06

Sommer 2024

aware

Nachhaltigkeit in der Software-Entwicklung

OKR: Management by Team

Bicep: Wir lassen die Muskeln spielen

Unsere Kollegin **Angela im Interview**

aware 06

Wir sind Expertinnen und Experten für digitale Lösungen rund um Web-Entwicklung, Apps, Cloud-Computing und künstliche Intelligenz.
Wir haben Spaß daran, komplexe Herausforderungen zu lösen und Abläufe zu vereinfachen.
Wir gehen mit Leidenschaft ans Werk und lieben Projekte mit Nervenkitzel und Impact.
Vor allem aber sind wir eines:
Ein Team aus tollen Menschen.

#verständnis

Mit Kreativität und rascher Auffassungsgabe tauchen wir in Ihre Welt ein. Wir stellen viele Fragen und wollen verstehen, was wir entwickeln.

#vielfalt

Es geht um mehr als nur Nullen und Einsen. In unserer täglichen Arbeit bekommen wir Einblicke in verschiedenste Themenbereiche: Vom Rettungswesen bis zu Finanzdienstleistern. Von der Bauwirtschaft bis zur Entsorgungstechnik. Vom Bioladen bis zum Großkonzern. **Diese Vielfalt zeigt sich nicht nur in unseren Lösungen, sondern auch in unserem Team.** Wir sind neugierige Menschen mit einer breiten Palette an Interessen und Perspektiven. Daran wollen wir andere teilhaben lassen.

#leichtigkeit

Es darf auch mal analog sein:
Wir schätzen **Momente der Entschleunigung** und hoffen, dass diese Seiten dazu beitragen, dem virtuellen Trubel kurz zu entfliehen.

softaware

Da sind wir wieder!

Nach über 5 Jahren Pause haben wir uns entschieden, unsere Firmenzeitung „aware“ wieder aufleben zu lassen. Weil wir viel zu erzählen haben. Weil die Anmut eines liebevoll gestalteten, angreifbaren Magazins besser zu uns passt als der hundertste E-Mail-Newsletter. Weil wir Menschen, die sich für unsere Arbeit interessieren könnten, regelmäßig informieren möchten.

Unsere Titelgeschichte dreht sich diesmal rund um nachhaltige Softwarelösungen. Nein, wir messen nicht den CO2-Verbrauch (auch das wäre spannend), sondern wir machen uns Gedanken, was die Lebensdauer von Software positiv beeinflussen kann.

Oft muss es am Beginn von Projekten schnell gehen: Endlich wurde nach Jahren die interne Entscheidung getroffen, dass eine Software in Auftrag gegeben wird – und dann müssen rasch Vergleichsangebote her, damit ein Anbieter ausgewählt werden kann. Für ein gegenseitiges Kennenlernen und das bewusste Auseinandersetzen mit den Anforderungen und dem Qualitätsverständnis bleibt zu wenig Zeit. Hauptsache, der Stundensatz ist bekannt. Ähnlich nachhaltig wie Plastikspielzeug – und ähnlich haltbar. Tobias liefert daher in seinem Artikel alternative Vorschläge für eine gelungene Vorprojektphase und ein nachhaltiges Ergebnis.

Technischer wird es bei Daniel und seinem Artikel über „Infrastructure as Code“. Für alle Technikerinnen und Techniker: Viel Spaß! Für alle anderen: Trotzdem lesen, Daniel erklärt auf nachvollziehbare Art und Weise, wie wir moderne und komplexe Anwendungen online bringen.

Außerdem erzählen wir noch, wie wir unser Unternehmen mit der Unterstützung des ganzen Teams kontinuierlich verbessern, wer bei uns die Rechnungen schreibt, was ein Restaurantbesuch mit Cloudcomputing zu tun hat und wie man aus einem Labyrinth herausfindet.

Ich freue mich über Feedback – und natürlich auch über neue Projekte, bei denen wir unsere Kompetenz beweisen können!

Viel Spaß beim Lesen und einen schönen Sommer!

Roman Schacherl

IMPRESSUM

Medieninhaber, Herausgeber, Verleger: softaware gmbh **Schriftleitung:** Roman Schacherl
Redaktion: Tobias Berei, Kathrin Enzenhofer, Gabriele Käferböck, Roman Schacherl, Daniel Sklenitzka
– alle: Ziegelweg 2, 4481 Asten, Telefon +43 7224 654 71
Gestaltung: Kathrin Enzenhofer, UI/UX Designerin der softaware gmbh
Die mit Namen gekennzeichneten Beiträge geben nicht unbedingt die Meinung der Redaktion wieder.

Angela Hofer

Angela, wie bist du zu softaware gekommen?

Ich war die erste Angestellte im Einzelunternehmen smilecompany, das mein Bruder Roman 2004 gegründet und 2012 in die softaware gmbh umgewandelt hat. Als mich Roman um Unterstützung bei der Buchhaltung gebeten hat, war alles noch klein und überschaubar. Seither wachse ich mit meinen Aufgaben...

Welche Aufgaben hast du bei softaware?

Ich bin für Buchhaltung und Rechnungswesen verantwortlich, d.h. ich erstelle und bezahle Rechnungen, verbuche die Belege am Monatsende, führe die Umsatzsteuererklärung durch und überweise ans Finanzamt, bereite den Jahresabschluss vor und (ganz wichtig!) überweise rechtzeitig die Gehälter [lacht]. Der zeitliche Aufwand ist schwankend, ich bin aber derzeit noch geringfügig beschäftigt und teile mir die Arbeit nach Bedarf frei ein.

Was machst du dabei am liebsten – und was nicht so gerne?

