Monolithische vs. Microservices-Softwarearchitektur

Die Wahl des richtigen Designs für Ihre App-Entwicklung

Willem L. Middelkoop

Mar. 3, 2020

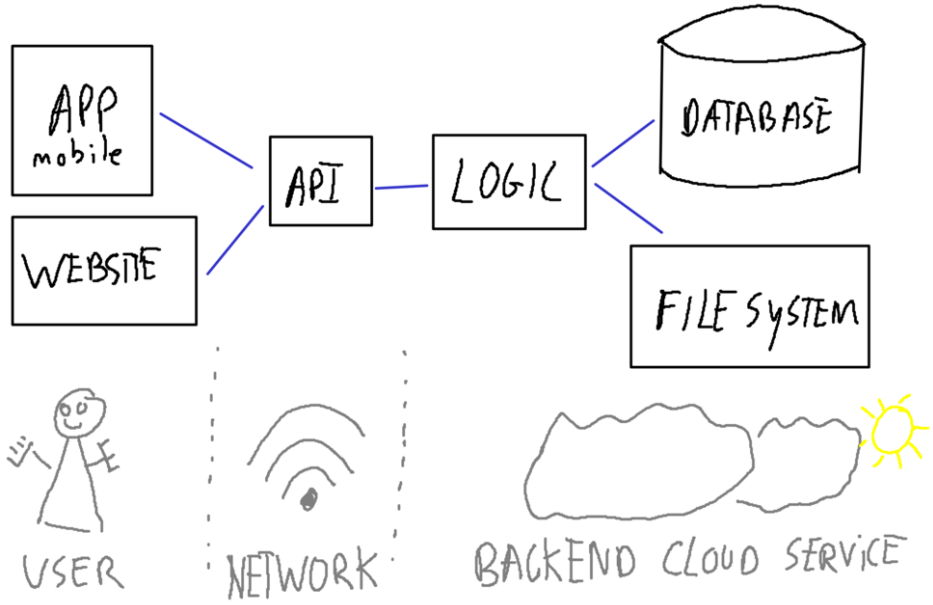
Diese Woche flog ich nach Göteborg, um mich mit Leuten einer großen internationalen Reederei zu treffen und über die Entwicklung von Unternehmenssoftware zu sprechen. Während des Meetings waren verschiedene Experten im Raum, einer von ihnen fragte mich nach der Wahl der richtigen Softwarearchitektur (für große, komplexe Unternehmensanwendungen). Eine sehr gute Frage, die einen Blogbeitrag wert ist.

Besuch einer großen internationalen Reederei, um über Unternehmenssoftware zu sprechen

