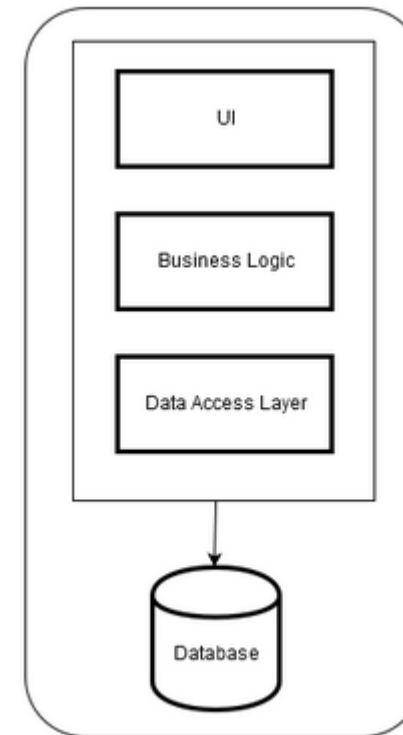


Microservices

Autor: Sebastian Janzen

Monolith

- Monolithische Anwendung wird als ein Ganzes entwickelt
- Drei-Schichten-Architektur
 - Clientseitige Applikation
 - Serverseitige Applikation
 - Datenbank



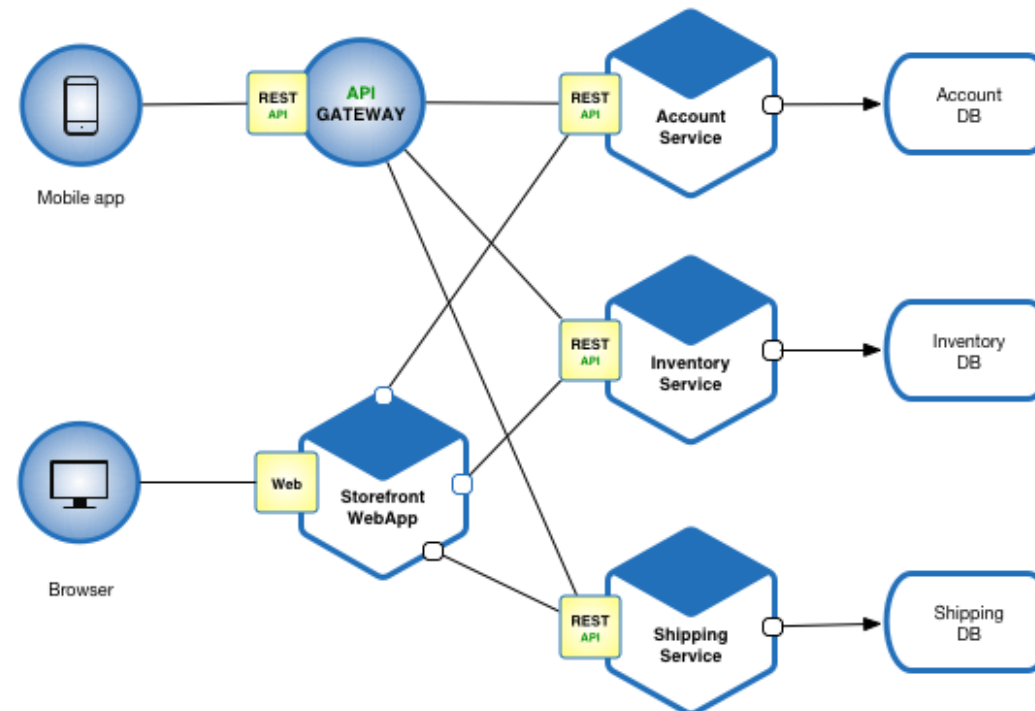
Monolithic Architecture

Monolith

- Zum Beispiel eine serverseitige Applikation ist zuständig für Zugriffe auf die Datenbank und Interaktionen mit dem Browser
- Eine Änderung im System führt zu einer neuen Softwareversion
- Die ganze Logik konzentriert sich in einer ausführbaren Datei
- Für neue Änderungen und Korrekturen muss das ganze System neu erstellt werden, das macht auf Dauer eine solche Anwendung schwierig zu entwickeln
- Es ist aufwändig die Modularität des Systems aufrecht zu erhalten, ohne das ungewollte Abhängigkeiten im System entstehen

Microservices

- Ein Microservice ist ein leichtgewichtiger autonomer Dienst der eine einzige Aufgabe erfüllt und mit anderen ähnlichen Diensten über eine gut definierte Schnittstelle kollaboriert.