Freude macht mir, den Überblick zu bewahren und wenn am Monatsende der Kassastand stimmt. Die pünktlich fällige Umsatzsteuererklärung macht mir manchmal Zeitstress, weil dafür alle Belege da sein müssen, was nicht immer ganz einfach ist.

Was machst du außerhalb von softaware gern, wie verbringst du deine Freizeit?

Mein weiteres Leben ist mit vielen anderen Aufgaben gefüllt: Meine wichtigste Rolle dabei ist die als Mutter unserer beiden jugendlichen Söhne. Außerdem engagiere ich mich viele Stunden ehrenamtlich, vor allem in unserer Pfarre, wo ich u.a. als Mesnerin tätig bin und die Familienmessen vorbereite und in der Flüchtlingshilfe, wo ich seit acht Jahren eine afghanische Familie mit drei Kindern unterstütze.

Dein liebstes Klischee über Softwareentwickler:innen?

Ich kenne alle unsere Entwickler:innen von Beginn an und habe hier nie diese typischen Einzelkämpfer erlebt, die still in ihrem Kammerl sitzen und nur digital kommunizieren. Im Gegenteil, es gibt in der softaware extrem viel Austausch, Spaß und Miteinander, das gefällt mir sehr.

A wer

[ugs.; dt.: **auch jemand**]

🎂 47 Jahre

🏡 Hofkirchen

💰 CAM
(Chief Account Manager)

Kaffee oder Tee?

Kakao

Haus oder Wohnung?

Passivhaus

Rad oder Auto?

stolze e-Up! Besitzerin seit fast 10 Jahren

Physik üben oder Belege buchen?

eindeutig Urlaub buchen 😉

16. Mai 2024

Daniel, Michi und Tobias sind auf der #sddconf in London! 🚀
In zahlreichen Workshops und Vorträgen werden die neusten Trends und Innovationen der Software-Entwicklung präsentiert. Wir sind gespannt auf ihre Erkenntnisse und wünschen ihnen jede Menge Spaß. Go team! 💻

#TechConference #LondonTech

20. Juni 2024

Ultimativer Rennspaß im Rotax MAX Dome! 🏎️💨
Nach intensivem Training und spannendem Qualifying hat Dominik Dörr im Rennen als Erster die Ziellinie überquert! 🏁🔥

#Speed #Racing #DomDoerr #RotaxMAXDome #Karting

08. Mai 2024

Ob uns unsere neue Siebträger-Kaffeemaschine einen eignen Post Wert ist? ... Absolut! 🤎☕
Wenn der Kaffee aus dieser Maschine fließt, schreibt sich der Code schon fast von selbst.

#Kaffeeliebe #Power

23. Mai 2024

Top-Company Award 2024! 🏆🎉
Ausschlaggebend für diese besondere Ehre sind die Bewertungen auf kununu, bei denen wir einen beeindruckenden Score von 4,9/5 erzielt haben. ⭐
Mit dieser Auszeichnung gehören wir zu den 5% der besten Arbeitgeber im deutschsprachigen Raum! 😍

#TopCompany #TopEmployer #Award #kununu

19. Juli 2024

Wir lieben Innovationen! 💛💡
Bei einem Werksbesuch bei der Doka durften wir sehen, wie innovative Produkte und effiziente Lösungen für die Bauindustrie entstehen. 🔨🚧
Vielen Dank für die Einladung!

#Schalungssysteme #Doka

@softawaregmbh

Die ersten Jahre sind die wichtigsten. Bei Software wie bei Kindern.

Dieser gemeinsame Nenner hat uns dazu bewogen, das Ennser Eltern- und Familienzentrum Bunter Kreis finanziell zu unterstützen.
Den privaten Verein gibt es seit mehr als 25 Jahren als fixen Bestandteil des sozialen Lebens in Enns und wichtigen Treffpunkt für Familien. Ziel des Ennser Eltern-Kind-Zentrums ist es, Eltern und Kinder auf ihrem gemeinsamen Weg von der Schwangerschaft bis zur Pubertät zu begleiten. Das finden nicht nur die Jungväter unter uns besonders unterstützenswert!

Wir waren dabei!

Wir sind der festen Überzeugung, dass diverse Teams besser funktionieren und haben uns vorgenommen Impulse zu setzen, um Frauen auf unseren spannenden Job aufmerksam zu machen.

Eine dieser Initiativen ist der jährliche Girls' Day, der heuer am 25. April stattgefunden hat. Mädchen im Alter von 13 bis 14 Jahren konnten bei uns einen spannenden Arbeitstag erleben, um am Ende sogar die mit Kathrin designte und mit Tobias entwickelte App am eigenen Smartphone ausprobieren.

Uns hat der Tag viel Spaß gemacht und wir freuen uns, falls ein paar Mädels ihre Begeisterung für einen zukünftigen Beruf entdecken konnten.

FC Perg Ladies

und unser Team teilen Technik und Teamgeist.

Wir sind stolz, Sponsor des ersten reinen Frauenfußballvereins im Mühlviertel zu sein!

„Insgesamt trägt der Frauenfußball dazu bei, sozialen Zusammenhalt durch Teamarbeit, Gemeinschaftsgefühl, Inklusion, Vielfalt und positive gesellschaftliche Wahrnehmung zu fördern. Diese Faktoren tragen dazu bei, eine unterstützende und inspirierende Umgebung für Frauen im Fußball zu schaffen."
- https://www.fc-perg-ladies.at

Der Schlüssel zu **nachhaltigen Softwarelösungen**

von Tobias Berei

Wenn wir von Nachhaltigkeit sprechen, denken wir nicht an Softwareprojekte, sondern an nachhaltig produzierte Kleidung oder umweltfreundliche Verkehrsmittel. Dennoch spielt dieses Thema in der Softwareentwicklung seit jeher eine entscheidende Rolle. Auch wenn die Softwarebranche schnelllebig ist, will man schließlich nicht alle zwei bis drei Jahre eine neue Software für dasselbe Problem entwickeln und einführen - ganz im Gegenteil. Wer besonderes Augenmerk auf nachhaltig entwickelte Softwarelösungen setzt, profitiert von wirtschaftlichen Vorteilen und geringeren Aufwänden für Wartung und Weiterentwicklung, wie wir nahezu täglich feststellen.

