernd richtige Lösung finden lässt. Um die Quadratwurzel aus der Zahl 2 zu berechnen, braucht man keinen Taschenrechner, nur Stift und Papier. Die Lösung muss größer als 1 sein und kleiner als 2, das dürfte ohne Weiteres nachvollziehbar sein. Also bildet 1 die untere Grenze des Intervalls und 2 die obere. Die Lösung muss irgendwo dazwischen liegen. Man teilt das Intervall in der Mitte in zwei Hälften und rechnet  $1,5^2$  aus. Ist das Ergebnis größer als

2, ist die Lösung im linken Intervall zu finden; ist das Ergebnis kleiner, findet man die Lösung im rechten Intervall. Es reichen vier Rechenschritte, um aus einer Menge von unendlich vielen Möglichkeiten eine zu finden, die schon sehr nah am Ergebnis ist:

$1,5^2 = 2,25$  -> Ergebnis muss zwischen 1 und 1,5 liegen  
 $1,25^2 = 1,56$  -> Ergebnis muss zwischen 1,25 und 1,5 liegen  
 $1,375^2 = 1,89$  -> Ergebnis muss zwischen 1,375 und 1,5 liegen  
 $1,4375^2 = 2,066$  -> Nach nur vier Rechenschritten schon sehr nahe am Ergebnis

Warum sollte dieses Verfahren nicht auch bei der Fehlersuche erfolgversprechend sein? Grundsätzlicher Ablauf eines Programms ist Eingabe – ein bis n Schritte Verarbeitung – Ausgabe. Setzen Sie in der Entwicklungsumgebung einen Haltepunkt dort, wo die Eingabe entgegengenommen wird, und einen, bevor die Ausgabe zurückgegeben wird, und ►



Umwege können zum Ziel führen (Bild 2)



dann einen in der Mitte aller Verarbeitungsschritte. Wenden Sie nun das eben beschriebene Verfahren an. Schnell werden Sie den Punkt gefunden haben, an dem der Fehler passiert.

Suchindizes funktionieren ähnlich: Sobald klar ist, dass in der einen Seite des Suchbaums nichts gefunden werden kann, wird in der anderen Seite weitergesucht.

### Umwege

Umwege können ebenfalls zum Ziel führen. In **Bild 2** sehen Sie eine durchsichtige spiralförmige Wand, im inneren Ende der Spirale liegt leckeres Futter. Ein Huhn wird es nicht schaffen, ans Getreide zu kommen, ein Affe hingegen schon. Letzterer ist in der Lage, auch einen Umweg in Kauf zu nehmen, um an die heiß begehrte Banane zu gelangen. Will heißen: Der direkte Weg ist nicht immer zielführend. Allerdings und zugegebenermaßen liegt die Kunst darin, dies zu erkennen.

### Loslassen

Das hat jeder schon mal erlebt, wagt der Autor zu behaupten: Man hat stunden- oder sogar tagelang ebenso verbissen wie erfolglos an einem kniffligen Problem gearbeitet. Am nächsten oder übernächsten Tag, wenn es so gar nicht in den Tagesablauf passt, kommt die Eingebung, wie das Problem zu lösen ist. Also lassen Sie das Problem rechtzeitig los. Entweder es löst sich von alleine – Andeutungen finden sich in den vorangegangenen Absätzen –, oder die Lösung findet Sie, um es mal mit Chuck Norris' Worten auszudrücken. Ganz von alleine. Entspannen Sie sich!

### 9 to 5

Ein kreativer Prozess, der die Entwicklung einer Anwendung unzweifelhaft ist, lässt sich nicht in ein Korsett pressen. Es

wird Menschen geben, die auf Knopfdruck ihr Potenzial abrufen können, andere können das nicht, ohne dabei auch nur einen Deut schlechter zu sein. Diesen individuellen Befindlichkeiten können gerade in der Softwareentwicklung durch eine geeignete Arbeitszeitgestaltung und passende Heimbüro-Regelungen Rechnung getragen werden. Dies gilt es zu

nutzen. Arbeitnehmer sollen sich nicht selbst ausnutzen – diese Gefahr besteht beispielsweise bei Vertrauensarbeitszeitregelungen ja grundsätzlich, und Gehalt und Arbeitsleistung sollen auch für den Arbeitgeber in einem fairen Verhältnis stehen. Optimalerweise entsteht so für beide Seiten eine Win-win-Situation.

### Sherlock Holmes

„When you have excluded the impossible, whatever remains, however improbable, must be the truth.“ Diesen Satz legt Sir Arthur Conan Doyle seinem Protagonisten in „Das Diadem von Beryll“ in den Mund. Übertragen auf das Thema dieses Artikels: Wenn Sie alle unmöglichen Fehlerursachen ausgeschlossen haben, muss das, was übrig bleibt, so unwahrscheinlich es auch klingt,

der Grund für Ihr Problem sein. Schließen Sie alles aus, was nicht der Grund für das Fehlverhalten Ihrer Anwendung sein kann.

Sehr wahrscheinlich hat sich die Anzahl der Code-Abschnitte, die Sie untersuchen müssen, nun beträchtlich reduziert. Am Beispiel eines völlig trivialen Mini-Sudokus ist diese Vorgehensweise in **Bild 4** illustriert.

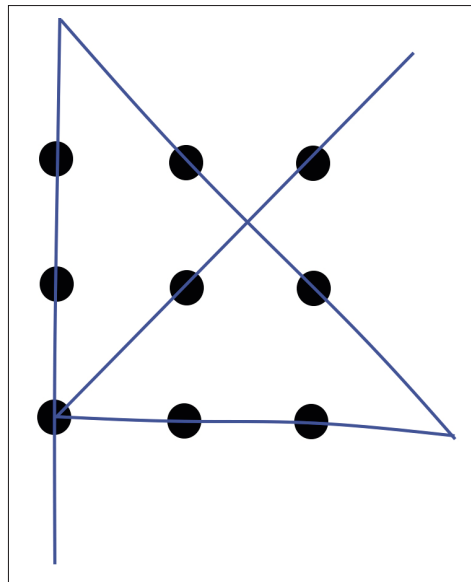
Apropos illustriert: Auch wenn Papier und Stift mittlerweile in Vergessenheit zu geraten scheinen, kann es immer sehr hilfreich sein, die Dinge mal ganz simpel und analog aufzuschreiben. Sei es eine Baumstruktur, die sich

seltsam verhält, oder ein Informationsfluss, der irgendwo stockt. Was man angucken oder anfassen kann, versteht man normalerweise besser, das begreift man im wahrsten Sinne des Wortes.

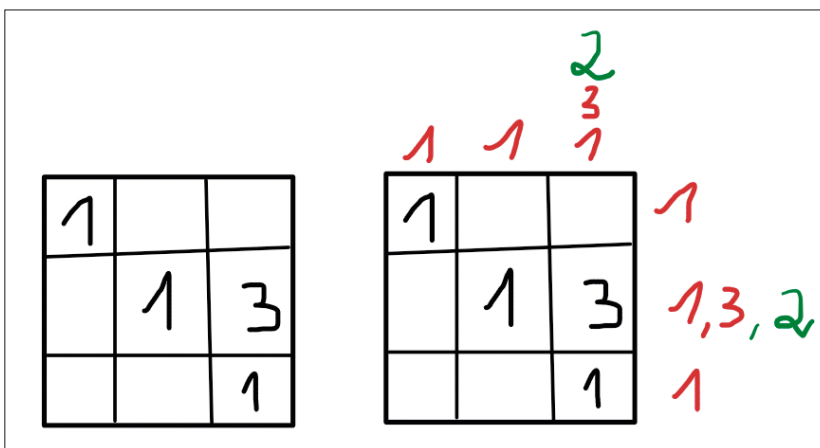
### Die letzte Stunde

Sie entwickeln ein neues tolles Feature. Es läuft wie geschmiert. Plötzlich verhält sich Ihre lokale Entwicklungsversion anders. „Irgendetwas“ funktioniert nicht mehr. Jetzt haben Sie drei Möglichkeiten:

- Sie verlieren den Fokus und sich selbst auf der Suche nach der Ursache dieses eigenartigen Verhaltens.
- Sie ignorieren das seltsame Verhalten und machen weiter.



Lösung des Bilderrätsels aus Bild 1 (**Bild 3**)



Links das „nackte“ Sudoku, rechts das gelöste. Die roten Ziffern sind die unmöglichen Lösungen, die grünen die sich daraus ergebenden möglichen Lösungen (**Bild 4**)

- Sie erkennen die Problematik, dass möglicherweise die von Ihnen vorgenommenen Änderungen zu dem Verhalten geführt haben.

Oft ist der Fehler in der Arbeit der vergangenen Stunde entstanden. Hierzu gab es auch mal einen Beitrag, der trotz aufwendiger KI-gestützter Internetrecherche nicht auffindbar war. Möglicherweise versteckt er sich in analoger Form in den unzähligen Stapeln alter Fachliteratur und Zeitschriften, die man üblicherweise auf dem Dachboden liegen hat und von denen man sich einfach nicht trennen kann ...

Um also nicht vollkommen aus dem Konzept gebracht zu werden, ist es ratsam, zu prüfen, ob der Fehler schon vorhanden war, als man begonnen hat, an dem tollen neuen Feature zu arbeiten. Wenn ja, ist der Fehler für Ihren Kontext aktuell irrelevant und wird, nachdem er beispielsweise im Ticket-System vermerkt wurde, ignoriert, sofern aus dem Bug nun kein Feature wird. Wenn nein, sind Ihre Änderungen der vergangenen Stunde die Ursache.

### Fazit

In diesem Artikel kommt das Wort „Lösung“ 21-mal vor, wohingegen nur 17-mal von „Problem“ die Rede ist. Man kann also belastbar behaupten, dieser Artikel sei ein eher lösungsorientierter. Handeln Sie im wirklichen Leben genauso, suchen Sie Lösungen, nicht Probleme.

„Hast du  $n$  Fehler gefixt, findet der erste Kunde den  $(n+1)$ . Fehler.“ Mit diesen Worten eines Arbeitskollegen soll dieser Beitrag enden, auf dass Sie Fehler  $n + 2$  vor dem nächsten Kunden finden! ■

[1] Eskimo-Wörter für Schnee, [https://de.wikipedia.org/wiki/Eskimo-W%C3%B6rter\\_f%C3%BCr\\_Schnee](https://de.wikipedia.org/wiki/Eskimo-W%C3%B6rter_f%C3%BCr_Schnee)

[2] Christian Zschke, 421 Mal Schnee, <https://www.sueddeutsche.de/panorama/schottisch-421-mal-schnee-1.2661222>

[3] Dilbert: I Can Fail at Any Speed You Like, <https://medium.com/vc-cafe/funny-dilbert-i-can-fail-at-any-speed-you-like-af1d042bf66a>

[4] Nietzsche, Friedrich: Also sprach Zarathustra. [Bd. 1]. Chemnitz, 1883, [https://www.deutschestextarchiv.de/book/view/nietzsche\\_zarathustra01\\_1883?p=35](https://www.deutschestextarchiv.de/book/view/nietzsche_zarathustra01_1883?p=35)



**Christian Sender**

ist bei der sycat IMS GmbH in Hannover als Softwareentwickler im Frontend und Backend unterwegs. Sie erreichen ihn unter [c.sender@sycat.com](mailto:c.sender@sycat.com).

# DWX ACADEMY

Hands-on Workshops  
und Weiterbildung für  
Software-Entwickler  
und -Architekten

## SOFTWAREQUALITÄT

## BEURTEILEN UND VERBESSERN

- Kodierrichtlinien
- Dokumentation
- Unit-Testing
- CI/CD
- Komponenten
- Composite Component Architektur 2.0

3 TAGE  
INHOUSE ODER  
REMOTE



**DAVID TIELKE**



**Deine Ansprechpartnerin: Susanne Herl**  
+49 731 88005-8835 • [susanne.herl@ebnermedia.de](mailto:susanne.herl@ebnermedia.de)

[developer-world.de/dwx-academy](https://developer-world.de/dwx-academy)

## CLEAN CODE UND ARCHITEKTUR, TEIL 2

# Abhängigkeiten und Architekturmuster

Hauptmerkmal von Architekturmustern sind die Abhängigkeiten. Und die erstaunliche Erkenntnis: Es gibt nur zwei Arten von Architekturmustern: DIP- oder IOSP-basierend.

**A**bhängigkeiten sind die größte Herausforderung in der Softwareentwicklung. Dafür gibt es zwei Gründe:

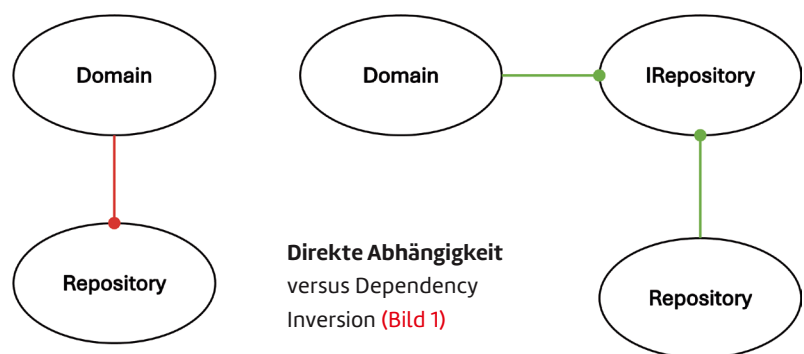
- Abhängigkeiten haben einen großen Einfluss darauf, wie leicht eine Softwarestruktur verstanden werden kann. Von der Verständlichkeit der Struktur hängt die Wandelbarkeit des Softwaresystems ab. Je leichter die Struktur durchschaut werden kann, desto leichter ist es, Änderungen und neue Features zu realisieren.
- Abhängigkeiten haben einen großen Einfluss darauf, wie leicht ein Softwaresystem automatisiert getestet werden kann. Erschweren die Abhängigkeiten die Testbarkeit, hat dies einen sehr negativen Einfluss auf die Korrektheit des Softwaresystems. Automatisierte Tests sind Stand der Technik und müssen durch die Abhängigkeitsstruktur optimal unterstützt werden.

Man kann schnell erkennen, dass Abhängigkeiten die beiden Werte Wandelbarkeit und Korrektheit stark beeinflussen. Aus diesem Grund haben sich viele Experten mit dem Thema auseinandergesetzt. Das Ergebnis sind Prinzipien wie das Dependency Inversion Principle (DIP) oder das Integration Operation Segregation Principle (IOSP), die sich unmittelbar mit der Struktur der Abhängigkeiten auseinandersetzen. Aber auch Prinzipien wie das Interface Segregation Principle (ISP) berühren das Thema Abhängigkeiten, da hier versucht wird, die Abhängigkeiten zu reduzieren, indem die Interfaces punktgenau nur so viele Details enthalten, wie unbedingt erforderlich ist.

## Entwurf

Weil Abhängigkeiten einen so großen Einfluss darauf haben, wie es um Wandelbarkeit und Korrektheit eines Softwaresystems bestellt ist, geht kein Weg daran vorbei, sie zu planen. Wer einfach draufloscodiert, landet nur zufällig bei einer Abhängigkeitsstruktur, durch die die beiden Werte gut unterstützt werden. Da beide Werte von zentraler Bedeutung sind, ist es logisch, dass auch der Entwurf der Struktur von zentraler Bedeutung ist.

Der Entwurf von Software scheint allerdings etwas aus der Mode gekommen zu sein. Ob dies daran liegt, dass die Agi-



lität mit der Aussage „No big design upfront“ davor warnt, zu viel zu entwerfen, sei dahingestellt. Die Betonung liegt hier auf dem Wort „big“. Möglich, dass die Aussage fälschlicherweise so interpretiert wurde, dass Software ohne Design erstellt werden soll. Möglich auch, dass die Idee des Test Driven Development (TDD) hier mehr geschadet als genutzt hat. Und natürlich sollte das Thema Entwurf pragmatisch angegangen werden. Dennoch: Ganz ohne geht es nicht.

Der Entwurf eines Softwaresystems gliedert sich in zwei Bereiche:

- Architektur ist verantwortlich für die grobe Struktur, bestehend aus Bibliotheken, Paketen, Komponenten und Microservices. Hier geht es eher um nichtfunktionale Anforderungen wie zum Beispiel Verteilung, Skalierung oder die Entscheidung zwischen strukturiertem Monolith versus Microservices. Ferner ist die Architektur verantwortlich für die Arbeitsorganisation.
- Das Detailed Design ist dagegen verantwortlich für die Struktur der Methoden und Klassen. Der Fokus liegt hauptsächlich auf den funktionalen Anforderungen. Wenn TDD überhaupt funktioniert, dann in diesem Bereich.

Den beiden Bereichen des Entwurfs, Architektur und Detailed Design, lassen sich auch die Verantwortlichkeiten für Wandelbarkeit und Korrektheit zuordnen. Das Detailed Design wirkt sich primär auf die Korrektheit aus. Lassen sich Methoden und Klassen leicht automatisiert testen, ist dies gut für die Korrektheit. Die Struktur aus Bibliotheken, Paketen et cetera haben dagegen einen eher geringen Einfluss auf die

Testbarkeit. Wenn Methoden und Klassen gut strukturiert sind, spielt es für die Testbarkeit mehr oder weniger keine Rolle, in welchen Bibliotheken die Klassen zusammengefasst sind.

Architektur, also die Struktur aus Bibliotheken, Paketen et cetera, wirkt sich dagegen mehr auf die Wandelbarkeit, die Wiederverwendung und die Arbeitsorganisation aus. Ist eine bestehende Funktionalität gut in einem Paket gekapselt, lässt sie sich leichter verstehen und damit wandeln. Auch die Wiederverwendung wird damit unterstützt.

Nicht zuletzt übernimmt die Architektur auch die Verantwortung für die Arbeitsorganisation. Entwickelt ein einzelner Entwickler an einem Softwaresystem, wird eine andere Grobstruktur entstehen, als wenn mehrere Teams parallel entwickeln.

Die Verantwortlichkeiten überlappen sich allerdings auch. Eine scharfe Zuordnung von Detailed Design zu Korrektheit und Architektur zu Wandelbarkeit ist nicht gegeben. Dennoch liegen die Zuordnungen eher in diesen beiden Bereichen. Und natürlich muss auch das Detailed Design die Wandelbarkeit unterstützen und die Architektur die Korrektheit.

## Prinzipien

Wenn über Prinzipien gesprochen wird, liegt der Fokus häufig darauf, wie der Code auszusehen hat. So wird das Prinzip Don't Repeat Yourself (DRY) in der Regel auf Codezeilen bezogen. Es soll im Code keine Dopplung geben. Selbstverständlich gilt dies auch auf der Ebene von Bibliotheken, Paketen et cetera. Auch hier soll es keine Dopplung geben. Pakete sind sogar eine gute Lösung, um Dopplungen zu reduzieren, da man in ihnen den an mehreren Stellen benötigten Code zusammenfassen kann. Die Frage lautet nun: In welchem Zusammenhang stehen die Clean-Code-Developer-Prinzipien mit der Architektur?

Einige Prinzipien können sinnvoll auf Architektur angewandt werden. Die beiden wichtigsten Prinzipien zur Abhängigkeitsstruktur von Softwaresystemen sind die beiden oben schon genannten:

- Dependency Inversion Principle (DIP)
- Integration Operation Segregation Principle (IOSP)

Sie haben einen so wesentlichen Einfluss, dass alle bekannten Architekturmuster auf ihnen basieren. Bei anderen Prinzipien ist kein Zusammenhang zur Architektur zu sehen, obwohl es beispielsweise von Bob C. Martin den Versuch gibt, die SOLID-Prinzipien auf Architektur anzuwenden. Das Extrembeispiel dürfte das Liskov Substitution Principle (LSP) sein. Es macht Aussagen über die Vererbung von Klassen. Architektur behandelt aber Bibliotheken und Pakete, bei denen es keine Vererbung gibt. Ähnliches gilt für das Open Closed Principle (OCP). Auch dieses Prinzip bezieht sich auf Klassen, und ein Transfer auf Architektur muss fehlschlagen. Bleiben wir also beim Thema Abhängigkeiten.

## Dependency Inversion Principle (DIP)

Die Kernidee des DIP ist es, zwei Klassen nicht unmittelbar voneinander abhängig zu machen. Stattdessen wird eine

Abstraktion in Form eines Interface dazwischengestellt. **Bild 1** veranschaulicht den Unterschied. Darin ist auf der linken Seite eine direkte Abhängigkeit zwischen Domain und Repository zu sehen. In diesem Beispiel enthält die Domain einen Teil der Domänenlogik des Softwaresystems, während das Repository dafür verantwortlich ist, auf die Datenbank zuzugreifen. Problematisch an dieser Struktur ist, dass die Domänenlogik unmittelbar von der Datenbanklogik abhängig ist. Das wird im Code dazu führen, dass innerhalb der Domänenlogik auf die Datenbank zugegriffen wird. Zum einen erschwert dies das Verständnis der Software. Zum anderen kann nun die Domänenlogik nur automatisiert getestet werden, indem für die Testfälle eine Datenbank bereitgestellt wird. Für Integrationstests ist dies zwar ohnehin erforderlich, jedoch soll die Domänenlogik nicht ausschließlich mit Integrationstests getestet werden.

Auf der rechten Seite der Abbildung ist zwischen Domain und Repository das Interface *IRepository* eingezogen. Die Domain ist vom Interface abhängig, weil sie es verwendet. Das Repository ist vom Interface abhängig, weil es dieses implementiert. Der Vorteil liegt nun darin, dass jetzt im Test mit Attrappen gearbeitet werden kann. Injiziert man im Test eine Attrappe des Repositories in die Domain, können die Tests ohne reale Datenbank durchgeführt werden (**Listing 1**).

Allerdings werden die Tests dadurch umfangreicher, weil das Verhalten der Attrappen für die Tests passend konfiguriert werden muss. So muss die Attrappe beispielsweise angewiesen werden, auf eine bestimmte Abfrage mit einem definierten Ergebnis zu antworten. Im Beispiel wird mit der Zeile *repository.Setup(...)* erklärt, dass die Attrappe ein ►

### Listing 1: Testen mit Attrappe

```
using Moq;

namespace architektur;

[TestFixture]
public class DomainTests
{
    [Test]
    public void Returns_error_if_name_not_set() {
        var repository = new Mock<IRepository>();
        repository.Setup(r => r.GetCustomer("123"))
            .Returns(new Customer { Id = "123",
                Name = "" });
        var sut = new Domain(repository.Object);

        var result = sut.ValidateCustomer("123");

        Assert.That(result, Is.EqualTo(
            "Name nicht gesetzt"));
    }
}
```



*Customer*-Objekt liefern soll, wenn nach der ID 123 gesucht wird. Ohne dieses Setup würde der Test fehlschlagen, weil die Attrappe dann null liefern würde.

Frameworks wie Moq oder NSubstitute erleichtern den Umgang mit Attrappen zwar, es bleibt jedoch dabei, dass die Tests relativ gesehen komplexer werden. Des Weiteren wird in Tests auf Interna der Implementation Bezug genommen. Wird später der Umgang mit dem Repository in der Domänenlogik verändert, kann es sein, dass Tests dann fehlschlagen, weil die Attrappen sich nicht mehr wie benötigt verhalten. Wäre dies alles alternativlos, würden wir damit leben und weiter auf das Dependency Inversion Principle setzen. Doch es gibt eine Alternative, welche diese Nachteile beseitigt.

### Integration Operation Segregation Principle (IOSP)

Das Integration Operation Segregation Principle (IOSP) wurde von Ralf Westphal und dem Autor entwickelt, um das Problem zu lösen, dass Strukturen beim Einsatz des DIP schwer verständlich und schwer testbar bleiben. Das Verständnis wird erschwert, weil trotz Dependency Inversion im Code ein unterschiedliches Abstraktionsniveau herrscht. Eine Methode, die eine andere Methode aufruft, integriert diese. Wenn sie sich gleichzeitig auch noch selbst um Details kümmert, kommt es zu einer Vermischung von abstraktem Code (Aufruf einer anderen Methode, die zum Softwaresystem gehört) und detailreichem Code (Aufruf von Methoden aus Runtime und Frameworks sowie Verwendung von Ausdrücken). Das Prinzip des Single Level of Abstraction (SLA) fordert jedoch, dass der Code einer Methode nur auf einem einzigen Abstraktionsniveau liegen sollte: entweder detailreich oder abstrakt.

Werden wie in Listing 2 gezeigt beide Niveaus vermischt, wird dadurch das Verständnis erschwert. Dies mag für kurze Methoden nicht dramatisch sein, doch sofern es eine Alternative gibt, sollten wir diese wählen. Im Beispiel oben ist der Aufruf der *Repository*-Methode auf einem höheren Abstraktionsniveau als der restliche Code der Methode.

Aus dem SLA folgt unmittelbar, dass mit dem DIP etwas nicht in Ordnung sein kann. Denn trotz DIP befindet sich der

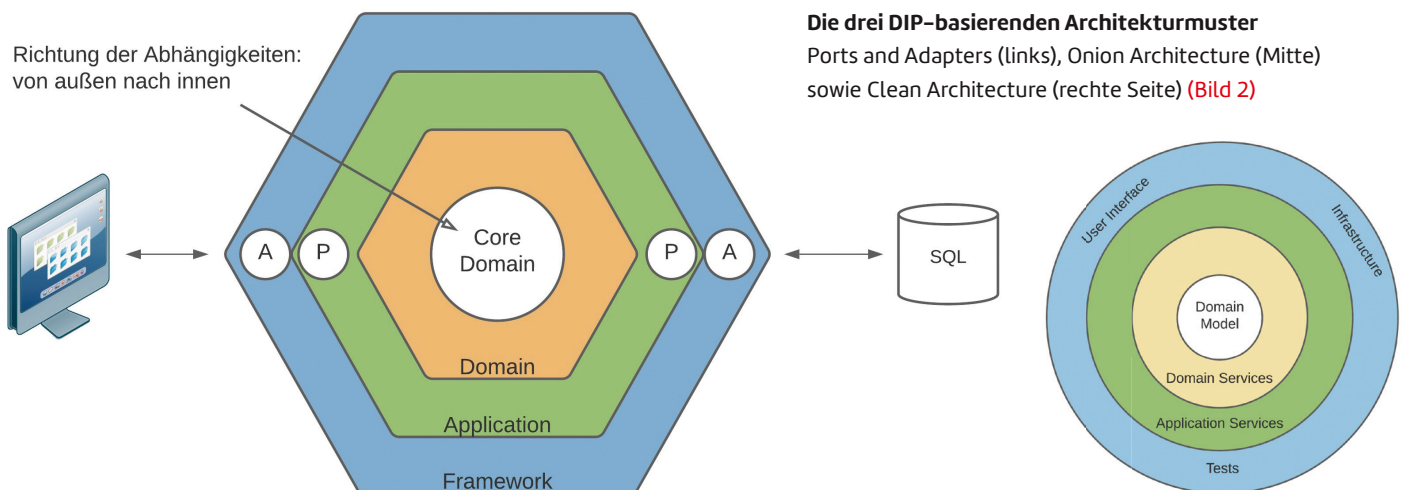
### Listing 2: SLA ist verletzt

```
public class Domain(IRepository repository)
{
    public string ValidateCustomer(string customerId)
    {
        var customer = repository.GetCustomer(
            customerId);
        if(string.IsNullOrEmpty(customer.Name)) {
            return "Name nicht gesetzt";
        }
        return "";
    }
}
```

Code in einer Methode immer auf unterschiedlichen Abstraktionsniveaus, sobald diese Aufrufe enthält und detailreichen Code. Und erstaunlicherweise ist dies zwingend immer der Fall, wenn das DIP zum Einsatz kommt, ohne dass es durch das IOSP ergänzt wird. Detailreicher Code, der die Dienstleistung einer anderen Methode in Anspruch nehmen möchte, muss diese aufrufen, ob nun direkt oder mittels DIP über ein Interface. Im Ergebnis befindet sich detailreicher Code mit abstraktem Code in ein und derselben Methode, was das SLA verletzt.

Auch ein weiteres zentrales Prinzip ist beim DIP verletzt: das Single Responsibility Principle (SRP). Eine Methode, deren Verantwortlichkeit darin liegt, ein Teilproblem des Softwaresystems durch detailreichen Code zu lösen, darf nicht gleichzeitig dafür verantwortlich sein, Teilaufgaben an andere eigene Methoden zu delegieren. Integration und Operation sind zwei Verantwortlichkeiten, die nicht vermischt werden sollten.

Die Lösung für dieses Dilemma liegt darin, die beiden Aspekte Integration und Operation zu trennen. Oder anders ausgedrückt: Detailreicher Code ist zu trennen von Aufrufen eigener Methoden. Für die Einhaltung des Integration Ope-



ration Segregation Principle (IOSP) sind zwei Unterscheidungen zu treffen:

- **Integration:** Erlaubt sind nur Aufrufe von eigenen Methoden, also solchen, die Bestandteil des Softwaresystems sind.
- **Operation:** Es sind nur Aufrufe fremder Methoden erlaubt, also solchen aus Runtime und Frameworks. Und es sind Ausdrücke erlaubt wie etwa  $x > 42$  oder  $y + 7$ .

Einige Beispiele:

```
if(x > 42) {
    y = Math.Abs(x - 56);
}
```

Dies ist Code einer Operation. Er enthält nur Ausdrücke und Aufrufe fremder Methoden (*Math.Abs*).

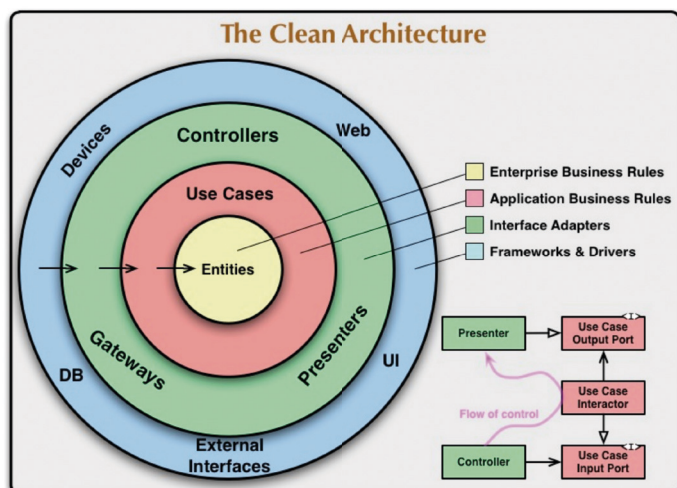
```
if(f(x)) {
    g(x);
}
```

Dies ist eine Integration, da der Code nur die beiden eigenen Methoden *f* und *g* aufruft.

```
if(x > 42) {
    g(x);
}
```

Hier sind Integration (Aufruf von *g*) und Operation (Ausdruck  $x > 42$ ) vermischt. Ferner liegt eine Verletzung des Prinzips Single Level of Abstraction (SLA) vor, da der Code Details enthält und gleichzeitig einen Aufruf auf höherem Abstraktionsniveau. Ferner ist das Single Responsibility Principle (SRP) verletzt. Der Code kümmert sich um seine eigene Verantwortlichkeit (Entscheidung, wann ein Aufruf zu erfolgen hat) als auch um die Integration eigenen Codes.

An den sehr simplen Codeschnipseln mag dies etwas abstrakt erscheinen. Im Kern ergibt sich bei Einhaltung des IOSP jedoch eine Codestruktur, die leichter zu verstehen ist und besser getestet werden kann.



### Listing 3: IOSP-basierende Lösung

```
public string ValidateCustomer(string customerId) {
    var customer = repository.GetCustomer(customerId);
    var message = CreateValidationMessage(customer);
    return message;
}

public static string CreateValidationMessage(
    Customer customer) {
    if(string.IsNullOrEmpty(customer.Name)) {
        return "Name nicht gesetzt";
    }
    return "";
}
```

Listing 3 zeigt, wie die Methode ganz leicht so refaktoriert werden kann, dass sie das IOSP einhält. Es entsteht eine Integrationsmethode *ValidateCustomer*, die ausschließlich dafür verantwortlich ist, die beiden Teilschritte aufzurufen. Zusätzlich befindet sich die eigentliche Validierung dann in einer Operation, die nun ohne Attrappen getestet werden kann.

### Architektur

Die beiden Prinzipien DIP und IOSP werden, wie oben dargestellt, üblicherweise im Zusammenhang mit Methoden und Klassen diskutiert. Doch sie haben auch einen zentralen Zusammenhang mit dem Thema Architektur: Aus ihnen sind alle bekannten Architekturmuster entstanden:

- **DIP-basierende Architekturmuster:** Ports and Adapters (Hexagonal Architecture), Onion Architecture sowie Clean Architecture
- **IOSP-basierendes Architekturmuster:** Integration Operation Data API Architecture (IODA Architecture)

### DIP-basierende Architekturmuster

Die Kernidee der DIP-basierenden Architekturmuster liegt darin, die Richtung der Abhängigkeiten umzukehren. Die drei Muster werden als konzentrische Sechsecke beziehungsweise Kreise dargestellt. Dabei verläuft die Richtung der Abhängigkeiten von außen nach innen (Bild 2).

