

Thomas Bley

Vom Monolithen zur Microservice-Shop- Architektur

betterCode PHP
2021

Bringmeister.de
Dein Online Supermarkt

Über mich

- Senior PHP Developer
- Linux, PHP, MySQL seit 2001
- Studium an der TU München
- arbeite bei Bringmeister in Berlin



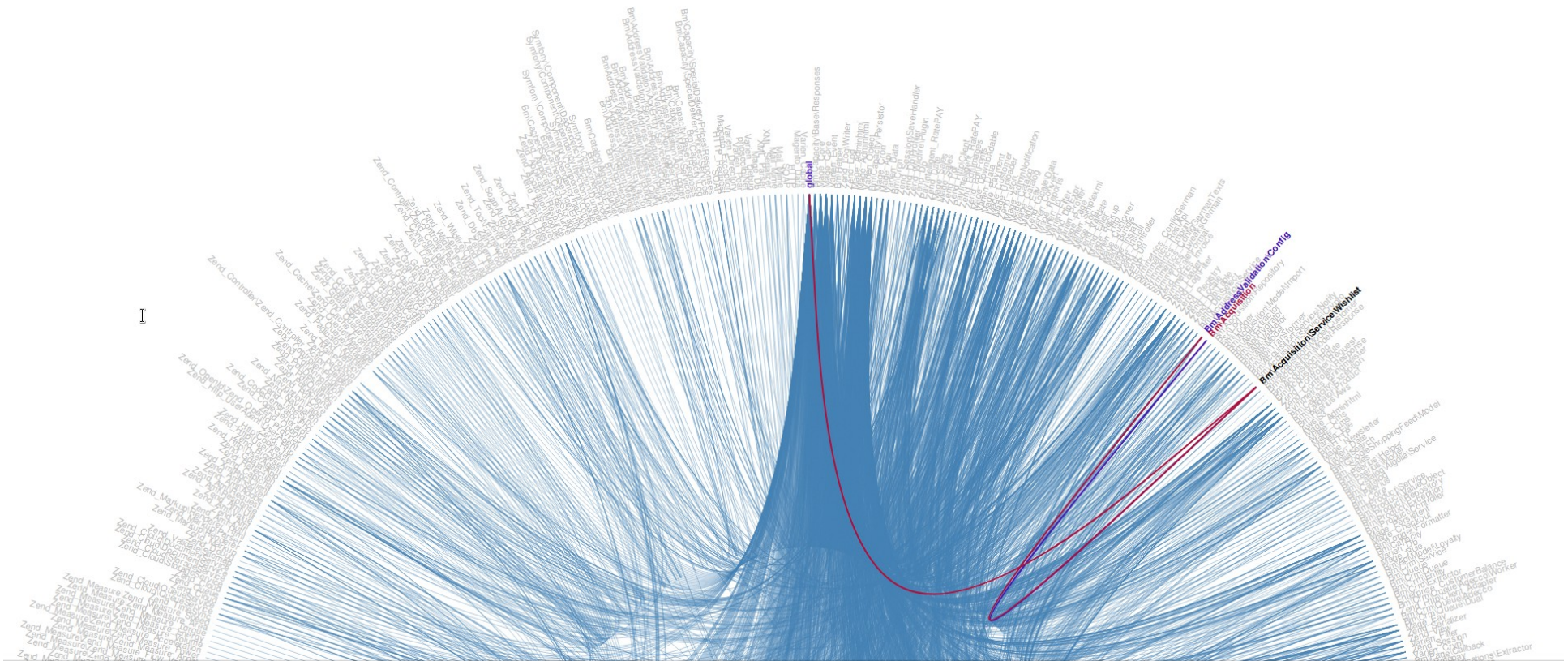
Der Monolith

- Magento 1 Enterprise: Supportende von Adobe angekündigt, Lizenzkosten
 - Große Code-Basis mit **1.8 mloc**, entwickelt über 10 Jahre mit 130 Entwicklern:
 - hohe Komplexität, starke Kopplung, mehrere API-Schichten
 - ungenutzter / unbekannter / schlechter Code, schwer zu pflegen, Bugs
 - Sicherheitsprobleme, nicht ausgelegt für DSGVO
 - viele Frameworks: Magento Varien, Zend Framework 1, Symfony 2, Laravel 5, ReactPHP, node.js, kein Composer
 - Probleme bzgl. Performance:
 - Produkt zum Warenkorb hinzufügen 400 Queries, **Bestellung platzieren 3200 Queries** (5 Transaktionen)
 - langsame Queries, Deadlocks, Probleme bzgl. Memory-Limit
 - Admin-Login >60s, Order-Suche im Admin 15s
 - Datenbank mit Entity-attribute-value-Modell entworfen (große Joins)
 - Aggregations-Tabellen synchron befüllt
- **schwierig das Business zu skalieren**