Wir beobachten häufig, dass der wesentlichste Aspekt für langfristig eingesetzte Software in ausreichender Planung und einer geeigneten Vorprojektphase liegt. Durch die umfangreiche Beleuchtung der Anforderungen sowie des Projektumfelds (Unternehmensdomäne, Anwender:innen, Arbeitsweisen, etc.) kann die Software an die speziellen Bedürfnisse angepasst werden, sodass nicht nur eine neue Software eingeführt wird, sondern auch eine echte Arbeitserleichterung oder Modernisierung von Abläufen stattfindet. Durch „Chemistry Meetings“, Anforderungsworkshops sowie eine UI/UX Designphase können hierbei beeindruckende Ergebnisse erzielt werden. Auf die traditionelle Herangehensweise, nämlich die aufwändige Erstellung eines Spezifikationsdokumentes, verzichten wir zunehmend und denken, dass die Zeit anders besser investiert ist. Anstatt unzählige Seiten mit schwer verständlichen Funktionsbeschreibungen zu produzieren, sprechen wir lieber ausführlich über die Anforderungen und tauchen tiefer in die Anwendungsdomäne ein, um danach die bestmögliche Lösung zu finden.

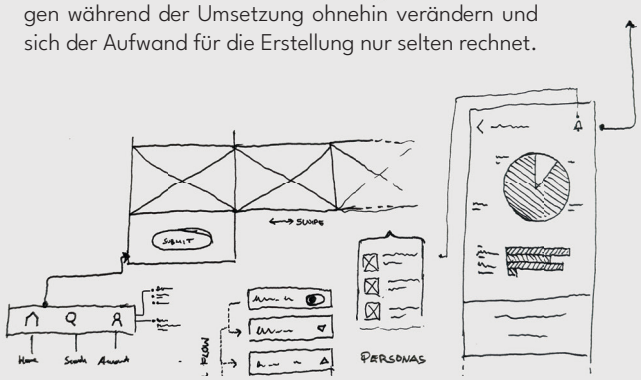
Ein Blick in die Glaskugel

Am Beginn von Softwareprojekten steht oft die Frage nach den erwarteten Kosten – und das völlig zurecht. Schließlich kaufen wir im täglichen Leben auch Waren, deren Preis wir im Vorfeld kennen oder vereinbart haben. Leider lässt sich dieser Gedanke nicht 1:1 für Softwareprojekte umlegen. Moderne Softwaresysteme und deren Anforderungen sind oft so komplex, dass eine exakte Schätzung vorab sehr viel Zeit in Anspruch nehmen würde. Neben der Frage, wer die Kosten dafür trägt, müssen wir uns auch fragen, ob diese Zeit wirklich gut investiert ist. Schließlich können sich Anforderungen auch während des Softwareprojekts ändern. Insbesondere wenn auch ein Forschungsanteil vorhanden ist, kann der benötigte Aufwand noch schwerer eingeschätzt werden. Gerade aufgrund dieser Rahmenbedingungen profitieren wir in der Durchführung vom agilen Vorgehensmodell, welches uns erlaubt auch während der Umsetzung neue Wünsche aufzunehmen oder auf Ideen zu reagieren. Gleichzeitig nehmen wir jedoch in Kauf, dass die Software fast nie vollständig spezifiziert und durchgeplant ist, weshalb sich dies erst während der Umsetzung ergibt. Spätestens jetzt wird klar, dass deshalb eine genaue Abschätzung der Kosten in frühen Phasen nur ein sehr grober Indikator sein kann (siehe auch „Cone of Uncertainty“ in Projekten).

Deshalb investieren wir unsere Zeit zu Beginn eines Softwareprojekts lieber in „Chemistry Meetings“. Dabei lernen wir uns gegenseitig kennen, hören uns die Wünsche sowie Aufgabenstellung an, stellen viele Fragen und tauchen sukzessive tiefer in das Themengebiet des Projekts sowie das Projektumfeld ein. So erhalten beide Parteien schon sehr frühzeitig einen umfangreichen Eindruck vom Gegenüber, um etwaige Hindernisse zu erkennen und zu beseitigen. Ziel ist es gegenseitiges Verständnis und Vertrauen aufzubauen, damit einer erfolgreichen Vorprojektphase nichts im Weg steht.

Erfolgsfaktor Vorprojektphase

Auch wenn die genaue Vorhersage der Kosten für die Umsetzung in frühen Phasen schwer ist, kann durch eine effiziente Vorprojektphase ein wertvoller Know-How-Aufbau stattfinden, wodurch in weiterer Folge eine genauere Abschätzung der Kosten möglich ist. Aus unserer Sicht ist die Vorprojektphase, bestehend aus Anforderungsworkshops und der Erstellung eines visuellen Prototyps, der Garant für die korrekte Ausrichtung des Softwareprojekts. Die Zusammenarbeit von Design und Entwicklung während dieser Phase ist besonders wichtig, da so die entwickelten Designideen nicht nur seitens des Projektpartners, sondern auch durch unser Entwicklungsteam begutachtet und diskutiert werden können. So stellen wir sicher, dass wichtige Fragen für die Umsetzung schon vor Entwicklungsstart geklärt werden. Die Erstellung von Pflichtenheften oder Spezifikationen gehört immer seltener zu unseren Vorprojektphasen, zumal sich die Anforderungen während der Umsetzung ohnehin verändern und sich der Aufwand für die Erstellung nur selten rechnet.