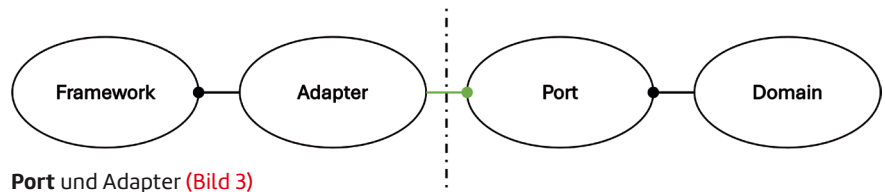
Die Zuordnung der einzelnen Aspekte zu „außen“ beziehungsweise „innen“ hängt von der Spezialisierung ab. Außen liegen allgemeine Aspekte. Je weiter man nach innen schaut, desto spezieller werden die Aspekte. Ganz innen im Kern liegt die Domänenlogik. Die ist so speziell, dass sie im Kern das Softwaresystem ausmacht. Hier muss alles selbst implementiert werden, da es keine fertigen Frameworks gibt, welche die Domänenlogik so umsetzen, wie es das Softwaresystem verlangt.

Ganz außen sind die Aspekte so allgemein, dass sie auf keinen Fall selbst implementiert werden. Hierfür kommen existierende Frameworks zum Einsatz, sei es Open Source oder zugekauft. ►

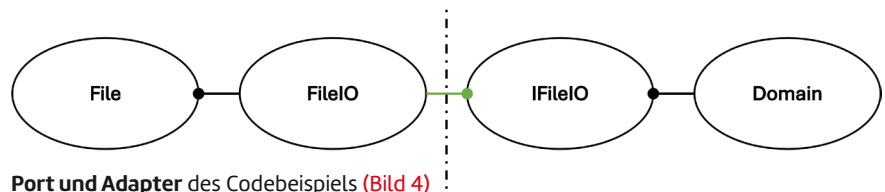
Nun besagt das Dependency Inversion Principle (DIP), dass die Abhängigkeiten vom Speziellen zum Allgemeinen verlaufen sollen. Details sollen abhängig sein von Abstraktionen. Im Fall der hier beschriebenen konzentrisch angeordneten Aspekte müssen also die inneren Aspekte abhängig sein von Abstraktionen, die wiederum von den äußeren Aspekten bereitgestellt werden. So verläuft dann die Richtung der Abhängigkeiten von außen nach innen.

Die Hexagonal Architecture enthält einen konkreten Vorschlag, wie diese Umkehrung der Abhängigkeitsrichtung technisch umgesetzt werden kann. Dazu werden sogenannte Ports und Adapters verwendet. Bild 3 zeigt deren Struktur.

Benötigt eine innenliegende Klasse der Software eine Dienstleistung einer weiter außen liegenden Klasse, definiert sie einen Port. Technisch gesehen ist dies ein Interface, welches die Methoden enthält, die aufgerufen werden sollen. Eine weiter außen liegende Klasse könnte nun dieses Interface implementieren. Dies ist jedoch nur in den Fällen möglich, in denen die Klasse zum eigenen System gehört. Soll eine Klasse eines Frameworks verwendet werden, kann dieses nicht veranlasst werden, einen Port (Interface) zu implementieren. Hier kommen die Adapter ins Spiel. Ein Adapter implementiert einen Port. Damit kann er in eine weiter innen liegende Klasse injiziert werden. Gleichzeitig macht sich der Adapter abhängig von der oder den benötigten Klassen, um mit ihrer Hilfe die benötigte Funktionalität bereitzustellen.



Port und Adapter (Bild 3)



Port und Adapter des Codebeispiels (Bild 4)

In Bild 3 ist zu sehen, dass die Domain einen Port definiert und sich von diesem abhängig macht. Diese Abhängigkeit bleibt innerhalb der Domain, im Kern des Architekturmusters. Eine Ebene weiter außen wird dann ein Adapter erstellt, der den Port implementiert. Aus diesem Grund verläuft die Abhängigkeit nun von außen nach innen (grün dargestellt). Damit der Adapter seine Arbeit verrichten kann, macht er sich abhängig von einem Framework. Auch diese Abhängigkeit ist unterhalb des Radars, da sie im selben Ring des Architekturmusters liegt. Listing 4 zeigt dies an einem Beispiel.

In diesem Codebeispiel muss die Domänenlogik auf Dateien zugreifen. Die Hexagonal Architecture schlägt für diesen Fall vor, dass die Domäne ein Interface (Port) *IFileIO* mit den Methoden bereitstellt, die sie vom weiter außen liegenden Adapter benötigt. Da wir das .NET Framework nicht zwingen können, unsere eigenen Interfaces zu implementieren, implementieren wir einen Adapter *FileIO*. Dieser implementiert das Interface und kann damit in die Domänenlogik injiziert werden. Gleichzeitig kann der Adapter das .NET Framework verwenden, um Dateien einzulesen, hier die Methode *File.ReadAllText*. Somit verläuft die Abhängigkeitsrichtung nun von außen nach innen (Bild 4).

#### ● Listing 4: Port und Adapter für Dateizugriff

```
public class Domain(IFileIO fileIO)
{
    public void DoSomething() {
        var content = fileIO.ReadAllText("file.txt");
        // work on 'content'
    }
}

public interface IFileIO
{
    string ReadAllText(string filename);
}

public class FileIO : IFileIO
{
    public string ReadAllText(string filename) {
        return File.ReadAllText(filename);
    }
}
```

### Onion und Clean Architecture

Onion Architecture und Clean Architecture basieren auf derselben Idee: Die Richtung der Abhängigkeiten soll von außen nach innen verlaufen. Es wird lediglich darauf verzichtet, zu konkretisieren, wie die Abhängigkeitsumkehr technisch vonstattengehen kann. Ferner werden an einigen Stellen andere Begriffe für die Verantwortlichkeiten verwendet. Von diesen Details abgesehen handelt es sich im Kern um die exakt gleiche Idee.

### IOSP-basierendes Architekturmuster

Wie wir weiter oben bereits gesehen haben, hinterlässt das Dependency Inversion Principle (DIP) diverse Probleme. Von einem pragmatischen Standpunkt aus betrachtet ist das größte Problem, dass wir bei Tests auf Attrappen angewiesen sind. Obwohl jeder Entwickler dies für normal und unvermeidbar hält, sollten wir prüfen, ob es eine Alternative gibt, bei der wir in Tests keine Attrappen benötigen.

Die Lösung liegt darin, Operationen von ihrer Integration zu trennen. Was wir automatisiert testen wollen, sind vor allem die Operationen. Das sind die Methoden, in denen die Logik der Anwendung steckt. Hier sind Framework-Aufrufe und Ausdrücke enthalten, deren korrekte Verwendung wir sicherstellen müssen.

Integrationstests gehen immer, völlig egal wie die innere Struktur einer integrierenden Methode aussieht. Die zentrale Frage ist daher, wie leicht sich der „Rest“ der Anwendung isoliert testen lässt.

Die Antwort der DIP-basierenden Architekturen lautet: Kehre die Abhängigkeitsrichtung um, indem du ein Interface einziehst. Dann können die Abhängigkeiten im Test durch Attrappen ersetzt werden.

Die Antwort des IOSP lautet dagegen: Trenne Integration und Operation, und du kannst im Test komplett auf Attrappen verzichten. Ich meine: Das klingt spannend genug, um es auszuprobieren.

Nehmen wir die Aufgabenstellung von oben, bei der unter anderem aus einer Datei gelesen werden musste. Die Lösung nach DIP bestand darin, den Dateizugriff durch ein Interface zu abstrahieren. Durch Herauslösen der Integration der beteiligten Aspekte kommen wir zu einer anderen Struktur, die in [Listing 5](#) gezeigt ist.

Hier sind die beiden Aspekte „Lesen aus der Datei“ und „Arbeiten mit dem Inhalt der Datei“ durch jeweils eigene

#### ● Listing 5: Beispiel gemäß IOSP

```
public class WordCount(IFileIO fileIO)
{
    public Dictionary<string, int>
        CountWords(string filename){
        var content = fileIO.ReadAllText(filename);
        var dictionary = CountTheWords(content);
        return dictionary;
    }

    public static Dictionary<string, int>
        CountTheWords(string content) {
        var words = content.Split([' ', '\n', '\r', '\t'],
            StringSplitOptions.RemoveEmptyEntries);
        var dictionary = new Dictionary<string, int>();

        foreach (var word in words) {
            var lowerWord = word.ToLowerInvariant();
            if(!dictionary.TryAdd(lowerWord, 1)) {
                dictionary[lowerWord]++;
            }
        }

        return dictionary;
    }
}
```

#### ● Listing 6: Test der Logik, ohne Attrappe

```
[TestFixture]
public class WordCountTests
{
    [Test]
    public void CountTheWords_InputIsEmptyString_
        ReturnsZero() {
        var result = WordCount.CountTheWords(
            string.Empty);
        Assert.That(result.Count, Is.EqualTo(0));
    }

    [Test]
    public void CountTheWords_InputHasOneWord_
        ReturnsOne() {
        var result = WordCount.CountTheWords("Hello");
        Assert.That(result["hello"], Is.EqualTo(1));
    }

    [Test]
    public void CountTheWords_InputHasMultipleWords_
        ReturnsWordCount() {
        var result = WordCount.CountTheWords(
            "Hello world! Hello this is a test.");
        Assert.That(result.Count, Is.EqualTo(6));
        Assert.That(result["hello"], Is.EqualTo(2));
    }
}
```

Operationen implementiert. Das Zusammenwirken der beiden Operationen wird durch die Integrationsmethode erledigt. So haben wir die Möglichkeit, die beiden Operationen isoliert zu testen, ohne dass dabei Attrappen zum Einsatz kommen ([Listing 6](#)). Ob man zusätzlich ein Interface einzieht, hängt davon ab, ob dies im Ausnahmefall für Integrations-tests hilfreich ist. Der Regelfall bei Anwendung des IOSP lautet jedoch: Verzichte auf das Interface so lange, bis du es tatsächlich benötigst. Da ein Interface mithilfe der Refactoring-Tools moderner IDEs schnell erzeugt ist, stellt dies keinen Nachteil dar.

### IODA-Architektur

Beim DIP haben wir die Idee, die zunächst auf Klassen angewendet wurde, auf Architekturmuster übertragen und sind dadurch zu den drei Modellen Hexagonal, Onion und Clean Architecture gekommen. Vollziehen wir das Gleiche mit dem IOSP. Wie sieht eine Architektur aus, wenn wir Integration und Operation voneinander trennen?

Zunächst entstehen, wie in [Bild 5](#) gezeigt, durch die Trennung in Integration und Operation die beiden oberen Ebenen der IODA-Architektur. Auf der obersten Ebene sind die Klassen, Bibliotheken, Pakete et cetera anzutreffen, deren Verantwortung in der Integration liegt. In einer .NET Solution ►



entspricht dies dem Projekt, welches die *Program.cs* beziehungsweise die *Main*-Methode enthält. Hier liegt der Einstiegspunkt in die Anwendung und damit der oberste Punkt der Integration.

Die *Main*-Methode trägt selbst in keiner Weise zur Lösung des Problems bei, für welches das Softwaresystem erstellt wird. Sie sorgt lediglich dafür, dass alles bereitgestellt wird, um die Anwendung zu starten. Hier werden beispielsweise dem DI-Container die benötigten Klassen bekannt gemacht.

Neben dem Einstiegspunkt braucht es dann weitere Integrationen, um die einzelnen Interaktionen der Anwendung bereitzustellen. Dabei besteht jede Interaktion eines Anwenders aus mehreren Schritten, die nacheinander auszuführen sind. Manchmal kommen dabei Fallunterscheidungen zum Einsatz, manchmal auch Schleifen. All das ist Aufgabe der Interactors-Klassen. Sie enthalten Integrationsmethoden, mit denen die Operationen aufgerufen werden, die in Summe die Anforderungen der einzelnen Interaktionen umsetzen. Die Interactor-Methoden erledigen die Arbeit nicht selbst, sondern delegieren sie an Operationen (oder in komplexen Szenarien auch an weitere integrierende Methoden).

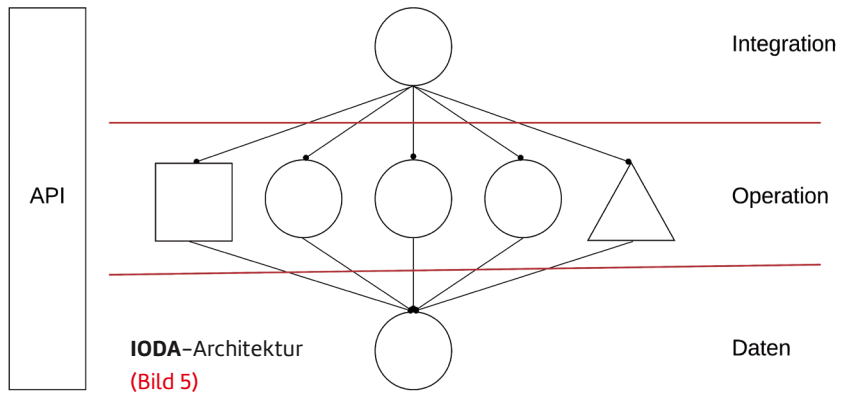
Bevor wir detaillierter auf die Interactors schauen, betrachten wir die zweite und dritte Ebene in der IODA-Architektur etwas genauer. Unterhalb der Integration liegt die Ebene der Operationen. Fundamental ist hier, dass zwischen den Operationen keinerlei Abhängigkeiten bestehen. Aus diesem Grund können hier auch die drei Hauptaspekte Benutzerschnittstelle (UI), Domänenlogik und Ressourcenzugriff vermischt werden.

Das Zusammenwirken der einzelnen Operationen erfolgt nicht dadurch, dass diese sich gegenseitig aufrufen. Stattdessen werden sie von den Integrationen aufgerufen. Das führt dazu, dass die Operationen frei von Abhängigkeiten bleiben und somit ohne Attrappen getestet werden können.

Beim Zusammenspiel der Operationen ist es wichtig, dass sie gemeinsame Datentypen verwenden können. So kann eine Benutzerschnittstelle ein Objekt vom Typ *Kunde* zurückliefern, welches dann von der Domänenlogik validiert und vom Ressourcenzugriff in eine Datenbank gespeichert wird. Die Datenstruktur *Kunde* ist hier eine reine Datenstruktur, die keinerlei Logik enthält. Auch dient sie keinem der Aspekte im Speziellen. Sie ist allgemein dafür zuständig, Daten zwischen den einzelnen Operationen bewegen zu können.

Aus diesem Grund liegen auf der untersten Ebene der IODA-Architektur die Datenstrukturen als Bestandteil eines gemeinsamen Kontrakts der Operationen.

Der vierte und letzte Aspekt der IODA-Architektur sind APIs, die in den Operationen und den Datenstrukturen verwendet werden können. Mit API meinen wir hier Methoden aus Runtime und Frameworks. Zu diesen existieren natürlich auch Abhängigkeiten. Diese unterscheiden sich maßgeblich von Abhängigkeiten zu eigenen Methoden: Wir sind nicht für



deren Korrektheit verantwortlich, sondern setzen diese voraus. Insofern werden Runtime- und Framework-Abhängigkeiten auch bei den DIP-basierenden Modellen nicht durch Interfaces und Attrappen behelligt. Jedes Softwaresystem ist von einer Basis abhängig, die als korrekt vorausgesetzt wird.

## Interactor

Im Kontext der IODA-Architektur wird in der Regel ein weiteres Muster angewandt: die Interactors. Damit werden Klassen bezeichnet, die Interactor-Methoden enthalten. Die Faustregel: Pro Benutzerinteraktion gibt es eine integrierende Interactor-Methode. Kann man in einem Dialog auf verschiedene Buttons klicken oder Menüpunkte auswählen, stellen diese jeweils eine Benutzerinteraktion dar. Jeder wird dann eine Interactor-Methode zugeordnet. Diese enthält die benötigten Aufrufe von Operationen, die notwendig sind, um die Anforderungen der Interaktion umzusetzen.

Das Elegante an dieser Vorgehensweise ist, dass sich aus einem Entwurf der Benutzerschnittstelle sehr leicht die Interaktionen ableiten lassen. Ferner lässt sich leicht bestimmen, welche Eingaben die Interactor-Methode erhält beziehungsweise welche Ergebnisse sie an den Dialog zurückliefert.

## Fazit

Es gibt im Prinzip nur zwei Architekturmuster: DIP-basierend und IOSP-basierend. Die drei DIP-basierenden Muster gehen im Kern auf dieselbe Idee zurück: Nutze Dependency Inversion. Daraus leitet sich aber, neben Prinzipienverletzungen, das Problem ab, dass Tests sehr häufig auf Attrappen angewiesen sind.

Trennt man dagegen Integration und Operation, entsteht besser verständlicher Code, der darüber hinaus auch leichter zu testen ist, da die Tests keine Attrappen benötigen. ■



**Stefan Lieser**

sucht ständig nach Verbesserung und neuen Wegen, um die innere Qualität von Software zu optimieren. Gemeinsam mit Ralf Westphal hat er die Clean Code Developer Initiative (<https://clean-code-developer.de>) ins Leben gerufen. <https://ccd-akademie.de>

# DWX ACADEMY

Hands-on Workshops  
und Weiterbildung für  
Software-Entwickler  
und -Architekten

REMOTE  
ODER  
INHOUSE

## Cloud und DevOps Hands-on Workshops

|                                                                                |          |
|--------------------------------------------------------------------------------|----------|
| Kubernetes: Ab in die Cloud                                                    | (2 Tage) |
| Erste Schritte mit Docker für .NET                                             | (2 Tage) |
| Angewandte DevOps-Prozesse für .NET<br>Core Services mit Docker und Kubernetes | (3 Tage) |
| Azure DevOps – CI/CD mit Azure Pipelines<br>und Git                            | (3 Tage) |
| Azure Pipeline – Multistage YAML Pipelines<br>für Kubernetes                   | (3 Tage) |
| Verteilte Anwendungen bei Azure<br>mit Docker und Kubernetes                   | (2 Tage) |
| DevOps mit GitHub                                                              | (3 Tage) |
| Git-Masterclass – Versionsverwaltung<br>für die Praxis                         | (3 Tage) |
| DevSecOps – Security in den Prozess integrieren                                | (2 Tage) |



Weitere Informationen auf [developer-world.de/dwx-academy](https://developer-world.de/dwx-academy)  
Preise auf Anfrage!

Deine Ansprechpartnerin: Susanne Herl • +49 731 88005-8835 • [susanne.herl@ebnermedia.de](mailto:susanne.herl@ebnermedia.de)



## BILDSCHIRMINHALTE AUFZEICHNEN MIT JAVASCRIPT

# Browserspionage

JavaScript kann nicht nur Kameras anzapfen, um an Videodaten zu gelangen, sondern auch den Bildschirm – Zustimmung vorausgesetzt.

Endlich ist es so weit: Unsere lose Folge spannender JavaScript-APIs findet eine Fortsetzung. Diesmal geht es um Möglichkeiten, bestimmte Videoinhalte aufzuzeichnen. Dafür gibt es in modernen Browsern bereits seit Längerem unterschiedliche Optionen. Der bekannteste Vertreter hierfür ist vermutlich `getUserMedia()`, eine Methode, mit der auf Videodaten zugegriffen werden kann, die über ein Hardwaregerät (etwa: Webcam) oder auch ein `<video>`-Element (sogar `<audio>` ist möglich, wenn es nur der Ton sein soll) dem Client zur Verfügung gestellt werden. Typische Nutznießer dieses API sind die Webinterfaces von Videokonferenzsystemen, die auf genau diese Weise überhaupt erst Audio- und Videodaten übertragen können. Wir hatten vor einiger Zeit dieses API bereits im Einsatz, als wir QR-Codes aus dem Browser heraus gescannt haben [1].

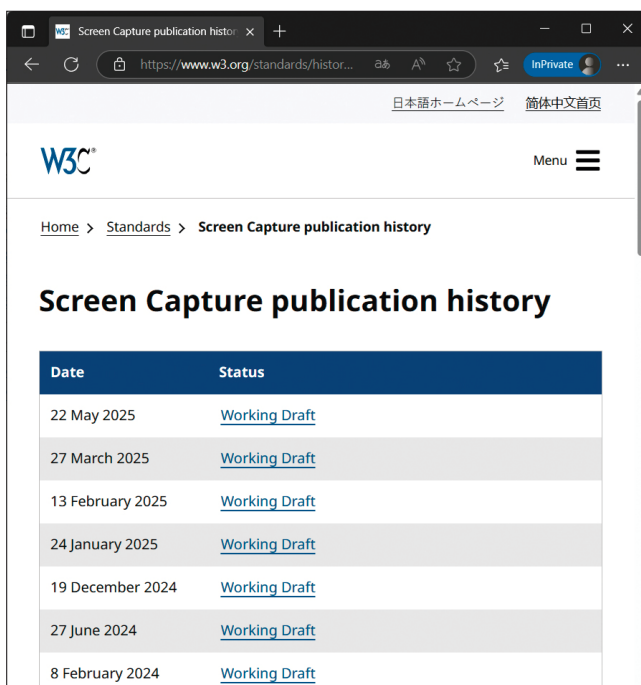
vom Bildschirm. Das kann der Inhalt eines Browserfensters beziehungsweise -tabs sein oder sogar alles, was zu sehen ist.

## Standards

Bevor wir in die Untiefen des API einsteigen, machen wir zunächst einen Realitätscheck: Wie sieht es überhaupt mit der Browserunterstützung aus? Beginnen wir mit der zugrunde liegenden Spezifikation, dem Screen Capture API [2]. Dieses entsteht unter der Ägide des World Wide Web Consortiums (W3C). Zum Erstellungszeitpunkt dieses Artikels (Mitte Juni 2025) handelte es sich bei der Schnittstelle noch um einen W3C Working Draft, also einen Arbeitsentwurf. Letztes Änderungsdatum ist Mai 2025, und ein Blick auf die Versionshistorie [3] beziehungsweise **Bild 1** zeigt, dass es alle paar Monate einen neuen (Zwischen-)Stand gibt. Negativ interpretiert: Da wird noch viel gebastelt. Positiver Spin: Da ist Bewegung drin. Und in der Tat: Das API selbst ist recht stabil, es sind die Details, an denen es immer noch Änderungen gibt.

Hoffnung macht die Liste der aktuellen Bearbeiter der Spezifikation. Mit der Leitung sind zwei Personen betraut, eine von Google, eine von Mozilla. Der sonst immer wieder festzustellende Trend – in der dotnetpro schon häufig moniert –, dass insbesondere Google neue APIs erstellt, die dann auch nur in Google Chrome und verwandten Browsern implementiert werden, scheint also hier nicht der Fall zu sein. Auch in der Liste der früheren Bearbeiter finden sich Google, Mozilla sowie zwei Mitarbeiter von Cisco, also einer Firma mit Erfahrung im Implementieren von Videokonferenztools.

Die aktuelle Situation des Browsersupports gestaltet sich dennoch eher unübersichtlich. Auf Caniuse.com verbirgt sich das API unter dem Namen der Hauptfunktion, die hier zum Einsatz kommt, nämlich `getDisplayMedia()` [4]. Dort ist die Clientunterstützung gut und für alle modernen Desktop-Browser (!) als grün markiert. Bei den MDN Web Docs (früher bekannt unter dem Namen MDN, Mozilla Developer Network) sieht es etwas anders aus [5]. In der Präambel der Seite heißt es, das Feature würde vom Edge-Browser unterstützt und von Chrome nur eingeschränkt. Die Kompatibilitätstabelle am Ende der Seite wiederum listet Chrome und Edge als ebenbürtig auf, aber Safari und Firefox bleiben außen vor. Bei den letztgenannten Browsern gibt es jedoch eine Ausnahme: `getDisplayMedia()` selbst wird unterstützt, also die Kernfunktion. Und so lassen sich die unterschiedlichen Angaben in Einklang bringen: Prinzipiellen Support bieten alle aktuellen Browser, aber bei Spezialthemen haben Chrome und Konsorten die Nase vorn.

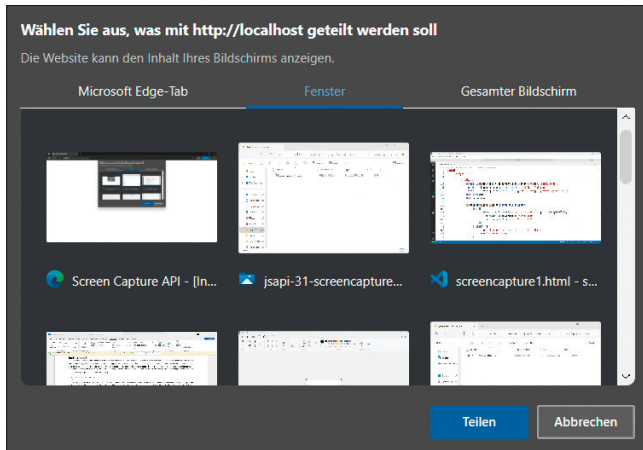


| Date             | Status                        |
|------------------|-------------------------------|
| 22 May 2025      | <a href="#">Working Draft</a> |
| 27 March 2025    | <a href="#">Working Draft</a> |
| 13 February 2025 | <a href="#">Working Draft</a> |
| 24 January 2025  | <a href="#">Working Draft</a> |
| 19 December 2024 | <a href="#">Working Draft</a> |
| 27 June 2024     | <a href="#">Working Draft</a> |
| 8 February 2024  | <a href="#">Working Draft</a> |

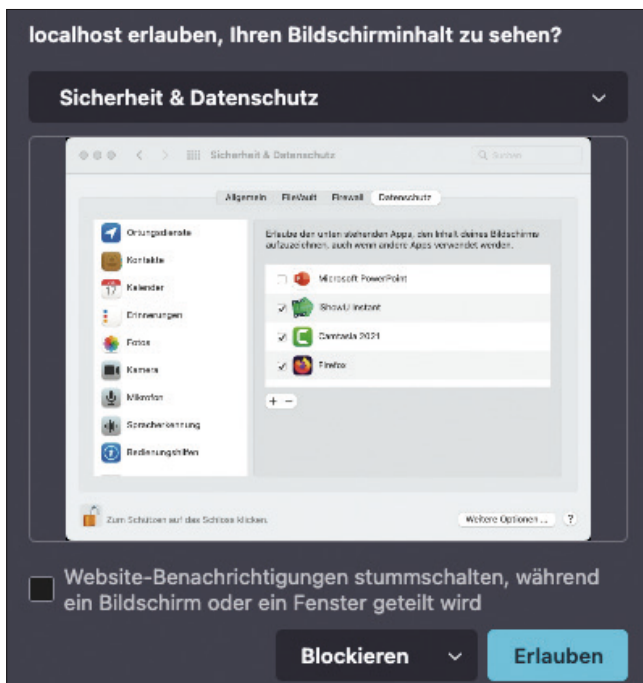
**Und es bewegt sich doch:** Status-Updates beim Screen Capture API (**Bild 1**)

Etwas abseits davon gibt es ein weiteres API, das ebenfalls in Videokonferenzsoftware zum Einsatz kommt, aber darüber hinaus zusätzliche praktische Einsatzzwecke aufweist: Das Screen Capture API erlaubt dem Browser Zugriff auf Dinge

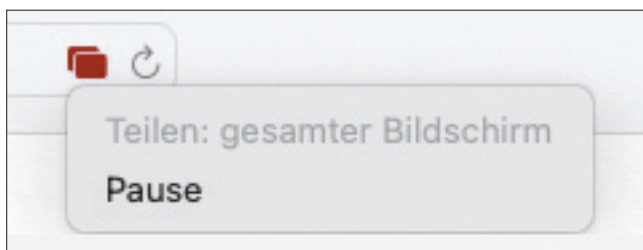




Aller .... Zugriffsanforderung in Edge (ähnlich: Chrome) (Bild 2)



... guten Dinge .... Zugriffsanforderung in Firefox, sogar mit Live-Vorschau (Bild 3)



... sind drei: Keine Zugriffsanforderung, aber eine Steuerung in Safari (Bild 4)

Wichtig ist, dass sich die Unterstützung aktuell auf die Desktop-Varianten von Browsern beschränkt. Zwar wäre es wohl technisch möglich, auch die Bildschirminformationen

von Mobilgeräten via JavaScript-API verfügbar zu machen, aber das ist in der Praxis schlicht ein exotischer Anwendungsfall und deshalb aktuell nicht vorgesehen. Es soll allerdings nicht verschwiegen werden, dass das Chromium-Projekt für die Unterstützung unter Android seit über zehn Jahren ein Ticket hat, und erst 2024 gab es insofern Bewegung, als ein Proof of Concept existiert [6]. Ob es zu einer Integration in den Browser kommt, scheint noch nicht festzustehen.

## Zugriff anfordern

Der erste Schritt zum Zugriff auf Bildschirm Inhalte ist ein höfliches Fragen. Aus sehr offensichtlichen Gründen kann eine Website nicht ohne explizite Zustimmung Anwendungsfenster filmen. Der Wächter, der dies einhält, ist der Webbrowser selbst. Die Aufforderung zur Zustimmung ist nur als Reaktion auf eine Nutzerinteraktion (in der Regel: Mausklick) möglich, wie auch bei anderen JavaScript-APIs (etwa Zugriff auf Zwischenablage oder Dateien oder Bluetooth-Geräte). Der folgende Code sorgt dafür, dass die Sicherheitsabfrage im Browser erscheint:

```
navigator.mediaDevices.getDisplayMedia();
```

Wie genau die Aufforderung zur Zustimmung aussieht, dürfen sich die Teams der Webbrowser jeweils aussuchen, dazu gibt es keine expliziten Vorgaben in der Spezifikation. Und so ist das UI auch teils sehr unterschiedlich, wie an den Beispielen von Edge (Bild 2) und Firefox (Bild 3) verdeutlicht. Safari teilt standardmäßig den kompletten Bildschirm, bietet aber in der Adressleiste eine Option an, das zu beenden (Bild 4).

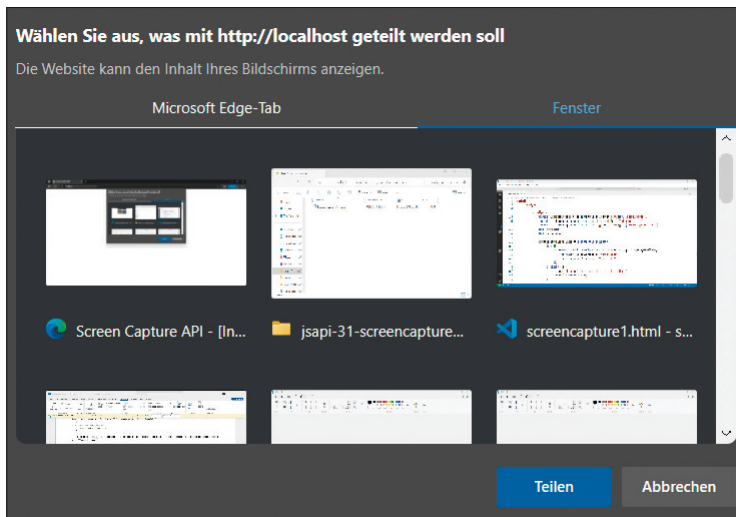
Allen UIs gemeinsam ist, dass es prinzipiell drei Ansätze gibt, was genau freigegeben werden soll: entweder ein spezifischer Browser-Tab oder ein Anwendungsfenster (nicht notwendigerweise der Browser!) oder ein kompletter Bildschirm. Bei Multi-Display-Setups kann also ein spezifisches Ausgabegerät ausgewählt werden. Dies erfüllt die typischen Anforderungen beim Screen-Sharing: entweder die eine Anwendung, mit der gearbeitet wird – wenn es eine Browser-App ist, dann unter Umständen nur der relevante Tab, die anderen gehen die Zusehenden nicht unbedingt etwas an –, oder eben der komplette Bildschirm, wenn zwischen Anwendungen gewechselt werden soll.

Die Möglichkeiten, die es in der Zugriffsanforderung gibt, können beim Aufruf von `getDisplayMedia()` innerhalb bestimmter Grenzen konfiguriert werden – allerdings aktuell nur in Chrome und verwandten Browsern (Edge, Opera, Brave und so weiter). Die Option `monitorTypeSurfaces` gibt an, ob das Teilen eines kompletten Bildschirms erlaubt sein soll; mit dem Wert `"exclude"` wird das unterbunden. Ein explizites Einschalten wird mit dem Wert `"include"` erreicht (und ist in allen aktuellen Browsern, die die Option unterstützen, der Standardwert).

Betrachten wir den folgenden Code:

```
navigator.mediaDevices.getDisplayMedia({
  monitorTypeSurfaces: "exclude"
});
```





**Kameraausschnitt:** Der komplette Bildschirm kann nicht mehr geteilt werden (Bild 5)

Wie Bild 5 zeigt, erscheint daraufhin der Tab für das Vollbild-Teilen (in Bild 2 noch zu sehen) nicht mehr im Rahmen der Zugriffsanforderung.

Eine weitere Reduktion der verfügbaren Optionen besteht durch das Setzen von `selfBrowserSurface: "exclude"`. Das führt dazu, dass der aktuelle Browser-Tab nicht in der Auswahl erscheint. Viele kennen aus Videokonferenzen den Effekt, der auftritt, wenn der Browser mit der Videokonferenz geteilt wird: Er zeigt somit sich selbst im Browserfenster an, was dann unendlich wiederholt wird. Seit Chrome und Edge in Version 112 ist deshalb `"exclude"` tatsächlich der Standardwert; soll der aktuelle Browser-Tab geteilt werden dürfen, muss `selfBrowserSurface` explizit auf `"include"` gesetzt werden. In älteren Browserversionen was es dagegen genau umgekehrt und `"include"` der Default.

```
navigator.mediaDevices.getDisplayMedia({
  monitorTypeSurfaces: "exclude",
  selfBrowserSurface: "include"
});
```

Der Terminus *surface*, also Oberfläche, tauchte bereits in den beiden vorgestellten Optionen auf und ist auch Bestandteil einer dritten Einstellungsmöglichkeit: wenn *surfaceSwitching* auf `"include"` gestellt wird, erhalten Benutzer ein UI zum Umschalten des zu teilenden Browser-Tabs. Diese Option ist standardmäßig deaktiviert (Wert `"exclude"`). Bild 6 zeigt, wie das Steuerelement in Edge aussieht.

```
navigator.mediaDevices.getDisplayMedia({
  monitorTypeSurfaces: "exclude",
```

```
selfBrowserSurface: "include",
surfaceSwitching: "include"
});
```

Für den Browser gibt es, wie bereits beschrieben, keine genauen Vorschriften, wie das UI zum Einholen der Zustimmung zum Bildschirmteilen auszusehen hat. Durch die Option `preferCurrentTab: true` kann jedoch der aktuelle Browser-Tab als vorausgewählte Option angegeben werden (in diesem Fall darf natürlich `selfBrowserSurface` nicht auf `"exclude"` gesetzt sein). Bild 7 zeigt, wie das in Edge aussieht – hier existiert nur eine einzige Option, nämlich der aktuelle Tab.

