Übungszettel – Microservices

Name: Michael Bogensberger

Datum: 15.06.2022

Inhalt

[2 Was sind Microservices? 3](#_Toc106385666)

[3 Was ist ein Message Broker? 3](#_Toc106385667)

[3.1 Was ist RabbitMQ? 3](#_Toc106385668)

[4 Inbetriebnahme 4](#_Toc106385669)

[5 Architekturbeschreibung 4](#_Toc106385670)

[5.1 API Gateway 4](#_Toc106385671)

[5.1.1 Spring Cloud Gateway 4](#_Toc106385672)

[5.2 Netflix Eureka 4](#_Toc106385673)

[5.3 Service-Discovery 5](#_Toc106385674)

[5.4 C4-Diagramme 6](#_Toc106385675)

[5.4.1 System Context 6](#_Toc106385676)

[5.4.2 Component View 6](#_Toc106385677)

[5.5 Codeauszüge 7](#_Toc106385678)

[5.6 Use-Cases 7](#_Toc106385679)

[5.6.1 Bestellung anlegen 7](#_Toc106385680)

[5.6.2 Payment verifizieren 7](#_Toc106385681)

[5.6.3 Packlistenitems abhaken 7](#_Toc106385682)

[6 Abbildungsverzeichnis 8](#_Toc106385683)

# Was sind Microservices?

Microservices sind ein architekturbezogener und organisatorischer Ansatz in der Softwareentwicklung, bei dem Software aus kleinen unabhängigen Services besteht, die über sorgfältig definierte APIs kommunizieren.

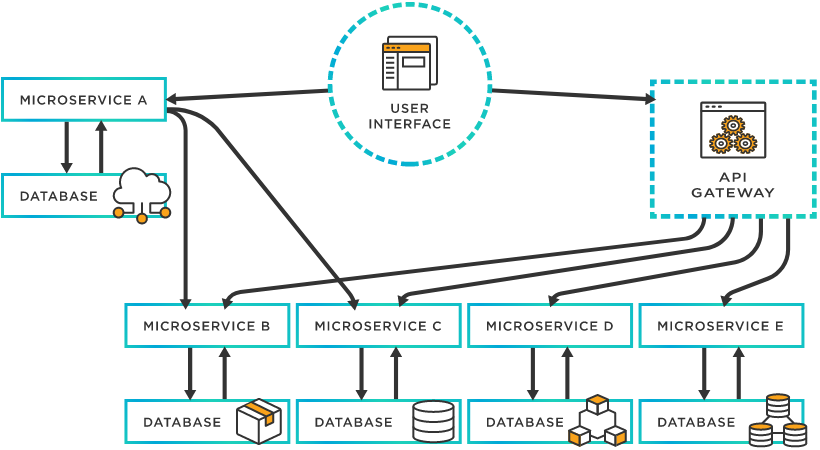
Sie sind ein Architekturkonzept für die Anwendungsentwicklung. Als Architektur-Frameworks sind Microservices verteilt und lose gekoppelt, sodass die Änderungen eines Teams nicht dazu führen können, dass die gesamte Anwendung nicht mehr funktioniert. Der Vorteil bei der Verwendung von Microservices liegt darin, dass Entwicklungsteams schnell neue Anwendungskomponenten bauen können, um sich ändernden geschäftlichen Anforderungen gerecht zu werden.

Abbildung Microservices Grafik

# Was ist ein Message Broker?

In der IT müssen ständig Nachrichten von einem Dienst zum anderen geleitet werden. Das muss auf eine kontrollierte Weise geschehen, sonst blockieren sich Nachrichten gegenseitig, es entsteht ein Stau und Prozesse können nicht optimal ablaufen. Damit Anwendungen problemlos miteinander kommunizieren, ist es sinnvoll, einen Mittelsmann einzuschalten – einen Dienst, der die Verteilung der Nachrichten übernimmt. Diesen nennt man Messaging Broker.

Zunächst hat der Message Broker als Middle Ware die Aufgabe, Nachrichten zu empfangen und an einen oder mehrere Empfänger weiterzuleiten. Nachrichten können Mails aber auch Informationen von Sensoren oder sonstige Daten sein. Dabei passt er das Nachrichtenformat an die jeweiligen Empfängersysteme an. Das hat den Vorteil, dass im Netz ein “Vermittler” etabliert wird, der ankommende Nachrichten zentral verteilt und diese dabei in die Sprache der Empfänger übersetzt.