Einen besonderen Mehrwert für das Projekt stellt die Entwicklung eines UI/UX Designs bzw. eines visuellen Klick-Prototypen dar. Fast immer wird die zu entwickelnde Software durch frühe UI-Sketches erstmalig „greifbar“ und vermittelt einen realistischen Eindruck vom Endprodukt. Aus diesem Grund entstehen zu diesem Zeitpunkt auch die meisten neuen Ideen und Wünsche – und das ist auch gut so! Da jetzt noch keine Zeile Quellcode getippt wurde, sind Änderungen und Erweiterungen der Anwendung vergleichsweise günstig (im Vergleich zur Entwicklung). Insbesondere wenn die Software über viele Jahre im Einsatz sein soll, ist die Entwicklung eines UI/UX Designs unerlässlich, sodass sich auch spätere Erweiterungen nahtlos in die Benutzeroberfläche einfügen können.

Kurz gesagt:

UI (User Interface) **ist das Aussehen,**
UX (User Experience) **ist das Gefühl!**

Immer wenn jemand mit einem (digitalen) Produkt interagiert, entsteht ein Nutzererlebnis. Es geht also darum, wie sich das Produkt anfühlt, denn die UX umfasst alle Wahrnehmungen und Emotionen während der Nutzung. Dazu gehören auch Aspekte wie User Research, die Erstellung von Wireframes und Workflows.
Die UI beschreibt dagegen, wie das Produkt aussieht. Das betrifft das visuelle und ästhetische Design, also Dinge wie Farben, Typographie, Grafiken und weitere visuelle Elemente.

Der Weg zur nachhaltigen Software

Wir denken, dass der Schlüssel für nachhaltige Softwarelösungen in einer auf das jeweilige Projekt ausgerichteten Vorprojektphase liegt. Auch wenn hier anfängliche Aufwände anfallen, bevor es mit der Umsetzung losgeht, rentiert sich diese Investition spätestens ab Inbetriebnahme oder der ersten Erweiterung der Software. Vielmehr müssen wir für eine realistische Einschätzung also den „Total Cost of Ownership“ über die gesamte Lebensdauer betrachten. Unsere Praxiserfahrung bestätigt, dass Lösungen, die unter diesen Gesichtspunkten entwickelt wurden, langfristig nicht nur kostengünstiger sind, sondern aufgrund der Designphase auch besser von den Anwender:innen angenommen und gerne genutzt werden. Durch den Einsatz moderner Technologien und geeigneter Softwarearchitekturen steht einer nachhaltig eingesetzten Softwarelösung somit nichts mehr im Wege.

Software und … **Kochen**

von Roman Schacherl

Ich koche gerne.
Vom Schweinsbraten bis zum Spinat-Feta-Strudel, von der Vorspeise bis zum Dessert. Nach getaner Arbeit die Kochlöffel schwingen und in Ruhe essen: Ja, Kochen ist mein Hobby.

Ich gehe aber auch gerne essen. In kleiner oder großer Runde, Altbekanntes oder Ausgefallenes, beim Wirt im Ort oder im Haubenlokal.

Zuhause selbst zu kochen hat unbestrittene Vorteile: Ich kenne alle Zutaten und deren Preis, ich entscheide über das Rezept und die Würzung und ob das anschließende Essen bei Kerzenlicht oder Partybeleuchtung stattfindet. Aber: Ich bin kein professioneller Koch. Und mein Kochen skaliert nicht: Weder meine Küche noch meine Wohnung sind ausgerichtet für ausgiebige Geburtstagsfeiern mit 30 Gästen.

Professionelle Köche arbeiten anders. Effizienter. Mit besseren Kochtöpfen und anderen Geräten. Das mir servierte Kürbisrisotto ist kein „Versuch“, der Küchenchef hat es schon hunderte Male gekocht und schrittweise verfeinert. Ich kann am Vorabend kurzfristig anrufen und einen Zweiertisch oder eine Tafel für 12 Personen reservieren. Und jeder Gast wählt sein Essen nach Belieben und bezahlt nur, was er bestellt – im Zweifelsfall eine günstigere Kinderportion.

> Selbst zu kochen ist On-Premise,
> Essen gehen ist Cloud.

Wir vertrauen als Software-Entwickler:innen seit 13 Jahren auf die Cloud. Wir schätzen die Professionalität der dort beschäftigten Köche, die große Speisekarte an verfügbaren Diensten, die unterschiedlichen Tischgrößen und die Art der Bezahlung. Wir können neue Technologien kosten oder auf das gewohnte Schnitzel setzen. Und es stört uns üblicherweise nicht, wenn am Nebentisch andere Gäste das Restaurant besuchen – gegen Aufpreis übersiedeln wir aber sogar in einen eigenen Raum.

Wer in der Software-Entwicklung Angst davor hat, dass seine Suppe versalzen wird, muss selber kochen. Dann aber bitte in ähnlicher Professionalität, mit ausgebildeten Köchen und entsprechenden Räumlichkeiten. Alle anderen sollten essen gehen – außer es geht ums Hobby:
Ich freu mich nämlich auf heute Abend.

Mahlzeit!

