

1 Einführung Softwarearchitektur

1.1 Was ist Softwarearchitektur?

Definiert die grundlegenden **Prinzipien und Regeln** für die Organisation eines Systems sowie dessen **Strukturierung in Bausteine und Schnittstellen** und deren Beziehungen zueinander wie auch zur Umgebung. → Beschreibt grobgranulare Strukturen wie: Komponenten mit Schnittstellen, Assoziationen/Interaktionen, Prinzipien & Muster (Pattern)

1.1.1 Bausteine

Beschreiben statische Struktur des Systems. Bestehen aus:

- Pakete: Fassen untergeordnete Pakete bzw. Komponenten zu einer Einheit zusammen
- Komponenten: Klar definiertes Verhalten. Können Schnittstellen anbieten/konsumieren

1.1.2 Architektur vs Entwurf (Design)

Architektur:

- Legt grundlegende Strukturen fest
 - B sp.: Mobile Native-App mit REST-Zugriff auf Web-Server
- Darstellung oft mit UML Komponentendiagramm

Entwurf:

- Legt feingranulare Struktur fest
 - B sp.: Aufbau der Native-App: GUI – Funktionsebenen – Zugriff auf Server
- Darstellung oft mit UML-Klassendiagramm

1.1.3 Rolle von Softwarearchitekt

Arbeitet eng mit dem Requirements Engineering zusammen (Schwerpunkt auf NFA). Ist verantwortlich für:

- Grobgranularen Strukturen
 - Querschnittskonzepte (z.B. Persistenz, Kommunikation, GUI)
 - Erkennen/Aufzeigen von Konsequenzen bei Architekturentscheidungen
- Erarbeitet/bearbeitet die Technical User Stories des Product-Backlog

1.2 Ziele von Softwarearchitektur

Ziel der Softwarearchitektur ist die Sicherstellung einer adäquaten Softwareproduktequalität.

- Nützlichkeit: Applikation erfüllt ihre Funktion
- Festigkeit: Software ist stabil und langlebig
- Ästhetik: User Experience ist ansprechend

1.2.1 ISO 25010 Softwareproduktequalitätsmodell

Benutzer:

- Funktionalität
- Performanz und Effizienz
- Kompatibilität und Interoperabilität
- Benutzbarkeit
- Verfügbarkeit

Betreiber & Entwickler:

- IT-Sicherheit
- Wartbarkeit
- Portierbarkeit

1.3 Modellierung

4 Sichten: Kontext, Baustein, Laufzeit und Verteilungssicht.

1.3.1 Kontextsicht

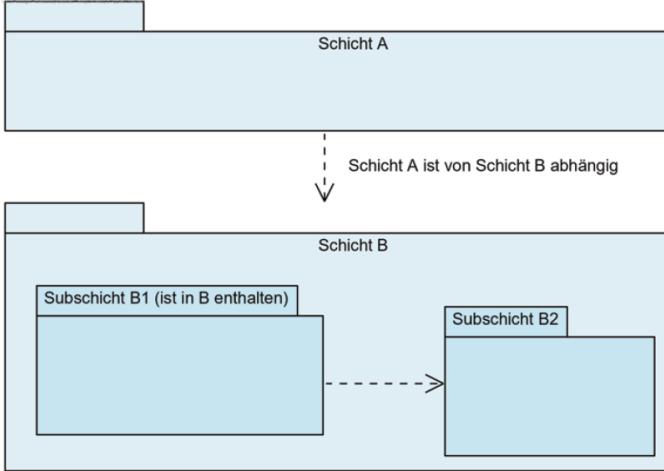
Das System steht im Zentrum als Black-Box. Alle Umsysteme und relevanten Stakeholder sind als externe Agenten dargestellt. Die Verbindungen von und zum System bilden Datenflüsse ab.

Diagramme: Kontext- oder UML-Diagramm

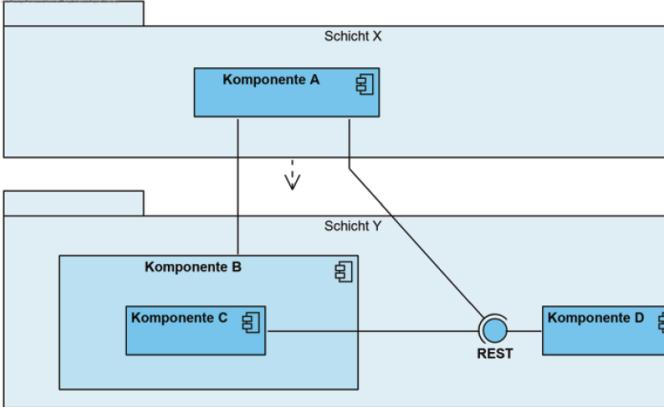
1.3.2 Bausteinsicht

Struktureller Aufbau des Systems

Paketdiagramm:



Komponentendiagramm:



1.3.3 Laufzeitsicht

Fokussiert auf die dynamischen Aspekte wie Prozessabläufe, Interaktionen zwischen Bausteinen und Verhalten.

Diagramme: Sequenz-, Aktivitäts- und Zustands-Diagramm

1.3.4 Verteilungssicht

Darstellen: Komponenten, welche in einer spezifischen Laufzeitumgebung installiert werden.

