

Programmierung eines Onlineshops mit Microservice Architektur

Philipp Honsel, Lennart Kampshoff

Inhalt

- ▶ Anforderungen
- ▶ Backend Architektur
- ▶ Frontend
- ▶ Inter-System Kommunikation
- ▶ Systemarchitektur
- ▶ Service Mesh

Anforderungen

Funktional:

- ▶ Startseite mit allen Produkten
- ▶ Warenkorb mit Kontaktfeldern
- ▶ Bestelldetails mit allen Daten

Nicht-Funktional

- ▶ Kommunikation über REST und JSON
- ▶ Service Urls dürfen nicht Hardcoded sein
- ▶ System muss auf Knopfdruck bereitstehen

Backend Architektur

- ▶ Technologien
- ▶ Warenkorb
- ▶ Generische Antworten
- ▶ Fehlerbehandlung
- ▶ Sicherheit

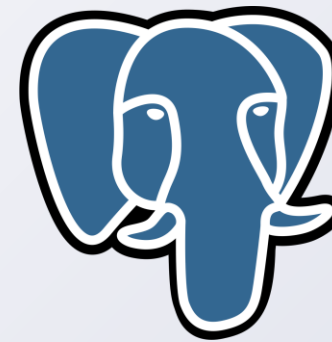
Backend Architektur - Technologien



ASP.NET Core



Entity Framework
Core



PostgreSQL



Seq Logging

Backend Architektur - Warenkorb



Backend Architektur - Generische Antworten

- ▶ Basis der Antwort ist immer im gleichen Format
- ▶ Übergreifen über alle Systeme
- ▶ Fehler werden immer gleich angegeben

```
1 {  
2   "success": true,  
3   "data": {  
4     "id": "123",  
5     "productName": "Beispielprodukt"  
6   },  
7   "error": null  
8 }
```

Beispielcode: Generisches Model bei erfolgreicher Abfrage

Backend Architektur - Generische Antworten

- ▶ Basis der Antwort ist immer im gleichen Format
- ▶ Übergreifen über alle Systeme
- ▶ Fehler werden immer gleich angegeben

```
1 {  
2     "success": false,  
3     "data": null,  
4     "error": [  
5         "E-Mail muss angegeben werden."  
6     ]  
7 }
```

Beispielcode: Generisches Model bei fehlgeschlagener Abfrage

Backend Architektur - Fehlerbehandlung



- ▶ In jedem Scope wird ein *NotificationHandler* Objekt angelegt
 - Hält Liste an Fehlern vor
- ▶ Tritt ein Fehler auf, wird dieser der Liste angehängen und die aktuelle Methode abgebrochen
- ▶ Im Controller wird geprüft, welches Format die Antwort haben muss
- ▶ Aufrufende Systeme brechen den Vorgang auch ab, so wird Atomarität der Transaktion gewahrt (in die aufrufende Richtung)

Backend Architektur - Fehlerbehandlung



Backend Architektur - Sicherheit

- ▶ Jeder Service erstellt beim Hochfahren einen JWT Token, in dem er claimt, welcher Service er ist
- ▶ Jeder Service hat eine Liste, von welchen Services er genutzt werden kann
- ▶ Einzelne Methoden oder ganze Endpunkte können dann geschützt und nur von erlaubten Services genutzt werden

 [Debugger](#) [Libraries](#) [Introduction](#) [Ask](#) Crafted by  auth0

Algorithm HS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmb3JlZXJ2aWNlIjoieY2hlY2tvdXQiLCJuYmYiOiJlE2NTg1MjQxOTUsImV4cCI6MjUzNDYmjk3MjAwLCJpYXQiOiJlE2NTg1MjQxOTUsImZlcyI6Imh0dHBzOi8vYXV0aGVudGljYXRpb24tc2Vydm1jZSJ9.aIFv_2NQsaRCRhJAXNdouYGTpGLWj-7_eIA206ChXYg
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```


PAYLOAD: DATA

```
{  "forService": "checkout",  "nbf": 1658524195,  "exp": 253402297200,  "iat": 1658524195,  "iss": "https://authentication-service"}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  61F7DCB32DDD1E816FBD81  
)
```

