Gegenüberstellung

Application Architektur

Standalone Software

One Tier Architecture

- Vorteile:
  - Anwendungslogik, Präsentation und Speicher liegen alle auf einem Gerät
- Nachteile:
  - Hoher Bedarf an Rechnungsleistung für aufwendige Operationen auf der Client Seite

Monolithic Architecture

Two Tier Architecture

- Vorteile:
  - Eng gekoppelte Komponenten
- Nachteile:
  - Updates müssen in Gesamtpaketen des gesamten Systems durchgeführt werden
  - Instandhaltung des Systems komplizierter
  - Abhängigkeiten von Frameworks fatal für das gesamte System

Three Tier Architecture

- Vorteile:
  - Besser Instandhaltbar
  - Besser skalierbare Ebenen (und somit zukunftssicherer)
  - Einfaches Setup
  - Kostengünstig
- Nachteile:
  - Eng gekoppelte Komponenten
  - Schwierig Instandzuhalten
    - Updates haben wahrscheinlich Abhängigkeiten zu anderen Komponenten

Service Architecture

Service Oriented Architecture

- Vorteile:
  - Leicht wartbar
  - Updates können seperat von anderen Service Modulen veröffentlicht werden
  - Lose gekoppelt und somit auch leichter zu skalieren
  - Durch Container sehr gut skalierbar und updates sind besser zu veröffentlichen
- Nachteile:
  - Aufwendiger zu implementieren

Microservice Architecture

- Vorteile:
  - Noch loser gekoppelt
  - Austauschbare Module
  - Noch bessere Skalierbarkeit
  - Fallback Module die Aufgaben übernehmen können falls andere Module ausfallen
- Nachteile:
  - Meistens nur für große Unternehmen vorgesehen auf Grund der Komplexität des Services

Scaling

Vertical Scaling

- Vorteile:
  - Komponenten erhalten mehr Leistung
- Nachteile:
  - Fällt eine Komponente aus, ist das gesamte System davon betroffen

Horizontal Scaling

- Vorteile:
  - Service erhält zusätzlich zu der Rechenkraft auch noch mehr Brandbreite und die Erreichbarkeit wird besser
  - Fällt ein Modul aus, so ist der Service nur verlangsamt aber nicht komplett unerreichbar
- Nachteile:
  - Möglicherweise kostenintensiver

Software Architektur

Peer to Peer Communication

- Vorteile:
  - Sicher
- Nachteile:
  - Komplexer
  - Eingeschränkte Bandbreite
  - Erreichbarkeit nicht gewährleistet

Event driven Communication (Message driven)

- Vorteile:
  - Lose Kopplung zwischen Komponenten und Ebenen
- Nachteile:
  - Geringe Datenmengen

Client Server Communication

- Vorteile:

    - Weit etabliert
    - Meistens erreichbar (nur in fatalen Ausfällen nicht erreichbar)
- Nachteile:
    - Anfällig für Angriffe
    - Verlust der direkten Kontrolle über die Nutzerdaten

Datenbase Architektur

SQL

- Vorteile:
    - Geeignet für Sozial Media Applications
        - Geeignet für die Darstellungen von Beziehungen
    - Geeignet für Finanzsystem
- Nachteile:
    - Große Abfragen brauchen länger

NoSQL

- Vorteile:
    - Geeignet bei vielen Lese/Schreibe Operationen
    - Geeignet wenn schnelle Skalierung erforderlich ist
    - Geeignet wenn viele Daten versendet/empfangen werden
    - Geeignet für dynmaische Inhalte
- Nachteile:
    - Komplexeres Setup

Quellen:

- https://www.ibm.com/cloud/blog/soa-vs-microservices#:~:text=The%20main%20distinction%20between%20the,when%20you%20neglect%20this%20difference.
- https://www.educative.io/blog/how-to-design-a-web-application-software-architecture-101#patterns
- https://www.educative.io/blog/mvc-tutorial
- https://www.educative.io/blog/how-to-design-a-web-application-software-architecture-101
- https://www.educative.io/blog/database-design-tutorial
- https://www.youtube.com/watch?v=r8JuAz9_V18
- https://www.youtube.com/watch?v=ccOzcjHQVos
- https://www.youtube.com/watch?v=KlHvRKSH4pk
- https://de.wikipedia.org/wiki/ACID
- https://de.wikipedia.org/wiki/CRUD
- https://github.com/Inf166/PPSS21Mai/issues/8