Charakteristiken von Microservices

Komponentisierung via Services

- Als Komponenten werden einzelne Services und Bibliotheken verstanden
- Der Vorteil eines Services ist in der Unabhängigkeit
- Wäre ein Service verändert wären andere Services unbetroffen
- Einzelne Softwarekomponenten sind besser voneinander getrennt

Charakteristiken von Microservices

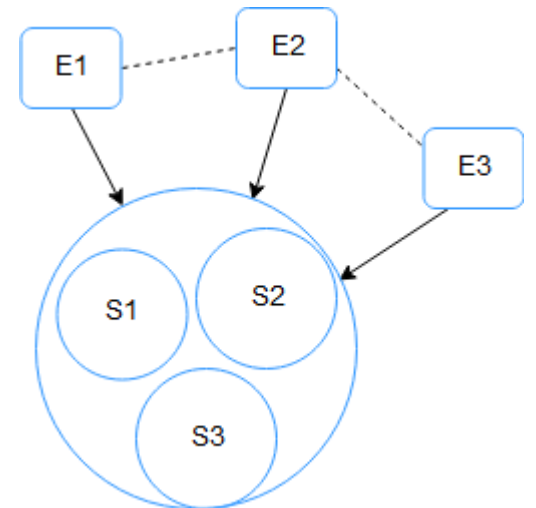
Aufbau um Business Capabilities

- Business Capabilities definieren wichtige Businessfunktionen
- Sie beschreiben was die Kernfähigkeiten eines Unternehmen sind
- Microservices sollen rund um Business Capabilities aufgebaut werden
- Gesetz von Conway besagt
 - „Jede Organisation die ein System entwirft bekommt am Ende ein Entwurf welches die Kommunikationsstruktur der umzusetzenden Organisation nachbildet “

Charakteristiken von Microservices

Aufbau um Business Capabilities

- Gesetz von Conway Beispiel
 - Entwicklerteams E1-E3 arbeiten zusammen am Projekt
 - Am Ende wird die Architektur aus Subsystemen S1-S3 bestehen
 - Qualität und Art der Schnittstellen zwischen den Subsystemen wird der Kommunikation zwischen den Teams entsprechen



Charakteristiken von Microservices

Service als Produkt

- Software sollte als ein Produkt angesehen werden
- Entwicklerteams sollten sich während des gesamten Lebenszyklus um ihr Produkt kümmern
- Vorteile sind ständiger Feedback von Nutzern und engere Bindung an das eigene Produkt
- Software wird mehr als ein Bündel von Funktionalitäten
- Es wird eine andauernde Beziehung aufgebaut, welches die Business Capability zu verbessert

Charakteristiken von Microservices

Smart endpoints and dumb pipes

- Applikationen der Microservices sollten abgekoppelt und zusammenhängend wie möglich sein (smart endpoints)
- Zum Beispiel: Empfangen einer Anfrage, Bearbeitung und senden einer Antwort
- Protokolle sollten hingegen unkompliziert sein
- Eine Möglichkeit wäre eine Infrastruktur zu nutzen, die nur als Router von Nachrichten fungiert (dumb pipes)

Charakteristiken von Microservices

Dezentralisierte Führung

- Erlaubt einzelnen Entwicklerteams ihre Stärken in verschiedenen Technologien anzuwenden und zusammen ein leistungsfähiges Produkt zu entwickeln
- Passende Technologie pro Service: **Polyglot Programming**
- Entscheidungen über Entwicklung liegen auf Ebene der Entwickler
- Entwicklung, Installation, Überwachung und Steuerung wird von Entwicklern übernommen