## Daten verarbeiten

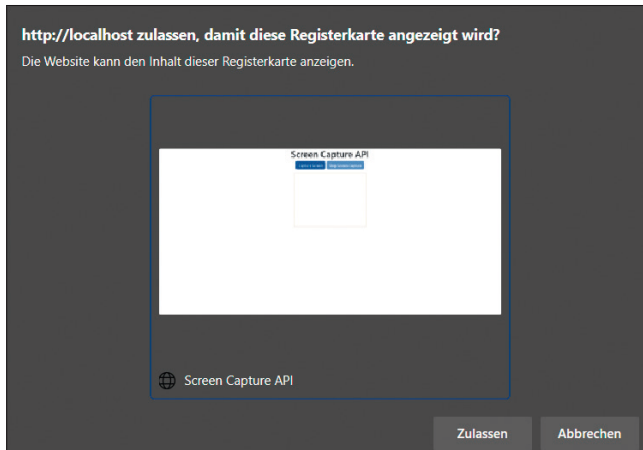
Darüber hinaus gibt es noch einige weitere Einstellungen, aber für die meisten Anwendungsfälle kommen wir hiermit schon recht weit. Es ist daher an der Zeit, die aufgenommenen Inhalte weiterzuverarbeiten. Der Rückgabewert von `getDisplayMedia()` ist, sofern der Bildschirmaufnahme zugestimmt wurde, ein Stream. Den können wir weiterverarbeiten, indem wir ihn beispielsweise in einem `<video>`-Element anzeigen. Ob wir aber tatsächlich Zugriff auf das enthalten, was im gewählten Fenster passiert, müssen wir zunächst feststellen. Ein `try-catch`-Block fängt eventuelle Ausnahmen – und dazu gehört auch die Verweigerung der Aufnahme – ab. Durch den Zeitverzug zwischen Fragen des Benutzers und Antwort ist es sinnvoll, den Aufruf von `getDisplayMedia()` asynchron durchzuführen. Im Erfolgsfall wird der gelieferte Stream an ein `<video>`-HTML-Element gebunden; im Fehlerfall erscheint eine entsprechende Meldung. Hier der relevante Codeausschnitt:

```
document.getElementById("btnCapture").onclick =
  async function() {
    try {
      let stream =
        await navigator.mediaDevices.getDisplayMedia({
          monitorTypeSurfaces: "exclude",
          selfBrowserSurface: "include"
        });
      videoOutput.srcObject = stream;
    } catch (ex) {
      alert("Error! See console for details.");
      console.error(ex);
    }
  };
```

Ein wichtiges Detail fehlt allerdings noch, und das nicht auf der JavaScript-Seite. Das Video-Element, in dem die Ausga-



**Linsenwechsel:** Schnelles Umschalten beim Teilen des Bildschirms (Bild 6)



**Selbstbezogen:** Der aktuelle Tab kann (als Einziges) geteilt werden (Bild 7)

be erfolgt, benötigt unbedingt das Attribut *autoplay*, damit die Inhalte dort auch angezeigt werden (ansonsten bleibt die Anzeige leer). Alternativ müssten wir nach dem Setzen von *videoOutput.srcObject* die Methode *videoOutput.play()* aufrufen. Außerdem ergänzen wir optional das Stummschalt-Attribut *<video muted>*, damit es nicht gegebenenfalls zu einem Audio-Echo kommt:

```
<video id="videoOutput" autoplay muted>
```

**Bild 8** zeigt die Anwendung im Browser; im Video-Element erscheint als Live-Bild die IDE mit dem Code, die auf einem anderen Monitor geöffnet war.

Ein begonnener Stream kann durch das UI, das der Browser anbietet, wieder beendet werden (beispielsweise in **Bild 8** unten zu sehen). Es ist jedoch auch möglich, diese Funktionalität direkt im Web-UI anzubieten. Dazu erstellen wir eine weitere Schaltfläche, die zu Beginn deaktiviert ist. Beim Starten des Streams wird sie dann aktiviert. Hier das Markup:

```
<button class="btn btn-primary"
  id="btnStopCapture" disabled>
  Stop Screen Capture</button>
```

Beim Klick auf diese Schaltfläche wird die *srcObject*-Eigenschaft des *<video>*-Elements auf null gesetzt, also die Anzeige gestoppt. Ein klein wenig aufwendiger ist tatsächlich das eigentliche Anhalten der Übertragung. Ein Stream kann mehrere Tracks enthalten, etwa Audio- und Videoinformationen. Per *forEach*-Schleife wird nacheinander jeder Track angehalten. Das sieht im Code so aus:

```
btnStopCapture.onclick =
  function() {
```

```
    videoOutput.srcObject = null;
    stream.getTracks().forEach(track => track.stop());
  };
```

## Aufzeichnung speichern

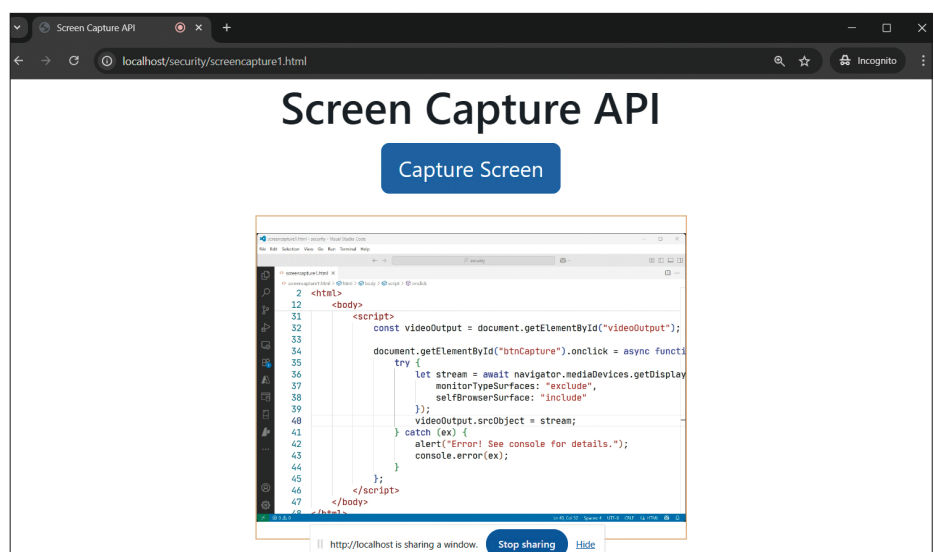
Das Screen Capture API ist aber nicht nur in der Lage, Bildschirminformationen als Stream in eine Website zu integrieren, sondern kann diese auch abspeichern. Bei Videokonferenztools wird der Stream an einen Server übertragen, während wir die Videodaten in unserem Beispiel direkt in eine lokale Datei schreiben.

Die Klasse *MediaRecorder* ist die zentrale Anlaufstelle, um einen Stream aufzunehmen. Wir initialisieren sie mit dem Stream und starten mit der *start()*-Methode die Aufzeichnung:

```
recorder = new MediaRecorder(stream);
recorder.start();
```

Die Daten des *MediaRecorder* können dann in eine Datei gespeichert werden. Hierzu bedienen wir uns der Techniken aus [7], insbesondere des (zumindest in Chromium verfügbaren) Datei-Picker für Schreibzugriff. Der folgende Code lässt den Benutzer einen Dateinamen wählen – als Dateiformat geben wir wegen seiner guten Verbreitung WebM vor – und erzeugt ein Datei-Handle mit Schreibrechten:

```
let fileHandle = await window.showSaveFilePicker({
  types: [
    {
      description: "WebM Video Files (*.webm)",
      accept: {
        "video/webm": ".webm"
      }
    }
  ]
});
```



**Spiegeln, Spiegeln:** Der Editor wird live in den Browser gestreamt (Bild 8)

```
const writeable = await
  fileHandle.createWritable();
```

Zurück zum *MediaRecorder*: Dieser unterstützt ein Ereignis *dataavailable*, das periodisch aufgerufen wird, wenn wieder ein Schwung Daten zur Verfügung steht – oder wenn die Übertragung beendet ist. Im zugehörigen Eventhandler schreiben wir etwaige Daten in die zuvor ausgewählte Datei weg. Ist die Übertragung gestoppt (der *state* des *MediaRecorder* hat dann den Wert *inactive*), schließen wir die Datei:

```
recorder.addEventListener(
  "dataavailable",
  async function(event) {
    await writeable.write(event.data);
    if (recorder.state === "inactive")
    {
      await writeable.close();
    }
  });
```

Die neue Fassung unseres Beispiels fragt nach einem Klick auf *Start Capture* zunächst nach der Erlaubnis der Bildschirmaufnahme und fordert dann zur Angabe eines Dateinamens

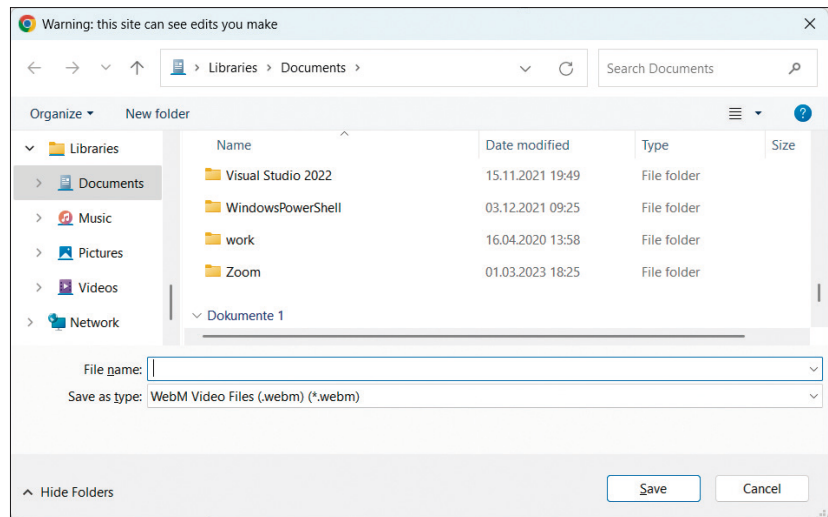
## ● Cross-Browser-Support

Die bisher erstellte Anwendung funktioniert auch in Firefox und Safari – bis auf zwei Aspekte. Zunächst einmal kennen beide Browser die Optionen nicht, die wir beim Aufruf von *getDisplayMedia()* setzen können, beispielsweise *monitorTypeSurfaces* oder *selfBrowserSurface*. Das ist aber kein entscheidender Mangel, denn eine Quellauswahl ist trotzdem möglich.

Ein größeres Manko ist allerdings das nicht funktionsfähige Speichern der Datei, denn die Methode *showSaveFilePicker()* gibt es dort (noch) nicht. Der Beispielcode zu diesem Artikel unter [9] umgeht das mit einer Feature-Erkennung – in Firefox und Safari wird also nichts gespeichert:

```
if (typeof window.showOpenFilePicker === "function")
{
  // Stream speichern
}
```

Es gibt jedoch hierfür Polyfill-Ansätze, die im Wesentlichen ein *<a download>*-Element erstellen und dieses dann per JavaScript anklicken. Eine bekannte ältere Bibliothek hierfür ist *StreamSaver.js* von Jimmy Warting [8], aber es gibt eine ganze Reihe von weiteren, wenngleich technisch ähnlichen Alternativen. Ein kompletter Polyfill für das File System API, inklusive weiterer Methoden und Typen, ist unter [10] verfügbar.



**Kein Recht auf Vergessen:** Was geteilt wird, landet in einer lokalen Datei (Bild 9)

zum Speichern auf, je nach Betriebssystem inklusive Warnmeldung in der Titelleiste des Fensters (siehe Bild 9). Nach Beendigung der Aufzeichnung befindet sich am angegebenen Ort dann tatsächlich eine WebM-Datei mit dem Inhalt des zuvor gewählten Fensters oder Bildschirms. ■

- [1] Christian Wenz, *Scannen im Browser*, dotnetpro 8/2021, Seite 62 ff.
- [2] Screen Capture API beim W3C, <https://www.w3.org/TR/screen-capture/>
- [3] Versionshistorie des Screen Capture API, <https://www.w3.org/standards/history/screen-capture/>
- [4] Screen Capture API bei Caniuse.com, [https://caniuse.com/mdn-api\\_mediadevices\\_getdisplaymedia](https://caniuse.com/mdn-api_mediadevices_getdisplaymedia)
- [5] Screen Capture API bei MDN, [https://developer.mozilla.org/en-US/docs/Web/API/Screen\\_Capture\\_API/Using\\_Screen\\_Capture](https://developer.mozilla.org/en-US/docs/Web/API/Screen_Capture_API/Using_Screen_Capture)
- [6] Diskussion zum Android-Support für das Screen Capture API, <https://issues.chromium.org/issues/40418135>
- [7] Christian Wenz, *Dateibrowser*, dotnetpro 8/2023, Seite 46 ff.
- [8] StreamSaver.js auf GitHub, <https://github.com/jimmywarting/StreamSaver.js>
- [9] Beispielcode zum Artikel, <https://eu-assets.contentstack.com/v3/assets/blt22276bff283176bd/blt683fafed3ed7a30f/jsapi-31-screencapture-code.zip>
- [10] Polyfill für das File System Access API, <https://github.com/use-strict/file-system-access>



### Christian Wenz

ist Softwarearchitekt, Berater und Spezialist für die Themen Performance und Web Security. Seit 2004 ist er Microsoft MVP. Seine neuesten Bücher sind „ASP.NET Core Security“ (erschienen bei Manning) und „C# und .NET 8“ (Hanser). [www.christianwenz.de](http://www.christianwenz.de)

# DWX ACADEMY

Hands-on Workshops  
und Weiterbildung für  
Software-Entwickler  
und -Architekten

REMOTE  
ODER  
INHOUSE

## Hands-on Workshops Webentwicklung

|                                                                 |          |
|-----------------------------------------------------------------|----------|
| Webgrundlagen: Programmierung mit HTML, CSS und JavaScript      | (2 Tage) |
| Next Level TypeScript                                           | (2 Tage) |
| Clean-Code-Entwicklung für JavaScript und TypeScript-Entwickler | (3 Tage) |
| Einstieg in Angular                                             | (3 Tage) |
| Angular für .NET-Entwickler                                     | (3 Tage) |
| Angular im Enterprise-Projekt meistern: Redux & NGRX            | (2 Tage) |
| Moderne React Patterns                                          | (3 Tage) |
| Einführung in Vue.js                                            | (2 Tage) |
| Vue.js 3 – What's new?                                          | (2 Tage) |
| Progressive Web App Bootcamp                                    | (3 Tage) |
| Einführung in GraphQL                                           | (2 Tage) |



Weitere Informationen auf [developer-world.de/dwx-academy](https://developer-world.de/dwx-academy)  
Preise auf Anfrage!

Deine Ansprechpartnerin: Susanne Herl • +49 731 88005-8835 • [susanne.herl@ebnermedia.de](mailto:susanne.herl@ebnermedia.de)





## POSTGRESQL

# Das Concurrency-Modell

Wie PostgreSQL parallele Abfragen mehrerer Benutzer bearbeitet.

Auch auf der Microsoft-Plattform hat die leistungsstarke Open-Source-Datenbank PostgreSQL an Bedeutung gewonnen. Grund genug, die Details der Datenbank näher zu betrachten. Im zurückliegenden PostgreSQL-Artikel [1] wurden Performance-Tuning-Techniken vorgestellt und dabei auf die Unterschiede zum SQL Server hingewiesen. Diesmal geht es darum, wie PostgreSQL Abfragen mehrerer Benutzer gleichzeitig bearbeitet.

## Multi-Version Concurrency Control

Das zeitgleiche Verarbeiten mehrerer Transaktionen spielt in Datenbanken eine entscheidende Rolle. So wird einerseits die Performance optimiert und zugleich die Datenkonsistenz sichergestellt. Viele moderne Datenbanken (auch PostgreSQL) nutzen dafür Multi-Version Concurrency Control (MVCC).

MVCC ist eine optimistische Methode zum Verwalten gleichzeitig ausgeführter Transaktionen, die verhindert, dass sich Lese- und Schreiboperationen gegenseitig blockieren.

Dieses Konzept beruht darauf, dass jede Transaktion eine eigene Version der Daten sieht, ohne dass sich die Transaktionen gegenseitig beeinflussen. PostgreSQL speichert also mehrere Versionen eines Datensatzes, um parallele Transaktionen effizient zu handhaben.

Der SQL Server verfolgt einen anderen Ansatz: Er bietet zwei Concurrency-Modelle an, die unterschiedliche Strategien zur Synchronisation von Transaktionen nutzen:

- Pessimistic Concurrency,
- Optimistic Concurrency.

Beim Pessimistic-Concurrency-Modell fordert jede schreibende Transaktion einen Exclusive Lock (X) an, und jede lesende Transaktion fordert einen Shared Lock (S) an. Wird auf dieselben Datensätze zugegriffen, führt das zu folgenden Problemen:

- Lesende Transaktionen blockieren schreibende Transaktionen.

- Schreibende Transaktionen blockieren lesende Transaktionen.
- Schreibende Transaktionen blockieren andere schreibende Transaktionen.

Abhilfe schafft das mit SQL Server 2005 eingeführte Optimistic-Concurrency-Modell, welches jedoch in On-Premises-Versionen von SQL Server zunächst deaktiviert ist.

Die Idee hinter Optimistic Concurrency ist, dass beim lesenden Zugriff auf Datensätze keine Shared Locks angefordert werden. Das vermeidet Blockaden zwischen lesenden und schreibenden Transaktionen.

Dabei werden bei schreibenden Transaktionen (*UPDATE*, *DELETE*) alte Datensatzversionen im Version Store gespeichert, der in der *TempDb* gehalten wird. Nicht mehr benötigte Datensatzversionen entfernt ein periodisch laufender Clean-up-Job.

Dieser Ansatz ist dem MVCC-Modell von PostgreSQL sehr ähnlich, welches bei jeder Änderung eines Datensatzes eine neue Datensatzversion erzeugt und in der Heap-Tabelle speichert. Für jede Datensatzversion notieren die internen Spalten *xmin* und *xmax*, für welche Transaktionen die Datensatzversion sichtbar ist:

- *xmin*: Gibt an, welche Transaktion die Datensatzversion erstellt hat.
- *xmax*: Gibt an, ab welcher Transaktion die Datensatzversion als gelöscht beziehungsweise aktualisiert gilt.

Daraus folgt, dass für eine bestimmte Transaktion immer nur eine logische Datensatzversion sichtbar ist. Physisch können aber in einer Heap-Tabelle mehrere unterschiedliche Datensatzversionen gespeichert sein.

Startet eine neue Transaktion, sieht diese nur diejenigen Datensätze, die auch schon vor ihrem Start gültig waren. Änderungen durch andere, noch nicht abgeschlossene Transaktionen sind für sie unsichtbar. Dadurch vermeidet PostgreSQL viele Deadlocks und Blockaden, die in Lock-basierten Daten-

|   | ctid  | state  | xmin      | xmax      | heap_hot_updated | heap_only_tuple | t_ctid |
|---|-------|--------|-----------|-----------|------------------|-----------------|--------|
| 1 | (0,1) | normal | 1292150 c | 1292151 c | t                | [NULL]          | (0,2)  |
| 2 | (0,2) | normal | 1292151 c | 1292152 c | t                | t               | (0,3)  |
| 3 | (0,3) | normal | 1292152 c | 1292153 c | t                | t               | (0,4)  |
| 4 | (0,4) | normal | 1292153 c | 0 a       | [NULL]           | t               | (0,4)  |

Vier Versionen desselben Datensatzes (Bild 1)

## ● Listing 1: Physische Datensatzversionen

```
-- Create a new database
CREATE DATABASE MVCC;

-- Create the pageinspect extension
CREATE EXTENSION pageinspect;

-- The following function returns the various
-- record versions of a page
CREATE FUNCTION heap_page(
    relname TEXT,
    pagenumber INT)
RETURNS TABLE (
    ctid tid,
    state TEXT,
    xmin TEXT,
    xmax TEXT,
    heap_hot_updated TEXT,
    heap_only_tuple TEXT,
    t_ctid tid)
AS $$
SELECT
    (pagenumber, lp)::TEXT::tid AS ctid,
    CASE lp_flags
        WHEN 0 THEN 'unused'
        WHEN 1 THEN 'normal'
        WHEN 2 THEN 'redirect to ' || lp_off
        WHEN 3 THEN 'dead'
    END AS state,
    t_xmin || CASE
        WHEN (t_infomask & 256) > 0 THEN ' c'
        WHEN (t_infomask & 512) > 0 THEN ' a'
        ELSE ''
    END AS xmin,
    t_xmax || CASE
        WHEN (t_infomask & 1024) > 0 THEN ' c'
        WHEN (t_infomask & 2048) > 0 THEN ' a'
        ELSE ''
    END AS xmax,
    CASE
        WHEN (t_infomask2 & 16384) > 0
            THEN 't' END AS heap_hot_updated,
        CASE WHEN (t_infomask2 & 32768) > 0
            THEN 't' END AS heap_only_tuple,
        t_ctid
    FROM heap_page_items(
        get_raw_page(relname, pagenumber))
    ORDER BY lp;
$$ LANGUAGE sql;

-- Create a new table
CREATE TABLE TestTable1
(
    Col1 INT GENERATED ALWAYS AS IDENTITY,
    Col2 VARCHAR(100)
);

-- Insert a new record into the table
INSERT INTO TestTable1(Col2)
VALUES ('Klaus');

-- There is one record
SELECT * FROM TestTable1;

-- Update the record a few times
UPDATE TestTable1 SET Col2 = 'Karin' WHERE Col1 = 1;
UPDATE TestTable1 SET Col2 = 'Philip' WHERE Col1 = 1;
UPDATE TestTable1 SET Col2 = 'Daniel' WHERE Col1 = 1;

-- There is still 1 logical record in the table
SELECT * FROM TestTable1;

-- But there are multiple row versions
SELECT * FROM heap_page('TestTable1', 0);

-- Shows the xmin and xmax columns
SELECT xmin, xmax, * FROM TestTable1;
```

banksystemen wie Microsofts SQL Server auftreten können. Nicht mehr benötigte Datensatzversionen werden zu einem späteren Zeitpunkt automatisch durch den *VACUUM*-Prozess gelöscht.

Sehen wir uns ein konkretes Beispiel dazu an: In **Listing 1** wird die Funktion *heap\_page()* erstellt, mit deren Hilfe über die Extension *pageinspect* die einzelnen physischen Datensatzversionen einer Data Page ausgelesen und aufbereitet werden. Im nächsten Schritt wird eine neue Tabelle angelegt, und es wird darin ein Datensatz eingefügt, der anschließend mehrere Male über ein *UPDATE*-Statement aktualisiert wird. In der Heap-Tabelle werden dadurch mehrere Datensatzversionen erzeugt.

**Bild 1** zeigt die vier Datensatzversionen, welche das *INSERT*-Statement und die nachfolgenden *UPDATE*-Statements generiert haben. Wie Sie sehen, wird über die Spalten *xmin* und *xmax* festgelegt, für welche Transaktionen eine Version sichtbar ist.

Es handelt sich dabei um Transaction-IDs in Form fortlaufender 32-Bit-Werte. Da PostgreSQL an dieser Stelle keine 64-Bit-Werte unterstützt, wird bei einem Überlauf ein sogenanntes Transaction Wraparound [2] ausgelöst.

## Transaction Isolation Levels

Beim gleichzeitigen Ausführen mehrerer Transaktionen spielen Transaction Isolation Levels eine entscheidende Rolle. ►

## ● Listing 2: Read Committed

```
-- Begin a new transaction, and update
-- a record
BEGIN TRANSACTION;
UPDATE TestTable1 SET Col2 = 'Klaus'
WHERE Col1 = 1;

-- Run the following SELECT statement in a
-- different session.
-- It will not block, and shows the current row
-- version ("Daniel").
-- The default isolation level is
-- Read Committed.
BEGIN;
SHOW transaction_isolation;
SELECT * FROM TestTable1;

-- Commit the transaction
COMMIT;

-- Run the following SELECT statement in a
-- different session. It will not block, and
-- shows the current row version ("Klaus").
-- This is a Non-Repeatable Read.
SELECT * FROM TestTable1;
COMMIT;
```

Sie definieren, wie und wann Änderungen, die während einer Transaktion vorgenommen wurden, für andere Transaktionen sichtbar werden.

PostgreSQL verwendet das MVCC-System zur Steuerung der Isolation, während der SQL Server traditionell auf Sperren (Locks) setzt und zusätzlich über Optimistic Concurrency eine Snapshot-Isolation implementiert. PostgreSQL unterstützt die folgenden vier standardisierten Transaction Isolation Levels nach SQL-Standard:

- *Read Uncommitted*,
- *Read Committed*,
- *Repeatable Read*,
- *Serializable*.

Eine Besonderheit ist der Level *Read Uncommitted*, da er intern wie der Isolation Level *Read Committed* behandelt wird.

## ● Listing 3: Repeatable Read

```
-- Run the following SELECT statement in a
-- different session.
-- It shows the current version of the row
-- ("Klaus"), and maintains that version during
-- the whole transaction
BEGIN ISOLATION LEVEL REPEATABLE READ;
SHOW transaction_isolation;
SELECT xmin, xmax, * FROM TestTable1;

-- Begin a new transaction, and update the
-- record. The new generated row version will
-- not be visible in the other transaction.
BEGIN TRANSACTION;
UPDATE TestTable1 SET Col2 = 'Karin' WHERE Col1 = 1;
SELECT * FROM TestTable1;

-- Rerun the SELECT statement in the other
-- session. It still shows the old row version
-- ("Klaus"), because of the isolation level
-- Repeatable Read.
SELECT * FROM TestTable1;

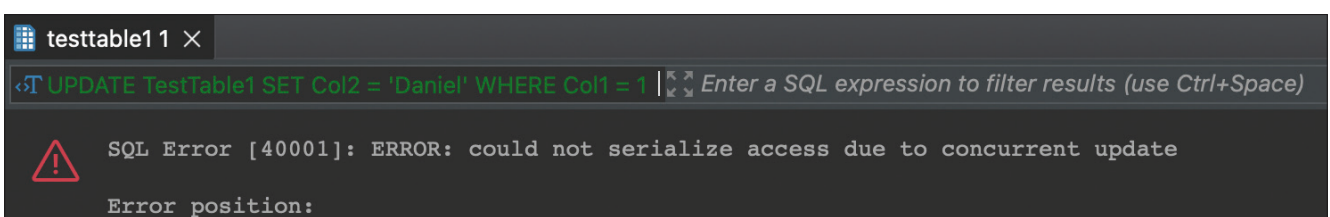
-- Commit the transaction
COMMIT;

-- Rerun the SELECT statement in the other
-- session. It still shows the old row version
-- ("Klaus"), because of the isolation level
-- Repeatable Read.
SELECT * FROM TestTable1;

-- Commit the transaction
COMMIT;
```

Das liegt daran, dass es beim Einsatz des MVCC-Systems niemals zu sogenannten Dirty Reads kommen kann. Sie sind in PostgreSQL technisch nicht möglich.

Der voreingestellte Isolation Level ist *Read Committed*, der Non-Repeatable Reads und Phantom Records erlaubt. Der große Unterschied zum Pessimistic-Concurrency-Modell des SQL Servers ist, dass sich lesende und schreibende Transaktionen nicht gegenseitig blockieren.



**Serialisierungsfehler** beim Aktualisieren desselben Datensatzes in mehreren Transaktionen im Isolation Level Repeatable Read (Bild 2)

Sehen Sie sich dazu bitte [Listing 2](#) einmal näher an. Wie Sie dem Beispiel entnehmen können, hat sich beim mehrmaligen Lesen des gleichen Datensatzes in der zweiten Session der Datensatzinhalt geändert.

Es handelt sich dabei um die zuvor angesprochenen Non-Repeatable Reads – Datensätze können sich innerhalb einer Transaktion folglich ändern.

Möchten Sie Non-Repeatable Reads vermeiden, müssen Sie den Transaction Isolation Level *Repeatable Read* nutzen, der diese verhindert. In [Listing 3](#) finden Sie ein Beispiel dazu.

Aktualisieren Sie im Isolation Level *Repeatable Read* den selben Datensatz in mehreren Transaktionen, generiert Ihnen PostgreSQL einen Serialization Error, wie Sie in [Bild 2](#) sehen können. ►

#### ● Listing 4: Serializable

```
-- Create a new table
CREATE TABLE Customers
(
    CustomerID INT GENERATED
        ALWAYS AS IDENTITY,
    CustomerName VARCHAR(100),
    SalesAmount MONEY,
    CustomerCountry CHAR(2)
);

-- 2 customers in Austria (AT)
INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 1', 100, 'AT');

INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 2', 200, 'AT');

-- 4 customers in Germany (DE)
INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 3', 300, 'DE');

INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 4', 400, 'DE');

INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 5', 500, 'DE');

INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 6', 600, 'DE');

-- 3 customers in Netherlands (NL)
INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 7', 700, 'NL');

INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 8', 800, 'NL');

INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 9', 900, 'NL');

-- Return the inserted records
SELECT * FROM Customers;

-- Return the sales amount for each country

SELECT SUM(SalesAmount), CustomerCountry
FROM Customers
GROUP BY CustomerCountry;

-- Begin a new transaction in the isolation
-- level Serializable
BEGIN ISOLATION LEVEL SERIALIZABLE;
SHOW transaction_isolation;

-- 2400
SELECT SUM(SalesAmount) FROM Customers
WHERE CustomerCountry = 'NL';

-- Begin a new transaction in the isolation
-- level Serializable
BEGIN ISOLATION LEVEL SERIALIZABLE;

-- This read is necessary, so that we establish
-- a read/write dependency between both
-- transactions.
-- 2400
SELECT SUM(SalesAmount) FROM Customers
WHERE CustomerCountry = 'NL';

-- Insert a new record for Netherlands
INSERT INTO Customers (CustomerName, SalesAmount,
    CustomerCountry) VALUES('Customer 10', 1000, 'NL');
COMMIT;

-- Still 2400, because we have Repeatable
-- Reads in Serializable
SELECT SUM(SalesAmount) FROM Customers
WHERE CustomerCountry = 'NL';

-- Insert a new record for Netherlands
-- Error: "Could not serialize access due to
-- read/write dependencies among
-- transactions"
-- "Hint: The transaction might succeed if
-- retried."
INSERT INTO Customers (CustomerName,
    SalesAmount, CustomerCountry)
VALUES('Customer 11', 1100, 'NL');
COMMIT;
```



```

BEGIN;
-- Create a new row version
UPDATE TestTable1 SET Col2 = 'Philip' WHERE Col1 = 1;

-- Returns the old row version of "Karin"
BEGIN ISOLATION LEVEL REPEATABLE READ;
SELECT * FROM TestTable1 WHERE Col1 = 1;

-- Commits the new row version "Philip"
COMMIT;

-- Returns the new row version "Philip"
SELECT * FROM TestTable1;

-- The following UPDATE statement references
-- an older row version which was already
-- updated in the meantime.
-- Error: "Could not serialize access due to
-- concurrent update"
UPDATE TestTable1 SET Col2 = 'Daniel'
WHERE Col1 = 1;
ROLLBACK;

```

Wichtig ist hierbei, dass von der aktiven Transaktion kein automatisches Rollback durchgeführt wird. Dies müssen Sie über den Befehl `ROLLBACK` selbst erledigen.

Sollen innerhalb einer Transaktion auch Phantom Records vermieden werden – also Datensätze, die auftreten und auch

```

-- Repeatable Read
BEGIN;
SHOW transaction_isolation;
ROLLBACK;

-- Reset the default isolation level
SET default_transaction_isolation
TO 'read committed';

```

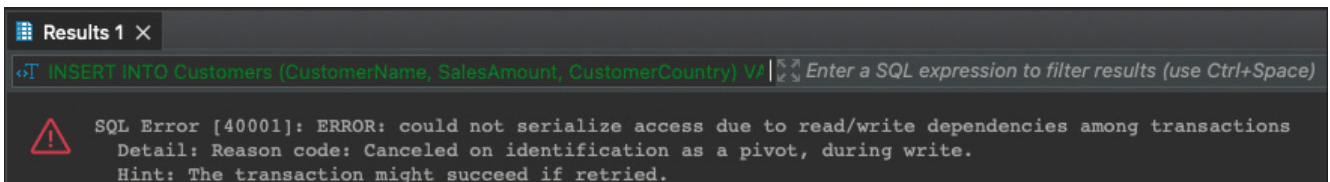
## Das Commit Log

Wie Sie gesehen haben, werden dank MVCC-System bei jeder Datensatzänderung neue Datensatzversionen generiert und in der Heap-Tabelle gespeichert. Eine weitere wesentliche Komponente von MVCC in PostgreSQL ist das Commit Log (CLOG), das für die Verwaltung und die Nachverfolgung von Transaktions-Status verantwortlich ist.