## Was ist RabbitMQ?

RabbitMQ basiert auf der Idee des Advanced Message Queuing Protocols (AMQP). Der große Vorteil von AMQP: Sender und Empfänger müssen nicht die gleiche Programmiersprache verstehen. Inzwischen hat sich der Messaging Broker etwas von AMQP gelöst und geht dank Plug-ins auch mit anderen Nachrichtenprotokollen wie STOMP oder MQTT um – die Idee bleibt aber die gleiche: Zwischen dem Produzenten und dem Konsumenten einer Nachricht liegt eine Warteschlange. In dieser werden die Messages zwischengelagert. Einen passenden Guide findet man hier[: IONOS-RabbitMQ-Guide](https://www.ionos.at/digitalguide/websites/web-entwicklung/rabbitmq/).

# Inbetriebnahme

Möchte man das Projekt in Betrieb nehmen, so muss man die docker-compose.yaml Datei ausführen. Hier ist nur wichtig darauf zu achten, dass die Ports nicht bereits belegt sind. Ansonsten kann man die Docker Container direkt starten. Nun kann man auch schon die jeweiligen Java Programme starten. In der Startreihenfolge ist die Service-Discovery optimalerweise Nr. 1. Danach folgen dann Orders, Stock und das API Gateway.

# Architekturbeschreibung

Die Applikation besteht aus folgenden Komponenten:

* Orders Microservice (Spring Boot Applikation nach einer Ports And Adapters-Architektur und mit Implementierung einiger taktischer DDD-Muster)
* Stock Microservice (Spring Boot Applikation ohne explizite Architektur)
* Delivery Microserver(Spring Boot Applikation ohne explizite Architektur)
* API Gateway (Spring Boot Applikation mit Spring Cloud Gateway)
* Service Discovery (Spring Boot Applikation mit Netflix Eureka)
* Vanilla JavaScript Frontend in Form von 3 Seiten (Webshop, Stock, Delivery), die die MS-Infrastruktur (Backend) über das API-Gateway verwenden.

## API Gateway

Das API Gateway sorgt für die Entkoppelung von Clients und Services. Dabei bildet es den alleinigen Kontaktpunkt für eingehenden und ausgehenden Traffic. Ein Microservices-Backend kann viele miteinander interagierende Services beinhalten, wobei diese eine heterogene Struktur hinsichtlich URLs und Protokollen aufweisen können.

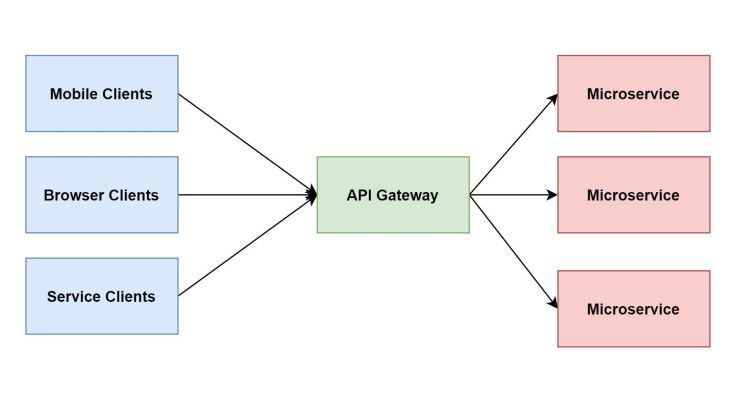


Abbildung API Gateway Grafik

### Spring Cloud Gateway

Das Spring Cloud Gateway bietet in Verbindung mit den anderen Spring Projekten wie Security, Service Discovery, Logging und Monitoring eine leistungsfähige Lösung und steht in Konkurrenz zu anderen API Gateway Lösungen. Spring Cloud Gateway bietet den Vorteil, dass es nahtlos integriert werden kann.

## Netflix Eureka

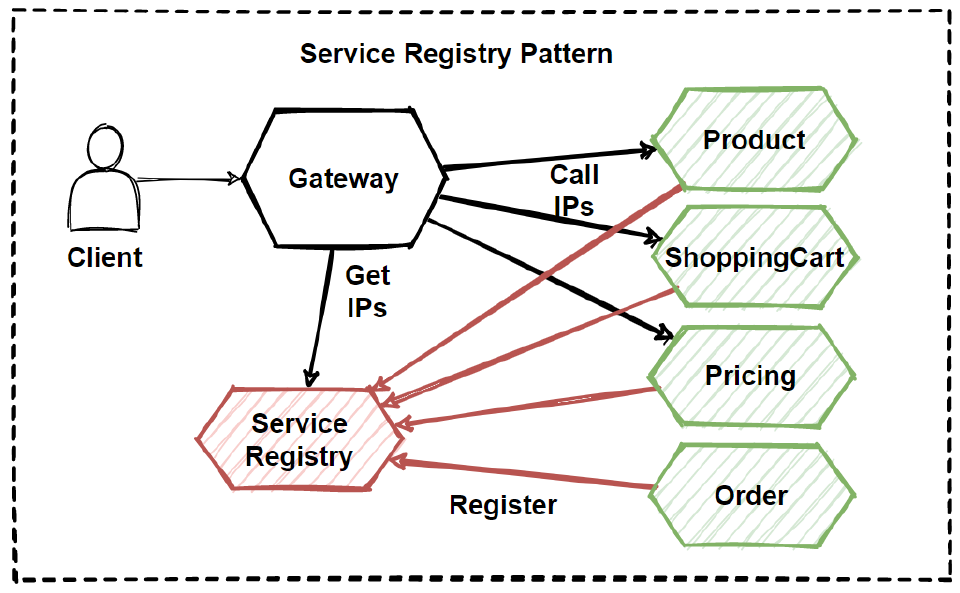
Netflix Eureka ist eine REST-basierte Middleware, die für die Erkennung und den Lastausgleich (load balancing) von Webanwendungen entwickelt wurde.

