

Verteilte Systeme Labor Aufgabe 2

Adrian Häuser 71141 haad1014@hs-karlsruhe.de,
Tom Ganz 71127 tomganzka@gmail.com

31. Oktober 2019

Inhaltsverzeichnis

1	Die Architektur	2
2	Swagger-Schemata	4

Kapitel 1

Die Architektur

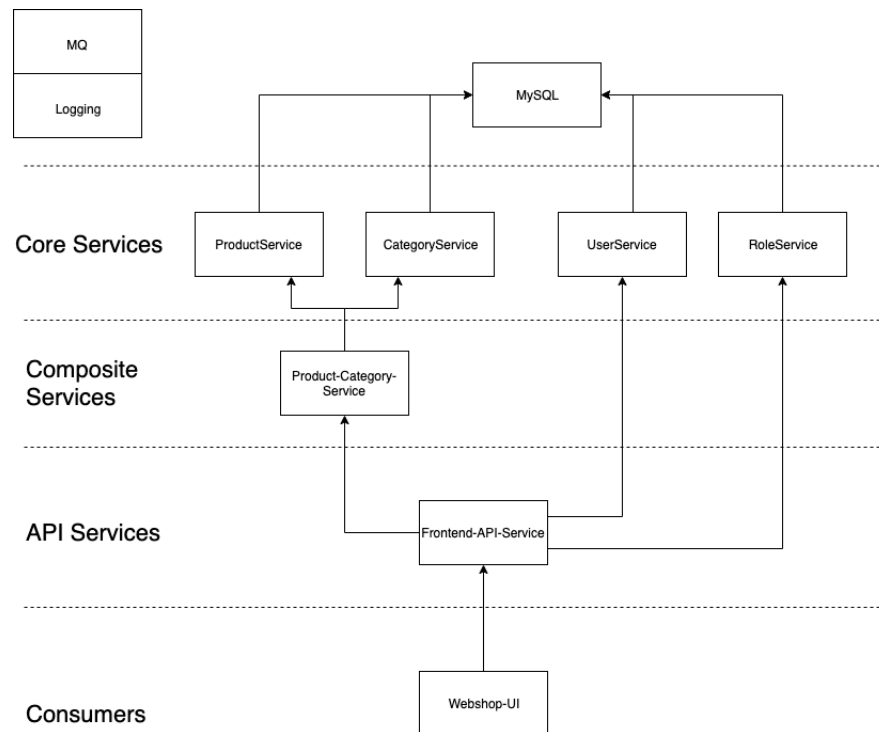


Abbildung 1.1: Die verteilte Microservice Architektur

Wie in Abbildung 1.1 zu erkennen, haben wir die Anwendung in vier Co-reservices unterteilt, eine für jedes DAO der Legacy-Anwendung. Diese führen CRUD Operationen auf einer gemeinsamen Datenbank-Server aus, auf dem sich für jeden Coreservice eine Datenbank befindet. Ein Composite Service existiert,

welcher Eigenschaften wie referenzielle Integrität erzwingt für ProductService und CategoryService. Der API-Service stellt wiederum ein einziges Interface für die Webshop-UI zur Verfügung.

Falls am Ende Kapazitäten von unserer Seite zur Verfügung stehen, wird ein globaler Logging-Service implementiert, welcher über eine Message Queue z.B. Apache Kafka Nachrichten entgegennimmt und persistiert.

Kapitel 2

Swagger-Schemata

user Operations available on user entity			▼
GET	/user	retrieves user	↶
POST	/user	creates an user	↶
DELETE	/user	deletes an user	↶
category Operations available on category entity			▼
GET	/category	retrieves all categories	↶
POST	/category	adds an category	↶
DELETE	/category	deletes a category	↶
product Operations available on product entity			▼
GET	/product/search	searches products	↶
PUT	/product	updates an product	↶
POST	/product	adds an product	↶
DELETE	/product	deletes a product	↶
GET	/product	retrieves product	↶

Abbildung 2.1: Die Endpunkte definiert durch Swagger

In Abbildung 2.1 sind die von uns spezifizierten Endpunkte und ihre Beschreibungen zu sehen. In Abbildung 2.2 sind die Modelle, die für die Eingabe

```

Product ▾ {
  id*           integer
                example: 0
  name          string
                example: Cat
  price         number
                example: 50
  details       string
                example: fluffy pet
  category      string
                example: Pet
}

```

```

Category ▾ {
  name*        string
                example: Pets
  id*          integer
                example: 0
}

```

```

User ▾ {
  id*           integer
                example: 0
  firstname*    string
                example: tom
  lastname*     string
                example: bismarck
  password*     string
                example: hidden
  role*         integer
                example: 1
}

```

```

Role ▾ {
  id*           integer
                example: 0
  type          string
  level         integer
}

```

```

LoginRequest ▾ {
  username*     string
                example: 0
  password*     string
                example: 0
}

```

Abbildung 2.2: Die Models definiert durch Swagger

und Ausgabe gebraucht werden definiert.