Der Monolith #2

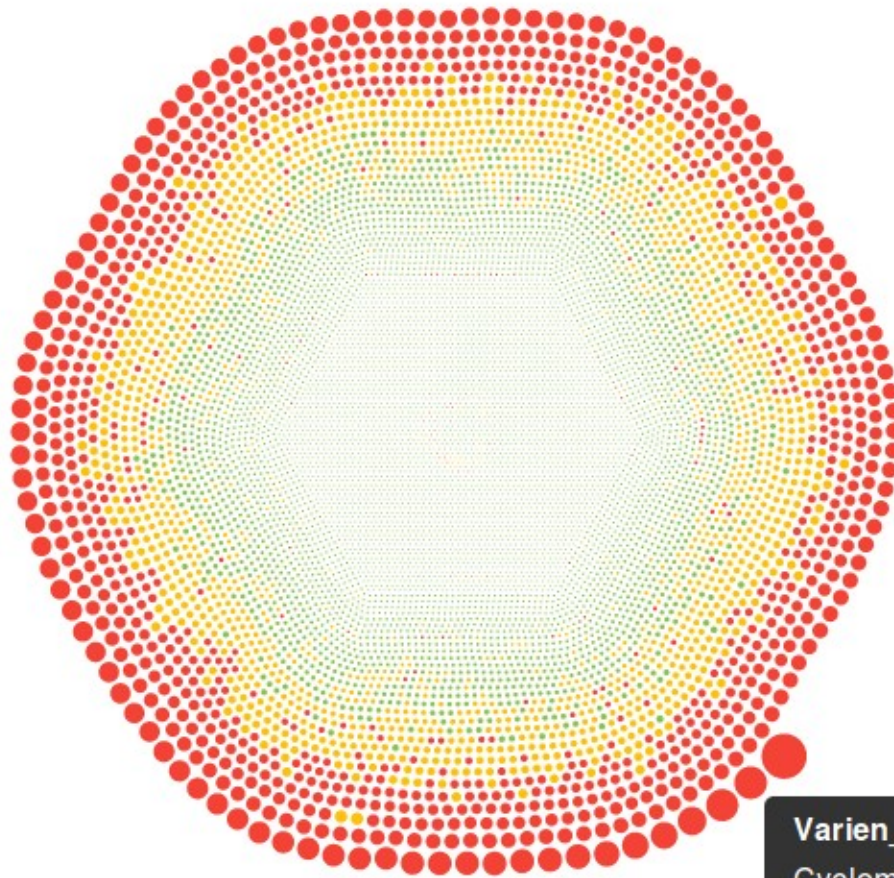
- Tests:
 - **Code-Abdeckung <10%**, Datenbank-Dump mit 7 GB nötig
 - viele Tests kaputt, werden nur lokal ausgeführt, CI nur für Deployment
 - viel manuelles Testing
 - Produktivität:
 - **langsame Entwicklung** neuer Funktionalitäten
 - Analyse von Problemen auf der Produktionsumgebung sehr komplex
 - Updates können nur manuell eingepflegt werden
 - Mehrere Quellen für Produktdaten, Daten häufig nicht synchron:
 - Datenbank
 - Solr, Algolia
 - Kleines Team: 5 Entwickler
- *Entwickler nicht glücklich, Management nicht glücklich, Kunden nicht zufrieden mit dem Online-Shop*

Wie sieht der Monolith aus?



Wie komplex ist der Monolith?

Maintainability / complexity



Each file is symbolized by a circle. Size of the circle represents the Cyclomatic complexity. Color of the circle represents the Maintainability Index.

Large red circles will be probably hard to maintain.

