Justin C.
David H.
Sanju P.

DRINKS SERVICE

Inhalt

Inhalt

Git-Repository: https://github.com/justindavidcalle/Micro-Shop

Einleitung

In diesem Projekt fokussieren wir uns auf die Entwicklung spezifischer Microservices, die für die Funktionalität unseres Shops wichtig sind. In unserem Shop verkaufen und liefern wir Alkohol. Jeder Microservice wird eine klar definierte Aufgabe erhalten und über REST APIs mit den anderen Services kommunizieren. Die Anwendung soll aus den folgenden Elementen bestehen:

Ein einfaches Frontend (React/Angular)

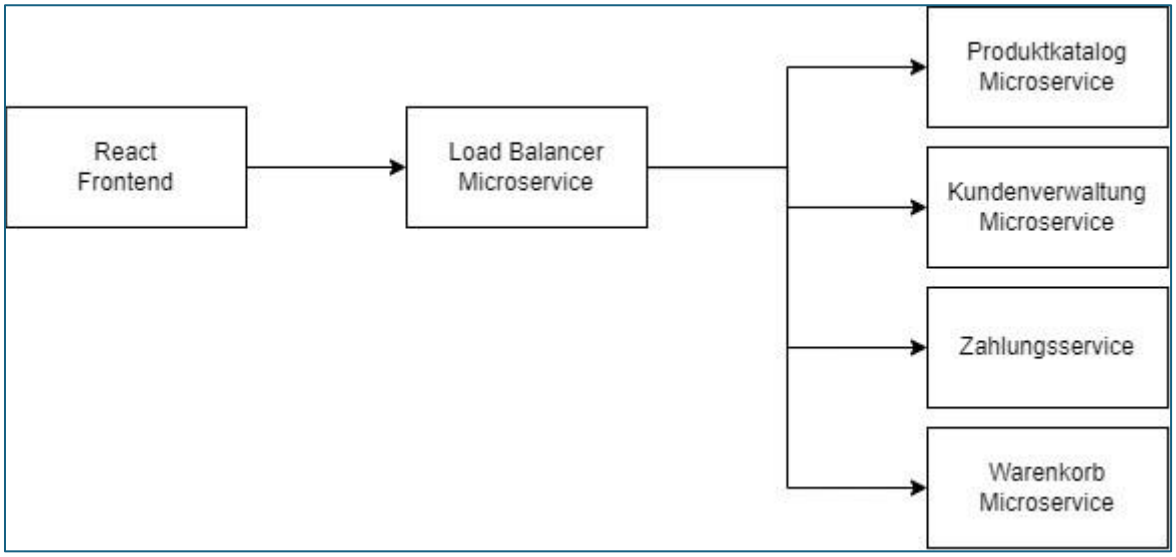
Ein Warenkorb-Microservice

Drei bis vier weitere Microservices

Unsere Shop-Anwendung soll in sich geschlossen "Sinn" ergeben, auch wenn sie nicht vollständig ausprogrammiert ist. Wir wählen zusätzlich zum Frontend und dem Warenkorb weitere Services aus. Wir haben uns für die folgende Microservices entschieden, den Produktkatalog, die Kundenverwaltung, den Zahlungsservice und den Bestellungsservice. Ziel ist es, eine Anwendung zu erstellen, die diese Services implementiert und anwendet.

Beispielsweise könnte ein Bestelldurchgang wie folgt aussehen:

Vom Gateway und Frontend zum Backend indem verschiedene Microservices aufgerufen werden.



Diese Dokumentation beschreibt unsere Herangehensweise bei der Entwicklung der Microservices, die Implementierung der gewählten Elemente und die Integration der verschiedenen Komponenten zu einer funktionalen Shop-Anwendung.

Services

Gateway

Unser Gateway ist mit Springboot aufgebaut. Auf https://start.spring.io kann jegliche Services erstellt werden. Für den Gateway brauchen wir folgende Dependencies:

- **Reactive Gateway**
- **Resilience4J**
- **Contract Stub Runner**

Produktkatalog

Der Produktkatalog-Service verwaltet alle Informationen zu den im Shop verfügbaren Produkten. Die Hauptaufgaben dieses Services umfassen:

- **Erstellen, Lesen, Aktualisieren und Löschen (CRUD) von Produkten**
- **Produkt-Suche und Filterung**

Dieser Service verwendet eine relationale Datenbank (z.B. MySQL oder PostgreSQL) zur Speicherung der Produktinformationen und stellt REST-APIs für den Zugriff auf diese Daten bereit.