Das Commit Log ist eine spezielle Datenstruktur, die Informationen über den Status von Transaktionen enthält. Anhand dieser Informationen kann man ermitteln, ob eine Transaktion abgeschlossen (committed), abgebrochen oder noch in Bearbeitung ist.

Das Commit Log wird in Pages zu je 8 Kbyte im Verzeichnis `$PGDATA/pg_xact` gespeichert, wobei der Status jeder Transaktion mit Hilfe von 2 Bits abgebildet wird:

- `00`: in Bearbeitung (noch nicht committed),
- `01`: abgeschlossen (committed),
- `10`: abgebrochen (Rollback),
- `11`: abgeschlossene Sub-Transaktion.



Fehlermeldung „could not serialize access“ (Bild 3)

wieder verschwinden –, können Sie den Transaction Isolation Level *Serializable* verwenden. Führen Sie die Abfragen der beiden Sessions in Listing 4 aus, werden auf Basis des Transaction Isolation Levels *Serializable* Phantom Records verhindert. Wie Sie im Listing sehen, würde das letzte `INSERT`-Statement von der zweiten Session einen Phantom Record erzeugen. Daher wird dieses Statement mit einer Fehlermeldung abgebrochen (Bild 3).

Eine weitere Option, die PostgreSQL anbietet, ist das Setzen eines Standard-Transaction-Isolation-Levels für die aktive Session:

```

-- Set the default isolation level to
-- Repeatable Read
SET default_transaction_isolation
TO 'repeatable read';

-- The new default isolation level is now

```

Warum wird das Commit Log überhaupt benötigt? Wie Sie bereits wissen, besitzt jede Datensatzversion die beiden Spalten `xmin` und `xmax`, die angeben, welche Transaktion für das Erzeugen (`xmin`) und das Löschen (`xmax`) der Datensatzversion verantwortlich war. Um jedoch innerhalb einer Transaktion zu prüfen, ob eine Datensatzversion sichtbar ist, muss PostgreSQL wissen, ob die dazugehörige Transaktion committed oder abgebrochen wurde. Und genau diese Informationen liefert das Commit Log:

- Ist eine Transaction-ID (XID) im Commit Log als *committed* markiert, ist die Datensatzversion für andere Transaktionen sichtbar.
- Ist die XID als abgebrochen markiert, wird die Datensatzversion ignoriert.
- Über eine weitere Datenstruktur mit dem Namen *ProcArray* kann PostgreSQL feststellen, ob eine Transaktion noch läuft. In diesem Fall wird eine Datensatzversion ebenfalls ignoriert.

## ● Listing 5: Commit Log

```
-- Create a new table
CREATE TABLE TestTable2
(
    Col1 INT GENERATED ALWAYS AS IDENTITY,
    Col2 VARCHAR(100)
);

-- Insert a new row into the table
BEGIN TRANSACTION;
INSERT INTO TestTable2(Col2) VALUES ('Klaus');

-- Returns the current transaction id
SELECT pg_current_xact_id();
SELECT pg_xact_status('1292203');

-- xmin and xmax are not yet committed
-- or aborted
SELECT * FROM heap_page('TestTable2', 0);

-- Commit the transaction & write the CLOG
COMMIT;
SELECT pg_xact_status('1292203');

-- Nothing has changed in the page, because the
-- transaction status has been written into the
-- so called "CLOG" - the "Commit Log
-- Structure", which is stored in
-- "$PGDATA/pg_xact".
SELECT * FROM heap_page('TestTable2', 0);

-- The bits on the page are only updated by a
-- "normal" running query, like the following:
SELECT * FROM TestTable2;

-- Now the bits on the page are also updated.
-- The xmin version is now in a committed state.
SELECT * FROM heap_page('TestTable2', 0);

-- Delete the record in a new transaction
BEGIN TRANSACTION;
DELETE FROM TestTable2;
SELECT pg_current_xact_id();
SELECT pg_xact_status('1292204');

-- The xmax column is now set to the current
-- transaction id, but the information bits are
-- not yet set.
SELECT * FROM heap_page('TestTable2', 0);

-- Commit the transaction & write the CLOG
COMMIT;

-- The xmax version is not yet in a committed
-- state.
SELECT * FROM heap_page('TestTable2', 0);

-- The bits on the page are only updated by a
-- "normal" running query, like the following:
SELECT * FROM TestTable2;

-- The xmax version is now also in a committed
-- state.
SELECT * FROM heap_page('TestTable2', 0);
```

Damit diese Überprüfungen nicht jedes Mal gegen das Commit Log durchgeführt werden müssen, wird der Status der Transaktionen im Header der einzelnen Datensatzversionen direkt in den Heap Pages gespeichert. Hierzu gibt es die beiden Hint-Bits *xmin\_committed* und *xmin\_aborted*.

Diese Bits werden jedoch nur von nachfolgenden *SELECT*-Statements gesetzt, da das Setzen dieser Bits am Ende einer Transaktion zu aufwendig wäre. Während einer Transaktion können die Bits nicht gesetzt werden, da zu diesem Zeitpunkt noch nicht bekannt ist, ob die Transaktion erfolgreich abgeschlossen wird. Und wenn die Transaktion committet wurde, ist es unklar, welche Datensätze und welche Pages von der Transaktion verändert wurden. Hat eine Transaktion viele Pages geändert, wäre es zu aufwendig, diese Pages zu protokollieren. Außerdem könnten auch einige dieser Pages nicht mehr im Buffer Pool zwischengespeichert sein, wodurch sie erneut aus dem Storage Subsystem gelesen werden müssten, um die Bits zu setzen. Dieses Vorgehen würde den Commit-Prozess spürbar verlangsamen.

In Listing 5 finden Sie ein Beispiel dafür, wie Transaktionen mit dem Commit Log zusammenarbeiten und wie nachfolgende *SELECT*-Statements die Hint-Bits der Datensatzversionen setzen.

## Snapshots

Wie Sie gesehen haben, verwendet PostgreSQL das Konzept von MVCC, um gleichzeitige Transaktionen ohne Zuhilfenahme von Sperren zu ermöglichen. Ein zentrales Element dieser Architektur sind Snapshots. Ein Snapshot definiert, welche Datensatzversionen für eine Transaktion zu einem bestimmten Zeitpunkt sichtbar sind.

Wie Sie bereits wissen, besitzt jede Datensatzversion Informationen darüber, wann (beziehungsweise durch welche Transaktion) sie erstellt und gelöscht wurde – identifiziert durch Transaction IDs (XID).

Ein Snapshot enthält eine Liste der aktiven Transaktionen zu einem bestimmten Zeitpunkt und nutzt diese Informationen, um zu entscheiden, ob eine Datensatzversion für eine ►

bestimmte Transaktion sichtbar ist. Wichtig zu wissen ist in diesem Zusammenhang, dass ein Snapshot ein logisches Konzept in PostgreSQL ist und zu keinem Zeitpunkt physisch persistiert wird.

Beginnt eine Transaktion, erzeugt PostgreSQL automatisch einen Snapshot der aktuell aktiven Transaktionen. Alle Leseoperationen dieser Transaktion basieren auf diesem Snapshot und bleiben bis zum Transaktionsende konsistent, selbst wenn andere Transaktionen zwischenzeitlich neue Daten schreiben oder bestehende Daten ändern.

Wann ein Snapshot erstellt wird, hängt vom gewählten Transaction Isolation Level ab (Bild 4):

- *Read Committed*: Es wird ein Snapshot am Beginn jedes Statements erzeugt.
- *Repeatable Read* und *Serializable*: Es wird ein Snapshot am Beginn der Transaktion erzeugt, der für die komplette Transaktion gültig ist.

Einen Einblick in den aktuellen Snapshot einer aktiven Transaktion bietet PostgreSQL über die Funktion `pg_current_snapshot()`. Wird sie aufgerufen, liefert sie die folgenden Informationen:

- *xmin*: Älteste Transaction-ID, die noch aktiv ist.
- *xmax*: Erste, nicht verwendete Transaction-ID.
- *xip\_list*: Liste aller Transaction-IDs, die zum Beginn des Snapshots aktiv waren.

Listing 6 zeigt ein Beispiel, wie Sie den aktuellen Snapshot mithilfe dieser Funktion einsehen können und wie andere aktive Transaktionen einen Snapshot beeinflussen.

## Visibility Map

Die Datenstruktur Visibility Map gehört zu jeder Heap Table und gibt darüber Auskunft, welche Pages für Transaktionen vollständig sichtbar sind. Das ist der Fall, wenn alle Daten-

### Listing 6: Snapshots

```
-- Create a new table
CREATE TABLE TestTable3
(
    Col1 INT PRIMARY KEY NOT NULL,
    Col2 VARCHAR(100)
);

-- Insert a new record - without committing
-- the transaction
BEGIN TRANSACTION;
INSERT INTO TestTable3(Col1, Col2)
VALUES (1, 'Klaus');
SELECT pg_current_xact_id();

-- Insert a new record in a 2nd session -
-- without committing the transaction
BEGIN TRANSACTION;
INSERT INTO TestTable3(Col1, Col2)
VALUES (2, 'Karin');
SELECT pg_current_xact_id();

-- Run the following transaction in a
-- 3rd session
BEGIN TRANSACTION;
INSERT INTO TestTable3(Col1, Col2)
VALUES (3, 'Philip');

SELECT pg_current_xact_id();
COMMIT TRANSACTION;

-- Run the following query in a 4th session
-- that creates a new snapshot.
-- The function pg_current_snapshot()
-- shows the following information:

-- "xmin:xmax:xip_list".
-- It shows 2 transactions on the "xip_list",
-- because the first 2 INSERT statements
-- are still running, and not yet committed.
BEGIN ISOLATION LEVEL
    REPEATABLE READ;
SELECT pg_current_xact_id();
SELECT pg_current_snapshot();

-- Commit the 2nd transaction
COMMIT TRANSACTION;

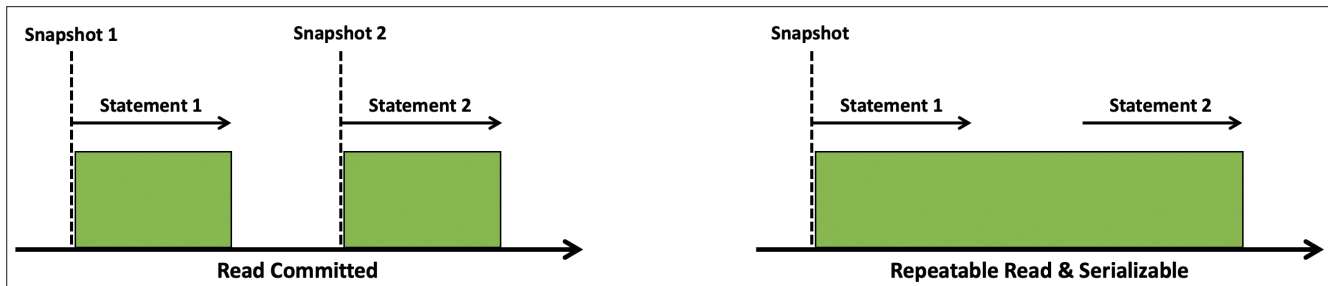
-- Commit the 1st transaction
COMMIT TRANSACTION;

-- Run the following query in the 4th
-- session that created the snapshot
-- The first row isn't shown in this
-- transaction, because it is part of the
-- "xip_list". By the time we started this
-- transaction, the INSERT of the 1st row
-- wasn't yet committed.
UPDATE TestTable3 SET Col2 = 'Daniel'
WHERE Col1 = 3;

SELECT ctid, * FROM TestTable3;
SELECT pg_current_xact_id();
SELECT pg_current_snapshot();

COMMIT TRANSACTION;

-- The table consists of 4 row versions
SELECT * FROM heap_page('TestTable3', 0);
```



Snapshots in PostgreSQL (Bild 4)

satzversionen dieser Page für alle aktuellen und zukünftigen Transaktionen sichtbar sind.

Solche Pages müssen zum Beispiel bei Index-only-Scans nicht erneut geprüft und gelesen werden. Die Visibility Map ist eine Optimierungstechnik von PostgreSQL, um die Performance bei Abfragen zu verbessern beziehungsweise den Wartungsaufwand bei VACUUM-Operationen zu reduzieren.

Die Visibility Map besteht aus einer Bitmap-Datei pro Heap Table, in der jede Page durch ein Bit repräsentiert wird. Ist das Bit gesetzt, bedeutet das, dass alle Datensätze dieser Page sichtbar sind, also von Transaktionen eingefügt wurden, die bereits abgeschlossen sind und seitdem nicht mehr verändert oder gelöscht wurden.

Die Informationen der Visibility Map werden durch VACUUM-Operationen gepflegt. Erst wenn eine VACUUM-Operation eine Page verarbeitet hat und dabei festgestellt hat, dass alle Datensätze darin sichtbar sind, wird das entsprechende Bit in der Visibility Map gesetzt.

Sobald ein Datensatz in einer Page aktualisiert oder gelöscht wird, wird das entsprechende Bit wieder gelöscht, da die Sichtbarkeit der Page nicht mehr gewährleistet werden kann.

Ein wichtiger Aspekt ist dabei, dass die Visibility Map nicht in Echtzeit aktualisiert wird, sondern im Rahmen von Wartungsprozessen. Das bedeutet, dass Index-only-Scans nicht sofort nach dem Einfügen neuer Daten effektiv sind. ►

## ● Listing 7: Visibility Map

```
-- Create a new table
CREATE TABLE VisibilityMap
(
    Id SERIAL PRIMARY KEY,
    Value TEXT
);

-- Insert 10000 rows, but these pages are not
-- visible in the Visibility Map
INSERT INTO VisibilityMap (Value)
SELECT 'Row ' || generate_series(1, 10000);

-- Create an index on the Value column
CREATE INDEX idx_VisibilityMap_Value ON Visibility-
Map(Value);

-- We have a Covering Index available for the
-- following query, but the Query Planner
-- produces a "Bitmap Index Scan" operation.
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT Value FROM VisibilityMap
WHERE Value = 'row 5000';

-- Let's run a VACUUM operation that generates
-- the Visibility Map
VACUUM VisibilityMap;

-- Rerun the query.
-- Now, we have the "Index-Only Scan"
-- operation. The "Heap Fetches" are 0.
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT Value FROM VisibilityMap
WHERE Value = 'row 5000';

-- Update a record
UPDATE VisibilityMap SET Value = 'updated'
WHERE Id = 5000;

-- Rerun the query.
-- The "Heap Fetches" are now 1.
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT Value FROM VisibilityMap
WHERE Value = 'updated';

-- Run the VACUUM operation again
VACUUM VisibilityMap;

-- Rerun the query.
-- The "Heap Fetches" are now 0.
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT Value FROM VisibilityMap
WHERE Value = 'updated';
```



## ● Listing 8: VACUUM-Operationen

```
-- Create a larger table
CREATE TABLE LargerTable
(
    ID INT GENERATED ALWAYS AS IDENTITY
        PRIMARY KEY NOT NULL,
    Number INT
);

-- Insert 10m records
INSERT INTO LargerTable (Number)
SELECT generate_series(1, 10000000);

-- The table has a size of around 350 MB.
SELECT
    pg_size_pretty(pg_relation_size
        ('largertable')),
    relname,
    reltuples,
    relpages, *
FROM pg_class WHERE relname = 'largertable';

-- Update every record in the table, and rollback
-- the transaction. This doubles the table size,
-- because of the additional generated row
-- versions.
BEGIN TRANSACTION;
UPDATE LargerTable SET Number = Number + 1;

-- The rollback is fast, because nothing happens
-- on the pages.
ROLLBACK;

-- We have now the double table size -
-- around 700 MB!
SELECT
    pg_size_pretty(pg_relation_size
        ('largertable')),
    relname,
    reltuples,
    relpages, *
FROM pg_class WHERE relname = 'largertable';

-- Runs the VACUUM command.
-- This removes the dead, old row versions, but
-- the pages will be not rewritten.
-- The number of pages and the size of the table
-- remains the same. We just have some free
-- space in the table, when new records must be
-- inserted.
VACUUM VERBOSE LargerTable;

-- Now we have the original size of around
-- 350 MB.
SELECT
    pg_size_pretty(pg_relation_size
        ('largertable')),
    relname,
    reltuples,
    relpages, *
FROM pg_class WHERE relname = 'largertable';

-- Runs the VACUUM command.
-- This removes the dead, old row version, and
-- rewrites the whole pages.
-- Therefore, the table will have its initial size.
VACUUM FULL VERBOSE LargerTable;

-- Shows the size of a given table
SELECT
    pg_size_pretty(pg_relation_size
        ('largertable')),
    relname,
    reltuples,
    relpages, *
FROM pg_class WHERE relname = 'largertable';
```

Die Visibility Map ist auch nicht feingranular, da sie auf Page-Ebene arbeitet. Wenn auch nur ein einzelner Datensatz in einer Page nicht sichtbar ist, wird die gesamte Page in der Visibility Map als nicht sichtbar markiert.

**Listing 7** zeigt ein Beispiel dafür, wie *VACUUM*-Operationen die Visibility Map aktualisieren und damit Index-only-Scans beeinflussen.

## VACUUM-Operationen

PostgreSQL kann dank der Datensatzversionen parallele Schreib- und Lesezugriffe auf Datensätze ohne Zuhilfenahme von Sperren erlauben und somit die Concurrency von gleichzeitigen Transaktionen erheblich verbessern. Der

Nachteil dieses Concurrency-Modells: Alte, nicht mehr benötigte Datensatzversionen müssen regelmäßig entfernt werden, damit diese keinen physischen Festplattenspeicher belegen und die Performance nicht beeinträchtigen. Diese Aufgabe übernimmt der *VACUUM*-Mechanismus.

In PostgreSQL wird jede Änderung eines Datensatzes nicht durch Überschreiben durchgeführt, sondern durch Anlegen einer neuen Datensatzversion. Die alte Version bleibt bestehen, bis sie von keiner aktiven Transaktion mehr benötigt wird. Gelöscht werden die alten Datensatzversionen erst später von *VACUUM*-Operationen.

Ohne regelmäßige *VACUUM*-Operationen würde die Datenbank mit der Zeit immer größer werden. Indizes würden

auf alte Datensatzversionen zeigen, Scans würden langsamer werden und der Speicherbedarf würde unentwegt wachsen.

PostgreSQL bietet folgende *VACUUM*-Operationen an:

- *Standard-VACUUM* entfernt alte Datensatzversionen aus den Pages und aktualisiert die Visibility Map. Es entsteht freier Speicher auf Pages, die Heap Table wird jedoch nicht kleiner.
- *VACUUM FULL* erstellt eine Heap Table von Grund auf neu, wodurch nicht benötigter Speicher an das Betriebssystem zurückgegeben wird.
- *Autovacuum* ist ein Hintergrundprozess, der Heap Tables automatisch überwacht und *VACUUM*-Operationen bei Bedarf anstößt.

Das Beispiel in **Listing 8** zeigt, wie ein einfaches *UPDATE*-Statement gegen eine Tabelle die Tabellengröße verdoppeln kann, da für jeden geänderten Datensatz eine neue Datensatzversion erzeugt werden muss. Daher ist es sehr wichtig, dass regelmäßige *VACUUM*-Operationen durchgeführt werden – was jedoch der stets aktive Hintergrundprozess *Autovacuum* gewährleistet.

### Fazit

Dieser Artikel hat das Modell Multi-Version Concurrency Control (MVCC) von PostgreSQL vorgestellt, das gleichzeitige lesende und schreibende Transaktionen erlaubt, ohne

dass dafür auf Sperren zurückgegriffen werden muss. Dadurch kann die Performance von Transaktionen deutlich erhöht werden – im Vergleich zum Pessimistic-Concurrency-Modell, das vom SQL Server verwendet wird.

Der nächste PostgreSQL-Artikel wird sich mit dem Write-Ahead Logging von PostgreSQL beschäftigen und es mit dem Transaction Log des SQL Servers vergleichen. Zusätzlich geht es um Backup- und Restore-Operationen und darum, wie Sie Point-in-Time-Recovery-Operationen in PostgreSQL durchführen. ■

[1] Klaus Aschenbrenner, *PostgreSQL, Performance Tuning*, dotnetpro 4-5/2025, Seite 112 ff.

[2] *Transaction Wraparound*,  
[www.postgresql.org/docs/17/routine-vacuuming.html#VACUUM-FOR-WRAPAROUND](http://www.postgresql.org/docs/17/routine-vacuuming.html#VACUUM-FOR-WRAPAROUND)



### Klaus Aschenbrenner

berät Unternehmen in Europa beim Einsatz des Microsoft SQL Server und beschäftigt sich mit Windows-Programmierung und .NET. Zweimal zeichnete in Microsoft für sein Engagement als Microsoft MVP aus. Er ist zu erreichen über [www.SQLpassion.at](http://www.SQLpassion.at).

# DWX ACADEMY

Hands-on Workshops  
und Weiterbildung für  
Software-Entwickler  
und -Architekten

3 TAGE  
INHOUSE ODER  
REMOTE

## EINFÜHRUNG IN SQL

- Daten abfragen
- Daten löschen
- Daten einfügen
- Optimierungen
- Daten modifizieren
- Transaktionssteuerung



THORSTEN KANSY



Deine Ansprechpartnerin: Susanne Herl  
+49 731 88005-8835 • [susanne.herl@ebnermedia.de](mailto:susanne.herl@ebnermedia.de)

[developer-world.de/dwx-academy](http://developer-world.de/dwx-academy)

## TIME-OUT

# (K)eine Frage der Schuld

Kritik am Arbeitsplatz sollte niemals persönlich sein. Oder doch? Verhindert eine Fokussierung auf systemische Ursachen die persönliche Entwicklung?



*Time-out Start*

Sie haben der in agilen Softwareentwicklung längst Kultstatus erreicht: die Retrospektiven. Am Ende jedes Sprints versammeln sich Entwickler, um gemeinsam Fehler im Prozess zu suchen und das Teamgefühl zu stärken. Warum konnte dieser Issue nicht fertig gemacht werden? Wie konnte es dazu kommen, dass wichtige Dinge nicht angegangen wurden? Und dann wird beraten, und als Lösung wird die Farbe des Labels „Urgent“ von Orange auf Rot gesetzt. Und das soll die Lösung sein? Wirklich? Besteht hier nicht die Gefahr, dass mit der stillen Annahme „Jeder tut immer sein Bestes“ individuelle Verantwortung hinter dem Deckmantel kollektiver Selbstoptimierung verschwindet? Dass auch ein rotes Label die Mannschaft nicht dazu bringt, unliebsame Dinge anzugehen, statt sich davor wegzuducken?

Ich will einmal eine gewagte These aufstellen: Wenn wir retrospektiv ausschließlich nach systemischen Ursachen forschen, senden wir ungewollt das Signal, dass persönliche Beiträge und Versäumnisse ohnehin keine Rolle spielen. Statt kritischer Selbstreflexion entsteht eine „Wir gegen das System“-Mentalität, bei der der Einzelne sich in Sicherheit wiegt und niemand mehr wirklich zur Rechenschaft gezogen wird. Für Softwareteams, in denen klare Verantwortlichkeiten und schnelles Feedback essenziell sind, ist dieser Kuschkurs nicht nur zu weich – er untergräbt Agilität, Motivation und den echten Willen zur Verbesserung.

Dabei hat der Ansatz „Jeder tut immer sein Bestes“ durchaus seine Berechtigung. Er kann wie ein Brennglas wirken, das aufdeckt, wie es überhaupt zu einem solchen Verhalten gekommen ist. Denn dass Retrospektiven darauf abzielen, den Prozess, das Umfeld, die Erwartungen und die Arbeitsweise in den Blick zu nehmen, ist hilfreich. Es gab ja durchaus diese „patriarchalischen“ Zeiten, in denen man einfach auf einseitige Schuldzuweisungen setzte, statt die wahren Ursachen von Fehlentwicklungen aufzudecken. Mikromanagement und das Abschieben von Verantwortung auf einzelne Teammitglieder waren an der Tagesordnung: Entwickler wurden eng getaktet, ihre Arbeitsschritte bis ins Detail überwacht und jeder vermeintliche Fehler sofort zum Anlass

genommen, den Finger zu heben. Die Folgen waren nicht nur ein dramatischer Verlust an Kreativität, sondern auch ein Anstieg von Fluktuation und Krankheitstagen. Wer sich permanent kontrolliert und entmündigt fühlt, verliert nicht nur die Freude an der eigenen Tätigkeit, sondern spürt förmlich, wie Vertrauen und Sinnhaftigkeit schwinden. In den Teams entstanden Gräben zwischen dem Management mit einem hohen Ergebnisdruck und denjenigen, die tagtäglich mit Coding und Tests für Sicherheit, Stabilität und Nachhaltigkeit sorgen wollten – ein Nährboden für Frust und Demotivation.

Weil man daraus gelernt hat und Nachhaltigkeit immer wichtiger wurde, wurden Retrospektiven entworfen und ein systemischer Ansatz gewählt, ein Ansatz, der das Team mit nach einer Lösung suchen lässt und es in die Regeln und Arbeitsweisen einbindet.

## Schuldig oder nicht schuldig?

Bevor wir uns aber in Allgemeinplätzen verlieren, würde ich gerne den Begriff Schuld etwas tiefer beleuchten, denn er betrifft uns existenziell, und Schuldgefühle oder Schuldzuweisungen sind ein tief sitzender Motor für unser Verhalten. Mehr noch als Regeln und Anweisungen. Und doch hängt beides zusammen.

Schuld hat für mich zwei Aspekte: einen persönlichen (ich fühle mich schuldig, weil ich meinen Erwartungen nicht gerecht geworden bin) und einen öffentlichen (der Mörder ist schuldig, weil er eine klare Regel der Gemeinschaft gebrochen hat). Ob sich ein Mörder tatsächlich persönlich schuldig fühlt, steht auf einem anderen Blatt, und um es vorwegzunehmen: Er tut es in der Regel nicht.

Der öffentliche Aspekt zerstört immer Beziehungen und fordert Wiedergutmachung, denn ohne diese Wiedergutmachung und das Festhalten an Fehlverhalten und Schuld könnte ein Team nicht innerhalb seiner Regeln weiter existieren. Das Vertrauen der Beteiligten muss wiederhergestellt werden. Regeln sind immer auch die Regeln der „Mächtigen“, insofern hat Nietzsche recht, wenn er von Schuld als einem Instrument der Macht spricht. Das kann die Macht des Ge-

schäftsführers, Teamleiters oder Inhabers in einer Firma sein. Oder eben die Macht des Teams, das einen Konsens in einem gewaltfreien Diskurs aller Beteiligten (siehe Habermas) herstellt, also gemeinsam Regeln und Standards entwirft.

Hat jetzt ein Mitarbeiter etwas getan, das nicht den Regeln, Erwartungen oder Standards des Teams oder Vorgesetzten entspricht, dann ist er schuldig im Sinne des Teams. Ob er selbst sich als schuldig empfindet, hängt dann stark davon ab, ob das besagte Verhalten von seinen eigenen Regeln, Standards und Erwartungen abweicht oder mit diesen konform geht. Und so ist eine Schuldzuweisung, die auf kein Schuldgefühl stößt, Symptom eines tiefer liegenden Problems: eine Diskrepanz zwischen Eigenerwartung und Fremderwartung, zwischen Eigenbild und Fremdbild und ein Zeichen für mangelhafte Kommunikation. Schuldzuweisungen ohne Schuldgefühle sind demotivierend und machen das Team auf Dauer krank.

Jeder Softwareentwickler baut über Jahre hinweg Kompetenzen, Werte und emotionale Strategien auf. Erfolge bestärken das Gefühl von Selbstwirksamkeit, Misserfolge hingegen nähren Zweifel und Selbstvorwürfe. Wenn nun Schuldgefühle aufkommen, können sie das ohnehin fragile Vertrauen in die eigene Kompetenz zunichtemachen. Unterschiedliche Erwartungshaltungen spielen dabei eine große Rolle: Was der Product Owner als „Dringlichkeit“ definiert, mag für den Entwickler nach Hau-ruck-Arbeit aussehen; für das Management sind Codequalität und Nachhaltigkeit schwer messbar. Die Kluft zwischen „Was ich leisten soll“ und „Wie ich es leisten soll“ weckt Schuldgefühle – mit der fatalen Botschaft: „Ich habe versagt.“

Auf der anderen Seite kann der Wunsch nach Codequalität zu einem Over-Engineering führen und die Firma als Ganzes gefährden, weil Abgabetermine gerissen werden. Mitarbeiter sollten das Vertrauen haben, dass Prioritätensetzungen des Product Owners Sinn ergeben und der Firma dienen. Treffen sie selbst die Entscheidung, was wichtig ist und was nicht, kommt es zu Konflikten.

Um dieser Dynamik entgegenzuwirken, müssen wir Transparenz schaffen und gemeinsame Erwartungsbilder entwerfen. Das beginnt bereits in der Planung: Kollegen sollten klare, realistische Zielvereinbarungen treffen, die technische Schulden ebenso berücksichtigen wie enge Release-Zwänge des Produktmanagements. Fühlen sich alle Beteiligten mitgenommen, verkleinert sich das Spannungsfeld, in dem Schuld entstehen kann. Auch eine Kultur des offenen Feedbacks – losgelöst von persönlichem Angriff – trägt dazu bei, frühzeitig Differenzen zu klären und Frust abzubauen.

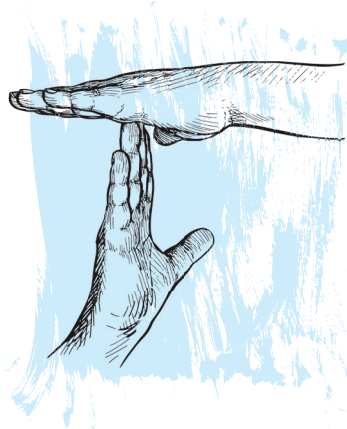
Wer schuldzuweisend konfrontiert wird, empfindet Misstrauen und Distanz – die Zusammenarbeit erstarrt. Vertrauen wiederherzustellen erfordert Mut zur Verletzlichkeit und ehrliche Dialoge. Ein Teamleiter könnte etwa zugeben: „Ich habe in der letzten Sprintplanung zu wenig Rücksicht auf eure Kapazitäten genommen. Das hat den Druck erhöht, und dafür übernehme ich die Verantwortung.“ Solche Gesten wir-

ken stärker als formelle Entschuldigungen, weil sie persönliche Bindung schaffen und den Weg für echte Wiedergutmachung ebnen.

Zudem helfen ritualisierte Wiedergutmachungsprozesse: Ein kurzes Commitment-Meeting, in dem Ursache, Auswirkungen und Maßnahmen klar benannt werden, ersetzt diffuse Schuldgefühle durch konstruktives Handeln. Wenn anschließend jeder Beteiligte – auch der vermeintlich Schuldige – aktiv an der Lösung mitarbeitet, entsteht ein neues Wir-Gefühl. Die emotionale Steuerung verlagert sich von „Ich habe versagt“ zu „Wir haben eine Chance, gemeinsam zu wachsen“.

Wenn es trotz dieser Maßnahmen dauerhaft zu Schuldzuweisungen kommt, die als ungerecht empfunden werden, kann es sein, dass die Erwartungshaltung des Mitarbeiters und die der Firma nicht kompatibel sind. Oder dass eine dauerhafte Diskrepanz zwischen Eigenwahrnehmung und Fremdwahrnehmung pathologische Ursachen hat. Es kann auch sein, dass die Erwartungen an den Mitarbeiter dessen Fähigkeiten übersteigen. Wie auch immer, klar sollte sein, dass nicht alle Maßnahmen immer zum Erfolg führen, dass es manchmal einfach der falsche Mann auf dem falschen Posten ist. Dann hilft letztlich nur die Trennung.

Wenn aber der Mitarbeiter ins Team passt, dann geht es um die bewusste Balance: Systemische Ursachen aufdecken und beseitigen, ohne die persönliche Verantwortung zu entwerthen. Indem wir Schuld nicht verteufeln, sondern sie als Signal verstehen, können wir sowohl Prozesse optimieren als auch die individuelle Entwicklung stärken. So entsteht ein agiles Arbeitsumfeld, in dem Softwareentwickler nicht nur mit Samthandschuhen behandelt werden, sondern mit klaren Erwartungen, ehrlichem Feedback und gesundem Vertrauen – die beste Basis für nachhaltige Leistungsbereitschaft und Innovationskraft. ■



*Time-out End*



**Dipl.-Theol. Bernhard Pichler**

war 20 Jahre Geschäftsführer der Softwarefirma informare, bevor er 2020 zum Leiter Entwicklung und Produktmanagement der DPS BS ernannt wurde. Die DPS BS ist Deutschlands größter Softwarepartner für betriebswirtschaftliche Software für den Mittelstand.



## SOFTWAREENTWICKLUNG IN KRISENZEITEN

# The Mindful Software Developer

Achtsamkeit als unverzichtbarer Beitrag für das Gelingen der agilen Transformation.

**M**indfulness war vor mehr als zehn Jahren ein angesagtes Thema. In einzelnen Softwarekonzernen ist es fest etabliert, aber in der Fläche vielfach nicht angekommen. Warum sollte man heute wieder darüber sprechen? Weil von Krisen erschöpfte Firmen an der agilen Transformation scheitern und manche Vorgesetzte ratlos auf autoritäre Führung setzen.