Varien_Db_Adapter_Pdo_Mysql
Cyclomatic Complexity : 411
Maintainability Index: 41.01

CI/CD, Betrieb

Was wir **haben**



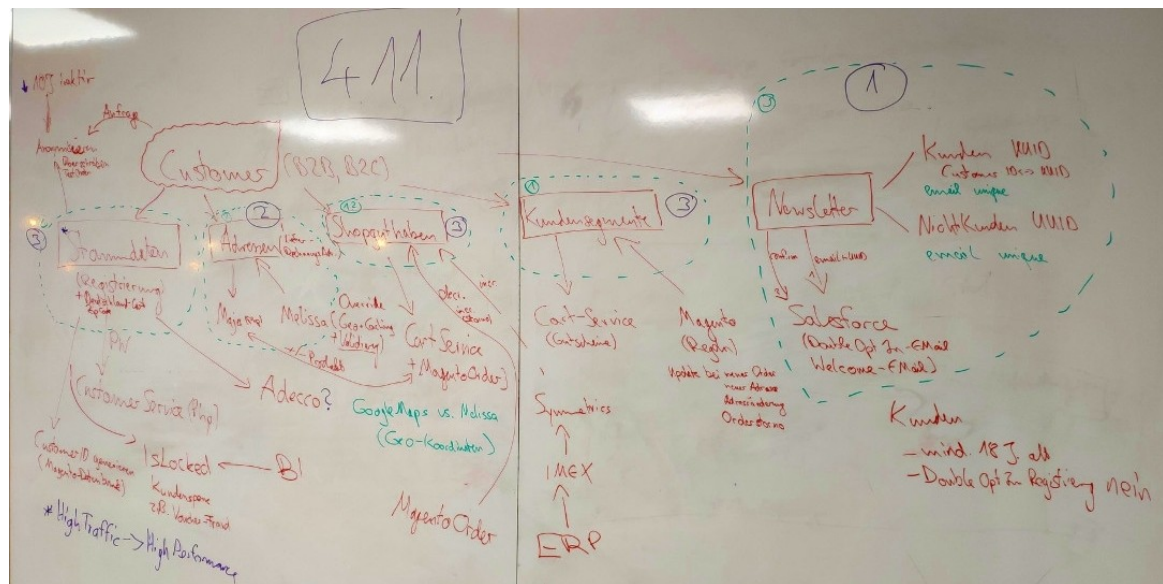
Was wir **brauchen**



Warum Microservices?

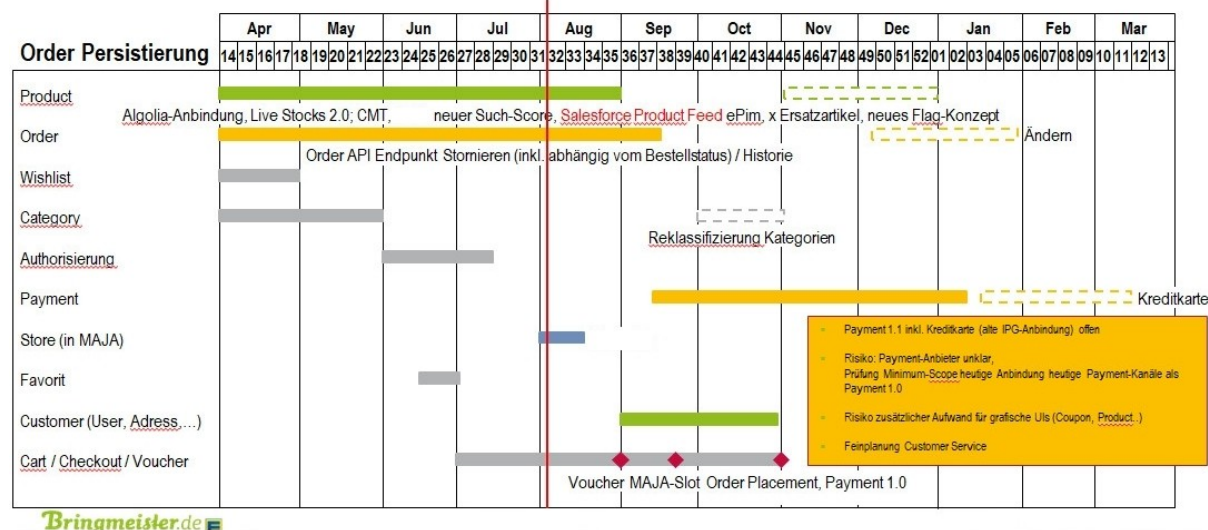
- Arbeiten mit einer kleineren Codebasis
 - einfacher zu entwickeln, einfacher zu ändern, einfacher zu warten
 - **weniger Komplexität**, Fokus auf Problemlösung
- Mehr Möglichkeiten bei der Auswahl von Datenbanken und Programmiersprachen
- Einfachere Aufteilung der Arbeit auf mehrere Teams
- **Harte Systemgrenze** mit standardisierter Kommunikation zwischen den Diensten
- Begrenzung der Auswirkungen durch Bugs und Fehler
- Isolierung der Daten
 - weniger Komplexität bei der Datenspeicherung
 - mehr Komplexität beim Zusammenführen von Daten (join)

