HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Blatt 8

Software Engineering 2

Luis Staudt

Aufgabe 1

Client-Server Architektur mit Microservices

Überblick der Architektur

Die vorgeschlagene Architektur folgt dem Microservices-Architekturstil und besteht aus mehreren unabhängigen Services, die über definierte APIs kommunizieren. Ein API-Gateway dient als zentraler Eingangspunkt für alle Client-Anfragen.

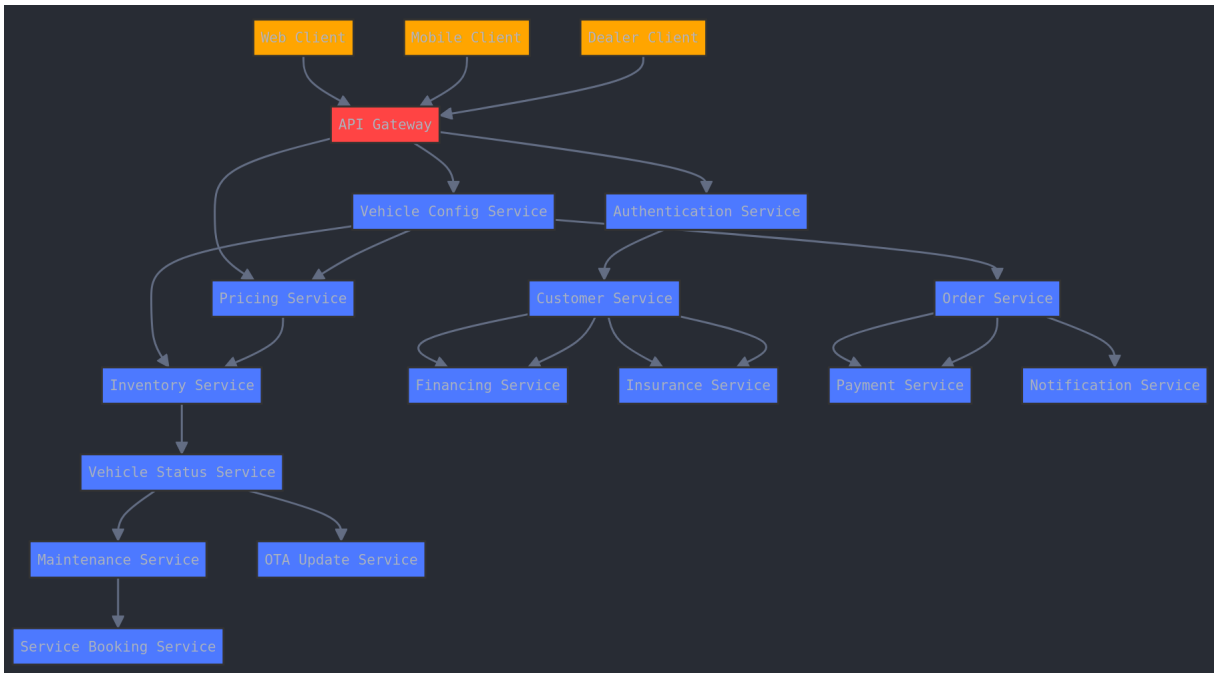

Abbildung 1: Überblick der Microservices-Architektur

Designaspekte der Microservices-Architektur

a) Dekomposition von Microservices

Logische Sicht der Microservices

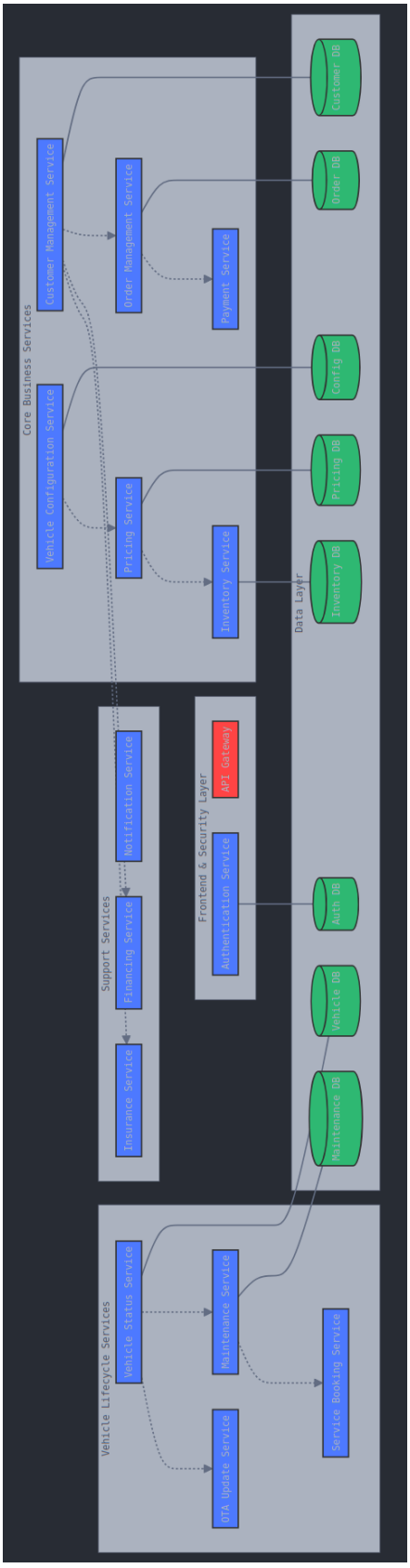

Abbildung 2: Logische Sicht der Microservices-Dekomposition

Beschreibung der Microservices

Authentication Service: Verwaltet die Benutzerauthentifizierung und -autorisierung für alle Clients (Kunden, Händler, Servicepartner). Stellt JWT-Token aus und validiert Berechtigungen für den Zugriff auf verschiedene Services. Implementiert Multi-Factor-Authentication für erhöhte Sicherheit.

Vehicle Configuration Service: Verwaltet die Fahrzeugkonfiguration und ermöglicht Kunden die Auswahl von Modellen, Motorisierungen, Farben und Ausstattungsmerkmalen. Stellt Konfigurations-APIs bereit und validiert die Kompatibilität der gewählten Optionen. Speichert Konfigurationsdaten und stellt diese anderen Services zur Verfügung.

Pricing Service: Berechnet Echtzeit-Preise basierend auf der Fahrzeugkonfiguration und aktuellen Marktbedingungen. Verwaltet Preislisten, Rabatte und Aktionen für verschiedene Märkte und Kundengruppen. Integriert externe Preisdatenquellen und führt dynamische Preisberechnungen durch.