Wir lassen die **Muskeln spielen**

von Daniel Sklenitzka

Bereit für die Cloud

95% der Anwendungen, die wir entwickeln, laufen mittlerweile in der Cloud, genauer gesagt in Azure[1]. Das bietet uns und unseren Kund:innen viele Vorteile: Viel höhere Sicherheit, als bei einem lokalen Server jemals möglich wäre, schnellere Entwicklungszyklen, mächtige Werkzeuge und natürlich die Möglichkeit, die Infrastruktur genau an den Bedarf der Anwendungen anzupassen: Ist die Software nur tagsüber in Betrieb? Kein Problem, dann fahren wir die Server nachts herunter und sparen so Betriebskosten. Sind in den nächsten Tagen Auslastungsspitzen zu erwarten? Okay, dann nutzen wir genau für den Zeitraum ein paar zusätzliche Server.

Damit so eine dynamische Skalierung reibungslos funktioniert, muss eine Software allerdings auch von Beginn an darauf ausgelegt sein, was eine gewisse Komplexität mit sich bringt: Wenn z.B. die Möglichkeit besteht, dass die Anwendung auf mehreren Servern gleichzeitig läuft, können Bilder oder Dokumente nicht einfach im Dateisystem des Servers abgespeichert werden, sondern müssen in einem externen Speicher abgelegt werden. Daneben gibt es meist zumindest noch eine Datenbank und je nach Projekt verschiedene weitere Systeme. Eine typische sogenannte „cloud-native" Anwendung umfasst also mehrere Komponenten (sogenannte Ressourcen), die alle korrekt miteinander „verdrahtet" werden müssen.

Handarbeit

Die einfachste Möglichkeit, eine Ressource in Azure zu erstellen, ist mittels der Benutzeroberfläche des entsprechenden Portals (Abbildung 1). Mit wenigen Klicks lassen sich hier Datenbanken, Web-Server & Co einrichten und konfigurieren. Diese Vorgehensweise mag zwar auf den ersten Blick schnell und einfach wirken, hat jedoch auch einige Nachteile:

Nachvollziehbarkeit: Änderungen sind später nicht oder nur schwer nachzuvollziehen: Wer hat wann warum welche Konfigurationseinstellungen vorgenommen?

Konsistenz: Die ad-hoc-Vorgehensweise führt beinahe zwangsläufig zu verschiedenen Schreibweisen bei der Benennung.

Wiederholbarkeit: Soll eine weitere Kopie der Anwendung zur Verfügung gestellt werden, z.B. für ein (zusätzliches) Entwicklungssystem, für Last- oder Security-Tests, die Produktivumgebung, etc., geht das nur durch mühsames manuelles Nachbauen.

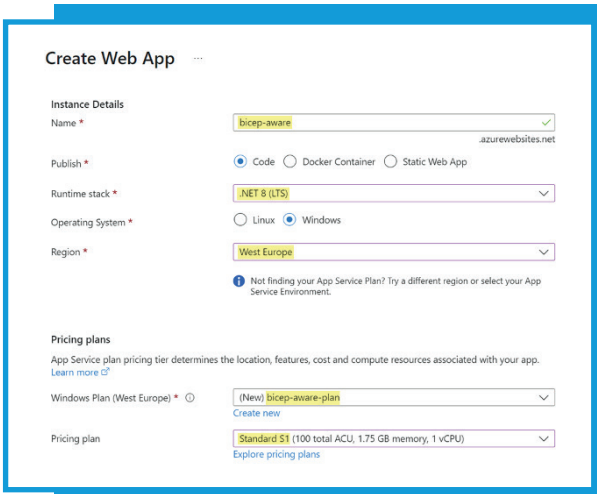


Abbildung 1:
Manuelles Erstellen einer Web-Anwendung über die Benutzeroberfläche

Infrastructure as Code

Besser ist es daher, die Infrastruktur-Komponenten einer Anwendung automatisiert zu erstellen. Man „programmiert" also nicht nur den Anwendungscode, sondern auch die Erzeugung der benötigten Infrastruktur – daher der Name „Infrastructure as Code" oder kurz „IaC".

Diese Vorgehensweise hat folgende Vorteile:

Versionierung: Der IaC-Code kann ebenso wie der Anwendungscode in die Versionsverwaltung eingecheckt werden. Dadurch sind sämtliche Änderungen nachvollziehbar und alte Versionen können jederzeit wiederhergestellt werden.

Dokumentation: Der IaC-Code dient gleichzeitig als Dokumentation der Architektur, weil auf einen Blick ersichtlich ist, welche Kompnenten es gibt und wie sie zusammenspielen.

Wiederverwendbarkeit: Die Erzeugung der verschiedenen Komponenten kann modular gestaltet und so leicht projektübergreifend wiederverwendet werden.

Konsistenz: Durch die Wiederverwendung der Templates ist auch die Konsistenz hinsichtlich Namensgebung und Konfiguration über mehrere Projekte hinweg einheitlich. Das hilft Entwickler:innen, sich in neuen Projekten schnell zurecht zu finden.

Effizienz: Darüber hinaus ist es natürlich auch effizienter, wenn man ein Setup aus einem anderen Projekt einfach übernehmen kann, anstatt wieder bei Null mit der manuellen Konfiguration zu beginnen.

Wiederholbarkeit: Neue Umgebungen lassen sich mit IaC auf Knopfdruck in Minuten erstellen. Damit lassen sich auch Fehler vermeiden, die auf unterschiedliche Einstellungen in Entwicklungs- und Produktivsystem zurückzuführen sind.

Es gibt verschiedene technische Möglichkeiten, IaC zu implementieren. Die Basis dafür bildet in Azure eine REST-Schnittstelle[2]. Diese kann entweder direkt, über die Azure Command Line oder über PowerShell aufgerufen werden. Dabei muss man jedoch imperativ vorgehen, die Ressourcen also nacheinander und in der richtigen Reihenfolge erzeugen.