das Projekt den Entwicklern beschreiben



das Projekt dem Management beschreiben

Meilensteinplan – Backend - BestCase



Die zukünftige Architektur definieren

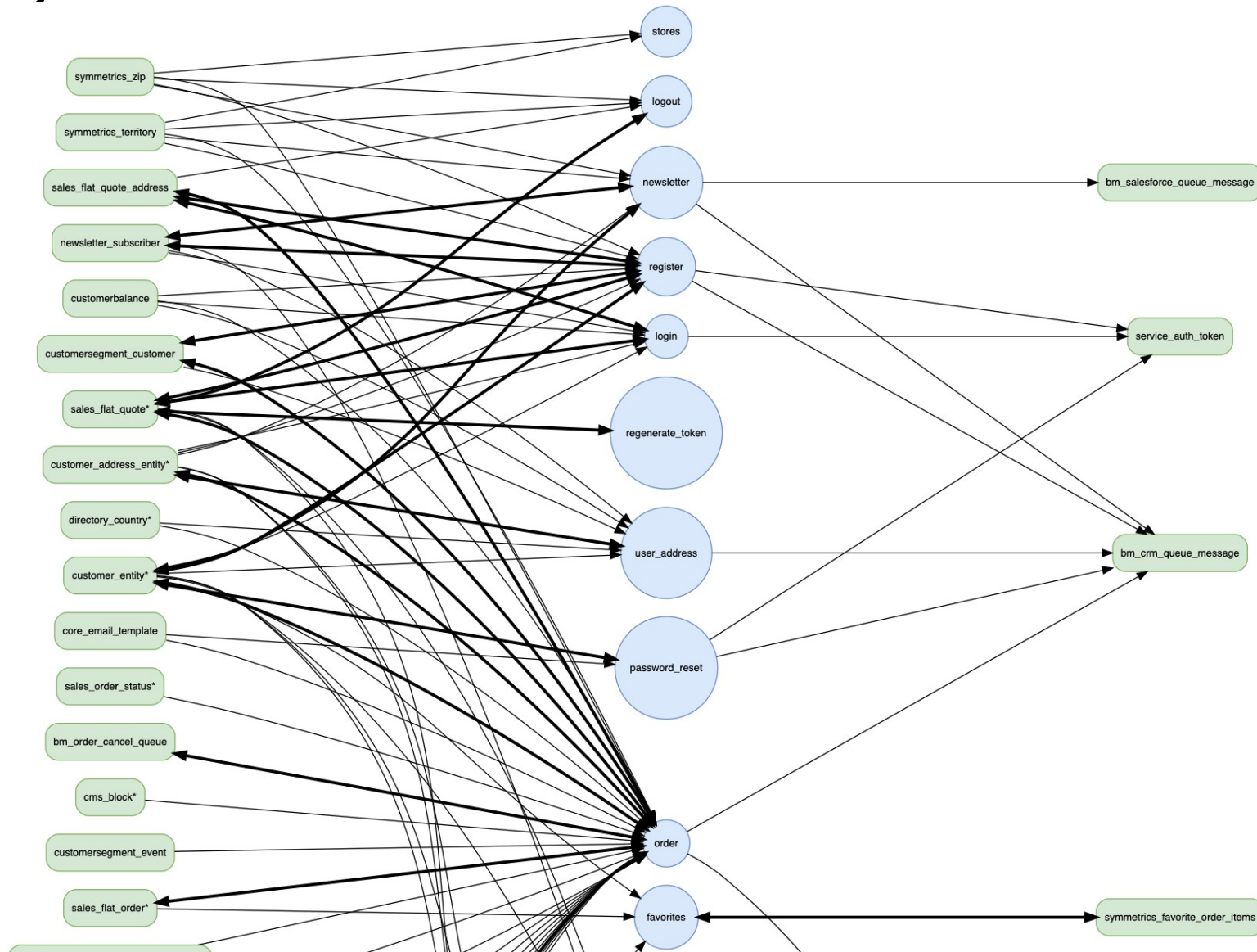
- Wir haben uns entschieden, bei PHP und MySQL zu bleiben
 - viel Erfahrung, alle notwendigen Bibliotheken und SDKs verfügbar, einfachere Portierung von PHP zu PHP
- Server-Infrastruktur behalten
- kein Fullstack-Framework
 - eigenes Mini-Framework (200 loc)
- Queries und Schema-Definitionen direkt in SQL, kein ORM, Joins durch JSON-Spalten reduzieren, keine Fremdschlüssel
- JWT statt Sessions
- 1 Mono-Repo für alle Dienste, jeder Dienst mit eigener Codebasis, eigener Datenbank, eigener composer.json, usw.
- Alle Produktdaten in Algolia
- SOLID, Kiss