Customer Management Service: Verwaltet alle Kundeninformationen, Profile und Präferenzen zentral. Stellt APIs für CRUD-Operationen auf Kundendaten bereit und gewährleistet DSGVO-Konformität. Integriert Kundensegmentierung und personalisierte Empfehlungen.

Order Management Service: Verarbeitet Bestellungen von der Konfiguration bis zur Auslieferung und verwaltet den gesamten Bestellworkflow. Koordiniert mit anderen Services für Preisberechnung, Lagerverfügbarkeit und Zahlungsabwicklung. Stellt Bestellstatus-Updates und Tracking-Informationen bereit.

Inventory Service: Verwaltet die Verfügbarkeit von Fahrzeugen, Komponenten und Ersatzteilen in Echtzeit. Synchronisiert Lagerbestände über verschiedene Standorte und Lieferanten hinweg. Stellt Verfügbarkeitsprüfungen für Konfigurationen und geschätzte Lieferzeiten bereit.

Payment Service: Verarbeitet alle Zahlungstransaktionen sicher und unterstützt verschiedene Zahlungsmethoden. Integriert externe Zahlungsdienstleister und gewährleistet PCI-DSS-Konformität. Verwaltet Zahlungspläne, Raten und Rückerstattungen.

Financing Service: Stellt verschiedene Finanzierungsoptionen bereit und berechnet Raten und Konditionen. Integriert externe Finanzpartner und führt Bonitätsprüfungen durch. Verwaltet Leasingverträge und Finanzierungsanträge.

Insurance Service: Bietet Versicherungsoptionen und ermöglicht den Abschluss von Versicherungen direkt im Portal. Integriert Versicherungspartner und berechnet Prämien basierend auf Fahrzeug und Fahrer. Verwaltet Versicherungsverträge und Schadensmeldungen.

