Spring Cloud Contracts

Übung zu: **Grundlagen der API Kompatibilität und wie man diese nachhaltig beherrscht**

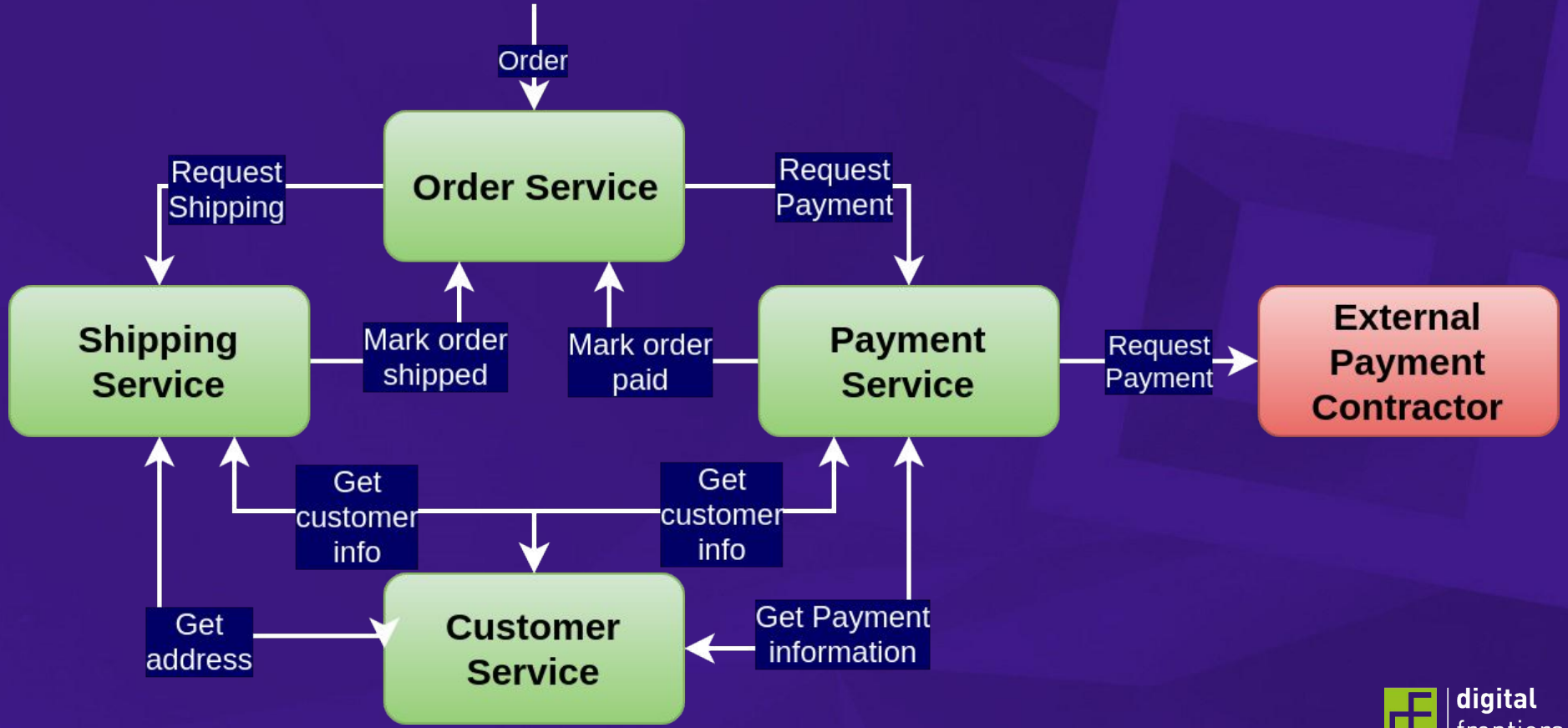
Szenario

- Shoplösung wurde in Microservices aufgeteilt
- Jeder Microservices wird von einem eigenen Team weiterentwickelt

Probleme:

- Aktuell benötigt jede Schnittstellenänderung viel Abstimmung
  - Welche Teams nutzen die API
  - Welche Features der API werden genau genutzt
- Kein Prozess für neue Schnittstellen
  - Beide Seiten entwickeln basierend auf ihrem Verständnis
  - Viele Änderungen auf beiden Seiten vor Livegang nötig
- Externe API wurden initial getestet
  - Unsicherheit ob externe Dienstleister ihre Schnittstellen anpassen

System Übersicht



Contracts für die APIs

1. Contracts für die Schnittstellen erstellen
2. Providertests gegen die Contracts
3. Consumertests gegen die Contracts

API Calls: Request Shipping

PUT /shipping/{orderId}/?articles=...&customer=...

Response:

200 OK

Body:

```
{
    orderId: long,
    articles: [ ],
    customer: string
}
```

Consumer?

→ Order Service

Provider?

→ Shipping Service

API Calls: Mark order paid

POST /order/{orderId}/shipped

Response:

200 OK

404 NOT FOUND: ungültige Order

Consumer?

→ Shipping Service

Provider?

→ Order Service

API Calls: Request Payment

PUT /payment/{orderId}/?articles=...&customer=...

Response:

200 OK

Body:

```
{
    orderId: long,
    articles: [ ],
    customer: string
}
```

Consumer?

→ Order Service

Provider?

→ Payment Service

API Calls: Mark order paid

POST /order/{orderId}/paid

Response:

- 200 OK
- 404 NOT FOUND: ungültige Order

Consumer?

→ Payment Service

Provider?

→ Order Service

API Calls: Get Customer Information

GET /customer/{customer}

Response:

- 200 OK
- 404 NOT FOUND: ungültiger Kunde
- Body:

```
{
    name: string,
    street: string,
    city: string,
    iban: string,
    bic: string
}
```

Consumer?

- → Payment Service
- → Shipping Service

Provider?

- → Customer Service

API Evolution

1. Shipping Service soll keine Details über Bezahlinformationen erhalten und benötigt eine eigene Schnittstelle für die Adresse
2. Payment Service benötigt die Adresse nicht und soll eine eigene Schnittstelle ohne Adresse erhalten
3. Implementieren neuer Contracts
4. Entfernen nicht mehr benötigter Contracts
5. Implementieren der neuen APIs
6. Rückbau nicht mehr benötigter APIs

New endpoint for address

GET /customer/{customer}/address

Response:

- 200 OK
- 404 NOT FOUND: ungültiger Kunde
- Body:

```
{
    name: string,
    street: string,
    city: string
}
```

Consumer?

→ Shipping Service

Provider?

→ Customer Service

New endpoint for payment information

GET /customer/{customer}/payment

Response:

- 200 OK
- 404 NOT FOUND: ungültiger Kunde
- Body:

```
{
    name: string,
    iban: string,
    bic: string
}
```

Consumer?

→ Payment Service

Provider?

→ Customer Service

3rd Party APIs

1. Aufsetzen eines Provider Proxys
2. Implementieren von Tests des Provider Proxys gegen den echten API Provider
3. Implementieren von Consumer Tests gegen die Stubs des Providers