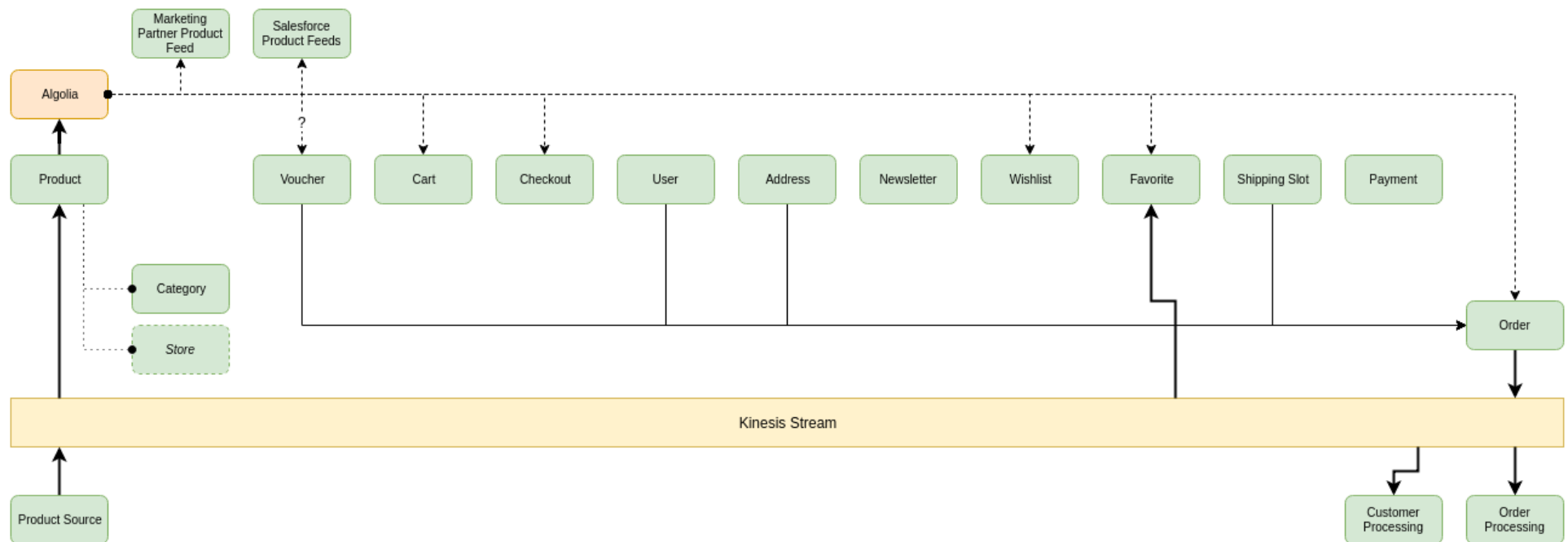
Quelle: https://www.youtube.com/watch?v=fCt2_AsCWKI

- 100% Testabdeckung, weiter Behat-Tests
- Statische Code-Analyse mit Psalm
- Enforce coding styles with PHP-CS-Fixer
- neue CI/CD-Plattform mit Bitbucket Pipelines