Softwarearchitektur

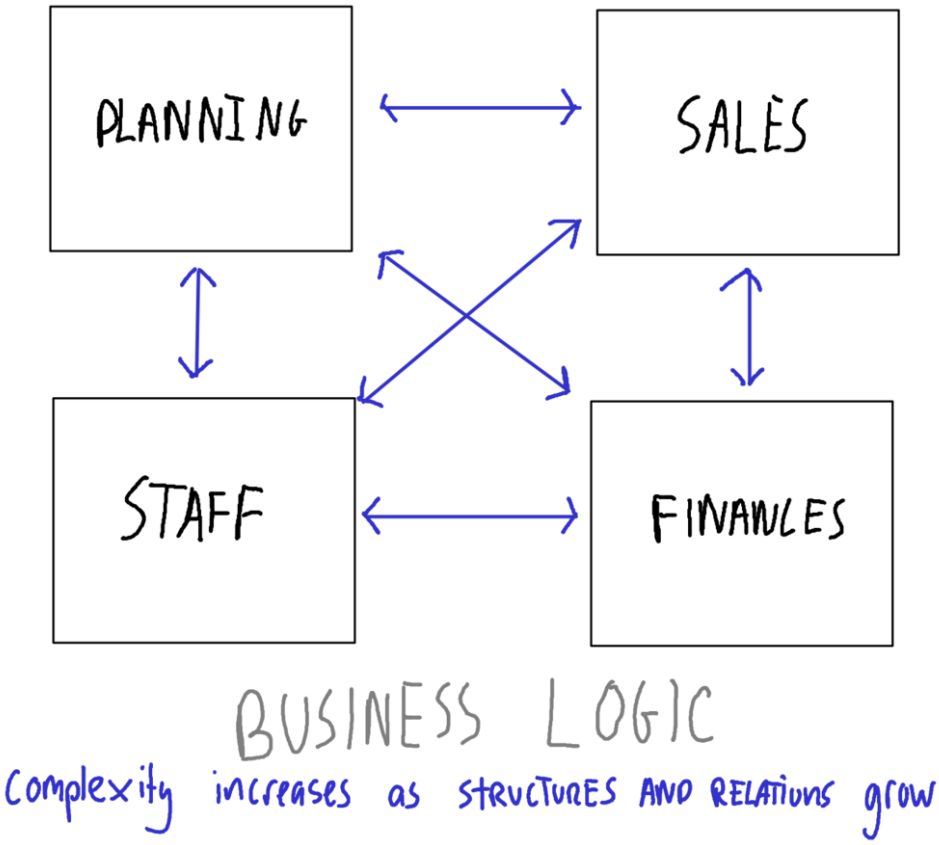
Beim Design von Apps sollten Sie nicht nur an das Aussehen und die Bedienung denken, gutes Design umfasst auch, wie alles zusammenarbeitet. Softwarearchitektur bezieht sich auf die grundlegenden Strukturen eines Softwaresystems. Jede Struktur umfasst Elemente, Beziehungen zwischen ihnen und Eigenschaften sowohl der Elemente als auch ihrer Beziehungen.

Sie ist analog zur Architektur eines Gebäudes. Sobald eine Architektur definiert und implementiert ist, wird es kostspielig, sie nachträglich zu ändern. Daher ist es eine gute Idee, die richtige Softwarearchitektur zu wählen, *bevor* Sie mit dem Bau beginnen.



Einfaches Beispiel für eine Softwarearchitektur - verschiedene Strukturen, die ein System mit unterschiedlichen Beziehungen zwischen ihnen bilden

Bei einfachen Apps kann die Softwarearchitektur offensichtlich sein, aber mit zunehmender Größe und Funktionalität Ihrer Anwendung kann die Anzahl der Strukturen in einem System schnell wachsen. Dies kann im Laufe der Zeit passieren, wenn die Bedürfnisse Ihrer Kunden wachsen. Je mehr Strukturen, desto mehr Beziehungen zwischen ihnen, desto komplexer wird es.