☐ secret base64 encoded

 Signature Verified

SHARE JWT

Frontend

Frontend



Angular



Nebular UI Kit



NGXS State Management



Ibu 800

1.10 EUR

[IN DEN WARENKORB](#)

Ibu 400

0.50 EUR

[IN DEN WARENKORB](#)

Ibu 600


0.75 EUR

[IN DEN WARENKORB](#)

1 Gegenstände im Warenkorb

WARENKORB LEEREN

WEITER EINKAUFEN

Produkt	Preis
 Ibu 400	0.50 EUR
= 0.50 EUR	

Kontakt


E-Mail muss im Email Format sein

Adresse

Postleitzahl muss mindestens 10000 betragen

Zahlung

KOSTENPFLICHTIG BESTELLEN

Produkt	Preis
 Ibu 400	0.50 EUR
Versandkosten	10.00 EUR
	= 10.50 EUR

Bestelldetails

Bestellnummer 46651947-c315-458a-b818-ce2839fd985c
Versandverfolgung YPEHEHDNTZNCNV4A
Versandkosten 10.00 EUR
Ausgeliefert false

Kontaktinformationen

E-Mail philipp.honsel@studmail.w-hs.de
Adresse Musterstraße, 17
46414 46414, Deutschland

Zahlungsinformationen

Kartennummer 0000000000000000
Ablaufdatum July 2023

Inter-System Kommunikation

- ▶ HTTP
- ▶ REST
- ▶ Ablauf einer Bestellung

Inter-System Kommunikation - HTTP

- ▶ Methoden
- ▶ Pfad
- ▶ Parameter
- ▶ Payload

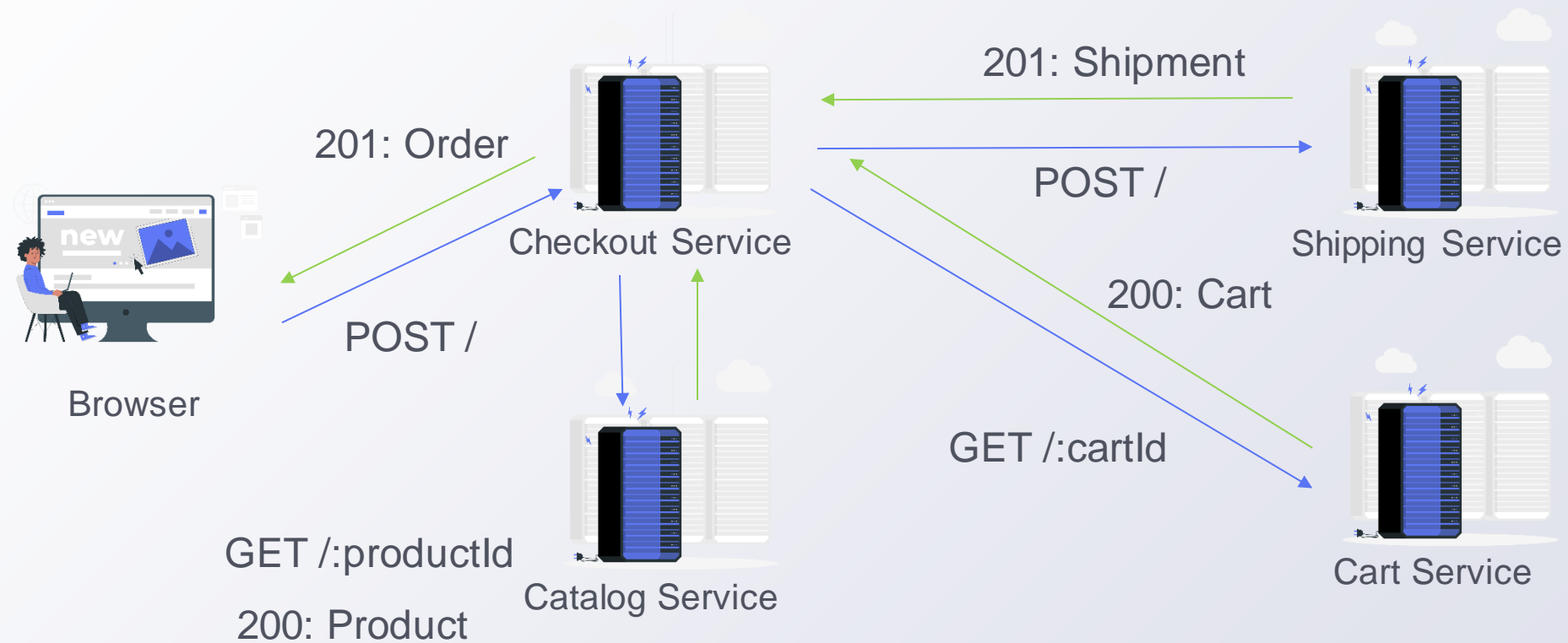
```
POST /path?parameter=value  
{  
    „data“: false  
}
```