Diagramm: Deploymentdiagramm

1.3.5 Werkzeuge

Textbasierte Dokumentation: Wiki für kollaboratives Erarbeiten der Inhalte, plus Versionierung. Bsp: Confluence

Modellierungswerkzeuge: Unterstützung von Diagrammspra-

chen wie UML, SysML. Bsp: Visual Paradigm

Code-Analysetool: Statische Analyse und Bewertung der technischen Qualität von Sourcecode. Bsp: SonarQube

Testtools:

2 Service Oriented Architecture

2.1 Synchrone Web Services

- Zustandsbehaftet

- S OAP - Simple Object Access Protocol

- Zustandslos

- R EST

- O Data

- G raphQL

- g RPC

2.2 Protokolle

E-Mail:

- SMTP (Versenden)

- POP (Empfangen)

- IMAP (synchronisieren von Mails)

Message Oriented Middleware (MOM, Message Broker):

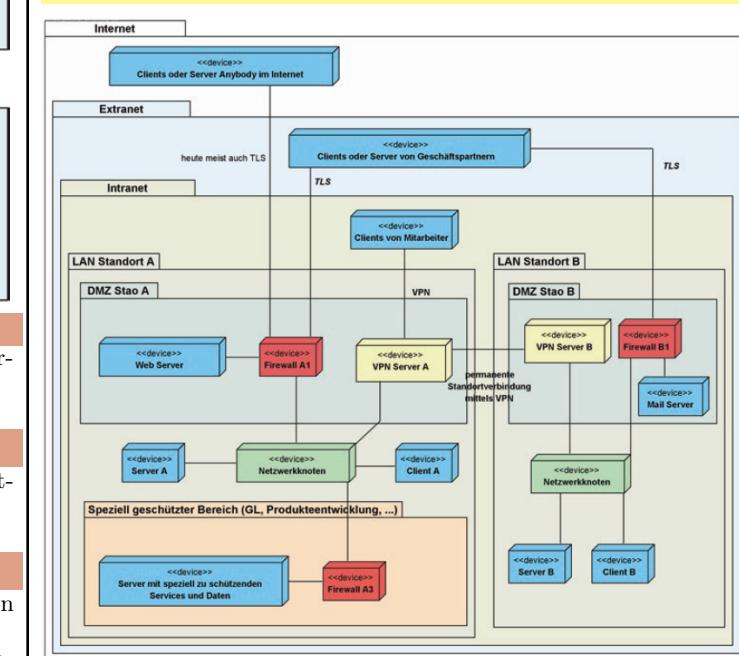
- Für asynchrone Kommunikation.. z.B. RabbitMQ

- AMQP

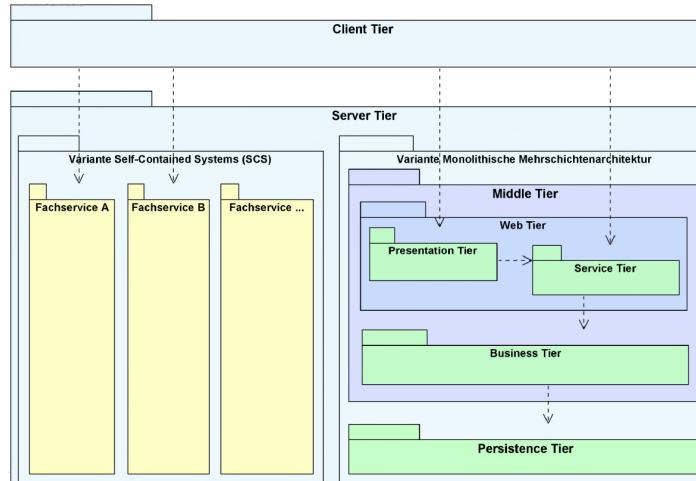
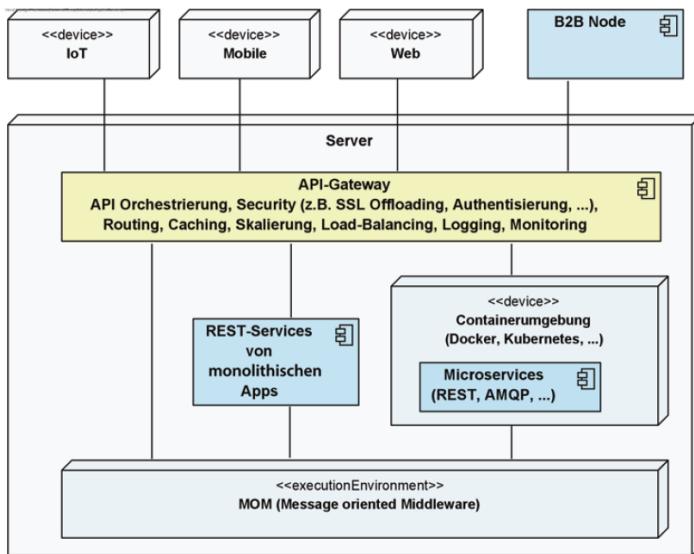
- JMS

- MQTT (IoT)