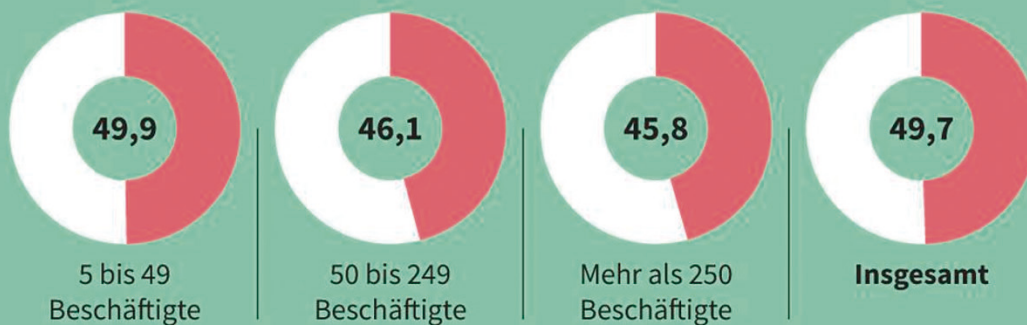
Wir leben in turbulenten Zeiten. Das ist wohl nicht übertrieben. Allein in den vergangenen fünf Jahren schienen die großen Herausforderungen Schlange zu stehen. Pandemie, bewaffnete Konflikte, politische Krisen, unsichere Lieferket-

se Verhaltensweisen haben eines gemeinsam: Sie sind (auch) der Versuch, sich der Komplexität zu entziehen, indem sie eine einfache Antwort geben. Doch komplexe Fragestellungen brauchen eine komplexe Antwort. Diese Antwort kann man durch Kooperation eher finden als durch Instruktion. Für die Softwareentwicklung, eine kreative, schöpferische Arbeit, gilt das erst recht.

Gleichzeitig ist Software – oder genauer: die Fähigkeit, sie zu entwickeln – eine Schlüsselkompetenz für unsere Wirtschaft. Software ist längst auch Bestandteil von Produkten, die

## Die Führungslücke

*So viel Prozent der Unternehmen dieser Größe in Deutschland fällt es zunehmend schwer, Führungspositionen zu besetzen, weil Beschäftigte daran seltener Interesse haben*



Befragung von Personalverantwortlichen in 1.029 Unternehmen im Rahmen des IW-Personalpanels von März bis Mai 2023

**Während immer weniger Beschäftigte Interesse an einer Führungsposition haben, wird autoritäre Führung wieder nachgefragt. Die Herausforderungen der aktuellen Zeit verlangen eher nach kooperativer Führung (Bild 1)**

ten, wirtschaftliche Konflikte, Fachkräftemangel, um nur ein paar zu nennen. Die deutsche Wirtschaft steht im OECD-Vergleich nicht gut da [1][2]. Vor diesem Hintergrund sind viele Firmen gestresst, sowohl auf Organisationsebene als auch auf der individuellen Ebene. Seit Jahren steigt die Komplexität, in der sich Unternehmen bewegen, stetig an. Das führt bei Menschen und Firmen zu Stress und Ungewissheit. Wie gehen sie damit um? Unter den vielen möglichen Reaktionen beobachte ich Dienst nach Vorschrift, eine „Wir machen es so wie beim letzten Mal“-Haltung und autoritäre Führung. Die-

gar nicht als digital wahrgenommen werden, und so stehen Unternehmen, denen die digitale Denke nicht in die Wiege gelegt wurde – und das sind die meisten –, vor der Herkulesaufgabe, in schwierigen Zeiten eine Fähigkeit zu erlernen oder zu verbessern, die nicht zu ihren Kernkompetenzen gehört.

Und damit landen wir bei dem sensiblen Thema der Agilen Transformation. Skeptiker können einwenden, dass Agilität wahlweise ein alter Hut oder gescheitert sei. Wenn „alter Hut“ bedeuten soll, dass wir alle doch längst agil transformiert sind, ist dies schlicht falsch. Viele Firmen, die sich agil

nennen, sind so flexibel wie eine Bahnschwelle. Am Scheitern hingegen ist etwas dran. Doch nicht die Agilität ist gescheitert, sondern so manche Firma, die es versucht hat. Die Gründe sind vielfältig und hier nicht das Thema. Um einem Streit aus dem Weg zu gehen, nehmen wir einfach mal an, Agilität sei tatsächlich gescheitert. Was folgt daraus? Ein „zurück zum schlecht gemanagten Wasserfall“ kann ja wohl keine Lösung sein. An einer plangetriebenen Vorgehensweise gibt es bei einem gut planbaren Projekt nichts zu bemängeln; es sei denn, man setzt das Vorhaben unprofessionell um. Also genau wie bei der agilen Vorgehensweise.

## Was braucht Softwareentwicklung in Krisenzeiten?

Wie geht es weiter? Wir befinden uns nun in einer äußerst komplexen Situation (siehe die oben genannten Krisen). Die wirtschaftliche Lage scheint uns mehr Challenges als Opportunities zu geben. Welcher Weg führt uns hier heraus? Ein starker Anführer, der von oben nach unten durchregiert, klingt für viele nicht schlecht [3]. Zu dumm nur, dass komplexe Probleme eher danach verlangen, zu experimentieren und die Kompetenz und Innovationsfähigkeit der ganzen Gruppe anzuzapfen, als sich auf die Ideen nur einer Person zu verlas-

sen. Ganz nah an die Perfektion gebracht werden, bevor sie den Markt erreichen. Nehmen wir an, die Firma ist einer dieser Hidden Champions, dem jetzt Krisen und Herausforderungen begegnen. Früher war die eigene Fertigungsqualität ein Bollwerk und deklassierte die Konkurrenz. In nur wenigen Jahren hat der Wettbewerb aus Fernost gelernt, eine Qualität zu liefern, die kaum noch von der eigenen zu unterscheiden ist. Das sorgt nicht nur für Absatzprobleme, es nagt auch am eigenen Wertgefühl. Und die Geschwindigkeit, in der die Konkurrenz den Anschluss hergestellt hat, ist Auslöser für bange Blicke in die Zukunft. Darüber hinaus sorgen schwächelnde Lieferketten sowohl für Unsicherheit als auch für unerwartete Aufwände. Nicht zuletzt verursacht der Fachkräftemangel weitere Sorgen über die Zukunft des Unternehmens.

Dem Champion stehen zwar die Schweißperlen auf der Stirn, aber er ist handlungsfähig. Software spielt in den Produkten eine immer wichtigere Rolle. Jetzt muss eine neue Vorgehensweise her, um Innovation hervorzubringen, Marktrisiken zu senken, schneller mit neuen Produkten auf den Markt zu kommen und näher am Kunden zu entwickeln. Die Unternehmensleitung hat viel von Agilität gehört und dass diese Vorgehensweise dabei helfe, ein paar der Risiken in der

Bei SAP zeichnet ein **Chief Mindfulness Officer** für die SAP Global Mindfulness Practice verantwortlich. Bereits vor rund zehn Jahren konnte der Walldorfer Softwarekonzern mit messbaren Erfolgen aufwarten, inklusive eines Return on Investment von 200 Prozent (Bild 2)

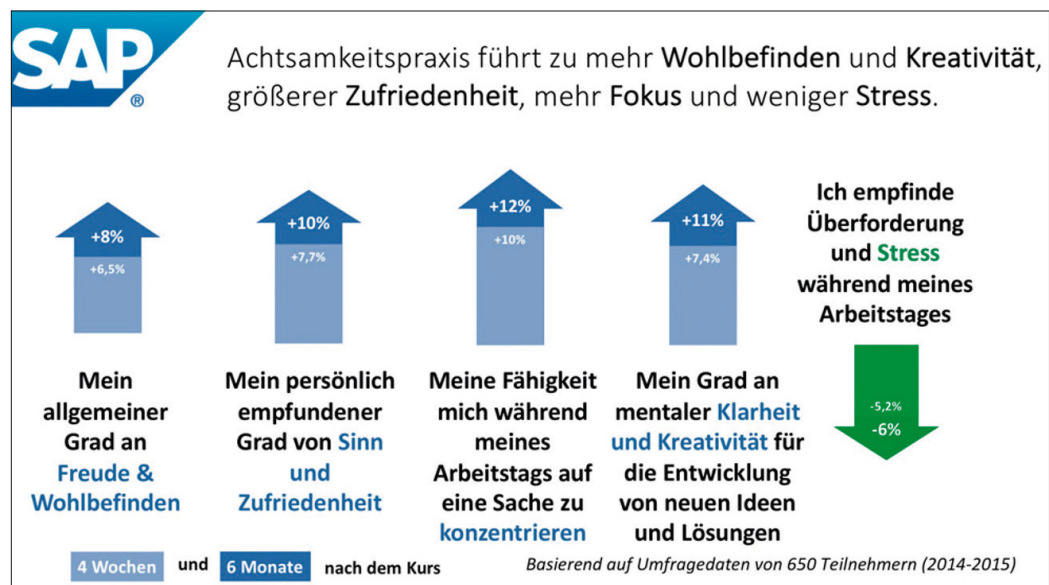


Bild: Kulturwandel.org [14]

sen. Wer Befehle erteilt, bekommt schnell „Dienst nach Vorschrift“ (Bild 1). Wir brauchen stattdessen crossfunktionale Teams, die in der richtigen Unternehmenskultur schnell Prototypen und Iterationen entwickeln, die man mit stetigem Nutzerfeedback zur Marktreife bringt. Das kann man nennen, wie man will. Ich nenne es agil.

Was passiert denn bei einer agilen Transformation, und wie läuft sie ab, wenn man aus der Vogelperspektive zuschaut? Stellen wir uns dazu exemplarisch eine grundsätzliche Firma vor, die ihre Kunden auf der ganzen Welt seit Jahrzehnten mit herausragend engineernden Produkten beliefert. Produkte, die in langen Entwicklungszyklen mit hoher Ingenieurskunst

Innovationsentwicklung zu reduzieren. Also kauft man ein wenig „Agilität“. Zuerst holt man einen Berater, der bei der Agilen Transformation hilft. Wie läuft diese Transformation, diese Verwandlung ab?

Um meinen Punkt deutlich zu machen, erlaube ich mir, stark zu verkürzen und ein wenig zu karikieren. Dann wird nämlich die Agile Transformation zu einer recht einfachen Sache: Der Champion führt ein paar neue Rollen ein, etwa Scrum Master, Product Owner, Release Train Engineer und Ähnliches mehr. Dazu ein neuer Entwicklungsprozess, der sich um einige neue Meetings wie Planning, PI Planning und Review entwickelt. Je nach eingesetztem Framework ►

schult man alle Mitarbeiter einmalig oder periodisch für den Rest ihres Lebens. Fertig ist die agile Transformation.

### Die Kultur macht den Unterschied

Doch halt! Irgendwas stimmt nicht. Da gibt es immer wieder Stimmen, die von Cargo Cult sprechen oder von Pseudo-Agilität. Eine Illusion von Agilität also? „Was fehlt denn noch, wenn wir doch alles nach Lehrbuch machen?“ Jetzt kommt der schmerzhafteste Teil. Es fehlt das richtige Mindset. Die richtige Haltung. Es fehlen die richtigen Glaubenssätze. Doch was soll das genau heißen? Wie kann unser Hidden Champion das verstehen? Ein Unternehmen, dessen wirtschaftliche Stärke im 20. Jahrhundert begründet wurde. Mit einer klaren Führung von oben nach unten. Mit klaren Zuständigkeiten. Mit sauber geschnittenen Abteilungen. Mit wohldefinierten Prozessen, in denen sich Arbeitsschritte an Quality Gates begegnen. Mit sorgfältig entwickelten Produkten, deren Qualität die Konkurrenz erlebigen lässt.

Das, was der Agility-Coach nun als „das richtige Mindset“ darstellt, widerspricht allem, was den Champion groß gemacht hat. Nun heißt es, man brauche mehr laterale und weniger vertikale Führung, Teamverantwortung statt individueller Zuständigkeit, Aufbrechen von Silos statt wohldefinierter Abteilungen, iterative MVPs statt makelloser Produkte. Eine „emerging practice“ statt einer „best practice“.

Nur noch das Mindset ändern? Nur noch? Also ob das eine Kleinigkeit wäre. Wie schwer fällt es jedem von uns, eine Grundüberzeugung, einen Glaubenssatz zu hinterfragen und einen neuen anzunehmen? Das ist schwer. Wenn wir das als Individuum versuchen, müssen wir Erlerntes verlernen und etwas Neues annehmen. Das bedeutet Unsicherheit, Orientierungslosigkeit, Ungewissheit, Stress und Angst.

Und diese Herausforderung übergeben wir nun dem so wieso schon erschöpften Unternehmen. Das kann nicht klappen. An diesem Punkt scheitert die Agile Transformation, denn sie bringt eine neue Belastung, einen neuen Stress mit sich. Es stellt sich die alles entscheidende Frage: Wie kann man es trotzdem schaffen?

### Energie vor Transformation

Die Grundidee ist simpel: Im Idealfall muss das Unternehmen aus seinem Erschöpfungszustand heraus, bevor es sich an irgendeine Art von Transformation wagt. Diese Idee umzusetzen ist aber nicht leicht. Wer verstehen will, an welchen Punkten man ansetzen kann – und wozu –, muss zuerst verstehen, wie Stress unsere Arbeit negativ beeinflusst. Vor allem in der Softwareentwicklung, in der Arbeit in skalierten Teams mit Beteiligten aus verschiedenen Abteilungen der Normalfall ist.

Stress verändert unseren Blick auf die Welt und wie wir Informationen verarbeiten. Er schränkt unsere Aufmerksamkeit ein und senkt unsere Kreativität und Produktivität. Im gestressten Zustand erinnern wir uns nicht mehr so gut an Informationen. Stress verschlechtert unsere Fähigkeit, mit anderen Menschen zu interagieren. Die Bereitschaft, offene Gespräche zu führen, geht zurück. In Retrospektiven oder in Refinements mit Business-Stakeholdern brauchen wir aber

genau das. Eine Agile Transformation basiert auf Interaktion, Offenheit und siloübergreifendem Austausch. Im gestressten Zustand halten uns andere Personen für weniger vertrauenswürdig, und deshalb sinkt die Wahrscheinlichkeit für Kollaboration. Wir neigen dazu, im Fehlerfall andere Personen zu beschuldigen, und wir sehen nicht so leicht über Spannungen im Team hinweg. Damit breitet sich der Stress eines Einzelnen im Team aus und senkt dessen Produktivität. In der Wissenschaft gilt es als belegt, dass gestresste Organisationen weniger agil sind. Sie sind nicht in der Lage, über den Tellerrand zu schauen und sich an verschiedene Kontexte und verändernde Märkte anzupassen [4]. Zusammengefasst



**Achtsamkeitsübungen** versprechen eine Reihe von positiven Effekten, die in der Softwareentwicklung insbesondere die agile Transformation unterstützen können (Bild 3)

lässt sich sagen, dass die Auswirkungen von Stress auf die Fähigkeit einer Firma, auf Veränderungen und Ungewissheit in der notwendigen Weise reagieren zu können, sehr negativ sind. Manager, die ihre Firma resilient für Krisen machen möchten, sind äußerst gut beraten, sich um das Wohlbefinden der Mitarbeiterinnen und Mitarbeiter zu kümmern.

Umgekehrt ist der Stress am Arbeitsplatz auch negativ für die Mitarbeiterinnen und Mitarbeiter; das ist schon fast eine Binsenweisheit. Deshalb hier nur ein erstaunlicher Vergleich,

der die Auswirkungen von Passivrauchen und Arbeitsstress vergleicht. Dass Passivrauchen schädlich ist, ist bekannt. So weist etwa das Ministerium für Arbeit, Gesundheit und Soziales des Bundeslandes Nordrhein-Westfalen auf seiner Webseite darauf hin, dass langjähriges Passivrauchen am Arbeitsplatz das Risiko, an Lungenkrebs zu erkranken, um rund 60 Prozent erhöht [5]. Wikipedia nennt für Deutschland die Zahl von 3300 Menschen, die pro Jahr an den Folgen des Passivrauchens sterben [6]. Dabei bezieht sich die Online-Enzyklopädie auf eine Studie des Deutschen Krebsforschungszentrums (DKFZ) aus dem Jahr 2005. Seither wurde der Schutz vor Passivrauchen punktuell verbessert.

Wie verhält es sich mit den Auswirkungen von Stress am Arbeitsplatz? Eine Metastudie vergleicht die Effekte mit denen des Passivrauchens und kommt hier zu folgenden Ergebnissen [7]: Die befragten Studienteilnehmer nehmen hohe berufliche Anforderungen und geringen Einfluss auf ihre Arbeit als nahezu genauso negativ für ihre körperliche Gesundheit wahr wie das Passivrauchen. Bei der mentalen Gesundheit stufen sie Ungerechtigkeit am Arbeitsplatz sogar als schädlicher ein. Wie sieht es aus, wenn man nicht nach subjektivem Empfinden fragt, sondern auf Daten des Gesundheitssystems zurückgreift? Dann bleibt die Aussage die gleiche: Ungerechtigkeit am Arbeitsplatz und hohe berufliche Anforderungen verursachen mehr Erkrankungen als das Passivrauchen; einschränkend sei gesagt, dass die Zahl der betrachteten Studien für die Ungerechtigkeit niedrig ist. Blickt man nicht auf Erkrankungen, sondern auf Sterblichkeit, sind die Auswirkungen von geringen Einflussmöglichkeiten auf die Arbeit größer als beim Passivrauchen. Kurz gesagt: Wer am Arbeitsplatz schlecht behandelt wird, nicht erreichbare Termine vorgesetzt bekommt und wenig Mitspracherecht über seine eigene Arbeit hat, wird häufiger krank und stirbt früher.

Für beide Seiten – Arbeitgeber ebenso wie Arbeitnehmerinnen und Arbeitnehmer – sollten diese Erkenntnisse genug Anlass geben, zu handeln. Viele Handlungsoptionen sind weithin bekannt. Eine (ordentlich umgesetzte) agile Arbeitsweise oder vergleichbare Maßnahmen geben den Mitarbeitenden zum Beispiel mehr Gestaltungsspielraum. Das ist für die individuelle Gesundheit hilfreich und für die Fähigkeit des Unternehmens, mit komplexen Veränderungen umzugehen, unverzichtbar. Weitere Erkenntnisse über Happiness at Work hat bereits ein vorangegangener Artikel dargestellt [8].

Was kann man nun gegen die geschilderten Probleme unternehmen?

### Maßnahmen gegen Firmen-Burn-out

In den vergangenen Jahren haben einige Firmen gute Erfolge bei der Stressbewältigung erzielt, indem sie begonnen haben, Mindfulness-Programme anzubieten. Es gibt Berichte von SAP, Bosch, Google, Intel und anderen Global Playern (siehe Bild 2). Mindfulness wird im Deutschen meist mit Achtsamkeit übersetzt. In der breiten Öffentlichkeit wird der Begriff oft in die Schublade der unwissenschaftlichen Esoterik eingeordnet. Das macht es nicht leicht, Mindfulness in erschöpften Unternehmen zu pitchten. Tatsächlich gibt es aber einige wissenschaftliche Untersuchungen von Mindfulness-

### ● STOP

Hinter dem Akronym STOP verbirgt sich eine sehr einfache Mindfulness-Methode, die vor allem bei spontanen Spannungen nützlich ist. Als Beispiel stelle man sich vor, dass in einer Besprechung in scharfen Worten die Qualitätsmängel und die hohe Zahl an Bugs in einer Software zur Sprache kommen, an der man mitgearbeitet hat. Eine impulsive Reaktion könnte sein, in ebenso scharfen Worten die eigene Kompetenz zu verteidigen.

Wer in einem solchen Moment Ruhe bewahren möchte, kann die folgenden vier Schritte von STOP verwenden:

**S** – Stop: Bedeutet innehalten, etwas Zeit verstreichen lassen.

**T** – Take a deep breath: Tief ein- und ausatmen.

**O** – Observe: Beobachten, wie es mir selbst geht, welche Gedanken und Emotionen ich habe, wie sich mein Körper anfühlt.

**P** – Proceed: Die besonnene Entscheidung, wie es nun weitergehen soll.

Der Vorteil der Methode liegt zweifelsohne darin, dass sie sehr leicht zu merken und anzuwenden ist. Ihr Nutzen besteht darin, eine unüberlegte Reaktion zu vermeiden und sich Zeit für eine bessere Handlungsweise zu geben.

Maßnahmen im beruflichen Kontext. Auch wenn die Studienlage unzureichend ist, gilt die Kernaussage, dass Mindfulness-Maßnahmen wirksam sind, gemeinhin als akzeptiert.

Doch was ist denn Mindfulness? Wir sprechen von einer inneren Haltung, was es so schwer macht, jemandem den Zustand zu beschreiben, der ihn selbst nicht kennt. Die folgenden zwei Charakterisierungen vermitteln eine erste Idee:

- Mindfulness bei der Arbeit bedeutet, dass wir unsere Gedanken, Gefühle, Körperempfindungen und unsere Umgebung bei der Arbeit mit einem wohlwollenden Blick wahrnehmen.
- Mindfulness ist das Bewusstsein, das entsteht, wenn man absichtlich, im gegenwärtigen Moment und ohne zu urteilen, aufmerksam ist.

Erstere Beschreibung stammt von Dr. Emiliana Simon-Thomas von der Uni Berkeley. Sie ist wissenschaftliche Direktorin des Greater Good Science Center an der Uni Berkeley. Das zweite Zitat stammt von Prof. em. Jon Kabat-Zinn, der an der University of Massachusetts Medical School geforscht hat (beides zitiert nach [9]).

Zusammengenommen kann man das wie folgt formulieren: Bei Mindfulness/Achtsamkeit am Arbeitsplatz geht es darum, unsere Gedanken, Gefühle und körperlichen Empfindungen wohlwollend und ohne Urteil wahrzunehmen.

Damit wir mit dem negativen Stress umgehen können, müssen wir uns als Erstes der unmittelbaren Auswirkungen bewusst werden. Unmittelbare Auswirkungen sind Gedanken, Gefühle und körperliche Empfindungen, die während ►



der stressigen Erfahrung oder kurz danach eintreten. Bei dieser Bewusstmachung ist es wichtig, einen wohlwollenden Blick und eine Urteilsfreiheit anzunehmen. Wenn uns also der herrschsüchtige Projektleiter eine sinnlose und ineffiziente Aufgabe gibt, die uns wütend macht, gilt es, die Wut zu erkennen, ohne sie zu verurteilen. So, wie wir auch anderen vielleicht zugestehen würden, in bestimmten Situationen Wut zu verspüren, sollten wir es auch uns erlauben. Dass „herrschsüchtig“, „sinnlos“ und „ineffizient“ unsere eigenen Interpretationen sind, ist zunächst egal. Der erste Schritt der Achtsamkeit besteht eben nur aus dem neutralen In-uns-Hineinhören und -Fühlen.

### ● Kritik an Mindfulness

Mindfulness ist ein riesiges Geschäft. Es gibt dazu unzählige Bücher, Kurse, Apps. Bereits 2015 hat die „Zeit“ geschrieben: „In vielen Konzernen ist Achtsamkeit zu einem Ritual geworden, dem man sich kaum entziehen kann.“ [12] Der zitierte Artikel ist lesenswert und beleuchtet einige der Auswüchse von „Achtsamkeit als Produkt“.

Weitere Kritiker bemängeln, dass Mindfulness aus seinem ursprünglichen, buddhistischen Kontext gerissen wurde. Ein Kontext, in dem es gerade nicht um weltliche Dinge und unbegrenzten Konsum geht. Insofern ist es verständlich, dass Menschen, die der buddhistischen Lehre nahestehen, die stark kommerzialisierte und aus ihrer Sicht banalisierte, säkulare Form der Achtsamkeit mit gemischten Gefühlen betrachten. In der Tat ist Mindfulness zu einem Lifestyle-Element geworden, das uns helfen soll, Gewicht zu verlieren, Ängste zu bekämpfen oder eben – und darum geht es ja auch in diesem Artikel – am Arbeitsplatz produktiver zu sein.

Darf man dann, angesichts der berechtigten Kritik, einen solchen Artikel schreiben? Als Autor erlaube ich mir eine persönliche Antwort: Ich habe vor rund 20 Jahren mit Meditation begonnen. Dabei habe ich mich bewusst für eine (ziemlich unkommerzielle) Meditationslehre entschieden, die keinen religiösen oder spirituellen Bezug hat, sondern mit jeder Weltanschauung vereinbar ist. Für mich war und ist das eine persönliche Bereicherung, die mir im Arbeitsalltag hilft. Die eigene Erfahrung und die subjektive Beobachtung, dass viele Softwareteams zu erschöpft wirken, um echte Verbesserungen in ihrem Arbeitsumfeld zu erzielen, ist mein Impuls für den Artikel. Und vielleicht kann ich damit einen Impuls auch für Firmen geben, mit denen ich nicht selbst arbeite.

Eine Grenze ist für mich überschritten, wenn man Mitarbeiterinnen und Mitarbeitern Kurse in Achtsamkeit und anderen Methoden zur Resilienzverbesserung anbietet oder gar nahelegt, aber die Ursachen für arbeitsbedingten Stress außer Acht lässt. Wenn der individuelle Stress von Mitarbeitenden durch unrealistische Ziele, Entmündigung durch Micromanagement, unprofessionelles, weil abwertendes Feedback und andere kulturelle Defizite entsteht, ist das Angebot eines Achtsamkeitskurses ein zynisches Beispiel von #CultureWashing.

Wie kann man die Fähigkeit zu dieser Form der Achtsamkeit erlernen und verbessern? So schwer es ist, Mindfulness zu charakterisieren, so einfach ist es, einige der Maßnahmen zu beschreiben, mit denen Unternehmen zur Achtsamkeit ihrer Mitarbeiter beitragen wollen. Es handelt sich dabei meist um kurze Meditationsübungen. Manche Unternehmen stellen geeignete Räume und Zeiten zur Verfügung, in denen Mitarbeiter die Übungen unter Anleitung erlernen können. Die Anleitungen sind zum Beispiel gesprochene Instruktionen, die die Teilnehmenden auffordern, auf ihre Atmung zu achten, ihre Gedanken fließen zu lassen oder ihre Aufmerksamkeit auf einen bestimmten Gedanken zu richten (siehe auch Bild 3).

Ein Kurs der Uni Berkeley unterscheidet bei den Übungen folgende Arten:

- **Körperfokussierte Übungen:** Hier geht es zum Beispiel darum, auf die eigene Atmung zu achten, oder Körperempfindungen wie Anspannungen wahrzunehmen.
- **Konzentrationsübungen:** In einer geführten Meditation konzentriert man sich etwa auf die Bilder, die während der Übung ins Bewusstsein geraten. Auch das konzentrierte Zählen von Atemzügen gehört dazu.
- **Achtsamkeit der Emotionen:** Diese Übungen fordern uns auf, unsere Emotionen ohne Wertung zu erkennen und eventuell festzustellen, wo wir eine Emotion im Körper spüren können. Wenn wir beispielsweise erkennen, dass Wut zu einer Anspannung der Schultern führt, können wir uns bei Bedarf für das bewusste Entspannen dieser Körperpartie entscheiden.
- **Achtsamkeit für unsere beruflichen Beziehungen:** Dabei geht es beispielsweise um Übungen, die das Mitgefühl mit Teammitgliedern verbessern sollen.

Für Übungen dieser Art sind jeweils nur fünf bis zehn Minuten pro Tag erforderlich. Darüber hinaus ist es nützlich, ein Repertoire an sehr kurzen Methoden aufzubauen, die dabei helfen, in plötzlich auftretenden Stressmomenten gelassener zu reagieren. Im Kasten **STOP** ist ein solches Methoden-Set beispielhaft erläutert.

Dass Mindfulness für viele Menschen hilfreich ist, kann man wohl aus den unzähligen Ratgebern im Buchhandel ablesen (siehe dazu auch den Kasten **Kritik an Mindfulness**). In diesem Artikel geht es aber nicht um den einzelnen Menschen, sondern um Unternehmen, die Software entwickeln. Was haben diese Firmen davon, wenn die Mitarbeiterinnen und Mitarbeiter solche Übungen ausführen?

Einen Teil der Antwort haben die ersten Absätze zu Herausforderungen, Krisen und (agiler) Transformation bereits vorweggenommen. Um die Frage noch besser zu beantworten, werfen wir einen Blick darauf, was Softwareentwicklung von uns verlangt. Welche Skills brauchen wir?

In der Regel müssen wir in der Lage sein, für uns neue Business-Anforderungen zu verstehen. Wenn wir bei der Umsetzung nicht einfach nur Fließbandarbeit verrichten wollen, müssen wir in der Lage sein, den Sinn hinter manchmal nicht präzisen Anforderungen zu erkennen. So mancher Business Owner denkt, er hätte die Softwareentwicklung so weit ver-

standen, dass er keine Business-Anforderungen formuliert, sondern bereits den größten Teil der technischen Lösung vorgibt. Als Entwickler müssen wir verstehen, was unser Gesprächspartner will und was er braucht; wir müssen zwischen beiden Dingen differenzieren können und unsere Rückfragen im richtigen Ton formulieren, sodass ein offenes Gespräch entstehen kann. In der eigentlichen Programmierung ist es unsere Aufgabe, zwischen erprobten und bekannten Vorgehensweisen auf der einen Seite und neuen Wegen auf der anderen Seite abwägen zu können. Wählen wir eine Architektur, weil sie immer so funktioniert hat, oder weil sie im konkreten Fall wirklich die beste ist? Sind wir offen für radikale Ideen unserer Juniorkollegin, oder fühlen wir uns in unserer Alpha-Position bedroht?

Und dann ist da noch das große Thema „Bugs“. Irgendwas läuft nicht wie gewünscht. Zunächst analysieren wir konzentriert das beobachtete und das erwartete Verhalten. Liegt überhaupt ein Bug vor? Haben wir die Anforderungen der Stakeholder richtig verstanden? Und wenn es unstrittig ist, dass es sich um einen Bug handelt, gibt es diese hartnäckigen Fälle, in denen die Bug-Ursache vollkommen unklar ist. Hier ist maximale „Open-mindedness“ gefragt. Wir dürfen nicht das Gleiche denken wie immer, denn das hat den Bug erst verursacht. Zur Lösungsfindung müssen wir um die Ecke denken.

Mindfulness verspricht, in allen genannten Fällen zu Verbesserungen zu führen:

- Im Gespräch mit dem Stakeholder bleiben wir gelassen und haben ein offenes Ohr für seine Ideen.
- Wir können leichter „out of the box“ denken und damit vielleicht bessere Architekturentscheidungen treffen.
- Regelmäßige Mindfulness-Übungen verringern Konflikte mit Teamkollegen.
- Und bei der Bugsuche und -lösung helfen uns verbesserte kognitive Fähigkeiten.

Des Weiteren sollen die Übungen positive Wirkung auf die Teamzusammenarbeit, Co-Creation, Kreativität und Innovationsfähigkeit besitzen. Mindfulness wird auch eine positive Auswirkung auf Leadership attestiert [10]. Die Art der Führung und die Unternehmenskultur spielen eine Rolle bei der Wirksamkeit von Achtsamkeit. So sorgt eine höhere Mindfulness eines Vorgesetzten für geringere emotionale Erschöpfung und höhere Zufriedenheit im Job bei den Mitarbeitenden. Das gilt aber nur, wenn die organisatorischen und kulturellen Voraussetzungen stimmen. Wenn grundlegende Bedürfnisse wie Autonomie und Verbindung zu anderen Menschen nicht gegeben sind, hilft es auch nichts, wenn der Vorgesetzte fleißig meditiert [11].

Und damit komme ich zu einem wichtigen Punkt des Artikels, den ich als Autor aus der subjektiven Perspektive darstellen möchte: Ich bringe die Themen Mindfulness und Happiness at Work bei unseren Kunden zur Sprache, weil ich darin Verbesserungspotenzial für die Firma sehe. In manchen Unternehmen, vor allem den größeren, bekomme ich zu meiner Überraschung die Antwort, dass man ja entsprechende Programme längst habe, die die Mitarbeiterinnen und Mitar-

beiter nutzen können. Viele Firmen werben mit einer ganzen Reihe von Benefits. Vom Obstkorb über Getränke bis hin zu Zuschüssen zum Kantinenessen, zu Fahrtkosten oder zu Kita-Kosten. In der konzerneigenen Academy gibt es Sprachkurse und Sportprogramme, Yoga und manchmal auch Mindfulness in irgendeiner Art. Die Firmen streichen auch mal eine Auszeichnung, wie den Corporate Health Award, ein.

Das sind unzweifelhaft gute Maßnahmen. Das Problem an der Sache: In den meisten Fällen, die ich gesehen habe, gehen diese Maßnahmen an der Realität im Tagesgeschäft vorbei. Es handelt sich um eine zweite Realität neben dem Arbeitsalltag. Wenn der Head of Spirit nette Mails über Resilien-

## ● Gefahr durch Meditation

Zu Risiken und Nebenwirkungen ... fragen Sie wahrscheinlich nicht Ihren Arzt oder Apotheker. Sollten Sie aber. Es stellt sich nämlich durchaus die Frage, ob Achtsamkeitsmeditation auch Gefahren birgt? Die Antwort lautet: Es ist nicht auszuschließen. Mitte 2024 ist ein Artikel von Miguel Farias, Associate Professor in Experimental Psychology an der Coventry University, erschienen, in dem er Belege für negative Auswirkungen von Meditation anspricht [13]. Die häufigsten negativen Symptome sind Angst- und depressive Zustände.