Vehicle Status Service: Überwacht den Status aller verkauften Fahrzeuge und sammelt Telemetriedaten. Stellt Echtzeitinformationen über Fahrzeugzustand, Position und Leistungsdaten bereit. Erkennt potenzielle Probleme und löst präventive Wartungsempfehlungen aus.

Maintenance Service: Verwaltet Wartungspläne, Servicetermine und Reparaturhistorien für alle Fahrzeuge. Sendet Erinnerungen für anstehende Wartungen und koordiniert mit Servicepartnern. Stellt Wartungsempfehlungen basierend auf Fahrzeugdaten und -nutzung bereit.

OTA Update Service: Verwaltet Over-The-Air-Softwareupdates für Fahrzeuge und stellt sichere Update-Mechanismen bereit. Führt Rollout-Strategien durch und überwacht Update-Erfolg. Ermöglicht Feature-Freischaltungen und Konfigurationsänderungen remote.

Service Booking Service: Ermöglicht die Buchung verschiedener Services wie Helpdesk, Notruf oder Zusatzleistungen. Verwaltet Service-Kataloge und Verfügbarkeiten von Servicepartnern. Koordiniert Terminplanung und Ressourcenzuweisung.

Notification Service: Sendet Benachrichtigungen über verschiedene Kanäle (E-Mail, SMS, Push) an Kunden und Partner. Verwaltet Benachrichtigungsvorlagen und Personalisierung. Stellt Event-basierte Benachrichtigungen für wichtige Systemereignisse bereit.

b) Kommunikation zwischen Microservices

Synchrone vs. Asynchrone Kommunikation

Für das Auto Portal wird eine hybride Kommunikationsstrategie vorgeschlagen, die sowohl synchrone als auch asynchrone Kommunikation nutzt. Synchrone Kommunikation wird für zeitkritische Operationen wie Preisberechnungen und Verfügbarkeitsprüfungen verwendet, da Kunden sofortiges Feedback erwarten. Asynchrone Kommunikation wird für nicht-zeitkritische Prozesse wie Benachrichtigungen, Wartungsempfehlungen und Datenreplikation eingesetzt. Diese Strategie optimiert die Benutzererfahrung durch schnelle Antwortzeiten bei kritischen Operationen und verbessert gleichzeitig die Systemresilienz durch lose Kopplung bei unkritischen Prozessen.