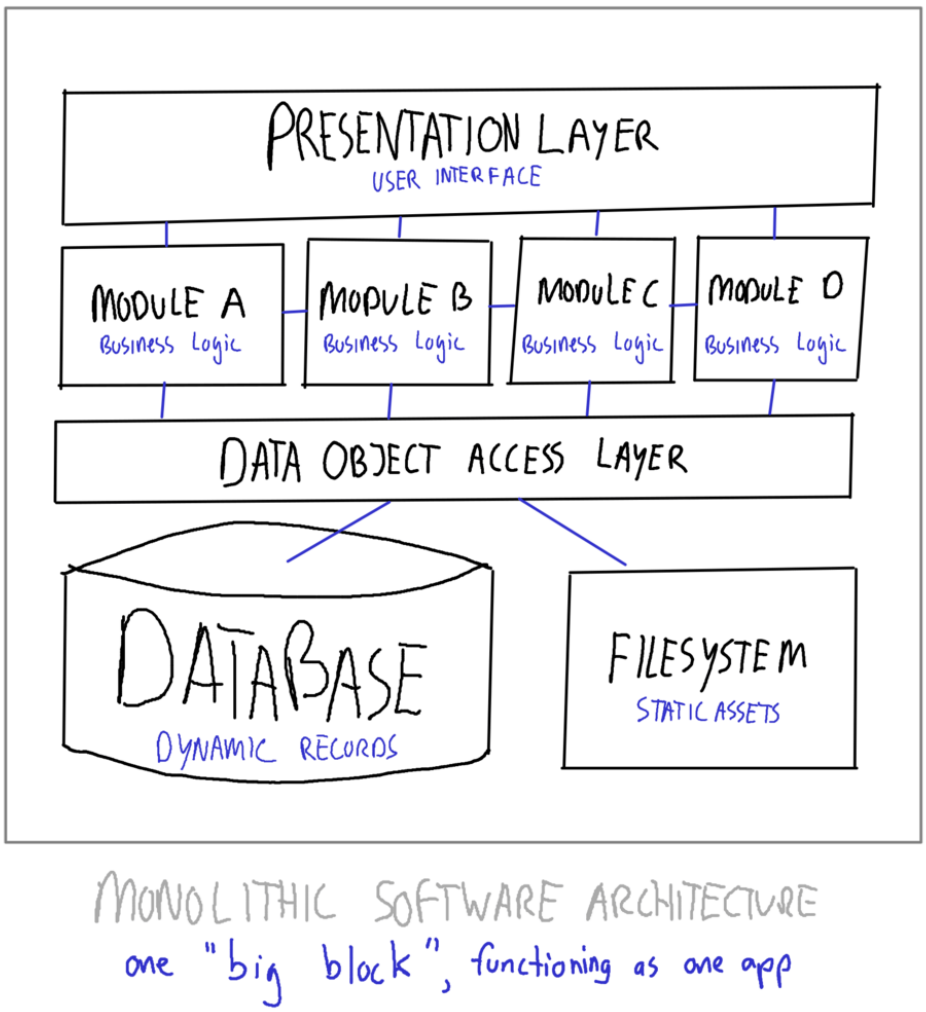
Zunehmende Komplexität mit steigender Anzahl von Strukturen (und ihren Beziehungen)

Umgang mit komplexem Softwaredesign

Wenn Sie erwarten, dass Ihre Software viele verschiedene Dinge handhabt, werden Sie sich wahrscheinlich mit Komplexität auseinandersetzen müssen. Aus architektonischer Sicht gibt es verschiedene Ansätze im Umgang mit dieser Softwarekomplexität.

Monolithische Architektur

Am besten als ”ein großer Block” beschrieben, ist die monolithische Architektur. Hier ist alles, was Ihre App tut, Teil derselben Codebasis.



Monolithische Softwarearchitektur

In einer "monolithischen Architektur" ist die Software oft geschichtet, wobei jede Schicht aus verschiedenen Arten von Komponenten besteht:

- **Präsentationsschicht**: Verantwortlich für die Benutzer- und/oder Anwendungsprogrammierschnittstelle (UI / API).
- **Geschäftslogik**: Hier werden die eigentlichen Geschäftsregeln definiert. Sie ist der Kern der Anwendung, der sehr eng mit den tatsächlichen Geschäftsprozessen abgestimmt ist.
- **Datenobjektzugriffsschicht**: Bietet eine gemeinsame Schnittstelle zwischen der laufenden Anwendung und persistenten Backend-Speichern.
- **Datenbank und Dateisystem**: Hier werden alle Daten und statischen Ressourcen (Dateien, Bilder usw.) gespeichert.

Je größer die Anwendung wird, desto größer wird jede Schicht. Dies macht es schwieriger, eine monolithische Architektur zu skalieren, ab einem bestimmten Punkt werden die Dinge einfach zu groß und komplex.