Die zum Teil sehr allgemeinen Begrifflichkeiten machen es nicht leicht, zu differenzieren. Der genannte Artikel betrachtet Meditation, was nicht per se das Gleiche wie eine Achtsamkeitsübung zwischen zwei stressigen Meetings ist. Es ist ein Unterschied, ob ich über Monate und Jahre täglich 45 Minuten meditiere oder vor der nächsten Besprechung drei Minuten auf meinen Atem achte. Trotzdem ist es wichtig, sich bewusst zu machen, dass Mindfulness eben etwas mit dem „Mind“ machen soll. Wer eine persönliche Vorgeschichte mit mentalen Störungen, Depressionen und Ähnlichen hat, sollte sich vor Beginn der Übungen mit einem Arzt abstimmen. Gleiches gilt für Meditation in Verbindung mit der Einnahme von Medikamenten, wie etwa Opioid-Analgetika zur Therapie chronischer Schmerzen. Man darf annehmen, dass die meisten Meditationslehrer weder ausgebildete Ärzte noch studierte Pharmakologen sind. Da die wissenschaftlichen Untersuchungen zu dieser Frage offensichtlich erst am Anfang stehen, darf man vom Meditationslehrer nicht zu viel verlangen. Einen Hinweis auf die nicht geklärte Problematik der Nebenwirkungen darf man aber erwarten.

ce schreibt, aber mein Teamlead ein Diktator ist, hilft mir die Mail auch nicht. Mit ausreichendem Zynismus könnte man unterstellen, dass die Mitarbeitenden an ihrer Resilienz arbeiten sollen, damit die Vorgesetzten nicht an ihren Führungsdefiziten arbeiten müssen. Das ist keine Kritik an einzelnen Personen, denn die Vorgesetzten haben ja auch Vorgesetzte. Es ist eine systemische, eine kulturelle Frage.

Es kommt vor, dass mir Teammitglieder im Kennenlerngespräch unter Tränen von einzelnen Begegnungen mit ►

## ● Testangebot

dotnetpro-Leser, die Mindfulness-Übungen ausprobieren möchten, finden ein kostenloses Angebot auf der Webseite des Autors unter: <https://kutura.digital/mindfulness>

einem Vorgesetzten oder einer Vorgesetzten berichten, deren Kränkung sie auch nach Monaten noch schmerzte. Bei einem Kunden gab es als Folge eines Rohrbruchs ein durch Fäkalien verunreinigtes Großraumbüro, das trotz ausreichender Zeit in der Corona-Pause nur notdürftig (wer erkennt die Ironie?) gereinigt worden war. Den Kunden zeigte man in den Filialen jedoch ein glänzendes, sauberes Image, und die CEO fantasierte auf der OMR über New Work. Mitarbeitende, die diesen Widerspruch erleben, entfremden sich, gehen in die innere Kündigung und machen Dienst nach Vorschrift.

Der Geschäftsführer einer anderen Firma meinte, man müsse „alle Teams auf Auslastung fahren“. Die Firma befand sich mitten in der Agilen Transformation. Zeit, um die Veränderung mit allen Beteiligten zu gestalten, gab es nicht.

Bei einem weiteren Kunden wurde mir vor wenigen Jahren in der Scrum-Master-Rolle vom vertikal agierenden Projektleiter in einem sogenannten agilen Umfeld lautstark unterstellt, mit dem Team überhaupt zu kommunizieren. Ich solle sie nicht von der Arbeit abhalten. Dass es aus diesem Team bereits Mobbing-Beschwerden über den Projektleiter gab, vervollständigte das Bild für mich, und ich beendete die Zusammenarbeit sehr schnell. Heute berichten die Medien über Standortschließungen dieser Firma, Kündigungen und einen neuen CEO, der als Sanierer bekannt ist. Ob es einen kausalen Zusammenhang gibt, kann ich nicht sagen, aber die Unternehmenskultur, die ich erlebt habe, war sicher keine Grundlage für Erfolg.

Unternehmen zeichnen gerne ein schönes und heiles Bild davon, dass sie sich um das Wohlbefinden der Mitarbeiterinnen und Mitarbeiter kümmern. Sie legen Fortbildungsprogramme auf, die vermutlich isoliert betrachtet sehr gut sind. Aber was bedeutet „Unternehmen zeichnen ein Bild“ oder „Unternehmen tun dies oder jenes“? Unternehmen können nicht handeln. Das können nur Menschen.

Wer sind die Menschen, die ihren Firmen diesen Anstrich geben? Es sind Personen, deren Job es ist, solche Programme zu initiieren. Ich bin zuversichtlich, dass die meisten von ihnen in der besten Absicht einen guten Job machen. Wenn aber gleichzeitig der Umgang im Arbeitsalltag und die Unternehmenskultur nicht den gleichen Gedanken folgen, handelt es sich bei den genannten Maßnahmen um #CultureWashing. Die oberste Leitungsebene steht in der Pflicht, sicherzustellen, dass wesentliche Unternehmenswerte, die man sich oben im Organigramm ausdenkt, auch wirklich durch das mittlere Management durchkommen und Einzug in den Arbeitsalltag finden.

Wenn Wertschätzung für Mitarbeitende einer dieser Werte ist, können Mindfulness-Angebote Vorteile sowohl für die

Menschen als auch für die Firmen bringen. Das gilt allerdings erst dann, wenn die Unternehmenskultur auch außerhalb des Meditationsraums in Ordnung ist. ■

- [1] *Deutschland ist Schlusslicht der Industrieländer*, [www.tagesschau.de/wirtschaft/weltwirtschaft/oecd-wachstum-deutschland-100.html](http://www.tagesschau.de/wirtschaft/weltwirtschaft/oecd-wachstum-deutschland-100.html)
- [2] *Deutsche Wirtschaft weiter abgeschlagen*, [www.tagesschau.de/wirtschaft/oecd-deutsche-wirtschaft-100.html](http://www.tagesschau.de/wirtschaft/oecd-deutsche-wirtschaft-100.html)
- [3] *Claudia Obmann, Autoritäre Chefs sind wieder gefragt*, [www.handelsblatt.com/karriere/fuehrung-sind-autoritaere-chefs-wieder-gefragt/100129837.html](http://www.handelsblatt.com/karriere/fuehrung-sind-autoritaere-chefs-wieder-gefragt/100129837.html)
- [4] *Dacher Keltner, Consequences of Stress at Work*, in: *Mindfulness and Resilience to Stress at Work*, BerkeleyX, [www.edx.org/learn/mindfulness/university-of-california-berkeley-mindfulness-and-resilience-to-stress-at-work](http://www.edx.org/learn/mindfulness/university-of-california-berkeley-mindfulness-and-resilience-to-stress-at-work)
- [5] *MAGS NRW, Passivrauchen*, [www.mags.nrw/informationen-zum-passivrauchen](http://www.mags.nrw/informationen-zum-passivrauchen)
- [6] *Passivrauchen in Deutschland*, <https://de.wikipedia.org/w/index.php?title=Passivrauchen&oldid=256425624#Deutschland>
- [7] *Joel Goh, Jeffrey Pfeffer, Stefanos A. Zenios, Workplace stressors & health outcomes: health policy for the workplace*, 2015, [https://behavioralpolicy.org/wp-content/uploads/2017/02/BSP\\_vol1is1\\_Goh.pdf](https://behavioralpolicy.org/wp-content/uploads/2017/02/BSP_vol1is1_Goh.pdf)
- [8] *Stefan Mintert, Happiness at Work*, dotnetpro 12/2024, Seite 6 ff.,
- [9] *Mindfulness and Resilience to Stress at Work*, [www.edx.org/learn/mindfulness/university-of-california-berkeley-mindfulness-and-resilience-to-stress-at-work](http://www.edx.org/learn/mindfulness/university-of-california-berkeley-mindfulness-and-resilience-to-stress-at-work)
- [10] *Richard Fernandez, Why Modern Workers Need Mindfulness*, [https://greatergood.berkeley.edu/video/item/why\\_modern\\_workers\\_need\\_mindfulness](https://greatergood.berkeley.edu/video/item/why_modern_workers_need_mindfulness)
- [11] *Jeremy Adam Smith, Three Benefits To Mindfulness at Work*, [https://greatergood.berkeley.edu/article/item/three\\_benefits\\_to\\_mindfulness\\_at\\_work](https://greatergood.berkeley.edu/article/item/three_benefits_to_mindfulness_at_work)
- [12] *Felix Stephan, Reguliere dich selbst*, [www.zeit.de/kultur/2015-09/mcmindfulness-silicon-valley-kapitalismus-buddhismus](http://www.zeit.de/kultur/2015-09/mcmindfulness-silicon-valley-kapitalismus-buddhismus)
- [13] *Miguel Farias, Meditation can be harmful – and can even make mental health problems worse*, <https://theconversation.com/meditation-can-be-harmful-and-can-even-make-mental-health-problems-worse-230435>
- [14] *Peter Bostelmann im Gespräch mit Sebastian Purps-Pardigol*, <https://kulturwandel.org/gespraech/sap-peter-bostelmann/>



### Stefan Mintert

begleitet Organisationen in der agilen Transformation und auf dem Weg zu Agile Leadership. Seine Mission sieht er darin, Menschen dabei zu helfen, sich im Rahmen ihrer Arbeit zu entwickeln und zu wachsen.

[stefan.mintert@kutura.digital](mailto:stefan.mintert@kutura.digital)

## UMGANG MIT SCHWIERIGEN KOLLEGEN

# „Der macht mich rasend ...“

Nicht alle Arbeitskollegen sind angenehm. Aber oft steckt mehr dahinter als bloß schlechte Laune – ein systemischer Blick hilft.

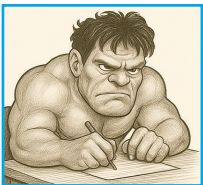
**W**er kennt sie nicht? Diese exzentrischen Arbeitskollegen, bei denen man unwillkürlich die Augen verdreht, sobald sie den Raum betreten oder im Meeting wieder ungefragt ihren Senf dazugeben. Diese Sorte Mensch, bei der man sich fragt, was in ihrem Leben wohl schief lief – und ob die privat auch so drauf sind. Haben diese Leute überhaupt Freunde oder Familie? Und wenn ja: Wie halten die das nur aus? Kurzum: Unerträglich.

Wie sich dieses Verhalten zeigt, ist vielseitig. Doch eines haben diese Kollegen gemeinsam: Sie bringen unsere emotionale Balance durcheinander. Und obwohl man sich fest vornimmt, souverän zu bleiben, gelingt das oft nicht.

## Die üblichen Verdächtigen

Im Lauf der Jahre konnte ich eine ganze Reihe von Kollegentypen beobachten, die durch ihr Verhalten regelmäßig für Augenrollen, Frust oder komplettes Unverständnis im Team sorgen. Einige davon tauchen in fast jedem Unternehmen auf – mal in abgeschwächter, mal in deutlich überzeichneter Form.

Die folgenden Beispiele sind bewusst pointiert dargestellt, um typische Kommunikations- und Verhaltensmuster sichtbar zu machen. Natürlich ist kein Mensch nur dieser eine Typ – aber oft hilft eine überzeichnete Karikatur dabei, das zugrunde liegende Prinzip besser zu erkennen.



### Hulk – der Choleriker

Beginnen wir mit einem besonders intensiven Exemplar: dem Kollegen – in meiner Erfahrung häufig im Management anzutreffen –, der bei jeder Kleinigkeit oder Unregelmäßigkeit schnell an die Decke geht. Wenn etwas schief-

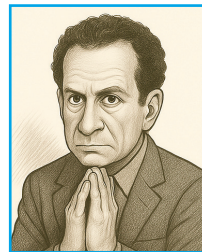
läuft, sucht er unmittelbar nach Schuldigen. Und wenn es gut läuft? Dann gibts wahlweise überschäumende Euphorie oder latent aggressives Schulterklopfen. Kurz: Ein Mensch, der stark von seinen Emotionen getrieben scheint.

Mit diesem Typ aneinanderzugeraten ist kein Hexenwerk. Solange alles glatt läuft, ist das Miteinander unproblematisch. Doch sobald Unvorhergesehenes eintritt – und das ist in der Softwareentwicklung eher die Regel als die Ausnahme –, kippt die Stimmung. Ganz gleich ob man eingeschüchtert reagiert oder dagegenhält: Die Situation wird unangenehm. In beiden Fällen schaukelt sich etwas auf, das eigentlich gar nicht um die Sache geht. Besonders tückisch ist, dass diese Ausbrüche oft überraschend kommen – manchmal aus banalen Gründen, manchmal aus völlig undurchsichtigen. Häufig

steckt gar kein sachliches Problem dahinter, sondern unterschiedliche Erwartungen oder schlicht: ein schlechter Tag.

#### Signalzitate:

- „Ist das dein Ernst?“
- „Und das hat jetzt so lange gedauert?“
- „Das hätte man doch vorher sehen müssen!“



### Monk – der Überkorrekte

Braucht in einem laufenden Projekt eine Komponente ein Redesign, wissen alle: Gib es bloß nicht an Monk, wenn du in absehbarer Zeit ein Ergebnis sehen willst. Monk verliert sich gern im Detail. Alles muss korrekt, zukunftsicher und elegant gelöst werden. Und während man auf ein Ergebnis wartet,

wird aus dem kleinen Bugfix ein nie endender Architektur-Entwurf. Selbst überschaubare Features werden mehrfach überarbeitet – nicht, weil sie kaputt wären, sondern weil sie noch „schöner“ gehen. Die Argumentation für diesen Mehraufwand? Meist hypothetische Anwendungsfälle in ferner Zukunft. Szenarien, die vielleicht nie eintreten – aber Monk schläft besser, wenn sie schon jetzt technisch vorbereitet sind.

#### Signalzitate:

- „Wenn wir das schon machen, dann bitte richtig!“
- „Wenn wir später mal dies und das machen wollen, ist alles vorbereitet.“
- „Das war vorher viel zu unsauber programmiert.“



### Sheldon – der Besserwisser

Das Ticket ist fertig, der Pull Request erstellt – der Kunde wartet dringend auf den Bugfix. Doch dann schlägt er zu: Sheldon. Als Reviewer eröffnet er eine Grundsatzdiskussion über die Position der geschweiften Klammer – obwohl dieses Thema längst entschieden und der Fix eigentlich dringend ist. Es geht ums Prinzip. Denn wenn wir jetzt inkonsequent werden, herrscht morgen Anarchie. Und an Softwareanarchie geht schließlich die Welt zugrunde.

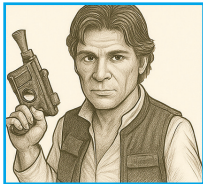
Sheldon ist der Typ Kollege, den man gerne um Rat fragt, wenn es um knifflige technische Details geht – er kennt jedes Paper, jede Spec und jede Fußnote im NuGet-Package. Nur diskutieren will man mit ihm lieber nicht. Aber das passiert schneller, als einem lieb ist. Sheldon ist diskussionssicher, hartnäckig und ausdauernd. Prinzipien sind ihm wichtiger ►



als Pragmatismus. Und wenn er sich am Ende eines Meetings noch zu Wort meldet, weiß das Team: Jetzt wird es zäh.

**Signalzitate:**

- „Aber in der Spec steht was anderes.“
- „Eine Untersuchung hat ergeben, dass Spaces besser sind als Tabs.“
- „Technisch gesehen hast du unrecht.“



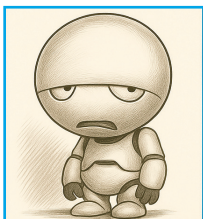
**Han Solo – der Cowboy-Entwickler**

Peng-peng – und fertig. Wenn es auf seiner Entwicklermaschine läuft und sein persönlicher Click-Test durchgelaufen ist, reicht das. Deployment zum Kunden? Klar. Tests, Dokumentation, Wartbarkeit? Uff, später vielleicht.

Han ist Fan von schnellen 70-Prozent-Lösungen. Nicht aus Faulheit, sondern aus Überzeugung: Das reicht oft schon – und ist wirtschaftlich effizient. Der Rest macht eh keinen Spaß. Während andere noch über Architektur nachdenken, ist Han längst beim nächsten Projekt. Das Problem: Diese 30 Prozent Rest sind oft die, die später knallen. Hans Code bringt Dinge ins Rollen – aber nicht immer in die richtige Richtung. Wartung, Bugs, Frust beim Kunden? Landen bei anderen. Han ist dann schon weitergezogen. Einsicht? Eher selten.

**Signalzitate:**

- „Ja, das passt doch so.“
- „Wozu braucht der Kunde denn sowas überhaupt?“
- „Jetzt darf da halt niemand was Falsches eintragen.“



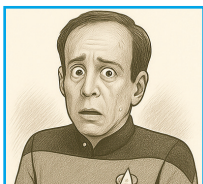
**Marvin – der Trotzige**

Marvin ist ein stiller Typ. Er liebt Technik, kann sich in Probleme hineinknien und liefert zuverlässig. Und doch schwingt bei jeder Antwort eine resignierte Grundhaltung mit. Nicht aus Faulheit – eher aus tief empfundener Gleichgültigkeit. Fragt man ihn, ob

Lösung A oder B besser sei, bekommt man ein „Wird eh schiefgehen“ zurück. Er interagiert kaum, zieht sich zurück und versprüht durch seine Körpersprache ungefähr so viel Elan wie ein leerer Akkupack. Manchmal ist er direkt, manchmal passiv-aggressiv, aber immer irgendwie... demotiviert. Und das überträgt sich auf das ganze Team. Niemand hat Lust auf konstruktive Diskussionen, wenn Marvin schon vorher die Welt abgeschrieben hat.

**Signalzitate:**

- „Am Ende machen wir ja eh wieder Mist.“
- „Und ich muss es wieder ausbaden.“
- „Ja, hättet ihr es mal von Anfang an so gemacht, wie ich gesagt habe.“



**Barclay – der People Pleaser**

Barclay ist der Typ Kollege, der immer freundlich ist, nie Nein sagt und sich irgendwie um alles kümmert – um die liegengebliebenen Reviews, die Küchenspülmaschine, das Teamfrüh-

stück und das ungeliebte Legacy-Modul, das keiner anfassen will. Auffällig ist das nur selten. Denn Barclay macht kein Aufhebens darum. Was er allerdings nicht macht: fertig werden. Er übernimmt alles, verspricht viel, wirkt immer wahn-sinnig beschäftigt – aber am Ende bleibt erstaunlich wenig greifbar. Die To-do-Liste wächst schneller als sein Output, Termine platzen, Tickets bleiben offen. Und trotzdem kann man ihm kaum böse sein, weil er ja „so nett“ ist. Aber irgendwie funktioniert es einfach nicht.

**Signalzitate:**

- „Klar, das kann ich übernehmen.“
- „Heute arbeite ich an Ticket X, Y, Z, AA und AB. Vielleicht schaffe ich heute Abend noch AC.“
- „Mir ist nur ein bisschen schwindlig, kein Problem.“

**Einordnung**

Sinn und Zweck dieser plakativen Auflistung von typischen „Freaks“ ist, dass jeder Leser zumindest zwei oder drei dieser speziellen Menschen in seinem Umfeld identifizieren kann – mal mit mehr, mal mit weniger starken Ausprägungen. In-sgeheim habe ich aber auch gehofft, dass durch die eine oder andere Charakterbeschreibung der geneigte Leser auch sich selbst erkennt (natürlich nur in gaaaanz schwacher Ausprägung – man ist ja kein Freak). Denn, und darauf zielt alles ab, es handelt sich in den meisten Fällen erst einmal um unterschiedliche Ausprägungen menschlicher Grundbedürfnisse, die sich in individuellen Lebensstrategien widerspiegeln.

Natürlich brauchen wir nicht darüber zu diskutieren, ob cholerische Ausbrüche eine akzeptierte Umgangsart im Sinne der freien Persönlichkeitsentfaltung sein sollten oder nicht. Aber das trifft auf alle extremen Ausprägungen zu – auf Kosten des Umfelds, aber auch des eigenen Wohlbefindens. Dazu aber später mehr.

Im ersten Schritt möchte ich über jene Gruppe von Menschen sprechen, bei denen sich das Unbequeme-Kollegen-Phänomen schlichtweg um einen Querschnitt der menschlichen Vielfalt handelt – Menschen, denen man oft mit etwas mehr Verständnis, Empathie und vielleicht sogar Humor begegnen kann. In meiner Erfahrung handelt es sich hierbei um den Großteil der sogenannten „schwierigen Persönlichkeiten“.

Aus diesem Grund stolpere ich auch regelmäßig über den Spruch „In unserem Büro herrscht arschlochfreie Zone“. Was auf den ersten Blick sympathisch klingt, meint in der Praxis oft: Wir wollen hier keine Reibung. Aber Reibung entsteht eben dort, wo Menschen mit unterschiedlichen Erfahrungen, Denkweisen und Werten aufeinandertreffen – also genau dort, wo echte Vielfalt beginnt.

Natürlich: Es ist angenehmer, wenn alle ähnlich ticken, ähnlich priorisieren und sich nicht ständig gegenseitig infrage stellen. Aber dieser Komfort hat einen Preis. Denn Teams, die nur aus Gleichgesinnten bestehen, mögen reibungslos laufen – sie bleiben aber oft auch hinter ihrem Potenzial zurück. Wirkliche Stärke entsteht nicht durch Übereinstimmung, sondern durch Ergänzung. Und das kann im Alltag durchaus anstrengend sein.

Als letzten Rückgriff auf die Popkultur möchte ich Captain Reynolds von der Firefly zitieren (aber Picard hat sicher auch

etwas Ähnliches gesagt): „We don't need crew that thinks alike – we need people who think to complement each other.“

## Perspektivwechsel

Jetzt, wo wir wissen, dass hinter schrägen Vögeln in den meisten Fällen nur andersgeartete Vögel stecken und es möglicherweise sogar gut ist, Menschen mit anderer Denkweise im Team zu haben, sollten wir uns mit der Frage auseinandersetzen, wie wir konfliktfrei mit deren Eigenschaften klarkommen. Natürlich ist es immer am bequemsten, wenn es mit dem Gegenüber keine Konfliktpunkte gibt, aber in einer diversen Gesellschaft ist das nun mal nicht möglich.

Ernsthafte Konflikte entstehen nicht nur durch unterschiedliche politische Meinungen, Lebensparadigmen oder Essgewohnheiten. Sie entstehen bei den banalsten Themen wie dem Wetter. Oder eben dort, wo feste Überzeugungen – selbst so triviale Details wie Spaces versus Tabs – unnachgiebig aufeinandertreffen und emotionale Reaktionen hervorrufen.

Meiner Erfahrung nach steckt hier eine der Kernbefürchtungen typischer in der IT tätiger Menschen: die tiefe Verbindung zwischen den fachlichen Fähigkeiten und der eigenen Selbstdefinition. Als Konsequenz daraus wird Kritik an der fachlichen Vorgehensweise als Kritik an der eigenen Person empfunden – man nimmt es persönlich.

Mit diesem Wissen im Hinterkopf schlage ich nun einen Perspektivwechsel vor: Wir versetzen uns dazu in eine der typischen Situationen, die einen zur Weißglut bringen, wenn besagter Kollege mal wieder seine Show abzieht.

Der ratgebergeplagte Leser wird jetzt vermutlich erwarten, dass wir uns in die Situation des anderen versetzen, um dessen Motivation zu verstehen. Ja, das ist grundsätzlich eine gute Idee – und wir werden das später im Text auch noch machen. Aber hier geht es erst mal in eine andere Richtung.

Denn wenn man davon ausgeht, dass Menschen eben sind, wie sie sind – und sich nicht einfach ändern, bloß weil man sich das wünscht –, dann bleibt im Umkehrschluss nur eine Frage: Was kann ich selbst anders machen, damit ich entspannter mit der Situation umgehen kann? Bevor ich aber so einfach mein Verhalten ändern kann, muss ich mir eines eingestehen: Für mich ist diese Situation ebenso emotional, sonst würde ich mein Gegenüber nicht als „blöden Kollegen“ beschimpfen. Bevor ich diesem Gefühl nicht auf den Grund gehe, bin ich nicht frei in meiner Handlungsfähigkeit.

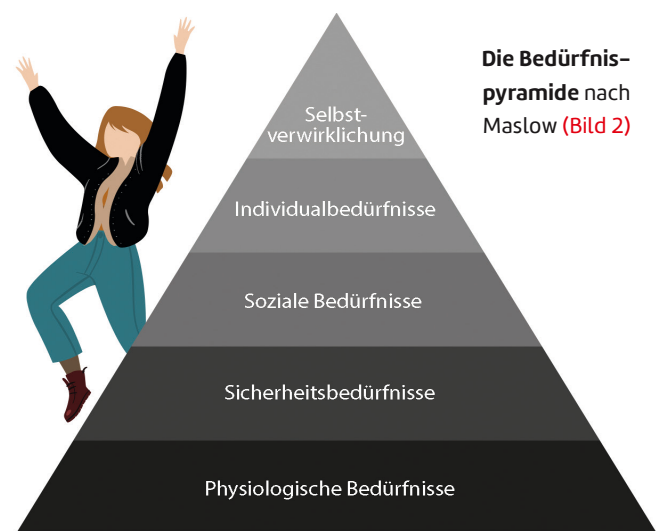
Die Kernfrage dieses Perspektivwechsels lautet also, warum mich das ganze Theater eigentlich so auf die Palme bringt. Anders formuliert: Warum geht mir das Verhalten einer anderen Person eigentlich so nahe – denn das ist es, was geschieht, wenn sich Emotionen regen.

## Die menschliche Natur

Bevor wir klären, warum uns bestimmte Situationen so stark triggern, lohnt sich ein kurzer Blick auf die Dynamik menschlicher Bedürfnisse. Keine Sorge, ich mache es kurz. Wer es genauer wissen will: In den Artikeln „Die unsichtbare Blockade“ [1] und „Als Coden noch Bock machte“ [2] habe ich die Themen Bedürfnisverletzungen und Schutzstrategien eingehender beschrieben.

Wichtig für uns hier ist: Jeder Mensch trägt ein Set an Grundbedürfnissen in sich – etwa nach Sicherheit, Zugehörigkeit, Einfluss und Selbstverwirklichung. Die Maslow'sche Bedürfnispyramide [3] bringt diese in eine verständliche Ordnung: von der Basis mit körperlichen Überlebensfaktoren (wie Essen, Wärme, Schlaf) über Sicherheit und soziale Einbindung bis zur Spitze, der Selbstverwirklichung (Bild 2).

Was simpel aussieht, ist in Wahrheit ein zentrales Steuerungssystem für unser Verhalten. Denn in der Kindheit – unserer sensibelsten Entwicklungsphase – wird geprägt, wie gut diese Bedürfnisse erfüllt wurden. Und unser Gehirn findet Wege, um sich in dieser frühen Abhängigkeit das zu sichern, was es zum Überleben braucht. Diese ersten Strategien werden nicht bewusst gespeichert. Sie landen tief im



Unterbewusstsein – und genau dort werden sie auch später wieder aktiviert, wenn etwas „nicht stimmt“.

In unserem Bewusstsein tauchen diese Warnsignale in Form von Emotionen wie Wut, Angst oder Scham auf. Ein innerer Schutzmechanismus springt an – oft lange bevor der Verstand überhaupt mitbekommt, was los ist.

Kritisch wird es, wenn aus einer frühen Bewältigungsstrategie eine Überlebensstrategie wird. Etwa dann, wenn ein Kind lernt: „Nur wenn ich funktioniere, bekomme ich Liebe.“ Nicht Lob, nicht Aufmerksamkeit – sondern echte, emotionale Zuwendung. Vielleicht waren die Eltern überfordert, kühl oder selbst unter Druck. Und Zuneigung gab es nur dann, wenn das Kind sich anpasste, Leistung brachte oder stillhielt. Für das kindliche System bedeutet das: „Meine Bindung hängt davon ab, dass ich keine Fehler mache.“ Und weil Bindung ein existenzielles Bedürfnis ist, wird diese Strategie tief verankert.

Im Erwachsenenalter reicht dann ein harmloser Satz wie „Das war nicht dein stärkster Beitrag“ – und das innere Alarmsystem schlägt an. Nicht, weil die Situation objektiv bedrohlich ist. Sondern weil sie an etwas Altes erinnert. Und plötzlich fühlt sich Kritik an wie Bindungsentzug.

Wir alle tragen solche Trigger in uns. Sie sind grundsätzlich nichts Schlechtes: Wut schützt unsere Grenzen, Angst ►

macht uns wachsam, Scham hilft uns, unser Handeln zu reflektieren. Schwierig wird es erst, wenn die emotionale Reaktion keine Wahl mehr lässt – und wir automatisch auf Situationen anspringen, die mit der ursprünglichen Bedrohung kaum noch etwas zu tun haben.

Zwei Dinge machen es besonders kompliziert: Erstens reagiert unser System nicht auf exakt dieselbe Situation wie damals, sondern auf etwas, das sich ähnlich anfühlt. Das sorgt dafür, dass alte Gefühle auch in ganz anderen Kontexten wieder auftauchen. Zweitens bleibt die gespeicherte Reaktion oft ein Leben lang aktiv – selbst dann, wenn sich das Umfeld längst verändert hat und keine reale Gefahr mehr besteht.

## Warum reagiere ich so?

Ist man an dem Punkt angelangt, dass ein Arbeitskollege als unerträglich empfunden wird, ist man bereits selbst in einem Gefühl – und das gilt es zu ergründen, bevor man sich der Situation rational nähern kann. Wir versetzen uns daher mal in eine der typischen Situationen, die einen beim betroffenen Kollegen zur Weißglut bringen. Was folgt, ist eine kleine Reihe von Warum-Fragen, die uns im Idealfall am Ende verraten, was das Kernproblem ist – und mit welcher meiner persönlichen Lebensstrategien diese Situation kollidiert.

Ich demonstriere das an einem fiktiven inneren Dialog, um die unterschiedlichen Ebenen der Fragekette aufzuzeigen:

- **Warum ist dieser Kollege so nervig?** – Weil er sich im Frühmooting mal wieder aufführt wie der große Macher – und am Ende wenig bei rumkommt.
- **Und warum ärgert dich das?** – Weil er so tut, als würde alles durch ihn laufen – aber eigentlich erledigen wir die Arbeit.
- **Warum trifft dich das?** – Weil ich den Eindruck habe, dass meine Leistung nicht gesehen wird. Ich bin verlässlich, ich liefere – aber es wirkt, als wäre das weniger wert als sein großes Getöse.
- **Wie fühlt sich das für dich an?** – Es fühlt sich ungerecht an. Ich bin wütend darüber, dass er sich so aufspielt und wir einfach unsere Arbeit machen – ohne den Applaus.
- **Kennst du das Gefühl von irgendwoher?** – Ja. So ähnlich war das früher mit meinem älteren Bruder. Er konnte besser reden, hat sich oft mit meinen Sachen gebrüstet – und meine Eltern habens ihm geglaubt. Ich war halt eher leise. Und irgendwann hab ich aufgehört, was zu sagen.

**Hinweis:** Der hier dargestellte Dialog ist stark verkürzt – in der Realität sind solche Antworten oft nicht sofort greifbar. Fragen wie „Was fühlst du gerade?“ oder „Woher kennst du dieses Gefühl?“ stammen aus der therapeutischen Praxis und brauchen oft Zeit, um eine ehrliche Antwort zuzulassen. Es ist völlig normal, wenn einem beim ersten Nachdenken keine klaren Zusammenhänge einfallen. Manchmal braucht es mehrere Anläufe, Stille – oder einfach ein bisschen Abstand, bis sich das Puzzle langsam zusammensetzt.

Natürlich weiß man in vielen Fällen rational längst, dass diese Situation objektiv betrachtet gar nicht so schlimm ist. Dass die eigene Leistung im Team durchaus gesehen wird. Dass niemand den Kollegen für einen Rockstar hält – außer vielleicht er selbst. Und trotzdem brennt es innerlich. Denn unser emo-

tionales Bewertungssystem zielt nicht auf die Sachlage – sondern auf das Gefühl, das sie in uns auslöst. Und das stammt häufig aus früheren Erfahrungen, in denen wir tatsächlich nicht gesehen, übergangen oder unfair behandelt wurden.

## Warum reagiert der andere so?

Hat man sich einmal bewusst gemacht, warum einen das Verhalten des Gegenübers so triggert, ist auch der Blick auf ihn freier. Nicht, weil wir ihn jetzt besonders nett finden müssten, sondern weil wir in der Lage sind, ihn mit mehr Abstand zu betrachten. Der Vorschlag an dieser Stelle: Wir versuchen, das Verhalten des Kollegen nicht mehr nur als Provokation zu lesen – sondern als Ausdruck eines Musters, das irgendwo herkommt. Nicht, um es zu entschuldigen. Sondern um zu verstehen, warum sich Menschen manchmal so verhalten, wie sie sich verhalten. Denn niemand wacht morgens mit dem Plan auf, möglichst anstrengend zu sein. Jedes Verhalten – so schräg es uns auch erscheinen mag – hat irgendwo seine Wurzeln. Und oft liegen die tiefer, als wir glauben.

Werfen wir also erneut einen Blick auf unsere sechs Typen. Nur diesmal nicht durch die genervte Teambrille, sondern mit der Frage: Was könnte dahinterstecken?



### Hulk

Bei Hulk vom Beginn dieses Artikels haben wir es mit einem Menschen zu tun, der durch seine vereinnahmende und dominante Art die volle Kontrolle über die Situation braucht. Durch die Verbreitung von Angst und Schuldzuweisungen wird jede gemeinsame Kommunikation auf Augenhöhe im Keim erstickt.