2.3 Netzwerk-Sicherheitsarchitektur:

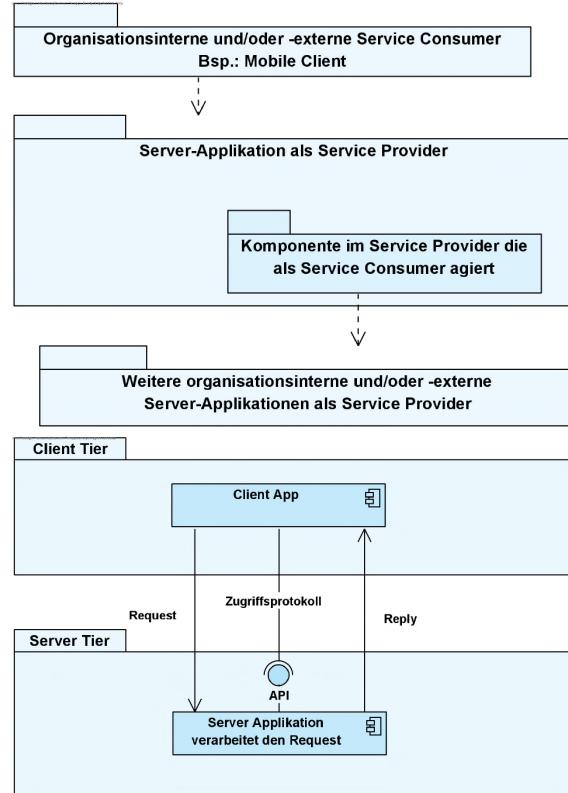


2.4 API-Gateway

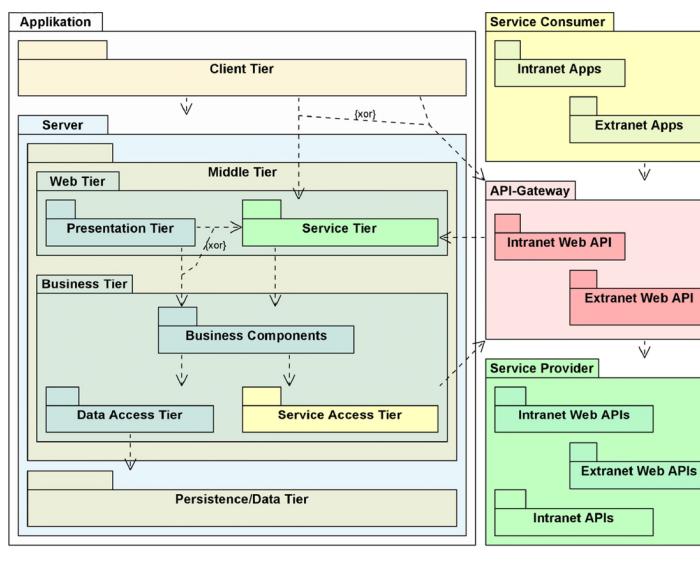


3 Mehrschichten Architekturen

3.1 Client-Server Grundmuster

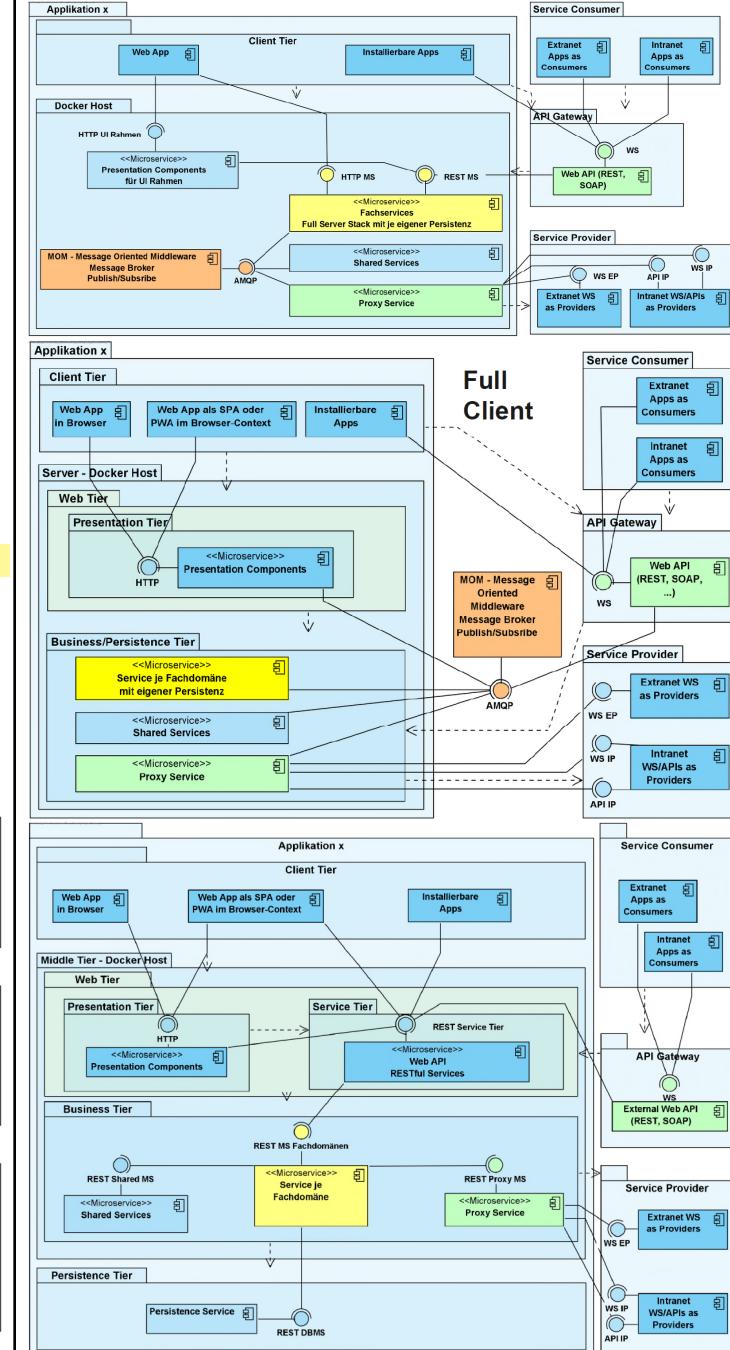


3.2 SOA-enabled Musterarchitektur