Besser deklarativ

Sogenannte ARM-Templates (für Azure Resource Manager) stellen im Gegensatz dazu einen deklarativen Ansatz dar. Man beschreibt also nicht im Detail, wie die einzelnen Ressourcen zu erstellen sind, sondern nur, was man alles benötigt – also den Zielzustand. Der Azure Resource Manager kümmert sich dann darum, diesen Zustand herzustellen – egal, ob dazu neue Komponenten erstellt oder bestehende anders konfiguriert werden müssen.

Bei den ARM-Templates handelt es sich um ein JSON[3]-basiertes Dateiformat, in dem die benötigten Ressourcen und ihre Einstellungen beschrieben werden. Das Format ist somit (zumindest theoretisch) nicht nur für Maschinen, sondern auch für Menschen einigermaßen gut lesbar. In der Praxis zeigt sich leider, dass das nur bedingt zutrifft. Die Syntax ist teilweise recht umständlich und beinhaltet viele Sonderzeichen, manche Dinge lassen sich darüber hinaus in der JSON-Struktur nicht besonders gut abbilden (Abbildung 2).

Was ist jetzt also der Königsweg?
Hier kommt nun endlich das titelgebende Bicep-Format in Spiel.

```json
"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
"contentVersion": "1.0.0.0",
"parameters": {
  "location": {
    "type": "string",
    "defaultValue": "westeurope"
  }
},
"resources": [
  {
    "type": "Microsoft.Web/serverfarms",
    "apiVersion": "2015-09-01",
    "name": "bicep-aware-plan",
    "location": "[parameters('location')]",
    "sku": {
      "name": "S1"
    },
    "properties": {
    }
  },
  {
    "type": "Microsoft.Web/sites",
    "apiVersion": "2022-09-01",
    "name": "bicep-aware",
    "location": "[parameters('location')]",
    "properties": {
      "serverFarmId": "[resourceId('Microsoft.Web/serverfarms', 'bicep-aware-plan')]",
      "siteConfig": {
        "metadata": [
          {
            "name": "CURRENT_STACK",
            "value": "dotnet"
          }
        ],
        "netFrameworkVersion": "v8.0"
      }
    },
    "dependsOn": [
      "[resourceId('Microsoft.Web/serverfarms', 'bicep-aware-plan')]"
    ]
  }
]
```

Abbildung 2:
Erstellen einer Web-Anwendung mittels ARM-Template

```
param location string = 'westeurope'

resource server 'Microsoft.Web/serverfarms@2022-09-01' = {
  name: 'bicep-aware-plan'
  location: location
  sku: {
    name: 'S1'
  }
}

resource webApp 'Microsoft.Web/sites@2022-09-01' = {
  name: 'bicep-aware'
  location: location
  properties: {
    serverFarmId: server.id
    siteConfig: {
      metadata: [
        {
          name: 'CURRENT_STACK'
          value: 'dotnet'
        }
      ]
      netFrameworkVersion: 'v8.0'
    }
  }
```

Abbildung 3:
Erstellen einer Web-Anwendung mit Bicep

Bicep

Bicep (zu Deutsch: Bizeps; sozusagen eine stärkere Version der ARM-Templates) ist eine sogenannte domänen-spezifische Sprache. Es ist also eine (Programmier-) Sprache, die für einen speziellen Zweck – nämlich die Definition von Azure-Ressourcen – optimiert wurde. Im Gegensatz zu den ARM-Templates, die auf das allgemeine JSON-Format setzen, gibt es hier eigene Konstrukte, die das Anlegen der verschiedenen Komponenten auf einfache Art und Weise ermöglichen. So können etwa Schleifen genutzt werden, um wiederkehrende Aufgaben durchzuführen. Die Syntax ist merkbar entschlackt (siehe Abbildung 3) und bietet alle Komfortfunktionen, die man sich als Entwickler:in wünscht (Syntax-Highlighting, Auto-Complete, ...).

Fazit

Bei allen Vorteilen bringen Cloud-Anwendungen auch eine gewisse Komplexität mit sich. Für einfache Projekte mag es zu Beginn ausreichen und sogar schneller sein, die notwendigen Ressourcen manuell anzulegen – sobald es etwas umfangreicher wird, setzen wir aber auf jeden Fall auf Infrastructure as Code. Dank einer mittlerweile umfangreichen Bibliothek von wiederverwendbaren Bicep-Templates können wir ohne großen Aufwand von den diversen Vorteilen, von Nachvollziehbarkeit über Konsistenz bis hin zu Zeitersparnis, profitieren.

[1]Azure ist der Name der Cloud-Umgebung von Microsoft. Derzeit umfasst Azure über 300 Rechenzentren weltweit, die durch ein globales Netzwerk verbunden sind.

[2]REST bezeichnet eine (mehr oder weniger) standardisierte Art von Schnittstellen, die aus jeder beliebigen Programmiersprache und Umgebung verwendet werden kann.

[3]JSON steht für „JavaScript Object Notation" und ist ein Dateiformat für den Datenaustausch zwischen Anwendungen. Da es sich um ein textbasiertes Format handelt, ist es – im Gegensatz zu binären Protokollen – grundsätzlich auch von Menschen lesbar.

Wie wir **planbar besser** werden

von Roman Schacherl

Es gibt Dinge, die sind dringend und müssen erledigt werden. Eine E-Mail-Antwort bis spätestens heute Abend. Ein Versions-Update Ende der Woche. Den Jahresabschluss bis Ende September veröffentlichen (wobei, das nehmen nicht alle so genau, wie wir aus den Medien entnommen haben). Und dann gibt es Dinge, die sind wichtig – aber es passiert auch nichts, wenn sie nicht sofort erledigt werden: Wir sollten unseren Social Media-Auftritt neugestalten. Wir möchten KI-Tools in unserer täglichen Arbeit etablieren. Wir würden gerne überlegen, wie wir Frauen für Technik-Jobs motivieren können. Es drängt uns dabei nichts: Wir könnten heute damit anfangen, oder in zwei Wochen. Und plötzlich ist ein Jahr vergangen und man denkt sich: Warum ist noch nichts passiert?