Jener Mensch könnte in Kindheitszeiten die Erfahrung gemacht haben, dass Wut der einzig verbleibende Kanal war, um überhaupt Gehör zu bekommen. Vielleicht hatte er Eltern, die emotional distanziert waren – immer beschäftigt, selten präsent, eher irritiert als interessiert, wenn das Kind auf sie zukam. Wer sich in einer solchen Umgebung ausdrücken will, braucht irgendwann andere Mittel. Und bekam man Aufmerksamkeit nur, wenn man laut wurde, dann prägt sich das als zuverlässiger Weg ein. Aus „Ich will Macht“ wird dann ganz schnell: „Ich ertrage keine Ohnmacht.“



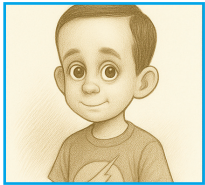
### Monk

Wie sieht es bei Monk aus? Da haben wir jemanden, für den es keine Kompromisslösungen gibt. Alles muss perfekt sein – bis ins letzte Detail. Und das dauert. Hier geht es also nicht bloß um Leistung, sondern um Perfektion. Was gut ist, reicht nicht. Es muss besser gehen. Und sicherer. Und noch ein Stück cleaner.

Möglicherweise ist dieser Mensch in einem Umfeld aufgewachsen, in dem Dinge nie einfach angenommen wurden. Wo ein gemaltes Bild kommentiert wurde mit: „Na, das kannst du aber schöner.“ Wo ein gutes Ergebnis nicht belohnt, sondern mit einem „Nicht schlecht – aber schau mal hier ...“ rela-

tiviert wurde. Lob war rar, und wenn es kam, hatte es einen Haken. Das Kind lernt: „Wenn ich dazugehören will, darf ich keinen Fehler machen.“ Und: „Solange ich etwas verbessern kann, bin ich noch nicht fertig.“

Heute im Team zeigt sich das in ewigen Redesigns, überambitionierten Architekturdiskussionen und dem Gefühl, dass noch nicht mal Prototypen fertig werden. Für ihn ist das kein Spieltrieb – es ist eine Frage von Sicherheit. Lieber zu viel als zu wenig. Lieber nochmal überarbeiten, als angreifbar zu sein.



#### Sheldon

Sheldon ist der, der in jedem Meeting noch eine Fußnote zur RFC einwirft, der bei Pull-Requests auf formale Details pocht, der sich nie zu schade ist, ein „Technisch gesehen hast du unrecht“ hinterherzuschreiben. Dabei

geht es ihm selten um das konkrete Problem – sondern ums Prinzip. Und darum, dass er derjenige ist, der es weiß.

Vielleicht hat dieser Mensch früh erfahren, dass er nicht gesehen wurde. Nicht in Form direkter Ablehnung, sondern durch einen subtilen Mangel an Interesse. Wenn Eltern mit seinem eigentlichen Wesen nichts anfangen konnten – sei es, weil es zu still, zu nerdig oder zu sensibel war –, sucht sich ein Kind Wege, trotzdem Anerkennung zu bekommen. Und Wissen funktioniert. Vor allem, wenn es überprüfbar ist. Wer klüger ist, liegt richtig. Und wer recht hat, hat Kontrolle.

Heute im Team erleben wir ihn als anstrengenden Besserwisser, der Diskussionen unnötig verkompliziert und Kritik nicht gut wegsteckt. Doch hinter der intellektuellen Schutzschicht steckt ein alter Wunsch: gesehen und ernst genommen zu werden – und zwar für das, was man kann. Auch wenn das, was man fühlt, lieber gut versteckt bleibt.



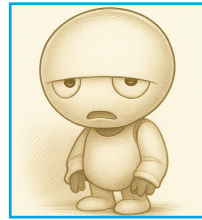
#### Han Solo

Han ist der Typ, der einfach macht. Schnell. Spontan. Direkt. Wer ihm zu lange mit Doku, Standards oder Tests kommt, wird weggewunken. „Wir brauchen jetzt eine Lösung, keine Wissenschaft.“ Und zack – schon ist

der Code live. Wartbarkeit? Wird später eh umgebaut. Hauptsache, es läuft.

Vielleicht war Han als Kind in einem Umfeld, in dem man funktionieren musste. Wo Verbindlichkeit Stress bedeutete. Vielleicht gab es zu Hause viel Unruhe – Umzüge, zerbrochene Beziehungen, ständige Wechsel. Wer in solchen Umständen aufwächst, lernt schnell, sich durchzuwursteln. Verantwortung wird vermieden, Nähe auch. Stattdessen zählt, dass man sich durchschlägt – egal wie. Lieber improvisieren als sich auf Regeln verlassen, die morgen schon wieder anders sein könnten.

Heute im Team wirkt Han wie der lockere Pragmatiker. Und ja, manchmal ist er das auch. Aber eben nur bis zu dem Punkt, an dem man ihn festnageln will. Dann wird es unangenehm. Denn was wie Coolness wirkt, ist oft nur die Flucht vor Verpflichtung – aus Gewohnheit, nicht aus Böswilligkeit.



#### Marvin

Marvin ist der stille Kollege, der nie direkt Nein sagt, aber auch nie wirklich Ja meint. Der bei Vorschlägen müde lächelt und bei Entscheidungen nur ein „Wird eh wieder Mist“ murmelt. Nicht destruktiv im klassischen Sinne – eher chronisch resigniert. Und mit

dieser Mischung aus Sarkasmus und Gleichgültigkeit zieht er jeden im Raum ein Stück runter.

Vielleicht ist Marvin in einer Umgebung groß geworden, in der Emotionen nicht willkommen waren. Wo auf Enttäuschung mit Achselzucken reagiert wurde, auf Hoffnung mit: „Stell dich nicht so an.“ Wer so etwas häufig erlebt, hört irgendwann auf, sich zu freuen – weil es sowieso wieder zerplatzt. Also schützt man sich: mit Rückzug, mit Ironie, mit möglichst geringer Fallhöhe. Und wer keine Erwartungen mehr hat, kann auch nicht mehr enttäuscht werden.

Heute begegnen wir Marvin als wandelnde Wolke aus Zynismus. Nicht, weil er anderen den Spaß verderben will – sondern weil er selbst längst nicht mehr glaubt, dass irgendwas gut werden kann. Und wenn man sich schon allein fühlt, dann wenigstens als der Einzige, der es von Anfang an gewusst hat.



#### Barclay

Barclay ist immer da. Immer freundlich, hilfsbereit, zuverlässig – jedenfalls auf den ersten Blick. Wenn jemand krank ist, springt er ein. Wenn die anderen ins Wochenende starten, macht er noch schnell die offenen Tickets fertig. Nur: So richtig fertig wird das alles nie. Denn Barclay sagt zu allem Ja – und zu sich selbst zu selten.

Vielleicht war Barclay schon als Kind zu oft „der Vernünftige“. Nicht, weil er das wollte – sondern weil es sein musste. Möglicherweise wuchs er in einem Umfeld auf, in dem die Erwachsenen selbst überfordert waren: psychisch, körperlich, finanziell. Und irgendjemand musste ja funktionieren. Also

hat das Kind früh gelernt, Verantwortung zu übernehmen. Es hat mitgedacht, mitgeföhlt, mitgetragen – viel zu früh. Und oft wurde er dafür sogar noch gelobt: „Du bist so selbstständig – auf dich kann man sich verlassen.“

Heute begegnet uns Barclay als jemand, der an allem mitarbeitet, aber in nichts sichtbar ist. Der nie Nein sagt, selbst wenn er längst am Limit ist. Der jede neue Aufgabe wie selbstverständlich aufnimmt – weil er tief verinnerlicht hat, dass Wertschätzung an Leistung gekoppelt ist. Und der oft selbst nicht merkt, dass sein freundliches „Kein Problem“ in Wirklichkeit längst ein stummes „Ich kann nicht mehr“ ist.

Jeder trägt ein Päckchen mit sich herum. Nicht jeder kann es gut verstecken. Doch was machen wir jetzt mit diesem Wissen, das in **Tabelle 1** nochmals zusammengefasst ist?

### Handlungsspielraum zurückerobern

Kommen wir also in die Handlung. Oder etwa nicht?

Das Perfide an diesem Thema ist ja, dass es mit einem halbseitigen Infotext über das menschliche Unterbewusstsein ►



nicht getan ist, um den Unmut zwischen zwei Menschen aus dem Weg zu räumen. Wir haben es hier mit sehr emotionalen Situationen zu tun, die sich auf rationaler Ebene meist nicht so einfach auflösen lassen – daher ja die ganze Vorrede.

Wie laufen solche Situationen oft ab?

Aus einer ganz normalen Arbeitssituation heraus reagiert der „doofe“ Arbeitskollege so, wie er nun mal reagiert. Als Folge davon fühlen sich n-1 andere Kolleginnen und Kollegen genervt bis getriggert – und reagieren entsprechend. Die Situation eskaliert, und die Dinge nehmen ihren Lauf. Was lernen alle? Der „blöde“ Kollege ist unverbesserlich und mit rationalen Argumenten nicht erreichbar. Außerdem lernt dieser „unverbesserliche“ Kollege, dass er mit seiner Art bei den anderen keine Freunde findet – und speichert die wiederum ebenfalls als Feindbild ab. Diese Dynamik ist vollkommen typisch – und das Problem besteht weiterhin.

Hier wird deutlich, dass das Verhalten von Menschen stark in Wechselwirkung zum Umfeld steht, in dem sie sich befinden. Das ist die Grundidee der systemischen Betrachtung von zwischenmenschlichen Dynamiken. Zum System gehört der besagte Kollege, zum System gehören aber auch das Arbeitsumfeld, die Kollegenschaft – und jeder einzelne von uns. Das eröffnet zwar ein wahnsinnig kompliziertes Geflecht an Parametern, zeigt aber auch, wie viele Stellschrauben es gibt.

Dass wir den Kollegen nicht unmittelbar beeinflussen können, ist offenkundig. Scheinbar hilft es nicht, ihm mitzuteilen, dass er sich nervig benimmt – ganz im Gegenteil: Das befeuert die Situation meist noch. Was wir aber immer ändern können, ist das eigene Verhalten. Und darauf läuft es im ersten Schritt hinaus.

Übliche Ratgeber beschränken sich meist auf das Verteilen von Handlungshinweisen: „Klopfen Sie Ihrem Mitarbeiter doch mal auf die Schulter, um sein Selbstbewusstsein zu stärken und ihm zu zeigen, wie sehr Sie ihn schätzen.“ Das ist zwar gut gemeint, bewirkt aber das Gegenteil, wenn die innere Überzeugung nicht zur Handlung passt. Menschen sind in diesem Zusammenhang nämlich verdammt feinfühliges Wesen. Wenn es eine Divergenz zwischen dem Gefühlten und dem Gesagten gibt, werden wir in der Regel misstrauisch – wir riechen den Braten.

Und jetzt wird auch klar, warum ich fünf Seiten damit verbracht habe, aufzuzeigen, was sowohl uns als auch unser Gegenüber antreibt, triggert oder belastet: Denn im Inneren sind alle Menschen auf der Suche nach guten und stabilen Beziehungen. Und die entstehen durch Wohlwollen und Vertrau-

en. Deshalb ist es wichtig zu wissen und zu erkennen, welche Verhaltensweisen mich persönlich ängstigen – und warum. Ebenso ist es für eine aufrichtige Begegnung essenziell, zu verstehen, zu akzeptieren und vor allem zu respektieren, dass andere Menschen durch ihre unterschiedlichen Erfahrungen auch unterschiedliche Wichtigkeiten und Werte mitbringen. Sollten diese Werte meinen eigenen krass widersprechen – so wie vermutlich im vorliegenden Fall –, wird es anstrengend. Ist einem das aber bewusst, können Kompromisse gefunden werden, ohne Respekt und Augenhöhe mit dem „Verhandlungspartner“ zu verlieren.

Das klingt jetzt alles sehr hochtrabend – aber wie fängt man damit an? Wie schon gesagt: Es gibt keine Schritt-für-Schritt-Anleitung. Es kommt nicht darauf an, die richtigen Dinge zu sagen – sondern auf die Haltung dahinter. Will ich wirklich mit dem Kollegen klarkommen, dann klappt das auch! Wir haben bereits herausgearbeitet, was uns konkret auf die Palme bringt. Sobald uns das bewusst wird, gelingt es uns, gelassener mit kritischen Situationen umzugehen und möglicherweise größere Kompromisse einzugehen. Das System ist hiermit geändert.

Wie wir gesehen haben, sind alle Überlebensstrategien im Kern Selbstschutzmechanismen. Sie helfen dabei, in einer vermeintlich gegnerischen Welt klarzukommen. Es geht immer darum, Bedürfnisse zu sichern, die sonst nicht erreichbar gewesen wären – oder schmerzhaft Situationen zu vermeiden. Die beste Heilung dagegen sind gegenläufige Erfahrungen. Das ist ein langwieriger Prozess – zugegeben. Aber es funktioniert: Indem Bedürfnisse auch ohne Kampf befriedigt werden. Doch dafür braucht es Vertrauen – Vertrauen darauf, dass man nicht in die Pfanne gehauen wird. Und mit einem wohlwollenden Umgang schafft man genau diese Atmosphäre.

Ich will an dieser Stelle nichts schönreden. Sich die eigenen Trigger bewusst zu machen – und sie im entscheidenden Moment auch noch zu unterbrechen, um eine angemessene Entscheidung im Hier und Jetzt zu treffen – braucht Übung, siehe auch den Kasten **Zwischen Reiz und Reaktion**. Und Geduld.

Aber ich verspreche: Diese Techniken helfen nicht nur im Umgang mit schwierigen Kollegen. Sie geben einem auch in vielen anderen Lebenssituationen das Gefühl zurück, wieder handlungsfähig zu sein – gerade dann, wenn die alten Reaktionen drohen, die Oberhand zu gewinnen.

Und das Beste daran: Hat man seine eigenen Muster erst mal im Griff, wird es oft gar nicht mehr so wichtig, ob der Kollege sich überhaupt noch verändert.

● **Tabelle 1: Übersicht über mögliche Zusammenhänge zwischen Verhalten und Bedürfnis**

| Verhalten                     | Was es bei mir auslöst       | Was womöglich dahintersteckt                                              |
|-------------------------------|------------------------------|---------------------------------------------------------------------------|
| Laut und bestimmend           | Ich fühle mich überfahren    | Vielleicht schützt er sich vor dem Gefühl, die Kontrolle zu verlieren     |
| Alles kontrollieren wollen    | Ich fühle mich klein gemacht | Vielleicht braucht sie Klarheit, um sich sicher zu fühlen                 |
| Überangepasst und hilfsbereit | Ich fühle mich genervt       | Vielleicht hat er gelernt, dass Zugehörigkeit nur durch Leistung entsteht |
| Rückzug und Funkstille        | Ich fühle mich abgelehnt     | Vielleicht schützt sie sich vor Überforderung oder Kritik                 |
| Rechthaberisch                | Ich fühle mich abgewertet    | Vielleicht fürchtet er, ohne Expertise nicht ernst genommen zu werden     |

## ● Zwischen Reiz und Reaktion

Wir haben gesehen, wie alte Erfahrungen unser Verhalten prägen. Wie Trigger blitzschnell Schutzmechanismen aktivieren, noch bevor wir bewusst reagieren können. Doch genau in diesem kurzen Moment zwischen dem Auslöser und der Reaktion liegt ein kleiner, aber entscheidender Handlungsspielraum.

Er ist leicht zu übersehen. Aber wenn wir unsere eigenen Trigger kennen, können wir lernen, diesen Moment wahrzunehmen. Nicht immer. Aber immer öfter. Hier beginnt Veränderung: Nicht, weil sich sofort alles ändert, sondern weil wir Schritt für Schritt anders wählen können.

## Die harten Fälle

Bis hierher klang vieles fast schon romantisch – ein bisschen Selbstreflexion, ein bisschen gegenseitiges Verständnis, und am Ende fassen wir uns alle an der Hand. Aber was ist, wenn der betreffende Kollege völlig unzugänglich bleibt?

Natürlich kann es sein, dass bestimmte Überlebensstrategien so stark ausgeprägt sind, dass das daraus resultierende Verhalten tatsächlich krankhaft wirkt. In solchen Fällen liegt nicht selten ein Trauma zugrunde – also eine unverarbeitete emotionale Überforderung, die in der Regel in früher Kindheit entstand. Entweder durch eine akute Grenzüberschreitung oder einen langanhaltenden Zustand innerer Überforderung. Das Unterbewusstsein sorgt dann mit allen Mitteln dafür, dass der Mensch nie wieder in eine solche oder auch nur eine ähnliche Situation gerät. Diese Schutzmechanismen zeigen sich mal laut, mal leise. Vielleicht in Form von Panikattacken, psychosomatischen Symptomen wie Kopfschmerz, Übelkeit oder diffuser Angst – oder als innere Überzeugung, etwas auf keinen Fall zu wollen, ohne dafür eine für andere nachvollziehbare Begründung zu haben.

Das Heimtückische: Den Betroffenen selbst ist die Verbindung zwischen Auslöser und Reaktion meist überhaupt nicht bewusst. Das auslösende Ereignis liegt tief vergraben – in extremen Fällen ist es vollständig aus dem Gedächtnis gelöscht. Denn der Schmerz ist so überwältigend, dass selbst die Erinnerung daran nicht auszuhalten ist. In solchen Fällen ist professionelle Begleitung notwendig.

Von außen lässt sich das Ausmaß kaum erkennen. Viele Betroffene wirken vollkommen entspannt – solange ihr Alltag so strukturiert ist, dass der innere Schmerz nicht berührt wird. Doch je nach Prägung lässt sich das nicht immer vermeiden. Und dann kommen sie, die „Marotten“, die impulsiven Ausbrüche, die nicht nachvollziehbaren Reaktionen. Jemanden in so einem Zustand zu konfrontieren ist in der Regel sinnlos. Um sich überhaupt offen und ehrlich mit solchen Themen auseinandersetzen zu können, braucht es für die Betroffenen einen absolut sicheren Ort – nicht Kritik im Büroflur. Eine betrieblich auferlegte Therapie ist in solchen Fällen keine Lösung, sondern ein Übergriff.

Veränderung kann erst dann beginnen, wenn die betroffene Person selbst dazu bereit ist. Und das passiert meist erst

dann, wenn die bestehenden Strategien nicht mehr funktionieren – oder der Schmerz des Vermeidens größer wird als der ursprüngliche Schmerz. Daher schleppen viele Menschen ihr Trauma ein Leben lang mit sich herum. Aber Hilfe ist möglich – und sie kommt nicht nur von Psychotherapeuten. Auch systemische Coaches oder ganzheitlich arbeitende Heilpraktiker können wertvolle Begleitung leisten, wenn sie dafür ausgebildet sind.

Sollten Sie selbst betroffen sein: Holen Sie sich Hilfe. Nicht weil Sie schwach sind – sondern weil Sie es verdient haben, dass das Leben leichter wird.

## Fazit

Der Umgang mit anderen Menschen kann anstrengend sein. Vor allem dann, wenn deren Art zu denken, zu arbeiten oder zu kommunizieren weit weg von unserer eigenen liegt. Aber genau dort beginnt Zusammenarbeit. Nicht im Gleichklang, sondern im Nebeneinander von Perspektiven.

Was uns in Konflikten trennt, sind oft keine Fakten, sondern Prägungen. Erfahrungen, Schutzstrategien, innere Bilder, die unser Gegenüber für uns schwierig erscheinen lassen – und umgekehrt. Unser Unterbewusstsein hilft uns dabei, uns zurechtzufinden. Aber manchmal bläst es Nebel ins Bild und lässt Situationen größer erscheinen, als sie tatsächlich sind.

Wenn wir lernen, diese inneren Reaktionen zu erkennen – und vielleicht sogar zu unterbrechen –, entsteht Raum für Verständnis, für neue Umgangsformen und für kleine, aber wirksame Veränderungen. Manchmal verändert sich dadurch der Kollege.

Oft reicht es aber schon, wenn sich die eigene Haltung verändert. Denn auch das gehört zur Wahrheit: Nicht jeder Mensch muss uns sympathisch sein. Aber jeder Mensch hat das gleiche Recht, gesehen und respektiert zu werden. Teams funktionieren nicht, weil alle gleich sind. Sondern weil Unterschiede ausgehalten und gestaltet werden. Und weil es Menschen gibt, die bereit sind, den ersten Schritt zu tun – nicht aus Selbstaufgabe, sondern aus Klarheit.

Am Ende ist es genau das, was gute Zusammenarbeit ausmacht: Nicht, dass es keine Konflikte gibt. Sondern dass wir wissen, wie wir ihnen begegnen. ■

[1] Tom Wendel, *Die unsichtbare Blockade*, dotnetpro 12/2024, Seite 15 ff.

[2] Tom Wendel, *Als Coden noch Bock machte*, dotnetpro 4-5/2025, Seite 130 ff.,

[3] Abraham Maslow, *Motivation and Personality*, Harper, 1954



**Tom Wendel**

ist freier Softwareentwickler und spezialisiert auf IT-Krisenberatung. Er unterstützt Unternehmen dabei, technische und organisatorische Herausforderungen zu lösen und ihre Entwicklerteams zu stärken.

[tom.wendel@wendelpunkt.de](mailto:tom.wendel@wendelpunkt.de)

# ARBEITSMARKT

## TRENDS UND JOBS FÜR ENTWICKLER

### Bitkom

#### Neue Jobs in der Digitalbranche

Trotz geopolitischen Unsicherheiten und schwierigem konjunkturellem Umfeld wachsen die Umsätze, und es entstehen neue Jobs in der Digitalbranche. So erwartet der Branchenverband Bitkom im deutschen Markt für IT und Telekommunikation (ITK) 2025 ein Umsatzplus von 4,4 Prozent auf 235,8 Milliarden Euro. Gleichzeitig soll auch die Beschäftigung in der Branche weiter steigen: um rund 9000 Jobs und damit von 1,345 auf 1,354 Millionen, siehe **Bild 1**.

Nach der aktuellen Bitkom-Prognose werden in Deutschland in diesem Jahr 161,3 Milliarden Euro mit IT umgesetzt, das entspricht einem Plus von 5,7 Prozent. Dominant ist dabei vor allem das Geschäft mit Software, das um 9,5 Prozent auf 52,7 Milliarden Euro zulegen soll. Innerhalb dieses Segments macht sich der Einfluss von Künstlicher Intelligenz und

Cloud Computing immer stärker bemerkbar.

Das Geschäft mit KI-Plattformen, auf denen KI-Anwendungen entwickelt, trainiert und betrieben werden können, wächst laut Bitkom um 50 Prozent auf 2,3 Milliarden Euro. Weit mehr als die Hälfte (58 Prozent) der Umsätze im Software-Markt werden bereits in Public Clouds generiert: 30,6 Milliarden Euro entfallen hierauf – ein Anstieg um 17 Prozent. Auch bei IT-Dienstleistungen machen Cloud-bezogene Services mit 19,3 Milliarden Euro bereits einen Anteil von 37 Prozent der Umsätze aus.

Bitkom-Präsident Dr. Ralf Wintergerst: „Die Cloud wird im Software- und Service-Markt zunehmend dominant.“

### Handelsblatt

#### Nachfrage nach KI-Skills steigt

Das „Handelsblatt“ hat über 760 000 Jobanzeigen der Agentur für Arbeit nach Erwähnung

gen der Begriffe „KI“ beziehungsweise „Künstliche Intelligenz“ gescannt und die Treffer nach Berufsgruppen geordnet. Das Ergebnis: In insgesamt 61 Berufsgruppen wurde bei Jobinseraten nach KI-Skills gefragt. Ganz oben in der Liste der gefundenen Nennungen von KI landeten in der Auswertung die Berufsgruppen Informatiker/in, IT-Berater/in und Softwareentwickler/in. Insgesamt fanden sich in den Top-25 zwölf Berufsgruppen mit IT-Bezug, vergleiche **Bild 2**. Oder anders gesagt: Wer einen IT-Job sucht, kommt schon heute kaum mehr ohne KI-Kenntnisse aus.

Zu ganz ähnlichen Ergebnissen kommt eine Auswertung der Jobbörse Indeed. Dafür wurde ausgerechnet, wie hoch der Anteil von Job-Anzeigen mit KI-Bezug für eine Berufsgruppe gemessen an der Gesamtzahl der Stellenausschreibungen zum Zeitpunkt der Erhebung war. Spitzenreiter hier: Softwareentwicklung vor Data Analytics und Informations-

management sowie IT-Support und IT-Infrastruktur.

### VDI

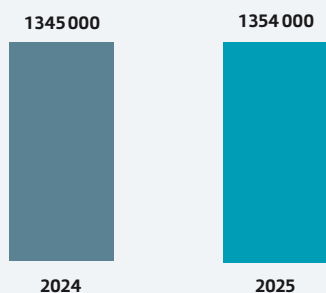
#### Engpass-Kennziffer sinkt

In seinem Bericht „Ingenieurmonitor 2024/IV“ hat der Verein Deutscher Ingenieure (VDI) auf die im Vergleich mit andern Ingenieurberufen besonders starke Zunahme der Arbeitslosigkeit bei Informatikberufen hingewiesen, für die zwischen September und Dezember letzten Jahres ein Indexanstieg um acht Punkte von 243,3 auf 251,3 ermittelt wurde. Zum Vergleich: Bei den Maschinen- und Fahrzeugtechnikberufen stieg der Index nur um knapp fünf Punkte von 111,6 auf 116,3, in Energie- und Elektrotechnik nur um 2,7 Punkte.

Gleichzeitig bleibt aber der Engpass an Ingenieuren weiterhin bestehen, weil deutlich mehr offene Stellen gezählt wurden als arbeitslos gemeldete Ingenieure. In den Informa-

### ITK-Arbeitsplätze

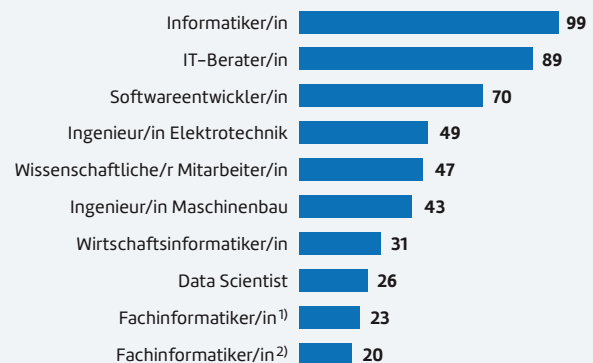
#### Prognose für 2025



Im laufenden Jahr soll die Zahl der Arbeitsplätze in der ITK-Branche laut Bitkom-Prognose um 9000 wachsen (**Bild 1**)

Quelle: Bitkom

### KI-Nennungen nach Berufsgruppen



Die meisten Nennungen von KI in Jobanzeigen wurden in Angeboten für Informatiker gefunden (**Bild 2**)

Quelle: Handelsblatt

<sup>1)</sup> Anwendungsentwicklung <sup>2)</sup> Systemintegration

# Zahl des Monats

Im vergangenen Jahr waren in Deutschland rund **1,52 Millionen** Fachkräfte in Berufen der Informatik, Informations- und Kommunikationstechnologie (IKT) tätig, das ist ein Zuwachs von vier Prozent. Dieser ist vor allem auf die sozialversicherungspflichtige Beschäftigung zurückzuführen. Mit **15 Prozent (165 000)** haben ausländische Fachkräfte hier einen relevanten Anteil am Wachstum.

Quelle: Bundesagentur für Arbeit

tikberufen ist die Engpass-Kennziffer – das Verhältnis zwischen Zahl der Arbeitslosen und den unbesetzten Stellen – in den letzten Jahren deutlich gesunken und liegt laut VDI mit 192 inzwischen unter dem Durchschnitt aller Ingenieur- und Informatikberufe.

Einen möglichen Ausweg aus der bestehenden Knappheit sieht der VDI darin, den bereits bestehenden Trend, Frauen für MINT-Berufe zu begeistern, weiter zu verstärken (Bild 3). Szenarioanalysen belegen laut VDI/IW-Ingenieurmonitor: Würde das Interesse von Mädchen an MINT-Berufen konsequent gefördert, könnte der Frauenanteil bis 2037 deutlich steigen und somit einen entscheidenden Beitrag zur Fachkräftesicherung leisten.

## Delphi

### Ein Umstieg könnte sich lohnen

In allen Branchen und Berufen in Deutschland steht ein Um-

bruch vor der Tür. Die geburtenstarken Jahrgänge – auch Boomer genannt – abgeleitet vom Babyboom in der Mitte der 1950er- bis zur Mitte der 1960er-Jahre – gehen alsbald in Rente und hinterlassen eine erhebliche Personallücke.

Geht es um Softwareentwickler, so lässt sich die Lücke dann leichter schließen, wenn der Neurentner mit Projekten befasst war, deren Technologien die Berufseinsteiger kennen, weil sie an der Uni gelehrt wurden – etwa Java oder C#.

Problematischer wird es für Personaler, die einen Nachfolger für jemanden suchen, der nicht mit einer weit verbreiteten Technologie gearbeitet hat, wie etwa Delphi. Wer sich rechtzeitig einarbeitet, könnte in dieser Nische eine gute Chance auf Verbesserung finden.

Um die Voraussetzungen für einen Umstieg zu schaffen, hat das Embarcadero-Team zusammen mit MVPs ein E-Book geschrieben und veröffentlicht, das Quereinsteigern, etwa von

C# und Java, einen Weg aufzeigen will, wie man schnell Delphi lernt. Das E-Book dient auch als Leitfaden für den Einstieg in RAD Studio und C++Builder.

Den kostenlosen Download des rund 130 Seiten umfassenden deutschsprachigen E-Books finden Sie im Web unter <https://lp.embarcadero.com/RADGettingStarted-DE> (Anmeldung erforderlich).

## Robert Half

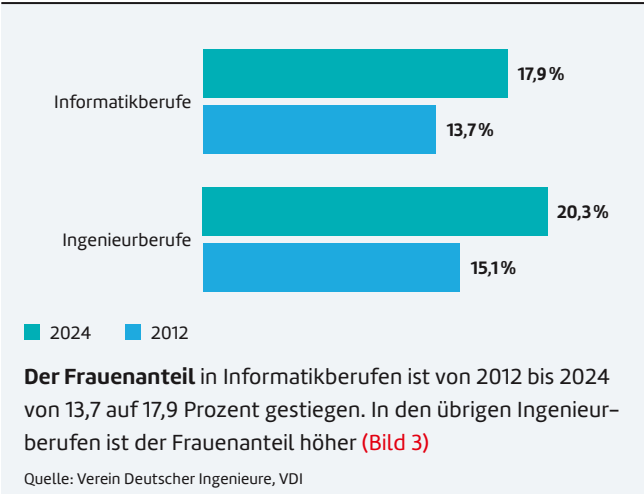
### Gehalt ist keine Privatsache mehr

Gehalt war lange ein individuelles Tabuthema. Doch dieser kulturelle Konsens bröckelt. Die Forderung nach mehr Transparenz in der Vergütung wird lauter. Das ist das Ergebnis einer repräsentativen Online-Umfrage unter 1002 erwerbstätigen Personen in Deutschland, welche der Personaldienstleister Robert Half zusammen mit dem Research- und Analystenhaus Techconsult durchgeführt hat. Die Ergebnisse: Über zwei Drit-

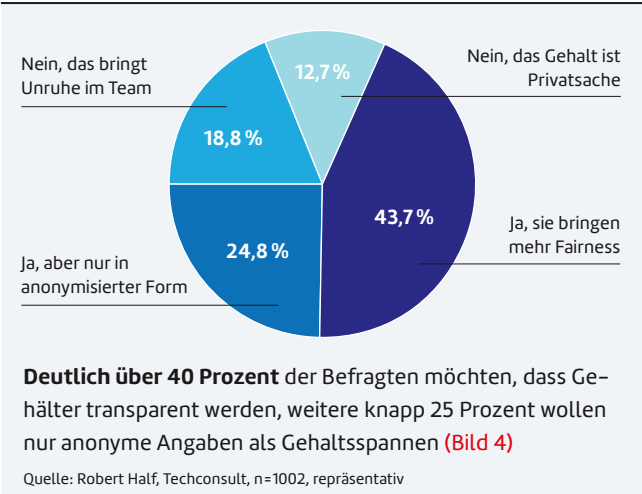
tel der Befragten sprechen sich für mehr Gehaltstransparenz aus – etwa 44 Prozent befürworten eine vollständige Offenlegung, rund 25 Prozent präferieren öffentlich einsehbare anonymisierte Gehaltsspannen, vergleiche Bild 4.

Besonders hoch ist der Wunsch nach Transparenz bei Millennials (Jahrgänge 1981 bis 1995), und Männer bevorzugen häufiger anonymisierte Modelle (30,1 Prozent) als Frauen (19,6 Prozent). Es gibt noch weitere Unterschiede zwischen den Generationen: Die Sorge, dass Offenheit zu Unruhe führt, ist bei Babyboomern (1956 bis 1965) und Generation X (1966 bis 1980) ausgeprägter als bei Millennials. Nur 12,7 Prozent der Gesamtstichprobe halten das Gehalt für eine private Angelegenheit – bei der Generation Z (ab 1995) sind es nur noch 8,7 Prozent im Vergleich zu 16,7 Prozent bei den Babyboomern. Der Trend ist eindeutig: Gehalt ist vor allem für die jüngeren keine Privatsache mehr.

## Frauen in Ingenieur- und Informatikberufen



## Transparente Gehälter





DWX 2025 IN MANNHEIM

# Neuer Standort, noch mehr Leidenschaft

1600 Entwickler:innen an vier Tagen, ein neuer Standort – und die Frage: Funktioniert das? Die Antwort lautet kurz und knapp: Ja, und wie!