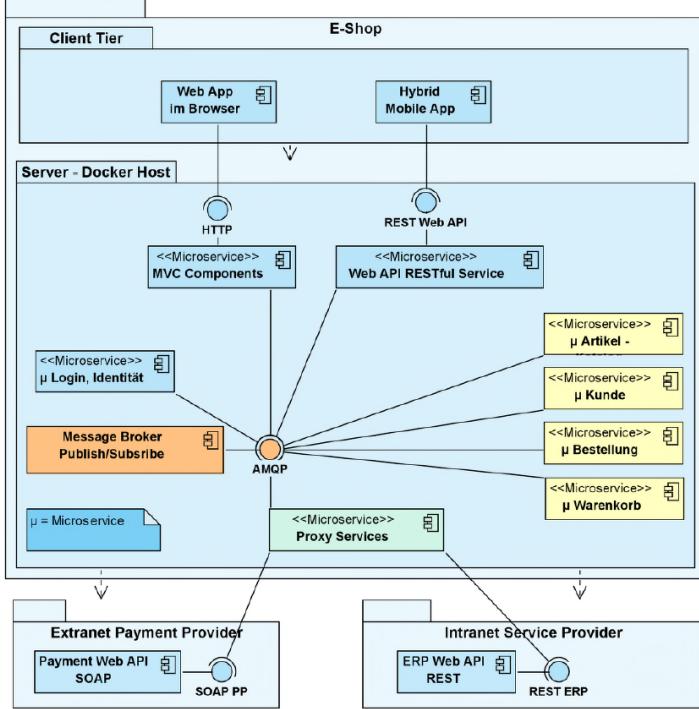


4 Microservices

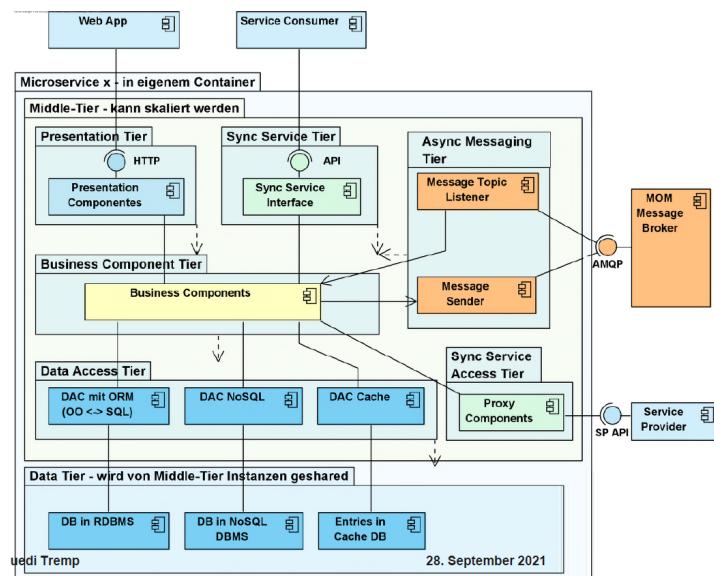
4.1 Referenz-Makroarchitekturen



4.1.1 Beispiel

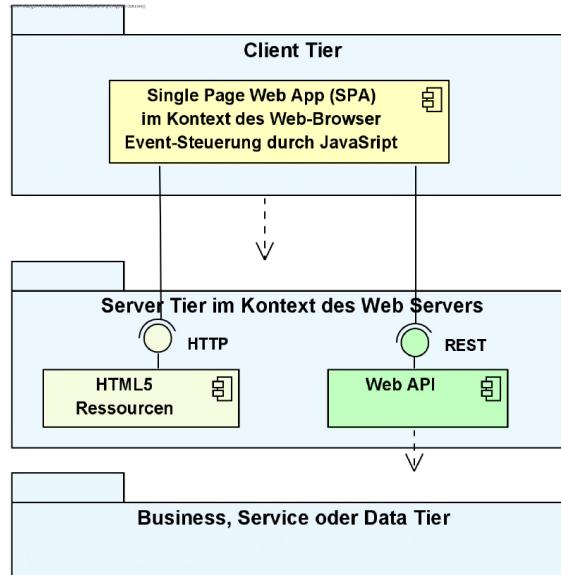


4.2 Referenz-Microarchitektur

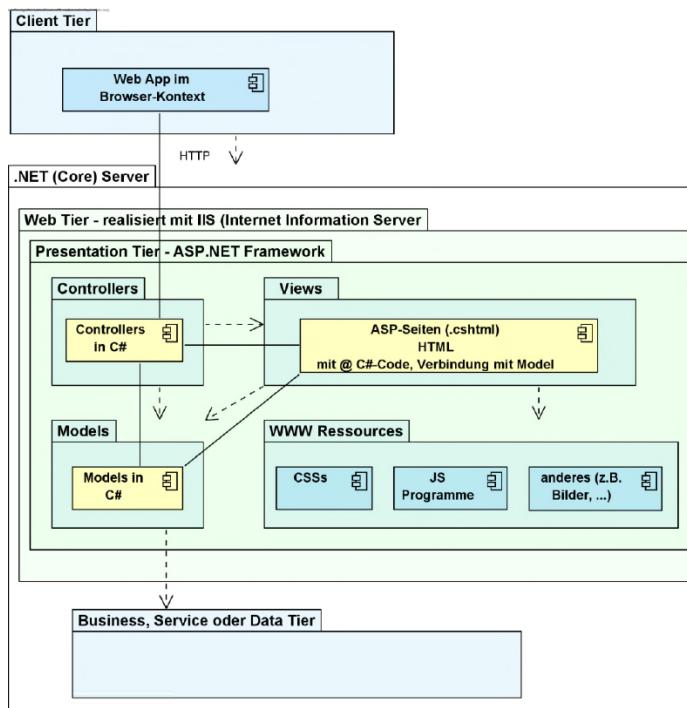