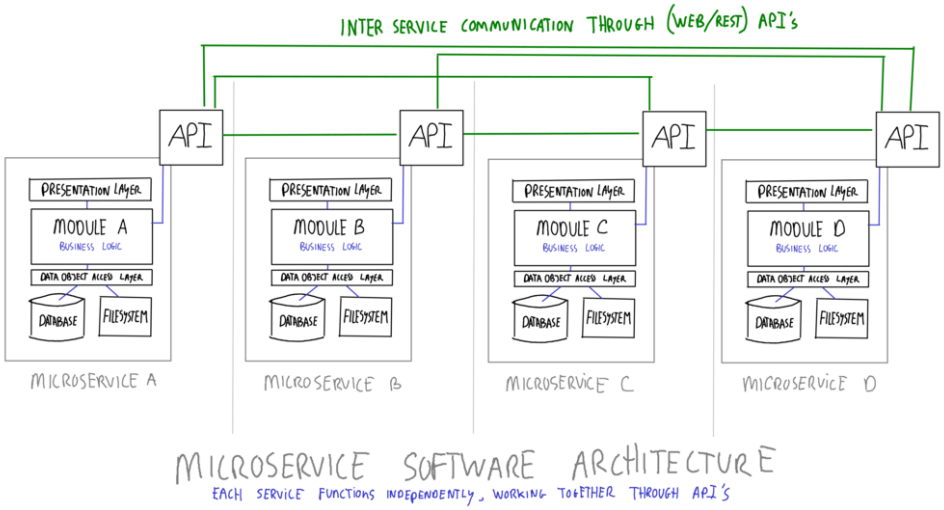
Jedes Mal, wenn Sie eine Änderung an der Anwendung vornehmen, wird die gesamte Codebasis betroffen. Bei (sehr) großen Anwendungen erschwert dies die Wartung der Software, da ein vollständiges Verständnis der gesamten Anwendung erforderlich ist.

Da die gesamte Anwendung voneinander abhängig ist, wirken sich Probleme in einem Teil der Anwendung oft erheblich auf das gesamte System aus. Die Kommunikation zwischen verschiedenen Teilen des Systems erfolgt intern auf Codeebene. Das bedeutet, dass die verschiedenen Teile in der gleichen (oder kompatiblen) Programmiersprache programmiert sein müssen. Dies ist ein Hindernis für die Einführung neuer Technologien (in der Zukunft), da Sie die Programmiersprache eines Teils des Systems nicht einfach ändern können.

Microservices-Architektur

Anstatt eine große monolithische Anwendung zu erstellen, können Sie Ihre Anwendung in kleinere, miteinander verbundene Dienste aufteilen. Jeder Microservice ist eine kleine eigene Anwendung, komplett mit eigener (aber kleinerer + einfacherer) Architektur, Geschäftslogik und Datenschichten.

Die verschiedenen Dienste kommunizieren miteinander über APIs, die unabhängig von der verwendeten Technologie definiert sind, oft in einer gemeinsamen "API-Struktur" wie REST, JSON, Redis und XML. Dies ermöglicht es der Kombination von Microservices, als "ein System" zu funktionieren.



Microservices-Softwarearchitektur

Durch die Zerlegung der "Big-Block"-Anwendung in kleinere Dienste wird es einfacher, die verschiedenen Teile des Systems zu verstehen. Dies erleichtert die Wartung des Systems, da jeder Microservice unabhängig entwickelt werden kann. Es reduziert auch die Hürde für die Einführung neuer Technologien, da Sie die Freiheit haben, die für einen bestimmten Dienst am besten geeignete Technologie zu wählen. Ein Teil des Systems kann in einer anderen Sprache programmiert werden als andere Teile. Dies ermöglicht es verschiedenen Teams (mit unterschiedlichem Fachwissen) zusammenzuarbeiten.

Die Kunst und Fähigkeit, die Microservices-Architektur richtig zu gestalten, liegt in der Fähigkeit, einen Microservice korrekt zu definieren. Zu breit oder zu granular kann die Architektur zerstören. Sie können Geschäftsprozesse, hierarchische oder Domänentrennung verwenden, um jeden Microservice zu definieren.

Die Herausforderung bei der Microservices-Architektur besteht darin, dass sie die Komplexität des gesamten Systems erhöht, da sie einen verteilten Kommunikationsmechanismus erfordert. Die verschiedenen APIs, die miteinander verbunden sind, können

schwierig zu testen und zu debuggen sein. Es wird schwieriger, Änderungen zu implementieren, die sich über mehrere Dienste erstrecken.

Das Definieren, Entwerfen und Implementieren der verschiedenen Interservice-APIs erfordert eine Vereinbarung über die ausgetauschten Daten. Dies kann schwierig zu erreichen sein, wenn Sie beabsichtigen, erweiterte und nicht-universelle Datenstrukturen gemeinsam zu nutzen.

