Technische Hochschule Ingolstadt

Fakultät für Elektrotechnik und Informatik

Zukunft in Bewegung

IT-Integrations- und Migrationstechnologien

Resilience Patterns für Integrations-Services

Prof. Dr. Bernd Hafenrichter 26.11.2023

Motivation

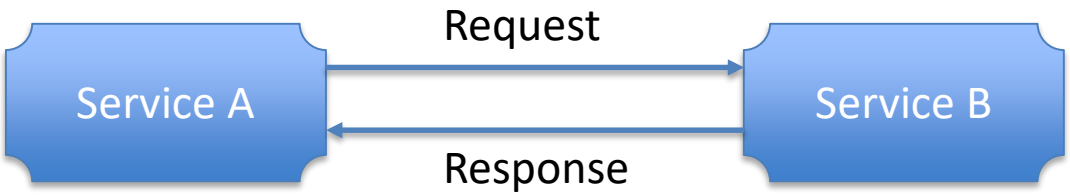
- Fehlerfälle sind inhärent im Design von verteilten/integrierten Systemen enthalten
- Auch bei der Verwendung von (Micro-)Integration-Services ist es notwendig über das Fehlerhandling nachzudenken.

- Partielle Ausfälle: eine Komponente fällt aus, wodurch ein Teil des Systems beeinträchtigt werden kann
- Ein wichtiges Ziel beim Entwurf verteilter Systeme: sie so aufzubauen, dass sie nach partiellen Ausfällen automatisch wiederhergestellt werden können
- Insbesondere sollte das System im Falle eines Fehlers akzeptabel weiterarbeiten, d. h. Fehler tolerieren

Motivation

Timeout

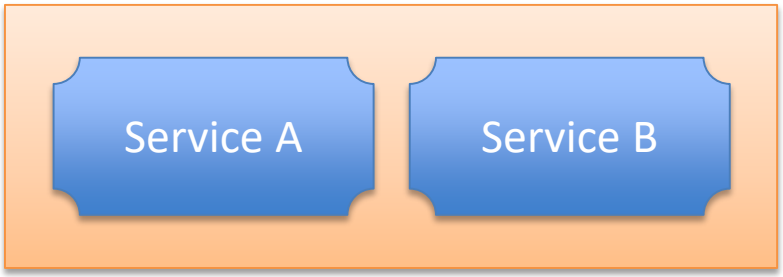
- Die Kommunikation zwischen Microservices erfolgt oft synchron. Ein Client wartet auf eine Antwort
- Die Idee: Ein blockierender Service soll den Aufrufer nicht negativ beeinflussen.
- Ein Timeout ist die Entscheidung, wann das Warten auf eine Antwort beendet werden soll
- Timeouts ermöglichen die Isolierung von Fehlern. Ein Fehlverhalten eines anderen Dienstes hat keinen Einfluss auf den aufrufenden Dienst
- Anständige Fehlerbehandlung: Implementieren Sie ein geeignetes Verhalten im Falle einer Zeitüberschreitung

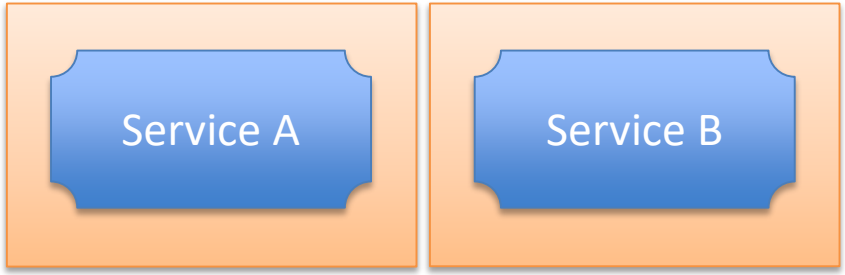


Motivation

Bulkhead

- Die Idee: Eine Applikation soll so partitioniert werden das der Fehler eines Microservices keine Auswirkungen auf andere Services hat.
- Ansatz:
  - Unabhängiges Deployment von Integration-Services
  - Fachliche Zusammenfassung von Integrationslogik in einem Service. Auftrennung unabhängiger Funktionen.
- Wird gegen das Prinzip verstoßen, können gemeinsam genutzte Ressourcen unabhängige Businessfunktionalität negativ beeinflussen
  - Threadpool, Datenbank, Speicher