Objectives and Key Results

Seit 2022 leben wir in unserem Team „OKR“. OKR steht für „Objectives and Key Results” (Ziele und Messwerte) und hilft uns dabei, kontinuierlich besser zu werden. Die Methode existiert seit den 1970er Jahren, wurde aber vor allem durch Google einem breiteren Publikum bekannt. Am Beginn stehen die Objectives: Es werden drei Ziele vorgestellt, die gerade jetzt für unser Unternehmen wichtig sind. Jede und jeder im Team ist anschließend eingeladen, über eine einfache Online-Umfrage Ideen und Bedenken zu sammeln – und sich zu überlegen, ob man bei diesem OKR-Thema gerne intensiver mitarbeiten möchte.

Das nach der Umfrage entstehende OKR-Kernteam trifft sich und entscheidet selbstständig über die Key Results: Wie sollen wir am Ende messen, ob wir besser geworden sind? Was wollen wir erreichen? Was einfach klingt, ist oft eine Herausforderung – und eine bereichernde Diskussion. Nicht alles, was sich messen lässt, ist ein gutes Key Result: Nur, weil wir fünf neue Blogposts online haben, bringt uns das nicht zwingend weiter – sie müssen auch gelesen werden. Key Results sollen keine „To-Do-Listen“ werden, sondern die tatsächliche Veränderung messen: Besser also tatsächliche Seitenaufrufe, als erstellte Blogposts zählen.

3 Monate, dann ist Schluss

Dann startet die eigentliche Umsetzung – und wieder ist das ganze Team eingeladen, sich zu beteiligen. OKR ist kein Werkzeug, um die Leistung einzelner Mitarbeiterinnen oder Mitarbeiter zu bewerten: Wir wollen gemeinsam besser werden. Die Umsetzungsphase dauert – und das ist das Besondere – nur drei Monate. Ein kurzer Zeitraum, in dem man fokussiert versucht, etwas „auf den Boden zu bringen“. Keine Luftschlösser, was irgendwann einmal besser gemacht werden könnte, sondern konkretes Anpacken.

Wir wollen gemeinsam besser werden!

Nach drei Monaten endet der Zyklus mit einem „OKR-Showdown“ im Rahmen eines Team-Frühstücks. Wir messen das Erreichte und halten fest, was wir gelernt haben. Oft ergeben sich aus den Themen so genannte „Health-Metriken“ – also Kennzahlen, die wir langfristig im Auge behalten wollen, damit die Verbesserung auch nachhaltig ist.

Nach OKR ist vor OKR

Drei Monate sind schnell vorbei. Wir haben uns entschieden, zwischen den Zyklen einen „Übergangs-Monat“ einzuführen, in dem die alten Themen sauber abgeschlossen und die neuen geplant werden. Es gibt somit von Jänner bis März, von Mai bis Juli und von September bis November drei Zyklen mit jeweils drei Themen – macht neun Themen pro Jahr. Die urlaubsbedingt ohnehin etwas reduzierten Monate August und Dezember sind bewusst ausgespart.

Einige Beispiele für Objectives der letzten Zeit:

- ✔ **Holen wir uns Leuchtturm-Projekte!**
- ✔ **Werden wir fit für die Auswirkungen von KI auf unser Unternehmen**
- ✔ **Lernen wir aus unseren Projekten**
- ✔ **Lernen wir, wie gut unsere Kundenbetreuung derzeit ist**
- ✔ **Verbessern wir unsere Attraktivität als Arbeitgeber für Frauen**

Eines der neuen Objectives, mit dem wir uns aktuell beschäftigen, ist:

- ✔ **Beweisen wir, dass wir skalierbare Anwendungen bauen können.**

Sollten Sie zu den Themen Anknüpfungspunkte haben, freuen wir uns über Ihre Kontaktaufnahme!

Was sich geändert hat

Viele aus unserem Team freuen sich, wenn sie etwas zur Unternehmensentwicklung beitragen können. Durch die kurze Zykluszeit ist das Engagement auch gut abschätzbar: Es ist keine Lebensentscheidung, ob man sich mehr in Richtung Management entwickeln möchte – es ist eine überschaubare Phase, in der ein paar Stunden pro Woche einem anderen Thema gewidmet werden.

Mit OKR haben wir eine Methode gefunden, die uns hilft, auch den zeitlich unkritischen, aber für den langfristigen Erfolg essenziellen Themen den notwendigen Raum zu geben.
Und plötzlich ist ein Jahr vergangen und man denkt sich:

▶ Wow, was wir alles geschafft haben.