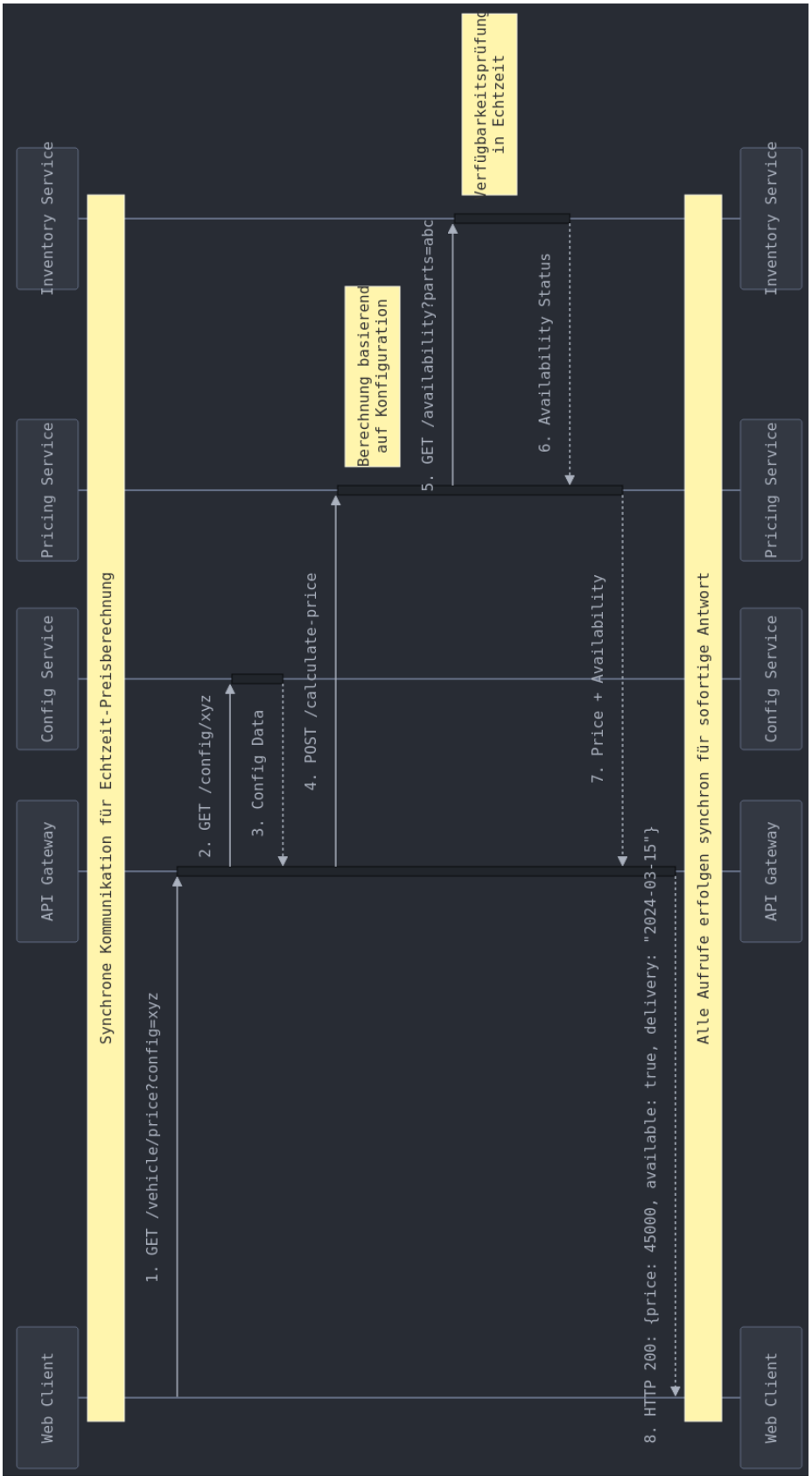

Abbildung 3: Synchrone Kommunikation - Preisberechnung mit Verfügbarkeitsprüfung

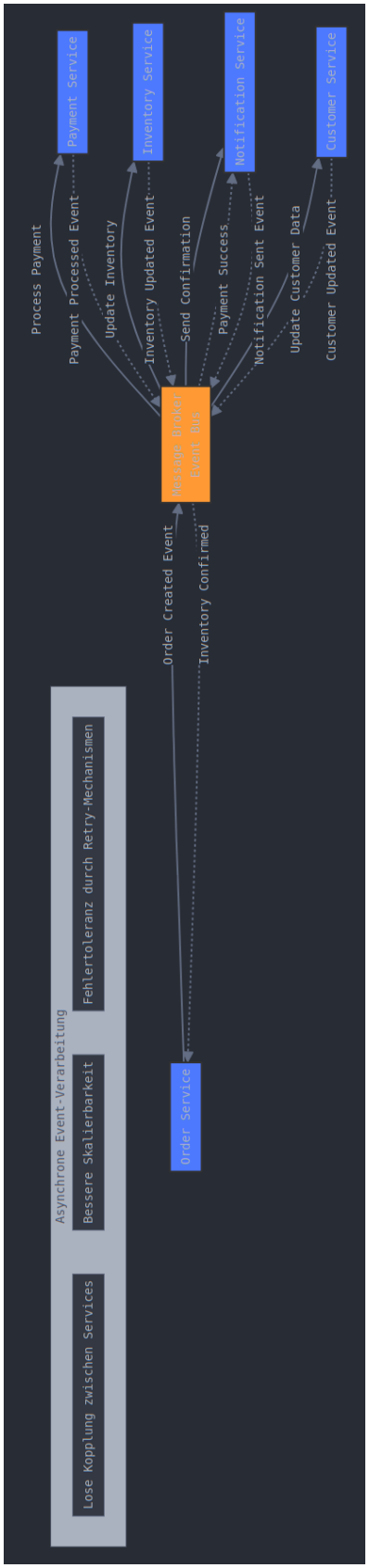

Abbildung 4: Asynchrone Kommunikation - Event-basierte Bestellverarbeitung

Direkte vs. Indirekte Kommunikation

Direkte Kommunikation wird für einfache Request-Response-Szenarien zwischen Services verwendet, bei denen eine klare 1:1-Beziehung besteht. Indirekte Kommunikation über Message Broker wird für komplexe Workflows eingesetzt, bei denen mehrere Services auf Ereignisse reagieren müssen. Der Message Broker entkoppelt Services voneinander und ermöglicht eine bessere Skalierbarkeit und Fehlertoleranz. Pattern wie Publish-Subscribe werden für Ereignisse verwendet, die von mehreren Services verarbeitet werden müssen, während direkte API-Aufrufe für spezifische Datenanfragen genutzt werden.