Die Überwachung und Bereitstellung einer Microservices-Architektur kann komplex sein, wenn die Anzahl der Dienste zunimmt. Es gibt einfach mehr zu verwalten, was mehr Planung und Koordination für jeden der Dienste erfordert.

Die Wahl der richtigen Architektur

Mein verstorbener Vater sagte mir einmal, dass man "am meisten mit den Dingen verdient, die man nicht tut". Komplexe Anwendungen zu erstellen ist von Natur aus schwierig, das Beste, was man tun kann, ist zu versuchen, die Dinge einfacher zu machen, indem man einfach nicht alles tut. Konzentrieren Sie sich auf die Dinge, die wirklich wichtig sind, machen Sie diese Dinge richtig.

Wenn Ihre Anwendung einfach und leichtgewichtig ist, ist eine monolithische Architektur der richtige Weg. Entwicklung, Bereitstellung, Wartung, alles ist einfacher, wenn die Dinge klein sind.

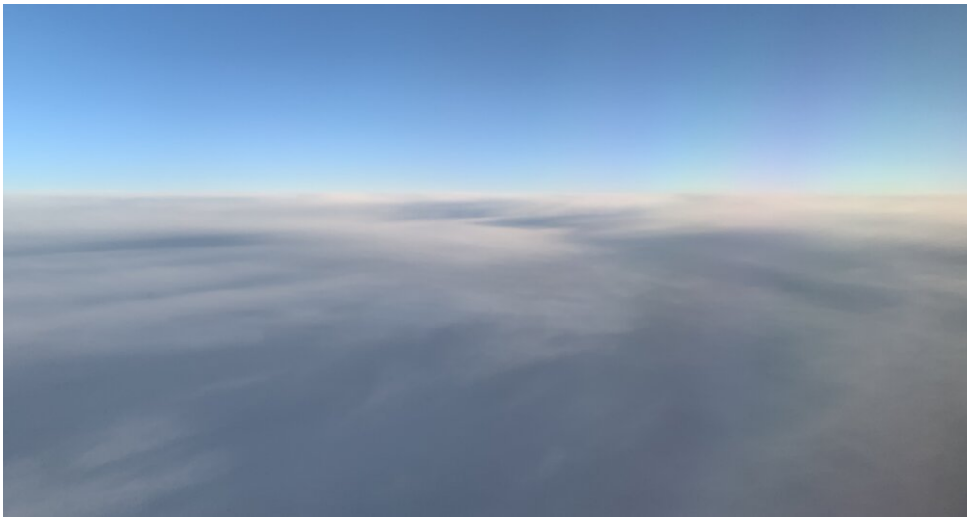
Wenn Sie der Komplexität nicht entkommen können, dann sollten Sie unbedingt die Microservices-Architektur in Betracht ziehen. Stellen Sie sich die folgenden Fragen:

- Sind Sie in der Lage, eine "gemeinsame Sprache" zwischen Ihren Microservices zu definieren?
- Gibt es klar abgegrenzte Geschäftsprozesse?
- Wie stark sind diese Geschäftsprozesse voneinander abhängig?
- Wie erwarten Sie, dass sich Ihre Anwendung im Laufe der Zeit entwickelt, erwarten Sie, dass sie wächst oder schrumpft?
- Wie viele verschiedene Programmiertechnologien planen Sie zu verwenden? Erwarten Sie, dass sich dies in Zukunft ändert?
- Wer wird die Software erstellen und warten, haben diese Erfahrung mit der Verwaltung einer solchen Architektur?

Fazit

Es gibt keine "Patentlösung" auf die Frage, welche Softwarearchitektur die richtige ist. Beide Ansätze haben ihre Vor- und Nachteile.

Das Beste, was Sie tun können, ist, eine bewusste - gut überdachte - Entscheidung zu treffen und diese zu verdoppeln!



„Alles ist möglich“, wenn Sie die gewählte Architektur beherrschen (ändern Sie sie nur nicht während des Fluges!)