Gefangen im **Labyrinth**

von Daniel Sklenitzka

Ein Roboter soll so programmiert werden, dass er ausgehend von einem Startpunkt den Weg durch ein Labyrinth zu einem vorgegebenen Ziel findet. Er kann sich in 90° Schritten nach links und rechts drehen (Befehle TURN LEFT bzw. TURN RIGHT) und (aus seiner Sicht) nach vorne bewegen (FORWARD). Ein erster Lösungsentwurf sieht folgendermaßen aus:

```
1  FORWARD        14 FORWARD
2  FORWARD        15 FORWARD
3  TURN LEFT      16 TURN LEFT
4  FORWARD        17 FORWARD
5  TURN RIGHT     18 TURN LEFT
6  FORWARD        19 FORWARD
7  FORWARD        20 FORWARD
8  TURN RIGHT     21 FORWARD
9  FORWARD        22 TURN RIGHT
10 FORWARD        23 FORWARD
11 FORWARD        24 TURN RIGHT
12 FORWARD        25 FORWARD
13 TURN RIGHT
```

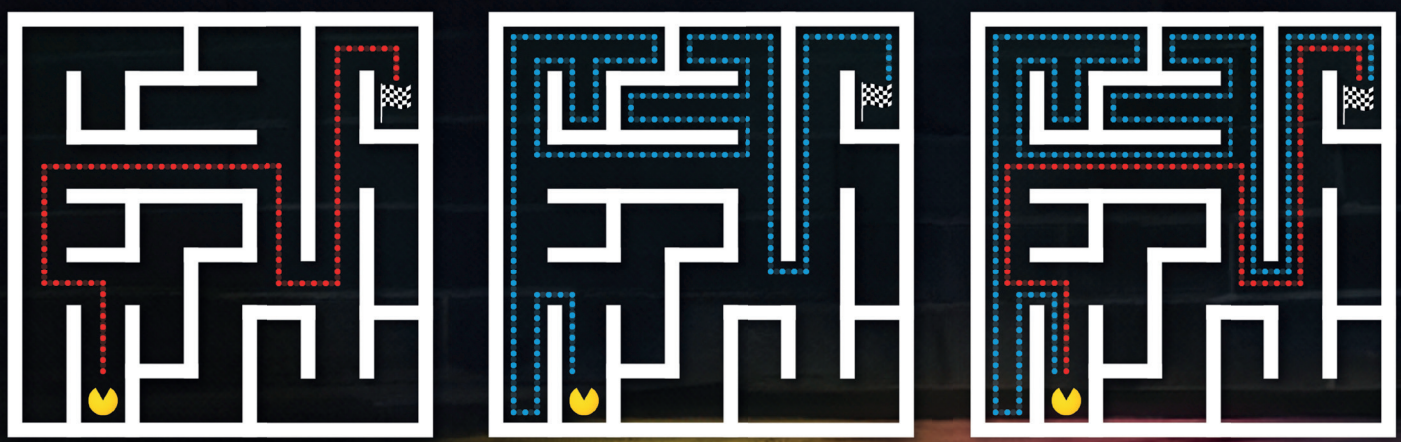
Ein anderer Ansatz unter Verwendung von zwei weiteren Befehlen ist kürzer, aber komplexer: Mittels IF CAN MOVE kann festgestellt werden, ob der Roboter direkt vor einer Wand steht. Der nächste Befehl wird nur dann ausgeführt, wenn kein Hindernis im Weg ist und er freie Fahrt hat. Mit dem Befehl GO TO kann zu einer beliebigen Stelle im Programm gesprungen werden. GO TO **4** sorgt also z.B. dafür, dass als nächstes der Befehl in Zeile 4 (und alle darauf folgenden) ausgeführt werden.

```
1  TURN LEFT
2  IF CAN MOVE
3  GO TO 8
4  TURN RIGHT
5  IF CAN MOVE
6  GO TO 8
7  GO TO 4
8  FORWARD
9  GO TO 1
```

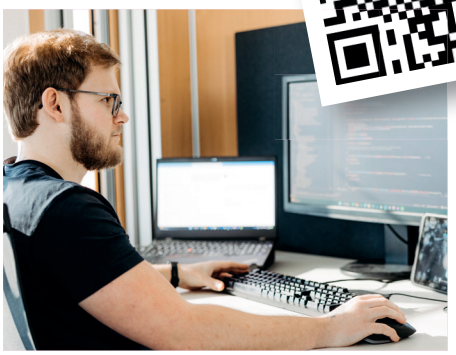
Die erste Variante ist genau auf die Problemstellung zugeschnitten, die zweite ist etwas allgemeiner und funktioniert auch in anderen Labyrinthen. Mit beiden landet der Roboter früher oder später im Ziel. Welcher Ansatz ist nun besser? Das kommt auf die Definition von „besser“ an:

- **Welche Version ist schneller entwickelt (und damit billiger)?**
- **Welche ist einfacher zu verstehen und zu warten?**
- **Wie lange braucht man nach einer Änderung zum Testen, um die Korrektheit sicherzustellen? *)**
- **Welche Variante ist fehleranfälliger (was passiert z.B., wenn der Roboter mit einer anderen Blickrichtung startet)?**
- **Mit welcher kommt der Roboter schneller ans Ziel? Ist das ein Kriterium?**
- **Was passiert, wenn sich eine Kleinigkeit am Layout des Labyrinths ändert? Was bei einem komplett neuen Labyrinth, das vielleicht zehn mal so groß ist?**

* Eine der beiden Versionen enthält übrigens einen Fehler. Haben Sie ihn bemerkt?



Lösung: Im ersten Lösungsansatz fehlt auf der langen Geraden kurz vor dem Ziel ein FORWARD Befehl (ab Zeile 19).

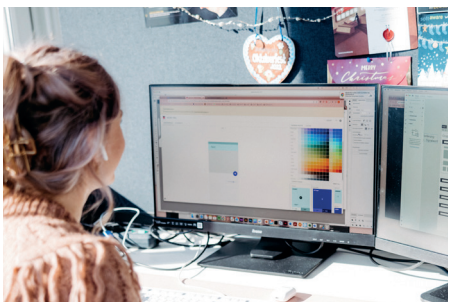


Haben Sie eine Aufgabe für uns?

Wir sind **Software-Expert:innen** und haben Spaß daran, **komplexe Herausforderungen** zu lösen und **Abläufe zu vereinfachen.**

www.**softaware**.at

✉ office@softaware.at



soft aware

Aware of your ideas.
Developing your software.