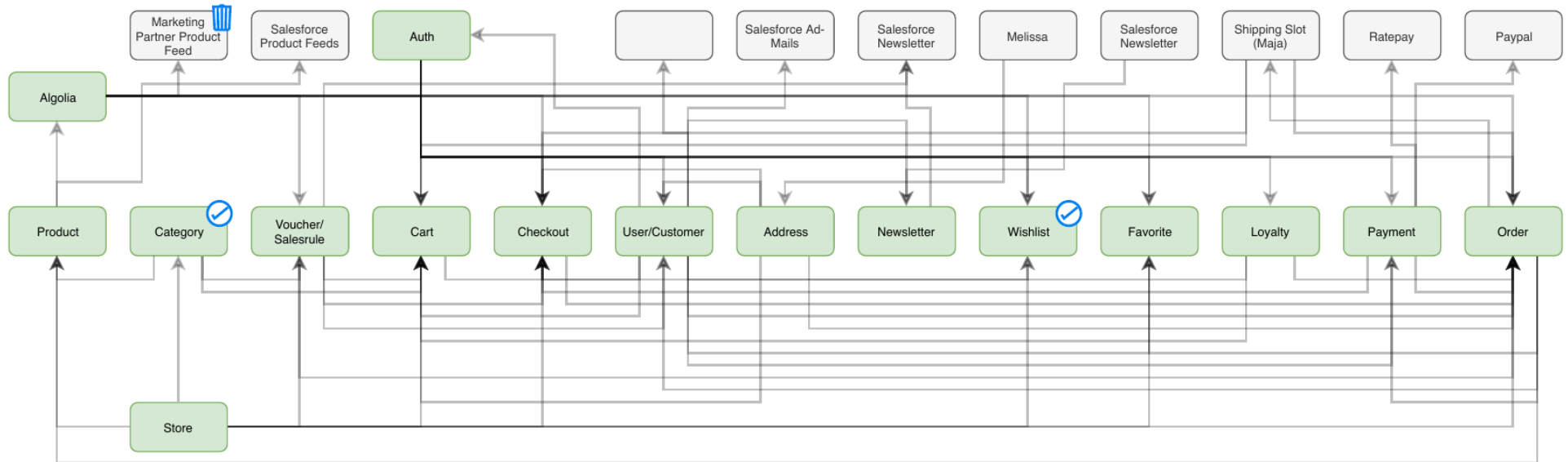
Abhängigkeiten auf Datenbankebene analysieren