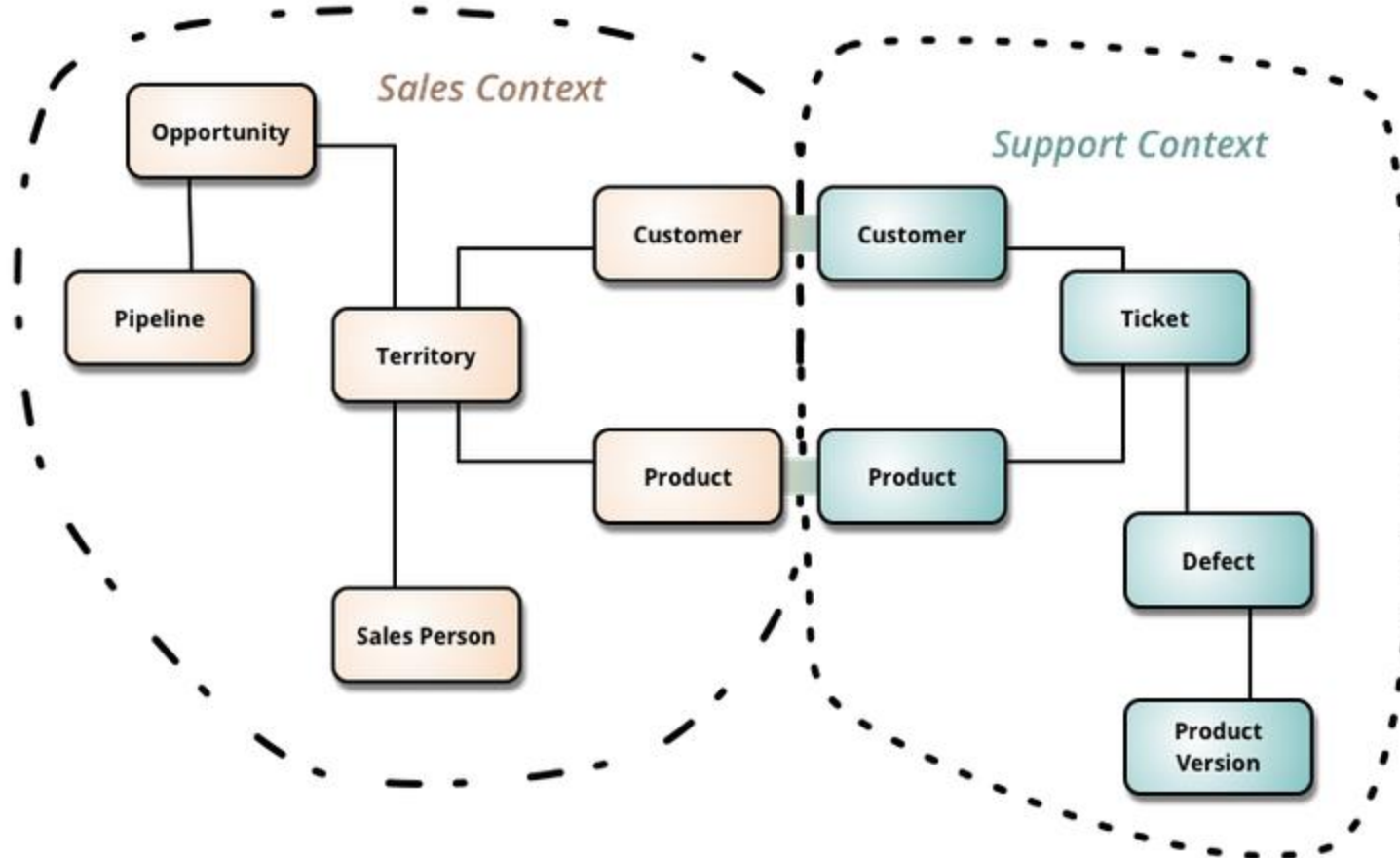
Charakteristiken von Microservices

Dezentralisiertes Datenmanagement

- **Bounded Context** beschreibt eine Abgrenzung in der ein bestimmtes Modell definiert und verwendet wird
- Es teilt komplexe Domänen in mehrere Kontextgrenzen und beschreibt die Beziehungen zwischen ihnen
- Microservices besitzen eine natürliche Korrelation zu Bounded Context, denn dieser verdeutlicht und verstärkt eine Trennung in verschiedene Kontextbereiche

Charakteristiken von Microservices

Dezentralisiertes Datenmanagement



Charakteristiken von Microservices

Dezentralisiertes Datenmanagement

- Es wird bevorzugt pro Service eine Datenbank zu haben
- Eine der Probleme von dezentralisiertem Datenmanagement ist Update Management
- Es ist schwierig Datenkonsistenz zu garantieren
- Mögliche Probleme werden von kompensierenden Operationen abgefangen (Vorgänge werden rückgängig gemacht)

Charakteristiken von Microservices

Infrastructure Automation

- Testautomatisierung und Ansätze wie Continuous Integration und Continuous Delivery helfen bei der Entwicklung von stabiler und hochwertiger Software