Abbildung 5: Direkte Kommunikation zwischen Services

c) Verteilung und gemeinsame Nutzung von Daten

Datenverwaltungsstrategie

Das System folgt dem “Database per Service“-Pattern, wobei jeder Microservice seine eigenen Daten verwaltet und andere Services nicht direkt auf diese Datenbank zugreifen können. Für gemeinsam genutzte Referenzdaten werden dedizierte Services bereitgestellt. Event Sourcing wird für kritische Geschäftsereignisse verwendet, um Konsistenz zwischen Services zu gewährleisten. CQRS (Command Query Responsibility Segregation) trennt Lese- und Schreibvorgänge für bessere Performance und Skalierbarkeit.

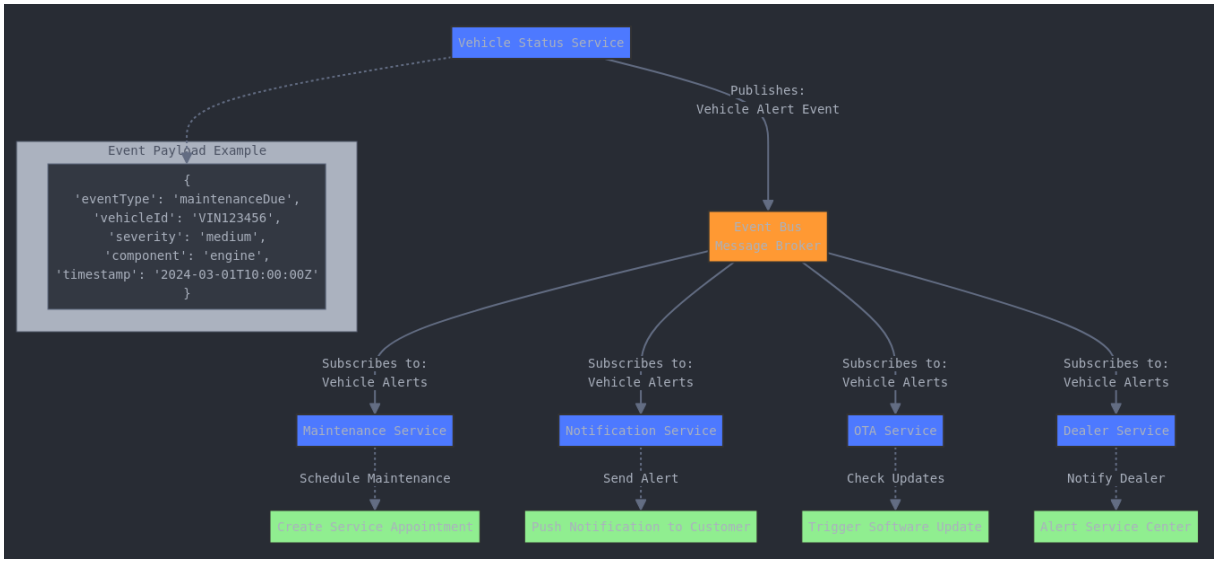

Abbildung 6: Indirekte Kommunikation über Message Broker

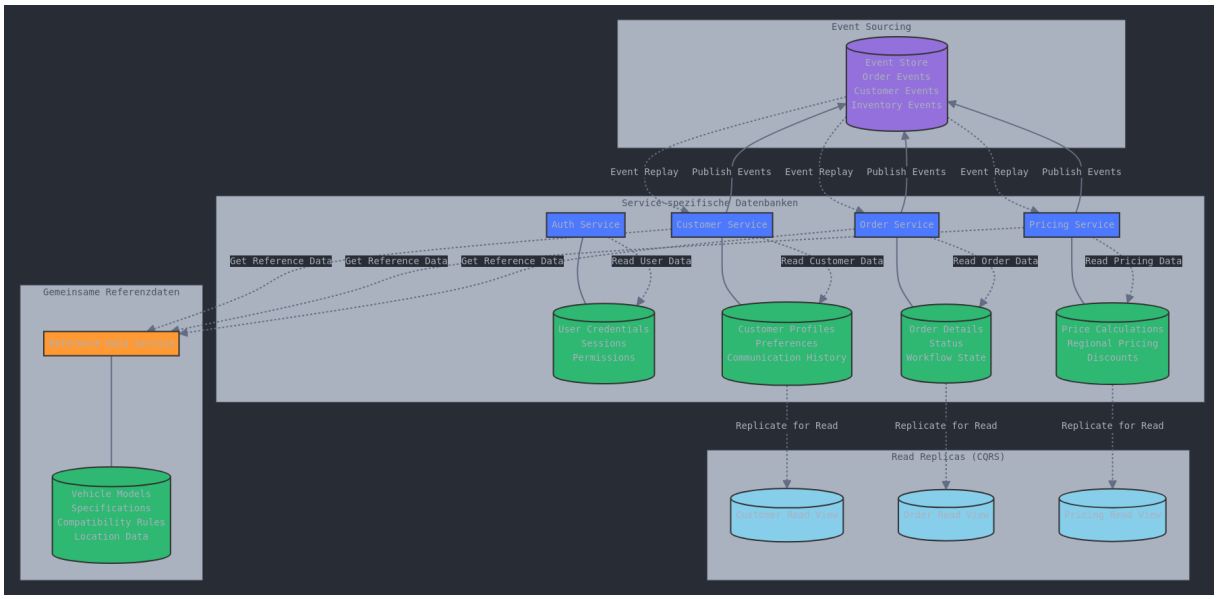

Abbildung 7: Datenverteilung und -sharing zwischen Microservices

Datenklassifizierung

Exklusive Servicedaten:

- Authentication Service: Benutzeranmeldedaten, Passwort-Hashes, Session-Tokens
- Customer Service: Persönliche Kundendaten, Präferenzen, Kommunikationshistorie
- Payment Service: Zahlungsinformationen, Transaktionsdetails, Kreditkartendaten
- Order Service: Bestelldetails, Status, Workflow-Zustand

Gemeinsam genutzte Daten:

- Vehicle Reference Data: Modelle, Spezifikationen, Kompatibilitätsregeln
- Pricing Data: Basispresse, Währungskurse, regionale Aufschläge
- Location Data: Händlerstandorte, Servicezentren, Lagerstandorte
- Configuration Rules: Gültige Kombinationen, Abhängigkeiten

d) Koordination von Microservices

Koordinationsstrategie

Für die Koordination der Microservices wird eine hybride Strategie aus Choreografie und Orchestration verwendet. Choreografie wird für einfache, lose gekoppelte Workflows bevorzugt, da sie die Autonomie der Services erhöht und Single Points of Failure vermeidet. Orchestration wird nur für komplexe Geschäftsprozesse eingesetzt, die eine zentrale Kontrolle erfordern, wie die Bestellabwicklung. Das API Gateway fungiert als zentraler Koordinationspunkt für Client-Anfragen, während interne Service-zu-Service-Kommunikation hauptsächlich event-driven abläuft. Service Mesh-Technologien unterstützen bei der Service Discovery und Load Balancing.