Folgende Depencies hat der Produktkatalog:

- Lombok
- Spring Web

Kundenverwaltung

Der Kundenverwaltungs-Service kümmert sich um die Speicherung und Verwaltung von Kundendaten. Zu seinen Hauptaufgaben gehören:

- **Kundenregistrierung und -authentifizierung**
- **Verwaltung von Kundenprofilen**

Dieser Service verwendet eine Benutzerdatenbank und stellt sichere REST-APIs für die Interaktion mit den Kundendaten zur Verfügung. Authentifizierung und Autorisierung können durch Integration mit Diensten wie OAuth2 oder JWT (JSON Web Tokens) realisiert werden.

Zahlungsservice

Der Zahlungsservice ist für die Abwicklung von Zahlungen zuständig und stellt sicher, dass Transaktionen sicher und effizient durchgeführt werden. Seine Hauptaufgaben umfassen:

- **Integration mit Zahlungsanbietern**
- **Verwaltung von Zahlungsvorgängen**

Der Service kommuniziert sicher mit den Zahlungsgateways und stellt APIs bereit, die vom Bestellservice aufgerufen werden können, um Zahlungen zu initiieren und deren Status zu überprüfen.

Warenkorb

Der Warenkorb-Service verwaltet die Artikel, die Kunden ihrem Warenkorb hinzufügen, bevor sie zur Kasse gehen. Zu den Hauptfunktionen dieses Services gehören:

- **Hinzufügen und Entfernen von Artikeln**
- **Verwaltung des Warenkorb-Inhalts**
- **Berechnung des Gesamtpreises**

Dieser Service kann eine In-Memory-Datenbank oder eine NoSQL-Datenbank (wie Redis) verwenden, um die Warenkorbdaten schnell und effizient zu speichern und abzurufen.

Bestellungsservice

Der Bestellungsservice ist für die Verwaltung und Verarbeitung von Bestellungen verantwortlich. Zu seinen Aufgaben gehören:

- **Erstellung und Verfolgung von Bestellungen**
- **Kommunikation mit dem Zahlungsservice**
- **Versand und Lieferung**

Dieser Service interagiert mit anderen Microservices wie dem Produktkatalog und der Kundenverwaltung, um alle notwendigen Informationen für die Bearbeitung von Bestellungen zu sammeln und bereitzustellen.

Architektur

Unsere Architektur ist auch auf Github zu sehen, im develop branch:

https://github.com/justindavidcalle/Micro-Shop/tree/develop

Es wäre durchaus von Vorteil, ein Docker-Compose.yml hinzuzufügen und Images zu erstellen, sodass alles in einem Container läuft. Wir konnten dies leider nicht ganz umsetzen, da wir nur wenig Zeit hatten.

Unser Frontend ist nur mit dem Gateway verbunden, somit können Zugriffe besser verwaltet werden. Dieses Gateway könnte man mit einem Eureka-Server verbinden, um die andere Services zu notieren, aber hier verwenden wir den Gateway direkt mit den Services. Unsere Services wurden alle mit Springboot erstellt und erweitert. Konzentriert man sich nur auf ein Tool, vereinfacht dies die Vielfalt und haltet alles möglich einheitlich.

Reflexion

Im Grossen und Ganzen hat uns das Projekt ziemlich viel Spass gemacht, jedoch hatten wir immer wieder unsere Schwierigkeiten, vorallem beim Gateway Microservice, da dieser sehr komplex und schwierig einzubauen war. Das Beste am Projekt fanden wir unsere Kommunikation und allgemeine Stimmung im Team. Wir haben uns daher sehr gut verstanden und konnten zusammen alle aufgetretenen Probleme lösen. Wir fanden das wir das Projekt trotz der Zeitkürzung gut umgesetzt haben.

Quellen

Titelblatt: www.canva.com