Beispielhafte HTTP-Anfrage

Inter-System Kommunikation - REST

- ▶ HTTP-Standard Verwaltung von API-Objekten
- ▶ Verwendet alle Teile einer HTTP-Anfrage
- ▶ Standardisierte Operationen
 - Abrufen (HTTP GET)
 - Erstellen (HTTP POST)
 - Aktualisieren (HTTP PUT/PATCH)
 - Löschen (HTTP DELETE)

Inter-System-Kommunikation - Bestellung



Systemarchitektur

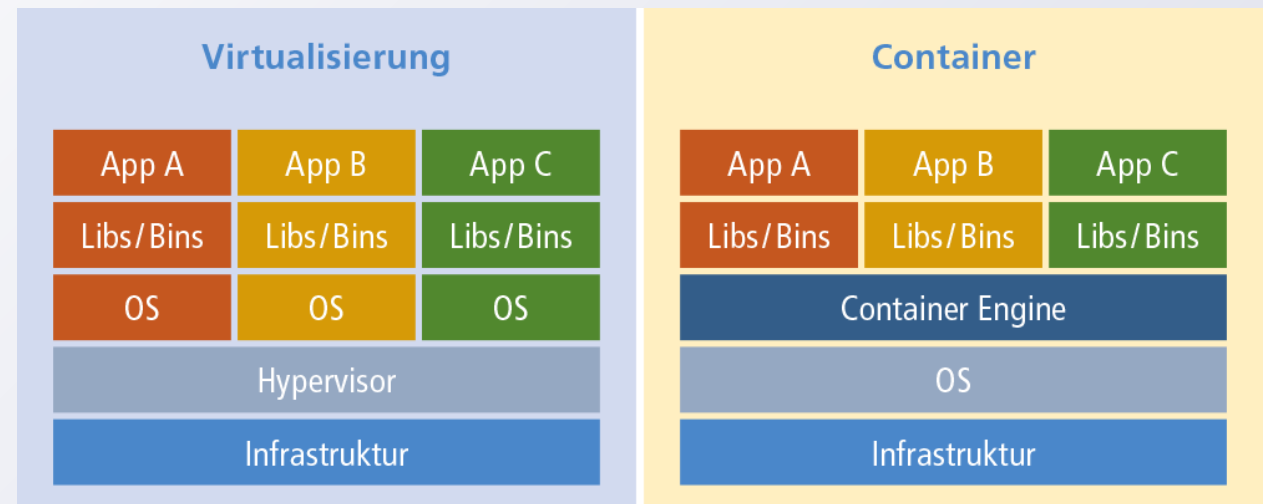
- ▶ Docker
- ▶ Vergleich zur virtuellen Maschine
- ▶ Kubernetes
- ▶ Helm

Systemarchitektur - Docker

- ▶ Container für Anwendungen
- ▶ Applikation abgekapselt
- ▶ Laufzeitumgebung wird ausgeliefert
 - Keine manuelle Installation von Abhängigkeiten
- ▶ Betriebssystem wird abstrahiert

Systemarchitektur - Vergleich zur virtuellen Maschine

- ▶ OS pro VM
 - Höherer Ressourcenverbrauch



Systemarchitektur - Kubernetes

- ▶ Große Docker Umgebungen unübersichtlich
- ▶ Hoher Verwaltungsaufwand bei mehreren Servern
- ▶ Zentrale Verwaltung des Clusters

Systemarchitektur - Kubernetes

- ▶ Desired State Principal
 - Gewünschter Status des Clusters wird angegeben
 - Kubernetes bringt das Cluster auf den gewünschten Stand