Charakteristiken von Microservices

Design for failure

- Ein Service sollte immer gegen mögliche Fehlerfälle ausgiebig getestet werden
- Komplexer bei Microservices als bei monolithischen Design
- Da nicht alle möglichen Probleme abgedeckt und vorhergesehen werden können, kommt Echtzeit-Monitoring zum Einsatz

Charakteristiken von Microservices

Evolutionäres Design

- Software so designen, dass statt es bei größeren Änderungen zu verwerfen, diese weiterentwickelt werden kann
- Die wichtigsten Eigenschaften eines Komponenten sind Austauschbarkeit und Erweiterungsfähigkeit
- Oftmals wird ein Service verworfen anstatt überarbeitet zu werden

Microservices: Vor- und Nachteile

Vorteile	Nachteile
Kompakter Quellcode	Größerer Ressourcenbedarf
Leichter Einstieg ins Entwicklerteam	Use Cases nicht eindeutig
Erhöhte Einsatzfähigkeit	Sehr komplex
Kürzere Entwicklungszeiten	Überhang an Services möglich
Autonome Services	Testfälle werden komplexer
Kleine dedizierte Teams	Abhängig von Schnittstellen
Resistent gegen Ausfall	Probleme mit Datenkonsistenz
Business Case abhängig	Aufwändig zu managen
Technologie unabhängig	Großer Kommunikationsaufwand pro Team
Wiederverwendbare Services	

Humane Registries

- Automatisierte Dokumentation
- Schreibt und aktualisiert Informationen in menschlich lesbaren Form
- Format sollte für alle lesbar und verständlich sein
- Entwickler haben selten Zeit eine Dokumentation zu pflegen
- Erweiterung der Informationen sollte unkompliziert sein
- Durchsucht den Quellcode des Systems und stellt detaillierte Informationen darüber bereit

Serverless

- Auf serverseitige Logik wird verzichtet, stattdessen Cloud-Services von Drittanbietern
- Zu solchen Cloud-Services zählen Datenbanken und Authentifizierungsmechanismen: **Backend as a Service**
- Ein anderer Bereich von Serverless ist **Function as a Service**
- Serverseitige Logik immer noch vom Entwickler geschrieben, jedoch verpackt in Container und von einer Cloud gemanagt

Micro Frontends

- Für Webanwendungen gewinnt Front-End immer mehr an Bedeutung
- Der Trend geht in Richtung einer 90 zu 10 Aufteilung zu Gunsten von Front-End
- Der aktuelle Trend heißt "Micro Front-Ends" und Unternehmen, wie Spotify und Zalando sind schon umgestiegen

