Microservice-Bingo

Und habe nun, ach, Microservices (BINGO) durchaus studiert? Viele kleine, eigenständige Module geformt und eine Applikation gebaut?

Da sie unabhängig voneinander agieren, kann man sie einzeln entwickeln, testen, austauschen, warten und erweitern. Das ist eigentlich das Idealbild der Softwareentwicklung. Clean Code (BINGO) vom Feinsten. Oder nicht?

So sexy dieses Design Pattern (BINGO) ist, denke ich doch immer gleich an den notwendigen Overhead (BINGO). Voneinander unabhängige Systeme müssen für alles selbst sorgen: Persistenz (BINGO), Message Queues (BINGO), Konfiguration (BINGO). Außerdem muss man sich genau überlegen, wie man sie orchestriert (BINGOBINGO). Würde jeder Service (BINGO) das selbst implementieren, würden die Micro- schnell zu Macroservices wachsen. Folglich bedarf es einer Laufzeitplattform (BINGO), die diese allgemeinen Dienste bereitstellt. Aber steht das nicht im Widerspruch zur Unabhängigkeit?

Mit Verlaub: Nein. Denn bei Microservices geht es um den Code, der für jeden Service zu schreiben ist. Der soll übersichtlich bleiben und deshalb die Anzahl an Code-Zeilen klein sein. Genannt werden Zahlen wie 250, 100 oder zehn.

Ein Microservice sollte also maximal zehn Zeilen Code umfassen? Lustig!

Dass er dann von weiteren Bibliotheken abhängig ist – muss wohl so sein. Frage: Ist das dann nicht hyperkonvergent? (BINGOBINGOBINGO)

Mächtig beeindruckt hat mich die Technologie, die Robert Eichenseer in seinem Artikel beschreibt. Service Fabric (BINGO) ist eine solche Laufzeitumgebung für Microservices. Das Besondere daran ist aber nun, dass es nicht nur – wie ja zu erwarten – in der Cloud auf Microsoft Azure läuft, sondern auch auf dem Windows Server 2016. Sie können die Laufzeitumgebung also auch bei sich in der Firma hosten. Damit aber nicht genug: Gemäß Microsofts neuer Offenheit (BINGO) gibt es diese Laufzeitumgebung auch für Linux (BINGO).

Wollen Sie also eine Anwendung in Betrieb nehmen, so können Sie das auf drei sehr unterschiedlichen Plattformen – mit denselben Microservices, wohlgemerkt. Die Freiheit (BINGO) schlechthin. Auf Visual Studio (BINGO) müssen Sie trotzdem nicht verzichten: Es gibt Templates (BINGO) für Modul-Projekte. Und: Ein Watchdog (BINGO) sorgt im System dafür, dass Einheiten, die sich aufgehängt haben, durch neu gestartete ersetzt werden. Scalability (BINGO) und Reliability (BINGO): Ist das nicht das, was alle wollen? Als Antwort würde ich sagen: Bingo.

Viel Spaß mit der dotnetpro wünscht Ihnen

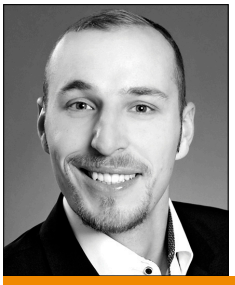
Tilman Börner
Chefredakteur dotnetpro

Christian Giesswein

erklärt, was es mit DataTemplates in WPF auf sich hat (S. 52)

Robert Eichenseer

stellt Service Fabric vor, eine Hosting-Umgebung für Microservices (S. 70)



Björn Steinbrink

zeigt, wie Sie Google Calendar mit C# nutzen (S. 106)