Motivation

Fail Fast

- Fehler sollten so schnell wie möglich erkannt werden
- Die Idee: Eine schnelle Fehlerreaktion ist viel besser als eine langsame Fehlerreaktion
- Fehlererkennung
    - Prüfen der Antwortnachricht
    - Überprüfung der Systemressourcen: Thread-Pools, Verbindungen, Socket-Limits, Datenbank

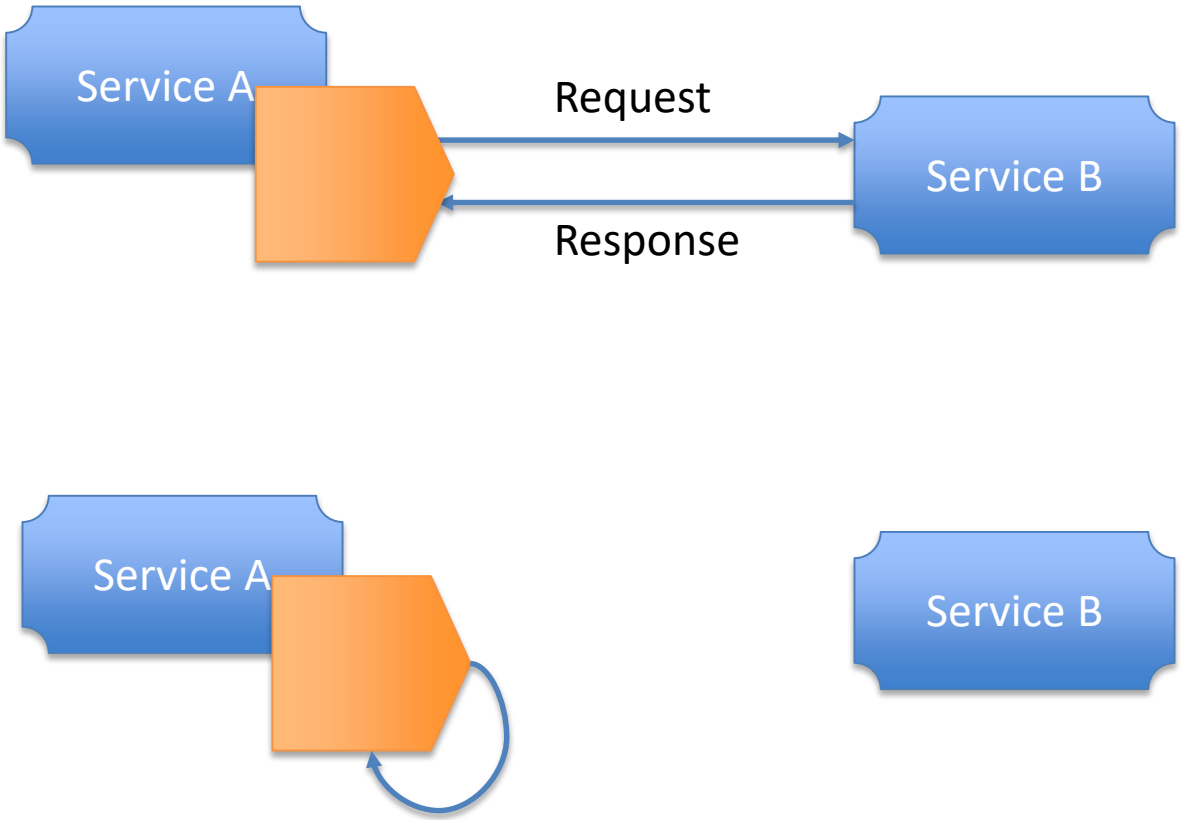
Motivation

Circuit Breaker

- Der Aufruf eines externe System/Service kann aus verschiedensten Gründen fehlerhaft sein
- Aufrufe können mit Hilfe eines Wrapper-Objectes überwacht werden und Fehlersituationen proaktiv versindern
- Ein Circuit Breaker ist ein Wrapper-Objekt
    - Aufrufe werden überwacht
    - Überschreigt die Anzahl der fehlerhaften Aufrufe einen Schwellwert verhindert der Circuit-Breaker weitere Aufrufe
    - Nach einer gewissen Zeit (Reset Timeout) prüft der Circuit Breaker of Aufrufe wieder möglich sind.

Motivation

- **Circuit Breaker**



Motivation

- **Circuit Breaker**