5 Clientseitige Architekturen

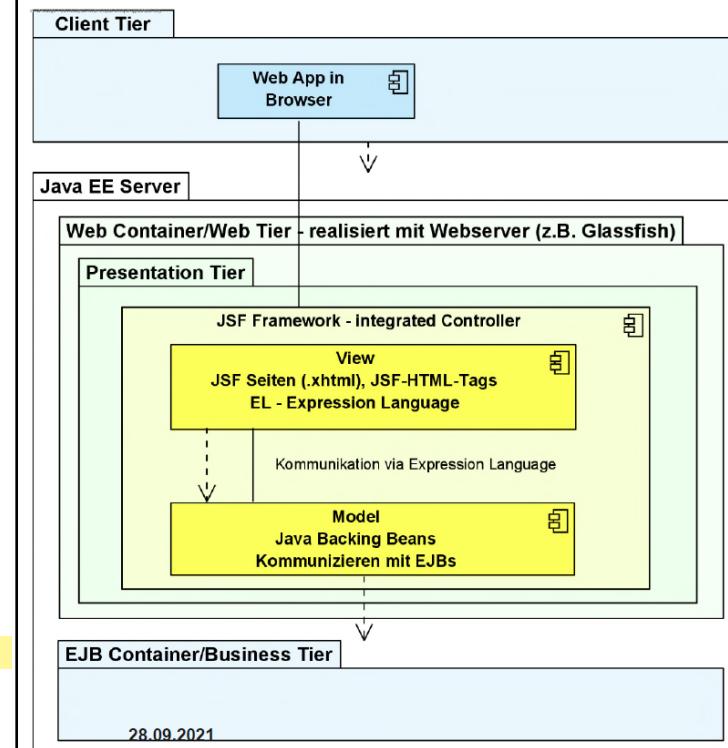
5.1 SPA



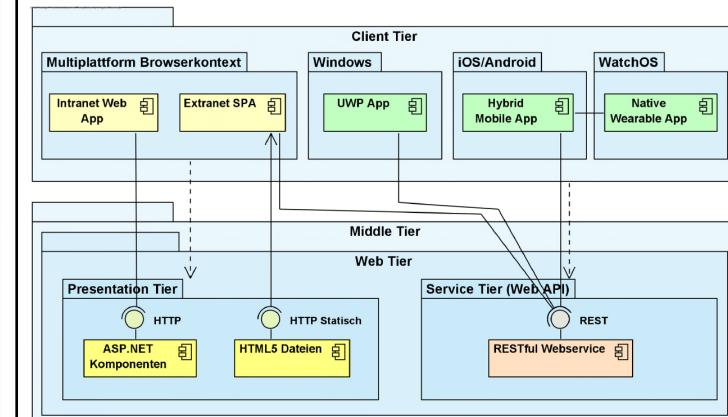
5.2 ASP.NET Core



5.3 Java EE Framework

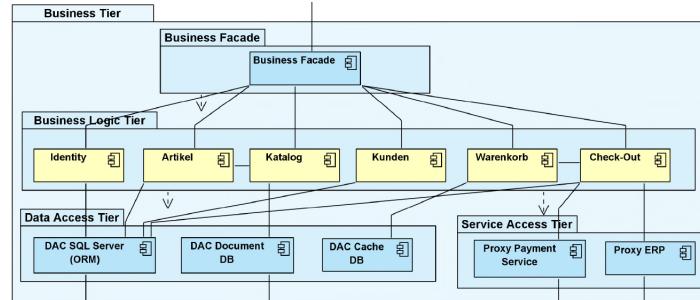


5.4 Makroarchitektur

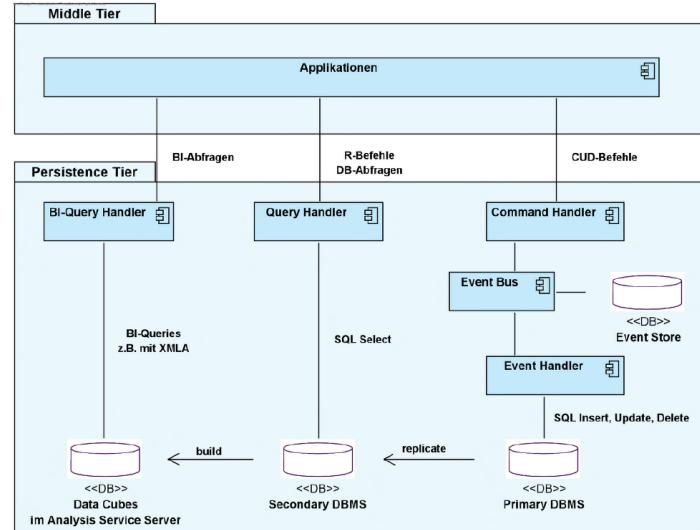