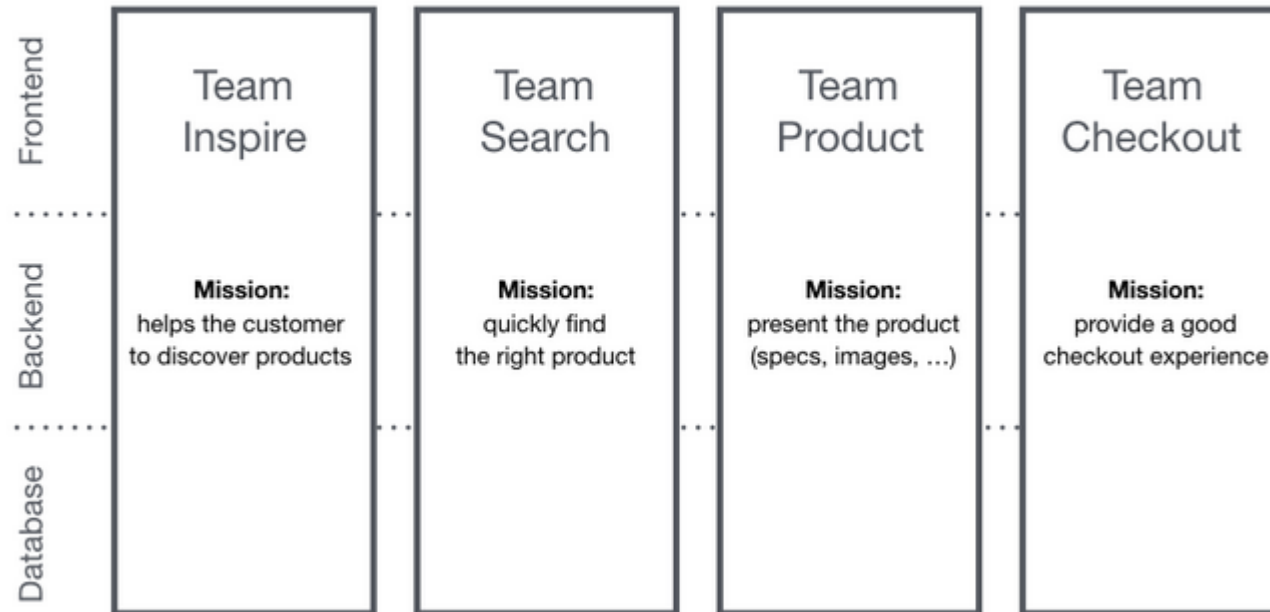
Micro Frontends

Vorteil	Nachteil
Unabhängig	Erhöhter Betriebsaufwand
Hohe Testbarkeit	Erhöhte Komplexität (z.B. Infrastruktur, Kommunikation)
Unabhängige Technologiestacks	Schlechtere Performance
Unabhängig im Fehlerfall	Restrukturierung kann sehr komplex werden
Parallele Entwicklung möglich	

Micro Frontends

- Vertikale Aufteilung

End-to-End Teams with Micro Frontends



Self-contained-Systems

- Architekturmuster mit Fokus auf Separation einer Funktionalität in mehrere unabhängige Systeme
- Aus der Kollaboration dieser Systeme entsteht ein neues logisches System
- Ähnlichkeit zu Microservices
 - Unabhängige Einheiten
 - Technologische Diversität
 - Dezentralisierte Infrastruktur

Self-contained-Systems

- Ein Microservices ist üblicherweise kleiner als ein SCS
- Weniger SCSs als Microservices
- Ein E-Commerce Shop kann 5-25 SCSs haben oder bis zu 100 Microservices.
- SCSs besitzen ein UI, während Microservices es üblicherweise auslagern

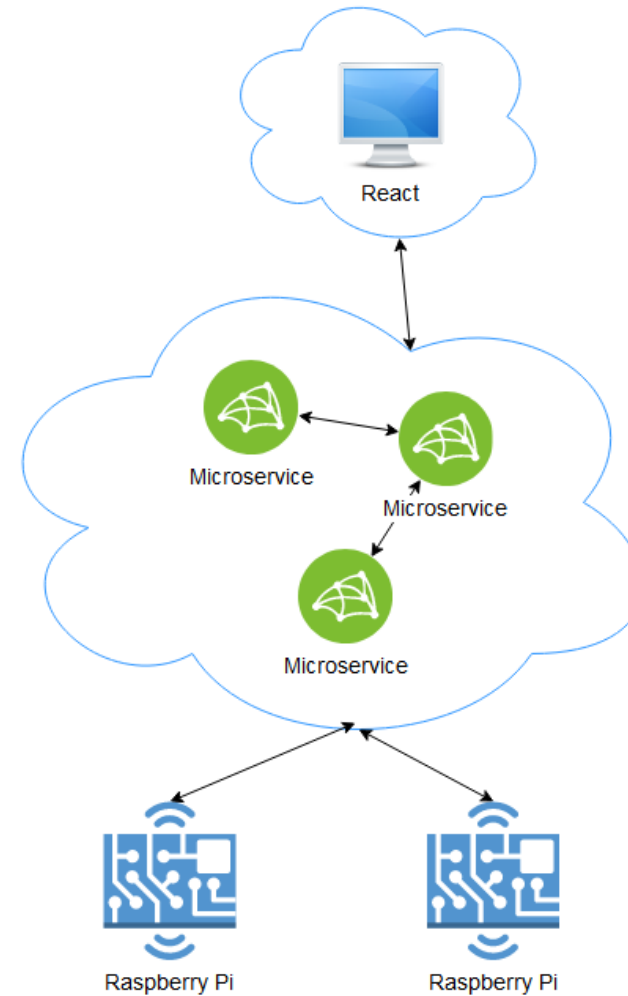
Demo

Story

- Smart Home Demo
- Zwei Raspberry Pis mit Sensoren und LEDs
- Microservices für Sensoren und LEDs
- Microservices sind in AWS Cloud
- React Dashboard zeigt Temperatur, Helligkeit, Luftfeuchtigkeit und kann die LEDs steuern

Architektur

- cote: Node.js Bibliothek
- Microservices in Javascript



Code

<https://github.com/janzse/microservices>

Quellen

- Abbildung Folie 3:
<http://microservices.io/patterns/microservices.html>

Fragen und Feedback