## Service-Discovery

Stellen wir uns mehrere Microservices vor, die eine mehr oder weniger komplexe Anwendung bilden. Diese werden irgendwie miteinander kommunizieren (z. B. API Rest ).

Eine auf Microservices basierende Anwendung wird normalerweise in virtualisierten oder containerisierten Umgebungen ausgeführt. Die Anzahl der Instanzen eines Dienstes und seiner Standorte ändert sich dynamisch. Wir müssen wissen, wo sich diese Instanzen befinden und wie sie heißen, damit Anfragen beim Ziel-Microservice ankommen. Hier kommen Taktiken wie Service Discovery ins Spiel.

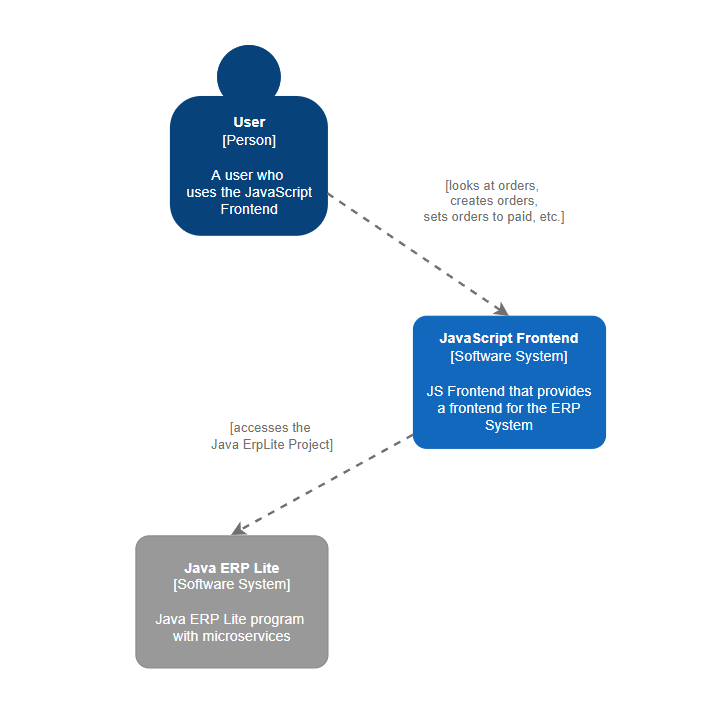
Der Service-Discovery hilft uns zu wissen, wo sich jede Instanz befindet. Auf diese Weise fungiert eine Service Discovery-Komponente als Register, in dem die Adressen aller Instanzen verfolgt werden. Die Instanzen haben dynamisch zugewiesene Netzwerkpfade. Folglich muss ein Client, wenn er eine Anforderung an einen Dienst stellen möchte, einen Service-Discovery Mechanismus verwenden.



## C4-Diagramme

In folgendem Abschnitt sind die jeweiligen C4-Diagramme zu sehen. Hier wird immer weiter ins System geblickt.

### System Context

In folgendem C4-Diagramm ist der System Context zu sehen. Sprich, hier sieht man das System von ganz außen.

### Component View

In folgendem

## Codeauszüge

Möchte

## Use-Cases

Möchte

### Bestellung anlegen

Möchte

### Payment verifizieren

Möchte

### Packlistenitems abhaken

Möchte

# Abbildungsverzeichnis

[Abbildung 1 Index Page – ERP-Lite 4](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171235)

[Abbildung 2 Svelte-Routing 5](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171236)

[Abbildung 3 place Order Modal 6](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171237)

[Abbildung 4 Order-Page 6](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171238)

[Abbildung 5 verify order Modal 6](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171239)

[Abbildung 6 packings-Page 7](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171240)

[Abbildung 7 packing to delifery Modal 7](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171241)

[Abbildung 8 add customer Modal 7](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171242)

[Abbildung 9 setPackedForPacking Mapping 8](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171243)

[Abbildung 10 packings Mapping 8](file:///D:\dev\tmp\SvelteERPliteFSE\Svelte_docu.docx#_Toc105171244)