6 Serverseitige Architekturen

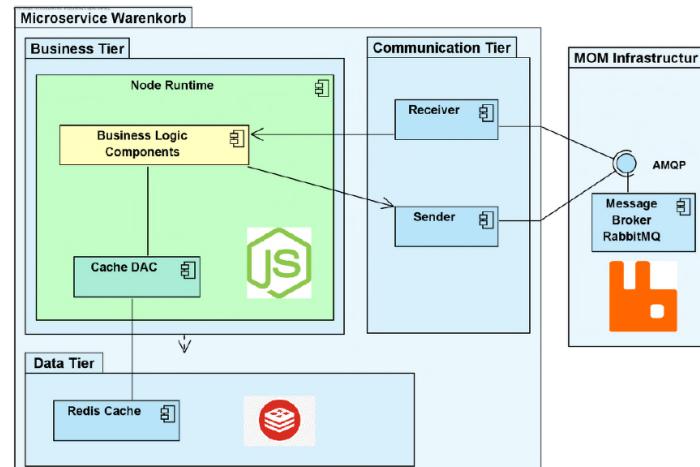
6.1 Beispiel E-Shop



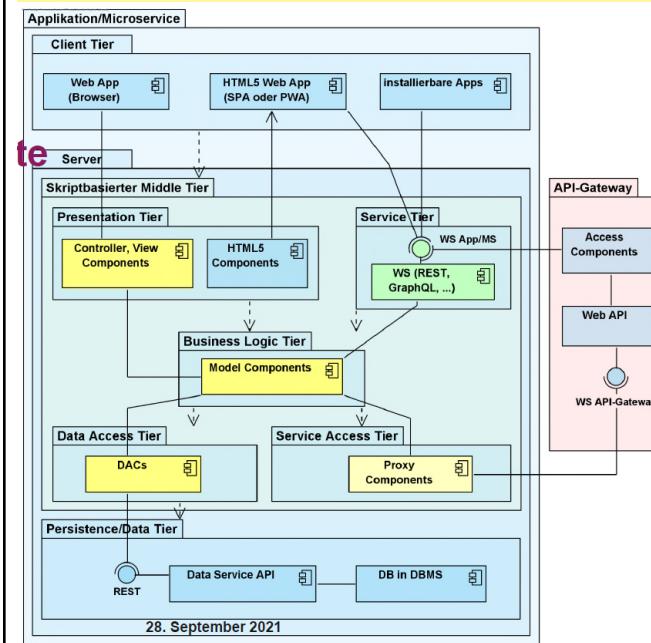
6.2 UML Komponentendiagramm



6.3 Beispiel Mikroarchitektur eines Microservice

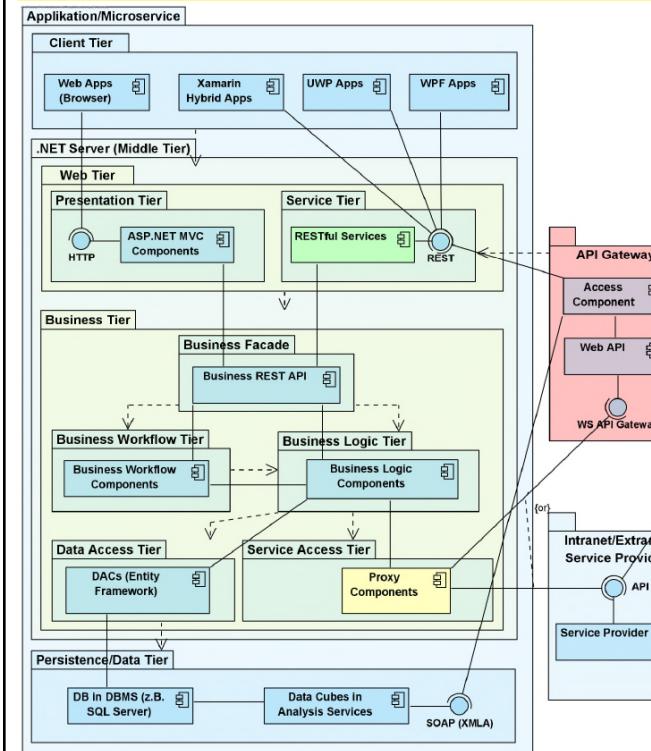


6.4 Referenzarchitektur Skriptbasiert