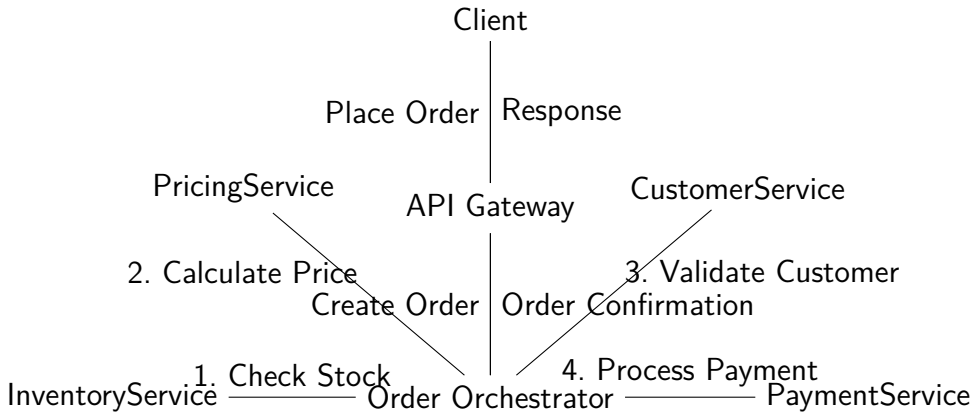

Abbildung 8: Orchestration Pattern für komplexe Bestellabwicklung

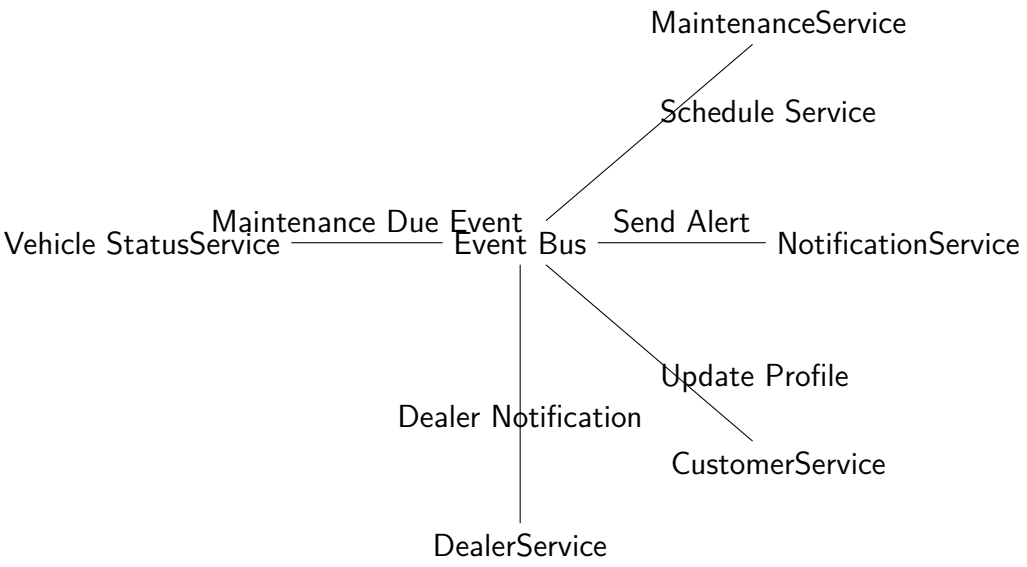

Abbildung 9: Choreography Pattern für Event-basierte Koordination

e) Fehlermanagement und Überwachung

Überwachungsstrategie

Das Überwachungssystem implementiert ein zentralisiertes Monitoring-Dashboard, das Metriken, Logs und Traces von allen Microservices sammelt und korreliert. Health Checks überwachen die Verfügbarkeit und Performance jedes Services kontinuierlich. Distributed Tracing verfolgt Anfragen durch das gesamte System und identifiziert Bottlenecks. Circuit Breaker Pattern verhindert Kaskadenausfälle bei Service-Fehlern. Alerting-Mechanismen benachrichtigen Operations-Teams bei kritischen Problemen automatisch.