**W**issen Sie, was passiert, wenn Entwicklerinnen und Entwickler vier Tage lang im Herzen Mannheims zusammenkommen? Es entsteht eine Art Paralleluniversum, in dem die Zeit anders tickt und jede Minute als doppelt so wertvoll erlebt wird. Genau das ist bei der DWX 2025 passiert.

Als wir vor Monaten beschlossen, die Developer Week erstmals nach Mannheim zu bringen, war das durchaus ein Wagnis. Würde die Community den neuen Standort annehmen? Würde die Infrastruktur stimmen? Und vor allem: Würde sich diese besondere DWX-Atmosphäre auch außerhalb des beeindruckenden NCC Ost der Messe Nürnberg entfalten? Antwort siehe oben: Sie hat. Und zwar volle Brause.

Denn die neue Location m:con Congress Center Rosengarten stellt Verbindungen auf ganz unterschiedliche Art her: Der vordere Teil wurde um 1905 herum erbaut, der hintere ist noch keine 20 Jahre alt – so kommen Geschichte und Gegenwart zusammen. Aber auch Alltag und Fortbildung: Der Rosengarten liegt im Zentrum Mannheims und bindet so die Wissensvermittlung auf höchstem Niveau an das tägliche Leben mit Hotels, Kneipen und Shopping.

Ob nun der eine oder andere Grund dominiert: Die DWX fühlte sich im neuen Zuhause wesentlich passender an. Das galt für die Workshops, für die drei Konferenztage mit bis zu zehn Themensträngen gleichzeitig, für die Ausstellung der Partnerunternehmen wie auch für die Abendveranstaltungen: Alles passte wunderbar in die Räumlichkeiten des Rosengartens.

Gebührend gefeiert wurde die neue Location mit der offiziellen Eröffnung der DWX: Mannheims Bürgermeister

**Grand opening:** Thorsten Riehle (Bürgermeister der Stadt Mannheim für Wirtschaft, Arbeit, Soziales und Kultur), Fernando Schneider (Conference Chair DWX), Marco Parrillo (Geschäftsführer Ebner Media Group) und Bastian Fiedler (Geschäftsführer m:con Congress Center Rosengarten) (von links) **(Bild 1)**

Thorsten Riehle, Bastian Fiedler, Geschäftsführer des m:con Congress Center Rosengarten, und Marco Parrillo, Geschäftsführer der Ebner Media Group, eröffneten die DWX 2025 ganz offiziell mit dem Durchschneiden eines Bandes **(Bild 1)**. „Willkommen in unserer Stadt“, sagte Bürgermeister Riehle, der in seiner kurzen Rede die Vielfalt und Offenheit von Mannheim herausstellte.

## Keynotes: Eine Zukunft, zwei Perspektiven

Nach der Eröffnung ist vor der ersten Keynote. Diese hielt Ricardo Cachucho von Pegasystems **(Bild 2)**. Er machte klar, worum es geht: „AI is here to stay.“ Nicht als Modewort oder Hype-Thema, sondern als fundamentaler Baustein unserer täglichen Arbeit. Seine Keynote „2030 – The Impact of Agentic Artificial Intelligence on Software and Low Code Development“ war eine Zeitreise in eine kaum mehr ferne Zukunft.

Besonders beeindruckend war seine Darstellung von Agentic AI – KI-Systemen, die nicht nur reagieren, sondern proaktiv handeln und Entscheidungen treffen. Ricardo zeigte auf, wie sich die Rolle von Entwicklern fundamental wandeln wird: weg vom reinen Coden, hin zum Orchestrieren intelligenter

Alle Fotos: Sofija De Mitri, Patrizio De Mitri, Event Wave

Systeme. „Wir werden zu Dirigenten digitaler Sinfonien“, brachte er es auf den Punkt. In seinem Kurzinterview nach der Keynote fasste er die drei wichtigsten Take-aways zusammen. Erstens: KI-Agenten werden unsere Arbeitsweise revolutionieren. Zweitens: Low-Code-Plattformen werden durch KI noch mächtiger. Und drittens: Wir Entwickler bleiben unverzichtbar, aber unsere Rolle wandelt sich vom Coder zum Architekten intelligenter Systeme.

Dr. Carola Lilienthal von der Hamburger WPS holte die Teilnehmerinnen und Teilnehmer in ihrer Keynote am zweiten Konferenztag dann wieder auf den Boden der Tatsachen zurück – auf sehr charmante und erhellende Art. „Die Devs sind immer schuld“, begann sie ihre Keynote und räumte gleich mit diesem ewigen Klassiker auf. Ihr Credo, das sie mit beeindruckenden Zahlen untermauerte: Je später ein Fehler entdeckt wird, desto teurer wirds.

Softwareentwicklung sei wie eine Expedition durch unbekannten Dschungel, erklärte sie: Man kennt das Ziel nicht genau, stolpert durch unwegsames Terrain und lernt unterwegs dazu. Deshalb ist Agilität da – nicht für die Iteration um der Iteration willen, sondern für echtes Lernen mit echtem Feedback. Früh testen, früh sprechen, früh korrigieren – das ist der Königsweg.

Ihr Plädoyer für automatische Tests und Tools wie archunit.org, um die Architektur sauber zu halten, kam bei den Zuhörern sehr gut an. Besonders ihre Vorstellung von Dual Track Scrum, um auch die User Experience in den agilen Prozess einzubinden, traf einen Nerv (Bild 2).

## Workshops: Systematisch tief eintauchen

Schon einen Tag vor der Eröffnung ging es im Rosengarten hoch her: Am Tag der Workshops konnten die Teilnehmer:innen zwischen Halb- und Ganztagesworkshops wählen. Von



Einen völlig neuen Blick auf Unit-Tests ermöglichte Stefan Dirschabel mittels 3D-Brillen (Bild 3)



**Keynoter** Ricardo Cachucho (links) betonte, dass die KI nicht mehr weggehen werde. Der Umgang damit sei Pflicht. In ihrer Keynote am zweiten Konferenztag hob Dr. Carola Lilienthal (unten) hervor, dass auch die User Experience von Anfang an mit in den agilen Prozess integriert werden muss (Bild 2)



Blazor über KI-Praxis bis hin zu Clean Code Development reichte die Bandbreite.

So begeisterten Christian Liebel und Marco Frodl mit dem Thema „AI-Praxis: Intelligente Business-Apps mit Angular, LangGraph und RAG“ die Zuhörer:innen – Hands-on, ganz praxisnah und mit dem gewissen Aha-Effekt.

Bei Marc Müllers Workshop „Mastering CI/CD Pipelines with Azure DevOps“ war das ähnlich. Kein Wunder – wer heute professionell entwickelt, kommt um solide CI/CD-Kenntnisse nicht herum. Marc führte die Teilnehmer von den Grundlagen bis zu fortgeschrittenen Deployment-Strategien.

Henning Schwentners halbtägiger Workshop zu Domain-Driven Design zeigte einmal mehr, warum DDD auch im KI-Zeitalter relevant bleibt. „Gerade in Zeiten von Microservices wird das Aufteilen der Domäne in Bounded Contexts immens wichtig. Domain-Driven Design ist hier ein entscheidender Schritt“, erklärte er treffend.

Neben diesen drei Sprechern lehrten aber auch noch Gregor Biswanger, Dr. Damir Dobric, Lukas Durovsky, Christian Giesswein, Thorsten Kansy, Peter Kröner, Lutz Marquardt, Jörg Neumann und David Tielke in kleinen Gruppen die Praxis der Softwareentwicklung.

## Themen: Die Qual der Wahl bei 150 Optionen

Für alle Teilnehmer:innen schlug an den drei Konferenztagen gnadenlos ein Luxusproblem zu: Bei täglich zehn parallelen Themensträngen und über 150 Sessions an drei Tagen hatte jeder die Qual der Wahl. Die Themen waren wie immer breit gestreut – von Software Architecture und Software Quality über CI/CD, agile Praxis und DevOps bis hin zu Web Frontend und Cloud Development. Einen großen Teil nimmt traditionell Microsofts .NET-Stack ein mit allen Tools und ►





Anhand einer Klopapierrolle zeigte Vicky Pirker, dass Klarheit der beste Freund des Users ist (Bild 4)

Frameworks. Sehr gut angenommen wurden aber auch die Vorträge zu Personal Growth und Sustainability. Und obwohl in vielen Sessions ebenfalls Thema, gab es zusätzlich selbstverständlich zwei explizite Tracks zu generativer KI.

Alle Sessions hier zu beschreiben würde den Rahmen dieses Berichts definitiv sprengen. Es seien deshalb exemplarisch nur vier herausgegriffen. Stefan Dirschnabels Session zu Test-Driven Development war ein multimedialer Kracher und zeigte, wie man auch trockene Themen spektakulär präsentieren kann – inklusive unerwartetem Drohnenflug im Saal und versteckten 3D-Brillen unter den Stühlen für eine völlig neue Sicht auf Unit-Tests (Bild 3). So macht Lernen Spaß!

Michael Kaufmanns „Deep-Dive into Git“ am letzten Tag war ein Muss für alle, die Git bisher eher per Trial-and-Error genutzt haben. Mike zeigte eindrucksvoll, dass auch altbekannte Tools noch ungeahnte Geheimnisse bergen. Von DAG über SHA-1 bis zu fortgeschrittenen Merge-Strategien – hier bekam man den nötigen Tiefgang verpasst.

Julia Kordicks Session „A (short) guide to Platform Engineering Excellence with Azure and GitHub“ traf den Zeitgeist. Sie erklärte überzeugend, warum wir DevOps neu denken müssen und wie Entwicklerteams durch passende Plattformen am besten unterstützt werden können.

Vicky Pirkers UX-Session wandte sich – mit einem Augenzwinkern – an alle, die glauben, Usability sei nur ein Nice-to-have. Sie legte charmant den Finger in die Wunde und wies auf Inkonsistenzen sogar in der neuen DWX-Website hin. Ihre Message: „Clarity is user's best friend“ – ein Grundsatz, den wir alle beherzigen sollten (Bild 4).

So vielfältig die Themen, so vielfältig waren auch die Herkunftsländer der Sprecherinnen und Sprecher. Sie kamen aus den USA, Kanada, den Niederlanden, Skandinavien und natürlich dem gesamten deutschsprachigen Raum. Die Mischung aus erfahrenen Veteranen und aufstrebenden Talenten sorgte für frische Perspektiven und neue Denkansätze.

Besonders schön war zu sehen, wie sich junge Talente neben etablierten Größen der Szene behaupteten. Die DWX ist schon lange nicht mehr nur eine Plattform für bekannte Namen, sondern ein Sprungbrett für die nächste Generation von Thought Leaders.

## Partner: Großartige Firmen, unzählige Innovationen

Die Ausstellung ist ein wichtiger Bestandteil der DWX. Mehr als zwei Dutzend Partnerunternehmen zeigten auf Ebene 1 ihre Produkte und Dienstleistungen: adesso SE, fecher, BenQ DACH, Bevel, Bundesverband Green Software, CODELUTION, Develappers, DevExpress, DICOTA, envite consulting, Everest Systems, Exxeta, Fast Reports, Impact Labs, Logitech, Massagesessel Welt, M&M Software, oose, pro-com DATENSYSTEME, Robot School, Rohde & Schwarz, Text Control, Uno Platform, UnternehmerTUM, Wendelpunkt, Xebia.

Für die Partner war der Impact außerordentlich – ein hundertprozentiger Match zur Zielgruppe der Developer brachte wertvolle Kontakte mit Menschen, die genau wissen, was sie brauchen. Moderne Frameworks, DevOps-Stacks, KI-Power und smarte Services, die den Alltag von Entwicklern, Architekten und Entscheidern richtig erleichtern. Alle Aussteller nutzten denn auch während der gesamten Konferenz die Gelegenheit, ihre Lösungen in ungezwungener Atmosphäre zu präsentieren. Ohne Zeitdruck und Terminstress entstanden so intensive Gespräche, die oft weit über das rein Fachliche hinausgingen.

Verlosungen von BenQ, Text Control, Fast Reports und DevExpress an den ersten zwei Abenden und in der Mittagspause des letzten Tages sorgten für Trauben von Teilnehmer:innen an den Ständen. So gingen unter anderem eine PlayStation 5, ein Rucksack mit Display und eine mobile Kaffeemaschine an die strahlenden Gewinner.

## Night Events: Der Developer lebt nicht nur von Wissen allein

Als es Abend wurde und die Aufnahmefähigkeit sank, war auf der DWX aber noch lange nicht Schluss. Die Casino Night am ersten Abend war ein Erlebnis der ganz besonderen Art. Echte Chips, professionelle Dealer und Entwickler:innen, die ihre besten Pokerfaces aufgesetzt hatten – eine Kombination, die für unvergessliche Momente sorgte (Bild 5).

Die routinierten Handbewegungen mancher Spieler:innen verriet: Hier waren durchaus ambitionierte Zocker am Werk, die ihre analytische Denkweise auch am Black-Jack-



Am ersten Abend schlug die Stunde der routinierten Spieler: In der Casino Night drehte sich alles ums Zocken bei Roulette, Black Jack und Poker (Bild 5)

Alle Fotos: Sofija De Mitri, Patrizio De Mitri, Event Wave



Ein anfangs skeptischer Dirigent Tom Wendel war zum Schluss von seinem Orchester mehr als begeistert (links). Beim anschließenden Partner Meet-up gab es viele Gespräche zwischen Teilnehmenden und Partnern (oben) (Bild 6)

Tisch anwendeten. Die Stimmung war ausgelassen, die Gespräche intensiv und die Vernetzung auf höchstem Niveau.

Am zweiten Abend mimte Tom Wendel in seiner Night Session „Code. Klang. Kontrollverlust“ einen Dirigenten, der alle Anwesenden zu Musikern eines Orchesters machte. Aber statt Violine, Horn oder Pauke gab es nur ein Instrument: das Smartphone. Tom brachte das gesamte Auditorium dazu, damit Musikstücke aus „Star Wars“ und „Fluch der Karibik“ zu spielen (Bild 6).

Die ersten Versuche schlugen fehl, was Tom zwar Applaus einbrachte, ihn jedoch auch kommentieren ließ: „Ich höre die Ironie im Klatschen bis hier hoch.“ Aber Aufgeben gab es nicht. Die Community ließ sich auf das Experiment ein, und zum Schluss war der Marsch aus „Fluch der Karibik“ klar zu erkennen.

Danach stand das Partner Meet-up an und rundete den Mittwochabend perfekt ab. In entspannter Atmosphäre konnten sich Teilnehmer:innen und Partner austauschen, neue Kontakte knüpfen und bei leckerem Essen und kalten Getränken die Eindrücke der vergangenen Tage Revue passieren lassen (Bild 6).

## Closing Panel: Überleben im KI-Zeitalter

Wo ein Anfang, da ein Ende: Im Closing Panel am letzten Tag wurde kontrovers über die Frage diskutiert: „AI without Getting Knocked Out – How to Thrive as a Software Developer in the AI Era?“ Sie beschäftigt alle Entwickler:innen in diesen Tagen. Christian Weyer (Thinktecture), Tobias Struckmeier (adesso), Mathieu Meisser (Logitech) und der selbstständige Coach und Consultant David Tielke diskutierten leidenschaftlich über unsere Zukunft als Developer. Dabei kristallisierte sich schnell heraus: Die Angst, von KI ersetzt zu werden, ist unbegründet – aber nur, wenn wir uns weiterentwickeln.

Die einhellige Meinung des Panels: Nur Prompts zu beherrschen reicht bei Weitem nicht. Wer zukunftsfähig bleiben will, muss verstehen, was unter der Haube passiert – sei es

bei Frameworks, Tools oder beim Code selbst. Mathieu Meisser brachte ein schönes Beispiel: „KI kann Code generieren, aber sie kann nicht verstehen, ob dieser Code auch in dein Geschäftsmodell passt.“

Christian Weyer schloss das Panel mit einem Satz ab, der noch lange nachwirken wird: „Embrace the new world.“ Nicht fürchten, sondern annehmen und aktiv gestalten. Das ist der Auftrag an die Entwickler-Community.

## Das Fazit: Ein Neuanfang, der nach Fortsetzung schreit

Mannheim war für viele eine echte Entdeckung. Die Stadt im Herzen der Metropolregion Rhein-Neckar überraschte mit ihrer Vielfalt, Lebendigkeit und exzellenten Verkehrsanbindung. Mittendrin das m:con Congress Center Rosengarten, das nicht nur alt und modern verbindet, sondern auch mit hervorragendem Essen, guter Technik und einem immer hilfsbereiten Team die Teilnehmenden versorgte.

Die DWX hat hier ein neues Zuhause gefunden und mit ihrem Programm, den Ausstellern und den Abendveranstaltungen die Teilnehmer:innen begeistert.

Viele neue Kontakte, intensive Gespräche mit Ausstellern, Sprecher:innen und Teilnehmer:innen, alte Bekannte und neue Gesichter – die DWX 2025 war ein voller Erfolg. Wer es nicht erlebt hat, hat etwas verpasst. Deshalb gilt: Wir sehen uns 2026 im Rosengarten. Das ist nicht nur ein Versprechen – das ist eine Herzensangelegenheit. ■



### Tilman Börner

war 20 Jahre Chefredakteur der dotnetpro. Heute unterstützt er das Team von Developer Education bei der Ebner Media Group. Daneben bringt er den Kolleginnen und Kollegen den Einsatz der KI nahe.

[tilman.boerner@ebnermedia.de](mailto:tilman.boerner@ebnermedia.de)





# LERNSTOFF

NEUE BÜCHER UND MEHR

## SCHRITT FÜR SCHRITT

### Copilot

Was mit generativer KI möglich ist, zeigt sich am schnellsten in der Praxis. Der Titel von Lisa Crosbie lädt dazu ein, es einfach mal zu probieren. Nach einer einleitenden Übersicht über diverse Versionen von Copilot und seine Einsatzbereiche im Web, auf mobilen Geräten und in Windows folgt eine intensive Lektion über das Prompten, also die Geschicklichkeit, der Maschine möglichst genau die Fragen zu stellen, die diese mit den gewünschten Ergebnissen erwidern kann. Einfache Fragen lassen sich dabei ergänzen durch Hinweise zum Kontext; hilfreich ist auch die Kombination mit Quellangaben oder Angaben darüber, wie das Ergebnis gestaltet sein soll. Dabei werden sowohl Prompts für Textergebnisse als auch für Bilder in vielfältiger Weise vorgestellt.

Im weiteren Verlauf geht es um den Einsatz von Copilot beim Surfen im Web und beim Chatten mit Mobilgeräten. Die Zusammenarbeit mit dem Edge-Browser, die Nutzung der Edge-Randleiste, des Designers und der Think-Deeper-Funktion für ausführlichere Ergebnisse werden anhand von Bei-



**Lisa Crosbie**

Microsoft Copilot. Schritt für Schritt  
mitp, 2025, 352 Seiten, Deutsch,  
ISBN 978-3-7475-1024-7, 34,99 Euro

Zielgruppe: KI-Entwicklung

Voraussetzungen: keine

Leseprobe: [www.mitp.de/out/media/9783747510247\\_Leseprobe\\_Microsoft\\_Copilot.pdf](http://www.mitp.de/out/media/9783747510247_Leseprobe_Microsoft_Copilot.pdf)

spielen vorgeführt. Die Copilot-App, die sich auch per Sprach-eingabe nutzen lässt, die Verwendung von Kamerabildern in Prompts und die Nutzung von Ergebnissen für weitere Nachfragen eröffnen weitere Optionen, von Copilot zu profitieren.

Der dritte Teil stellt die Kombination von Copilot mit Word, Excel, PowerPoint, Outlook und OneNote vor. Texte lassen sich nicht nur generieren, sondern auch bearbeiten, zusammenfassen oder analysieren. Copilot kann in Excel beispielsweise Vorschläge für Formeln machen und bei der Visualisierung von Daten helfen. (vh) ■

## SCHNITTSTELLEN ENTWERFEN

### API-Design

Das Buch bietet eine praxisnahe Einführung in das facettenreiche Feld des API-Designs. Es vermittelt nicht nur grundlegende Konzepte und strategische Überlegungen, sondern liefert auch konkrete Hinweise und Codebeispiele zur Gestaltung von Java-APIs. Ein besonderes Augenmerk liegt zudem auf Remote-Schnittstellen, etwa RESTful HTTP, GraphQL und Messaging, die in modernen Webanwendungen und verteilten Systemarchitekturen wichtig sind.

Nach einer Einführung zu Nutzen und Struktur von APIs werden API-Qualitätskriterien beschrieben – darunter Konsistenz, Erweiterbarkeit und intuitive Nutzbarkeit – sowie ein strukturierter Entwurfsprozess vom Szenario bis zum Review vorgestellt. Im Java-Bereich behandelt das Buch API-Typen, saubere Schnittstellengestaltung und fortgeschrittene Techniken wie Fluent Interfaces und Immutability mit Blick auf Wiederverwendbarkeit und Robustheit. Auch auf Ebene der Remote-APIs wird das Wissen vertieft: REST-Prinzipien, URI-



**Kai Spichale**

API-Design. Praxishandbuch für Java- und Webservice-Entwickler

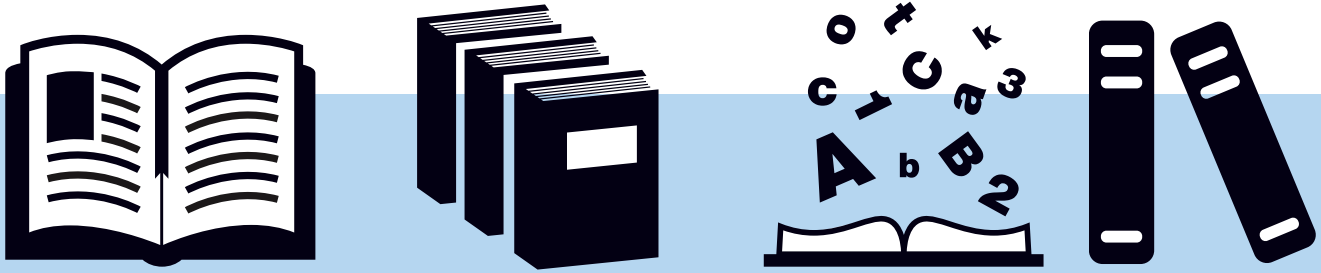
dpunkt.verlag, 3. Auflage 2025,  
414 Seiten, Deutsch,  
ISBN 978-3-98889-034-4, 44,90 Euro

Zielgruppe: API-Entwickler

Voraussetzungen: Java-Kenntnisse

Leseprobe: [www.assets.dpunkt.de/leseproben/14167/Leseprobe.pdf](http://www.assets.dpunkt.de/leseproben/14167/Leseprobe.pdf)

Design und Sicherheit stehen hier ebenso im Fokus wie die Integration asynchroner Kommunikation über Messaging-Systeme. Abgerundet wird das Ganze durch übergreifende Themen wie zielgerichtete Dokumentation, strategisches Caching und API-Management. In der dritten Auflage sind unter anderem der API-first-Ansatz und Sicherheitsthemen wie Authentifizierung, API-Keys, Distributed Denial of Service (DDoS) und Injection-Angriffe hinzugekommen. (bl) ■



## VON KONZEPTION BIS BETRIEB

# Mobile Systeme

Dieser Titel stellt in wohldosierter Form die grundlegenden Kenntnisse zusammen, die für Einrichtung und Betrieb mobiler Lösungen mit Smartphones, Tablets, Clouds beziehungsweise Datenbanken benötigt werden. In den Blick nimmt der Autor dabei den gesamten Lebenszyklus mobiler Systeme und alle wichtigen Dimensionen – nicht nur die technologischen Grundlagen, sondern auch die Aspekte der Sicherheit, der Ethik und der Verträglichkeit mit der Umwelt.

Die ersten Kapitel beschreiben die vielfältigen Komponenten, Technologien und Dienste, die für mobile Lösungen zusammenspielen müssen. Die verschiedenen Gerätetypen bis hin zu IoT und die marktbeherrschenden Betriebssysteme werden kompakt charakterisiert.

Die aktuell verfügbaren Entwicklungsframeworks für Apps und die Oberflächengestaltung sind der nächste Schwerpunkt des Buches, der ergänzt wird durch eine Einführung in das Application Lifecycle Management sowie die Gestaltung der Distribution.



**Florian Bliesch**

Mobile Systeme. Konzeption, Entwicklung und Betrieb

Hanser, 2025, 579 Seiten, Deutsch, ISBN 978-3-446-48276-0, 49,99 Euro

Zielgruppe: Softwareentwicklung

Voraussetzungen: Keine

Leseprobe: [www.bic-media.com/widget/?isbn=9783446483910](http://www.bic-media.com/widget/?isbn=9783446483910)

Das Thema Sicherheit wird anhand einiger exemplarischer Beispiele behandelt. Die Bereiche, in denen KI bereits ganz selbstverständlich zur Anwendung kommt, werden kurz vorgestellt. Die Aussichten mobiler Geschäftsmodelle werden anhand verschiedener Systeme skizziert.

Zudem ist auch dem Thema Metaverse noch ein eigenes Kapitel gewidmet, und auch Green IT sowie Green Coding werden anhand von zwei Fallstudien demonstriert. Abschließend wirft der Autor einen Blick auf die Chancen und Risiken der mobilen Zukunft. (vh)

## Kurz vorgestellt

**Bernhard Schloß, Christian Botta**

### Das Methodensystem für Projekte

Systematische Darstellung von Projektmanagement-Methoden. Alle Methoden, Werkzeuge und Frameworks werden nach den Kriterien Aufwand, Schwierigkeit und Wirksamkeit bewertet. Mit grafischer Übersicht sowie Tipps und Tricks.

Hanser, 2025, 330 Seiten, Deutsch, ISBN 978-3-446-48367-5, 39,99 Euro

**Maria Korneeva**

### Barrierefreie Webentwicklung

Anhand anschaulicher Beispiele werden die Grundlagen der digitalen Barrierefreiheit und die relevanten rechtlichen Rahmenbedingungen und Richtlinien vermittelt, darunter das Barrierefreiheitsstärkungsgesetz (BFSG), die Web Content Accessibility Guidelines (WCAG) und die EN 301 549.

O'Reilly, 2025, 416 Seiten, Deutsch, ISBN 978-3-96009-253-7, 39,90 Euro

**Kaveh Bashiri**

### Quantencomputing

Der Leitfaden gibt einen Einblick in Grundlagen und Anwendungsszenarien des Quantencomputings. Der Leser lernt das theoretische Fundament kennen, bevor es um Quantenalgorithmen geht. Versucht wird ein praxisorientierter Zugang mit Beispielen, die man in Qiskit nachvollziehen kann.

Rheinwerk, 2025, 373 Seiten, Deutsch, ISBN 978-3-8362-9848-3, 39,90 Euro

**Kristian Köhler**

### Software design und Entwurfsmuster

Das Handbuch vermittelt, wie sich rund 30 Patterns und Designprinzipien auf der Code-Ebene mit Ansätzen auf Architektur-/Strukturebene verbinden lassen – ergänzt durch Code-Beispiele, Best Practices und Diagramme.

Rheinwerk, 2025, 517 Seiten, Deutsch, ISBN 978-3-367-10259-4, 39,90 Euro

# Highlights in der Ausgabe 10–11/2025



## Focus:

.NET – die Nummer 10

## Weitere Themen:

- API Contracts perfekt im Griff
- Semantic Flow
- Agentic AI in DevOps-Pipelines
- AI-infused Test Data Builders
- SQL Server 2025
- PostgreSQL
- Kubernetes und Minikube
- Gamification in der Praxis

Foto: Shutterstock / JJ-studio

developer-world.de



## Inserentenverzeichnis

Ebner Media Group

www.ebnermedia.de

2, 53, 57–61, 97, 105, 111, 121, 147, 148

## Impressum

### dotnetpro by DWX

#### MEDIENHAUS

Ebner Media Group GmbH & Co. KG  
Karlstraße 3  
89073 Ulm

info@ebnermedia.de  
www.ebnermedia.de

#### GESCHÄFTSFÜHRUNG

Annabel Ebner, Marco Parrillo

DE147041097  
Sitz: Ulm, Amtsgericht Ulm, HRA 1900

Persönlich haftende Gesellschafterin der  
Ebner Media Group GmbH & Co. KG  
ist die Ebner Ulm MGV GmbH, Ulm.

Sitz: Ulm, Amtsgericht Ulm, HRB 576

#### OFFICE

Ebner Media Group GmbH & Co. KG  
Office Köln  
Helmholtzstraße 29 – 31  
50825 Köln  
Telefon: +49 731 88005-8000  
www.dotnetpro.de

#### VERANTWORTUNG CONTENT

##### Chefredaktion

Fernando Schneider  
fernando.schneider@ebnermedia.de

##### Redaktion

Tilman Börner  
tilman.boerner@ebnermedia.de

##### Externe Autor\*innen

Klaus Aschenbrenner, Florian Bader,  
Daniel Basler, Stefan Bechtel, Tilman Börner,  
Christian Giesswein, Olaf Heß, Bernhard Lauer,  
Florian Lenz, Stefan Lieser, Stefan Mintert,  
Robert Mühsig, Bernhard Pichler, Tobias Richling,  
Lars Roith, Dr. Holger Schwichtenberg,  
Christian Sender, David Tielke, Helmut Vonhoegen,  
Tom Wendel, Christian Wenz

### Art Direction

Maria-Luise Steinkühler  
malu.steinkuehler@ebnermedia.de

### Gestaltung

EMG DESIGN UNIT  
DesignUnit@ebnermedia.de

### Publisher

Marco Parrillo  
marco.parrillo@ebnermedia.de

### MEDIA

#### Media Sales

Thomas Deck  
thomas.deck@ebnermedia.de

#### Client Success (Disposition)

Anja Büttner  
clientsuccess@ebnermedia.de

#### Stellenmarkt

Sabine Vockrodt  
jobs@ebnermedia.de

### HERSTELLUNG, VERTRIEB & KUNDENSERVICE

#### Leitung Herstellung, Vertrieb & Kundenservice

Thomas Heydn  
thomas.heydn@ebnermedia.de

#### Kundenservice

Telefon: +49 731 88005-8205  
kundenservice@ebnermedia.de

#### Bezugspreise

Die jeweils gültigen Bezugspreise für unsere  
Print- und Digitalausgaben sowie für eventuelle  
Abonnements entnehmen Sie bitte den Angaben auf  
unserer Website oder den jeweiligen Bestellseiten.  
Alle Preise verstehen sich inklusive der gesetzlichen  
Mehrwertsteuer und zuzüglich ggf. anfallender  
Versandkosten, sofern nicht anders angegeben.

Preisänderungen bleiben vorbehalten. Bei  
bestehenden Abonnements gelten die bei  
Vertragsschluss vereinbarten Konditionen, sofern  
keine anderweitige Regelung getroffen wurde.

### Widerrufsrecht

Sie haben das Recht, binnen vierzehn Tagen ohne  
Angabe von Gründen diesen Vertrag zu widerrufen.

Die Widerrufsfrist beträgt vierzehn Tage ab dem Tag,  
an dem Sie oder ein von Ihnen benannter Dritter,  
der nicht Beförderer ist, die erste Ware in Besitz  
genommen haben.

Um Ihr Widerrufsrecht auszuüben, genügt  
eine eindeutige Erklärung, mit der Sie uns über  
Ihren Entschluss, diesen Vertrag zu widerrufen,  
informieren. Bitte richten Sie diese an:

Ebner Media Group GmbH & Co. KG  
Office München  
Brienner Straße 45a-d  
80333 München  
Telefon: +49 731 88005-8205  
E-Mail: hilfe@ebnermedia.de

Sie können dafür z. B. einen Brief oder eine E-Mail  
verwenden. Zur Wahrung der Widerrufsfrist reicht es  
aus, dass Sie die Mitteilung über die Ausübung des  
Widerrufsrechts vor Ablauf der Frist absenden.

### Urheberrecht / Copyright

Alle Inhalte, Werke und Publikationen – ob online  
oder im Print – der Ebner Media Group GmbH & Co. KG  
unterliegen dem deutschen Urheberrecht.

Jegliche Vervielfältigung, Bearbeitung, Verbreitung,  
öffentliche Zugänglichmachung oder sonstige  
Nutzung außerhalb der gesetzlichen Schranken des  
Urheberrechts ist ohne ausdrückliche schriftliche  
Genehmigung durch die Ebner Media Group GmbH  
& Co. KG nicht gestattet. Downloads, Scans oder  
Kopien sind ausschließlich für den privaten, nicht  
kommerziellen Gebrauch erlaubt.

Soweit Inhalte nicht vom Verlag selbst erstellt  
wurden, werden die Urheberrechte Dritter beachtet  
und entsprechend gekennzeichnet.

### DRUCK

Silber Druck GmbH & Co. KG  
Otto-Hahn-Straße 25  
34253 Lohfelden



DIE KONFERENZ FÜR AI,  
CLOUD, WEB & .NET DEVELOPMENT

# DWX

29. JUNI BIS  
02. JULI 26

M:CON CONGRESS CENTER  
ROSENGARTEN | MANNHEIM

SAVE  
THE DATE!

DEVELOPER  
WORLD



developer-world.de | #DWX



# .NET DEVELOPER CONFERENCE

Die Konferenz für .NET-Entwickler:innen

**24. BIS 27.  
NOVEMBER '25**  
Pullman Cologne Hotel

**10%  
Rabatt**  
mit  
DNPAboSpecial

Software Design | Software Architecture | Software Quality  
Software Testing | Frontend Development | GenAI and Agentic  
Data & Analytics | .NET | Approaches | Backend Technologies  
Cloud & Infrastructure