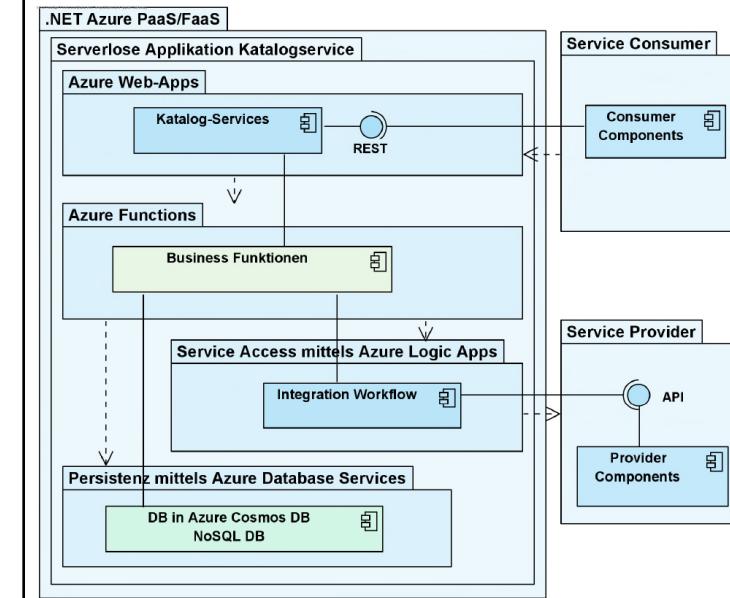


7 Serverseitige Architekturen: .NET

7.1 Referenzarchitektur

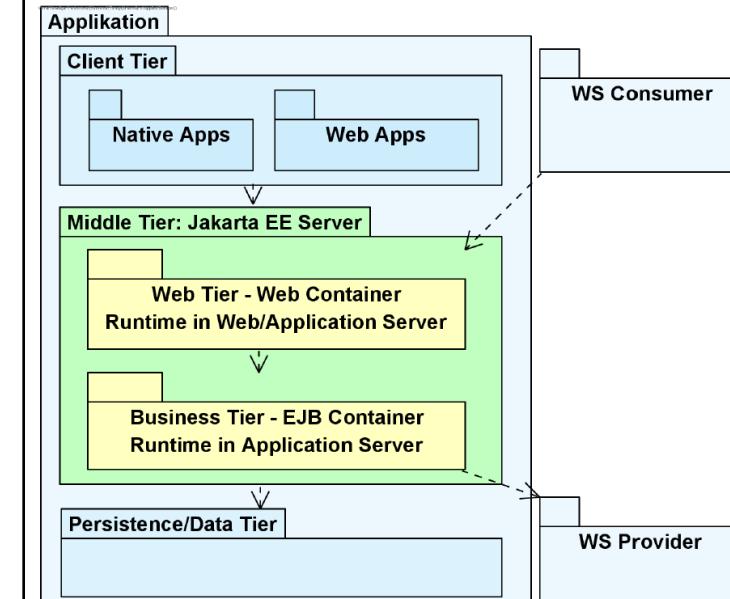


7.2 Mikroarchitektur (Azure)

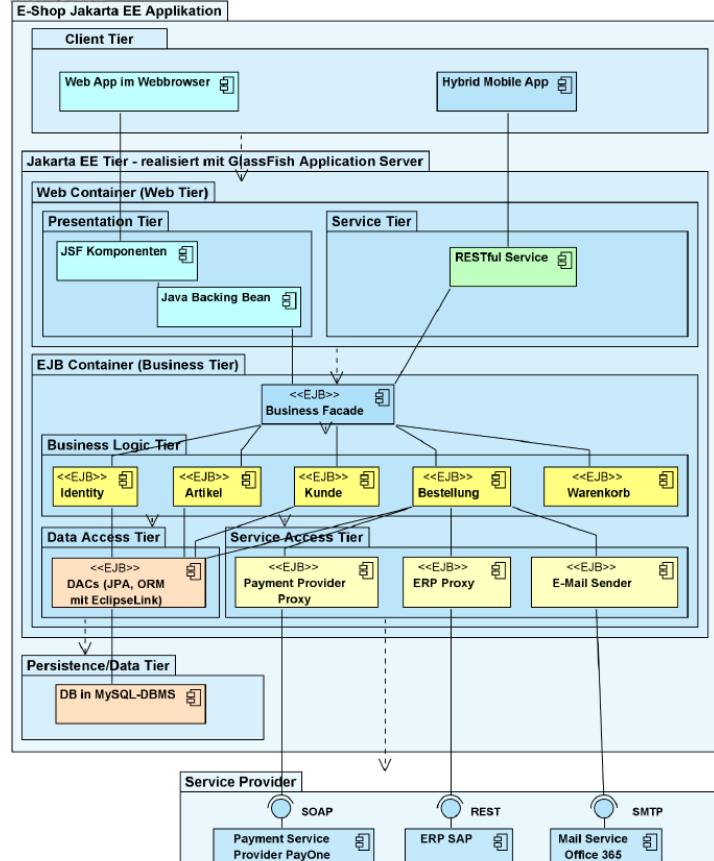
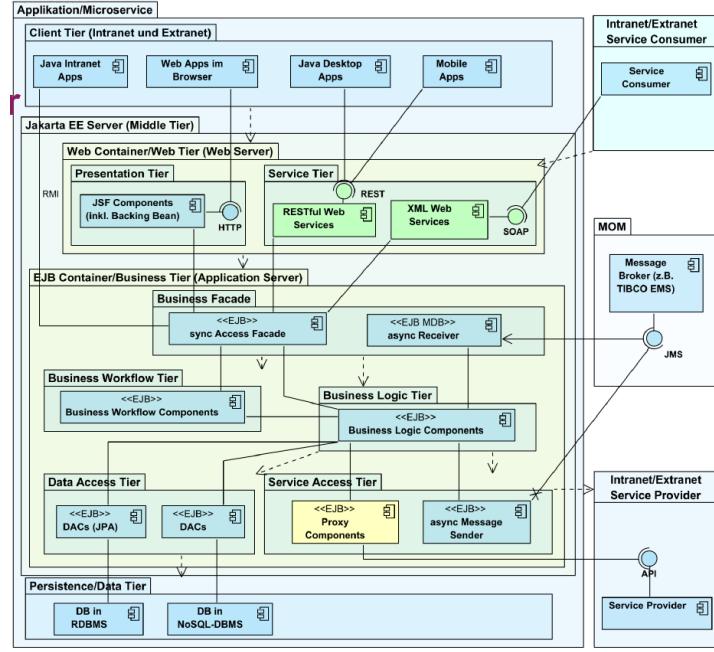