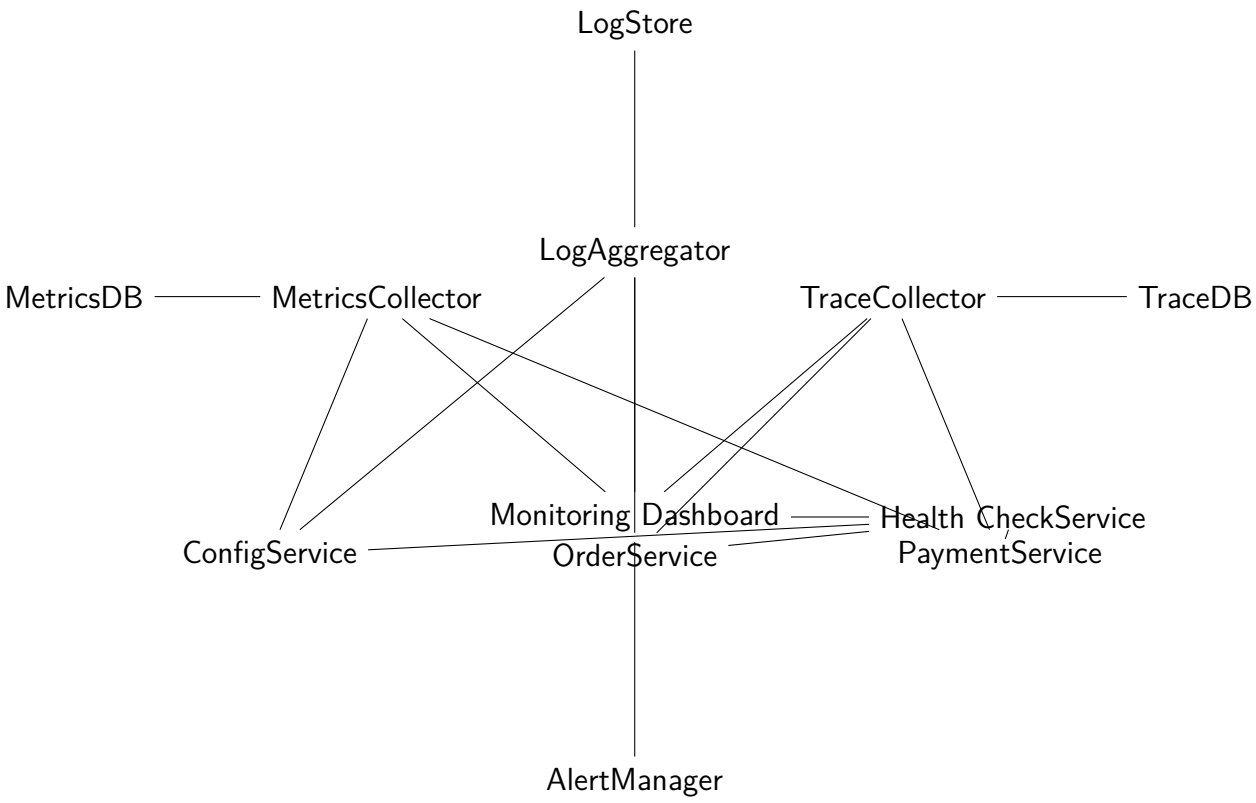

Abbildung 10: Integriertes Überwachungs-Dashboard für Microservices

Fehlerbehandlungsstrategien

1. **Circuit Breaker Pattern:** Verhindert wiederholte Aufrufe fehlgeschlagener Services und ermöglicht schnelle Wiederherstellung.

2. **Retry mit Exponential Backoff:** Automatische Wiederholung fehlgeschlagener Anfragen mit zunehmenden Wartezeiten.

3. **Bulkhead Pattern:** Isolierung kritischer Ressourcen zur Verhinderung von Kaskadenausfällen.

4. **Graceful Degradation:** Reduzierte Funktionalität bei Teilausfällen statt kompletter Systemausfall.

5. **Chaos Engineering:** Proaktive Testung der Systemresilienz durch kontrollierte Fehlerinjektionen.

Fazit

Die vorgeschlagene Microservices-Architektur für das Auto Portal bietet eine skalierbare, resiliente und wartbare Lösung für die komplexen Anforderungen des Online-Automobilverkaufs.

Durch die sorgfältige Dekomposition in fachlich abgegrenzte Services, eine durchdachte Kommunikationsstrategie und robuste Überwachungsmechanismen wird eine Architektur geschaffen, die sowohl aktuelle als auch zukünftige Geschäftsanforderungen erfüllen kann.

Die hybride Ansätze bei Kommunikation und Koordination ermöglichen es, die Vorteile verschiedener Patterns zu nutzen und gleichzeitig die Komplexität überschaubar zu halten. Das umfassende Monitoring- und Fehlerbehandlungskonzept gewährleistet hohe Verfügbarkeit und schnelle Problemidentifikation.

Diese Architektur bildet eine solide Grundlage für die digitale Transformation des Automobilvertriebs und unterstützt innovative Services wie Over-The-Air-Updates und präventive Wartung.