Services identifizieren

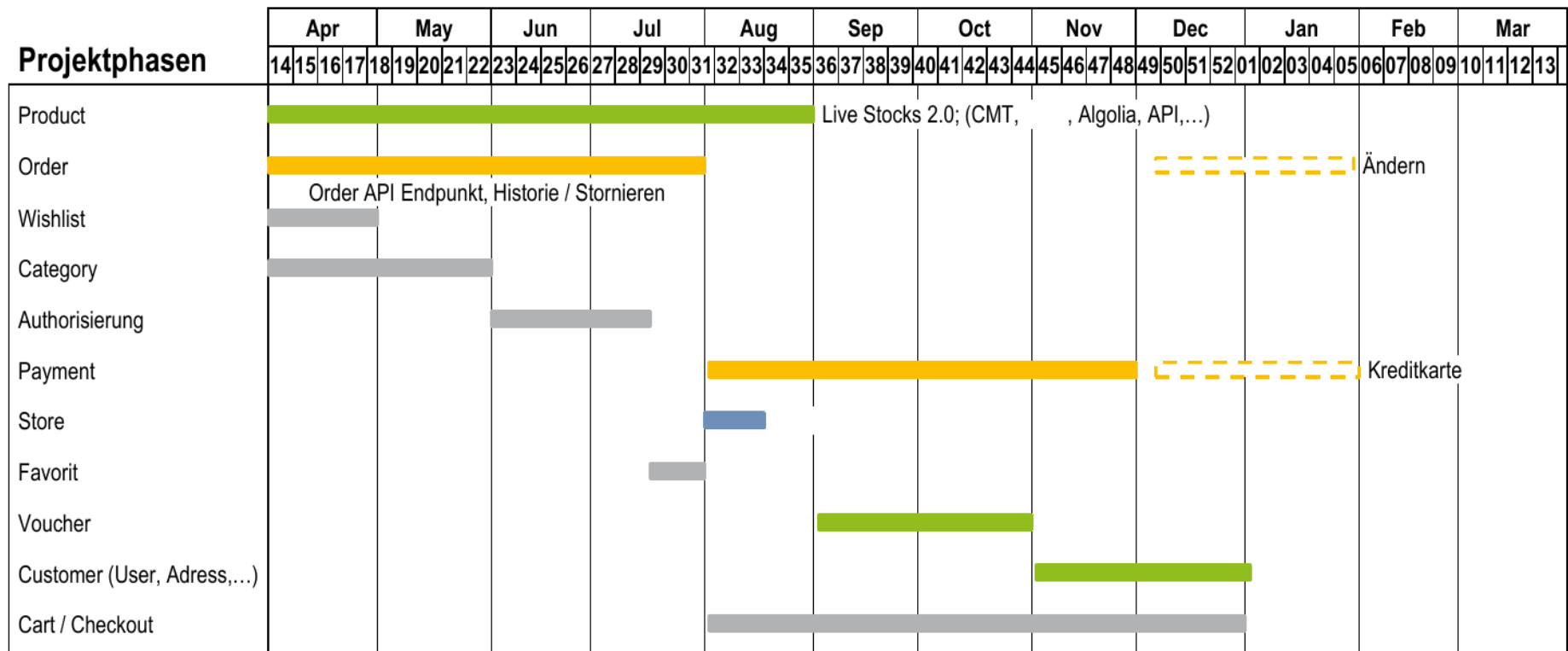


Mit einfachen Services beginnen



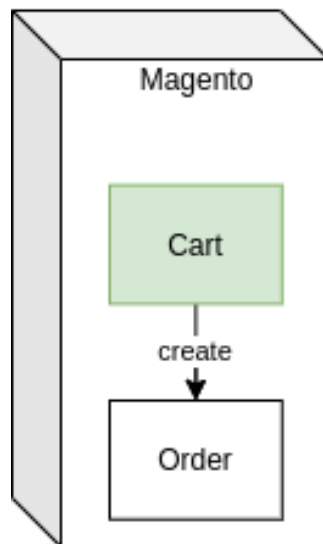
Services nacheinander implementieren und live nehmen

Meilensteinplan – Backend - BestCase

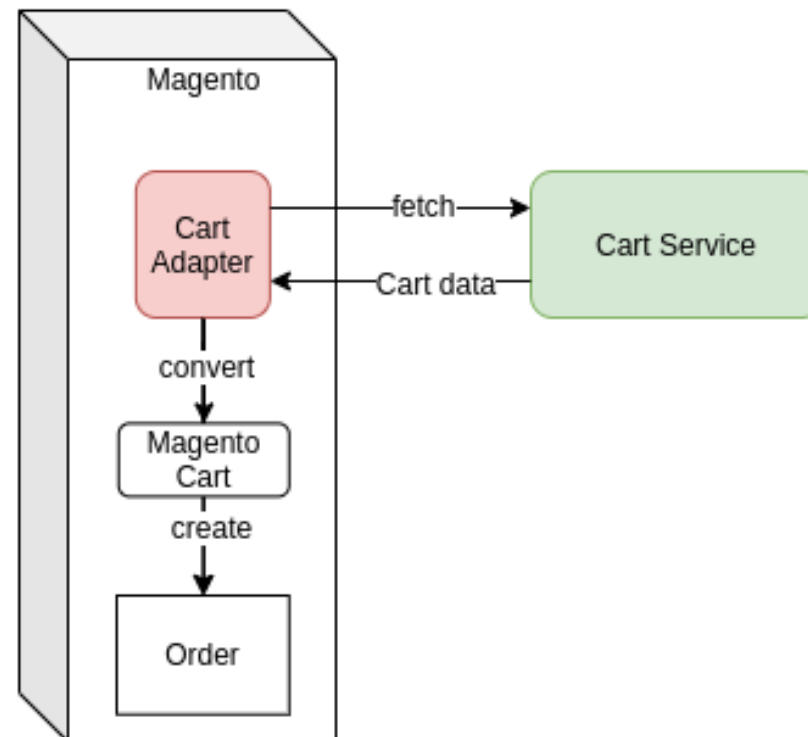


Komponenten mit starken Abhängigkeiten mit Adaptern ausbauen

Vorher:



Nachher:



Kommunikation zwischen Microservices und mit externen Dienstleistern

- **Synchron** mit REST, Load-Balancer (starke Konsistenz)
 - Kundenstammdaten, Adressen
 - Warenkorb, Gutscheine
 - Produktdaten
 - Zahlungsdienstleister
- **Synchron** mit SOAP (starke Konsistenz)
 - Altsysteme (Tourenplanung)
- **Asynchron** mit Ereignissen (Kinesis)
 - Bestellungen, Protokollierung
- **Asynchron** mit REST und Queues
 - externe Dienstleister (CRM, Kundensupport, etc.)
- JWT-Token des Kunden zwischen den Diensten weiterleiten