8 Serverseitige Architekturen: Jakarta EE

8.1 Basisarchitektor

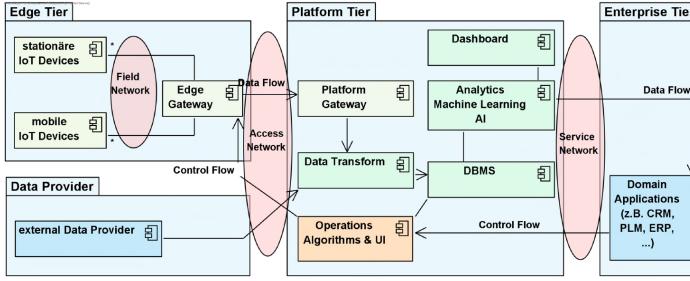


8.2 Musterarchitektur (Full Stack)

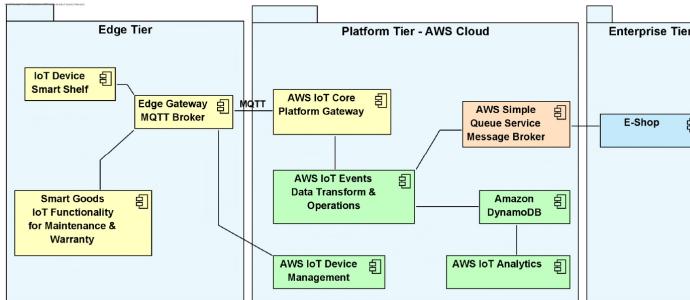


9 IoT Architecture

9.1 Referenzarchitektur



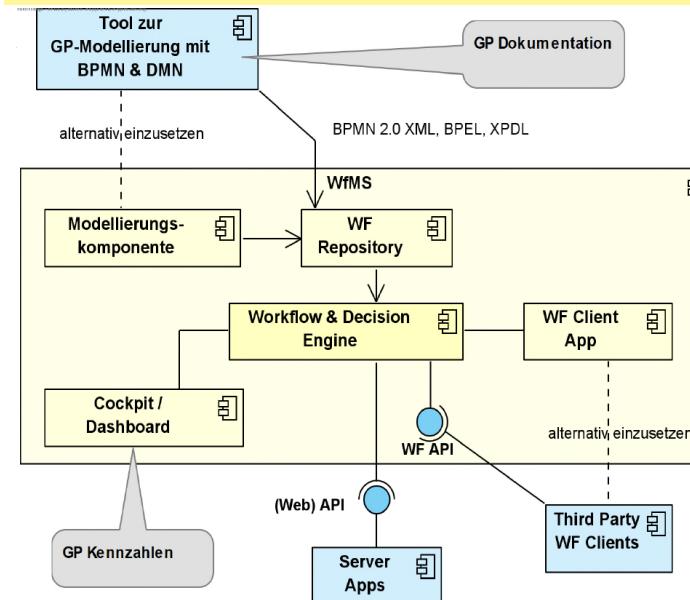
9.2 Beispiel: E-Commerce-Kontext



10 Einführung in die Anwendungsintegration

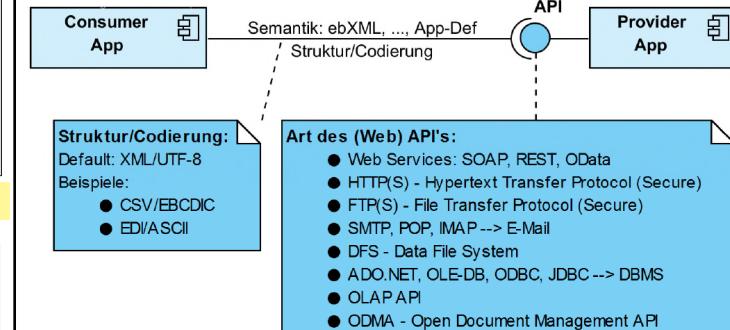
11 Integration auf Präsentationsebene

11.1 Referenzarchitektur Workflow Management System (WfMS)



12 Integration auf Applikationsebene

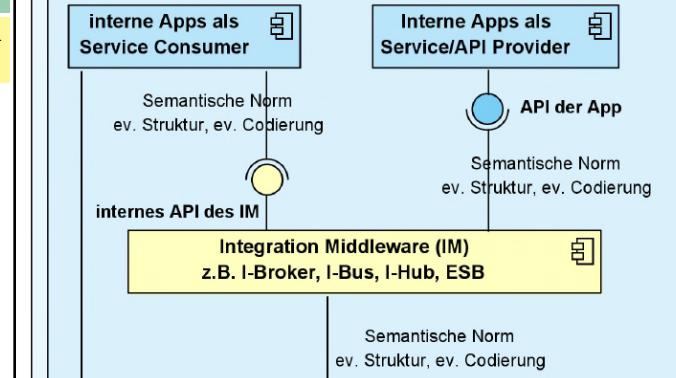
12.1 Beschreibung einer Applikationsschnittstelle



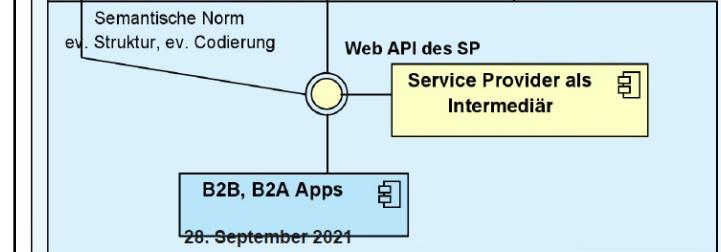
12.2 Referenz Integrations-Architektur

Integrationsarchitektur

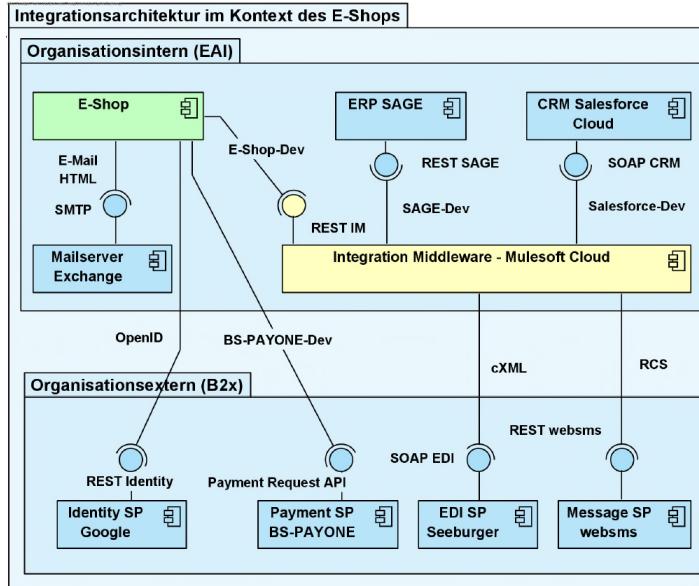
Organisationsinterne Applikationen (EAI)



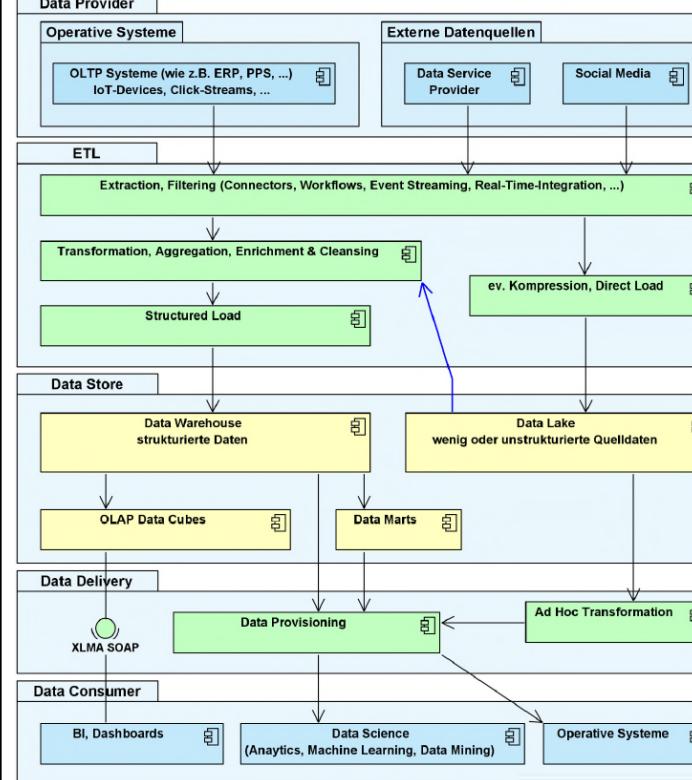
Organisationsexterne Applikationen (B2x)



12.3 Beispiel Integrations-Architektur



13.2 Zentrales Data Warehouse & Data Lake



13 Integration auf Datenebene