Systemarchitektur - Kubernetes

- ▶ Alles im Cluster ist ein API-Objekt
 - Deployment/Pod (Container)
 - Service
 - ConfigMap/Secret
 - Horizontal Pod Autoscaler
- ▶ Verwaltung über CLI (kubectl) oder REST-API

Systemarchitektur – Kubernetes - Autoscaler

- ▶ Horizontales Autoscaling fährt neue Pods hoch, wenn mehr Ressourcen benötigt wird
- ▶ Metriken:
 - Min: 1 Pod
 - Max: 5 Pods
 - Target CPU Percent: 75%

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
cart-service	Deployment/cart-service	0%/75%	1	5	1	82s
catalog-service	Deployment/catalog-service	0%/75%	1	5	1	82s
checkout-service	Deployment/checkout-service	1%/75%	1	5	1	82s
shipping-service	Deployment/shipping-service	1%/75%	1	5	1	82s

Abbildung: Auslastung vor Lastspitze

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
cart-service	Deployment/cart-service	76%/75%	1	5	2	9m45s
catalog-service	Deployment/catalog-service	57%/75%	1	5	1	9m45s
checkout-service	Deployment/checkout-service	73%/75%	1	5	1	9m45s
shipping-service	Deployment/shipping-service	69%/75%	1	5	1	9m45s

Abbildung: Auslastung während Lastspitze

Systemarchitektur - Helm

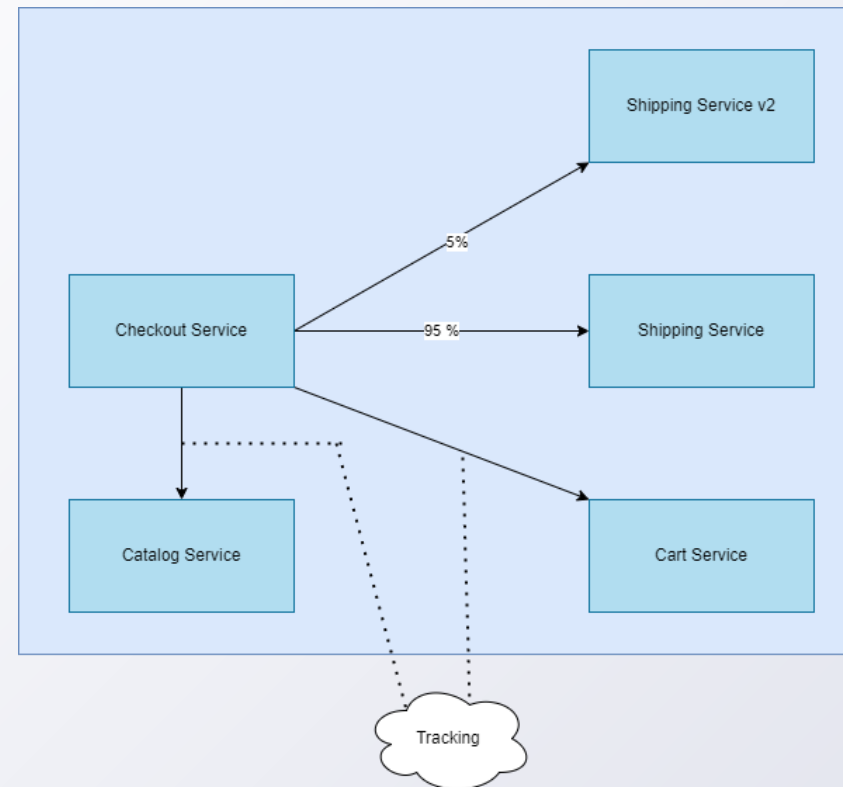
- ▶ Helm ist ein Packet-Manager für Kubernetes
- ▶ Zentrale Speicherung aller Kubernetes-Definitionen für eine Anwendung
- ▶ Trennung von API-Objekten und deren Werten (Template)
 - Template-Scripting möglich
- ▶ Versionierung des Cluster-Layouts

Service Mesh

Service Mesh

- ▶ Regulär unregelmäßiger Netzwerverkehr
- ▶ Service Mesh erzeugt neue Netzwerkschicht
- ▶ Kontrolle, Steuerung und Überwachung des Netzwerkverkehrs

Service Mesh



“ Programmierung eines
Onlineshops mit
Microservice Architektur.

Ende. Fragen?