Datenmigration (EAV → JSON)

Mit SQL die Daten vom alten in das neue Schema kopieren:

```
INSERT INTO customers.customer (email, attributes, created)
SELECT * FROM magento.customer_entity
LEFT JOIN magento.customer_entity_varchar as firstname on
    firstname.entity_id = customer_entity.entity_id and firstname.attribute_id = 5
LEFT JOIN magento.customer_entity_varchar as lastname on
    lastname.entity_id = customer_entity.entity_id and lastname.attribute_id = 7
LEFT JOIN magento.customer_entity_datetime as birthdate on
    birthdate.entity_id = customer_entity.entity_id and birthdate.attribute_id = 11
SET email = customer_entity.email,
    attributes = json_object(
        'first', firstname.value,
        'last', lastname.value,
        'birth', date(birthdate.value),
        'orders', (select count(*) from magento.sales_flat_order
                    where customer_id = customer_entity.entity_id)
    ),
    created = customer_entity.created_at;
```


Daten-Anonymisierung

Eine eigene Datenbank mit Views für die Daten-Anonymisierung verwenden:

```
SELECT * FROM customers.customer WHERE id = 1234;
```

id: 1234

email: foo.bar@baz.com

attributes: { "first": "Thomas", "last": "Bley", "birth": "1930-02-01", "orders": 42 }

created: 2021-05-06 12:41:11

```
CREATE or REPLACE VIEW customers_anonymized.customer AS
```

```
SELECT id,
```

```
    concat('invalid_', id, '@bringmeister.de') AS email,           # invalid_1234@bringmeister.de
```

```
    json_object(
```

```
        'first', concat('first_', id), 'last', concat('last_', id), # first_1234, last_1234
```

```
        'birth', '1980-01-02', 'orders', attributes->>"$.orders"    # 1980-01-02, 42
```

```
    ) AS attributes,
```

```
    created
```

```
FROM customers.customer;
```

Ergebnisse

- Projekt **pünktlich** und **qualitativ** erfolgreich abgeschlossen (März - Nov. 2019)
- 10 Microservices, 3 Admin-Interfaces
- 99.99% Testabdeckung mit Unit- und Integration-Tests
- Codegröße reduziert auf **100 kloc** (vorher 1.8 mloc)
- Datengröße in der Datenbank um 80% reduziert
- System-Performance und Umsatz deutlich gesteigert
- Hardwarekosten um 50% reduziert
- Externen Sicherheitsaudit bestanden
- Tests, Build und Deployment in < 10 Minuten
- Entwicklung neuer Funktionalitäten und Wartung viel einfacher und schneller

→ *Entwickler glücklich, Management glücklich, Kunden zufrieden mit dem Online-Shop*

Microservices wie einen Monolithen implementieren?



Quelle: twitter.com/ddprtt/status/1425418538257428488

Was haben wir gelernt?

- Kleiner ist besser
- Probleme nicht ignorieren
- Anforderungen sind wichtig
- Dokumentation
- Komplett neu schreiben ist viel Arbeit, aber es lohnt sich
- Unterstützung vom Management ist wichtig
- Monitoring (insbes. externe Dienstleister, wir benutzen Datadog)
- Testing, testing, testing (PHPUnit, Psalm, Trivy, manuell, usw.)
- Performance, performance, performance (ein 10x-System bauen)

php 8.1

PDO MySQL: Ganz- und Gleitkommazahlen in den Resultsets werden jetzt mit nativen PHP-Typen zurückgegeben, wenn emulierte prepared Statements verwendet werden. Das vorherige Verhalten kann mit der Option **PDO::ATTR_STRINGIFY_FETCHES** wiederhergestellt werden.

```
$options = [  
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,  
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,  
+    PDO::ATTR_STRINGIFY_FETCHES => true,  
];
```

Quelle: github.com/php/php-src/blob/php-8.1.0RC6/UPGRADING

Danke fürs Zuhören!

Fragen?

Slides zum Download:

github.com/thomasbley/talks

folgt mir auf Twitter:

twitter.com/thbley

