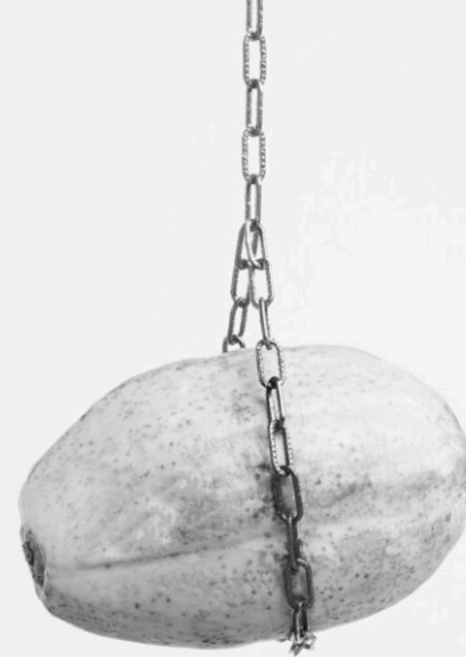
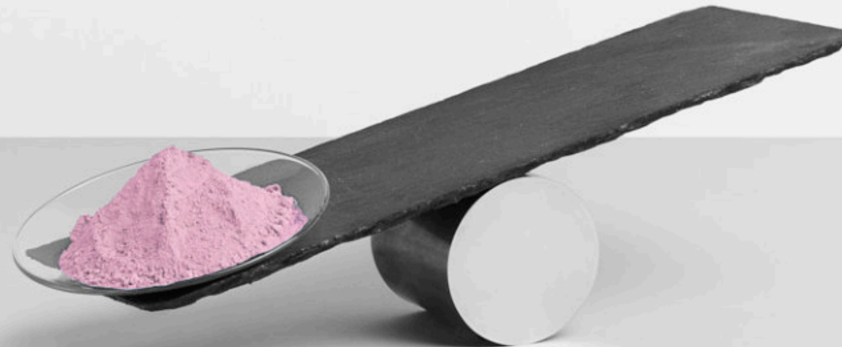
COFINPRO

Vom Distributed Monolith zu Self-Contained Systems: ein Erfahrungsbericht





https://speakerdeck.com/mscholtz/vom-distributed-monolith-zu-self-contained-systems-ein-erfahrungsbericht

COFINPRO

Wer sind wir

Marcos Scholtz

Marcos.Scholtz@cofinpro.de

Gregor Tudan

Gregor.Tudan@cofinpro.de

COFINPRO

Unser Projekt

- Robo-Advisor für die vollständig digitale Geldanlage
- White-Label Plattform der genossenschaftlichen Finanzgruppe
- mehrere hundert Banken angeschlossen
- 5 verschiedene Produkte

COFINPRO

Was wollen wir erzählen?

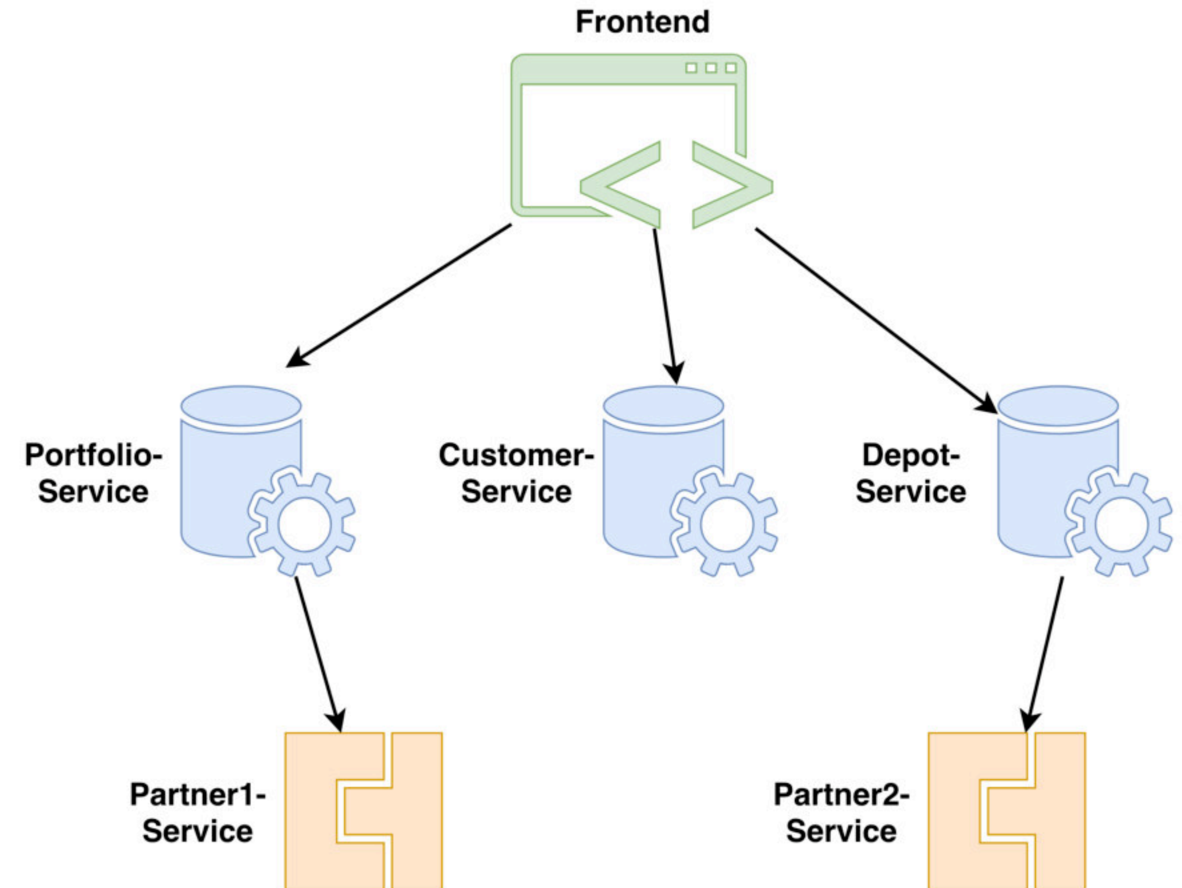
- Vorher – unser Distributed Monolith
- Unser Ziel – Self-Contained-Systems
- Unser Weg – wie sind wir vorgegangen
- Was ist noch zu tun
- Was haben wir gelernt

COFINPRO

Wie war es vorher:
unser „Distributed Monolith“

Am Anfang war alles schön…

- neue Web-Anwendung
- Microservices in zwei Schichten
  - Business-Logik
  - Integration
- 1 AngularJS Frontend
- 1 Kunde, ein Produkt
- 1 Scrum-Team



Die Welt war in Ordnung

COFINPRO

Der Preis des Erfolgs

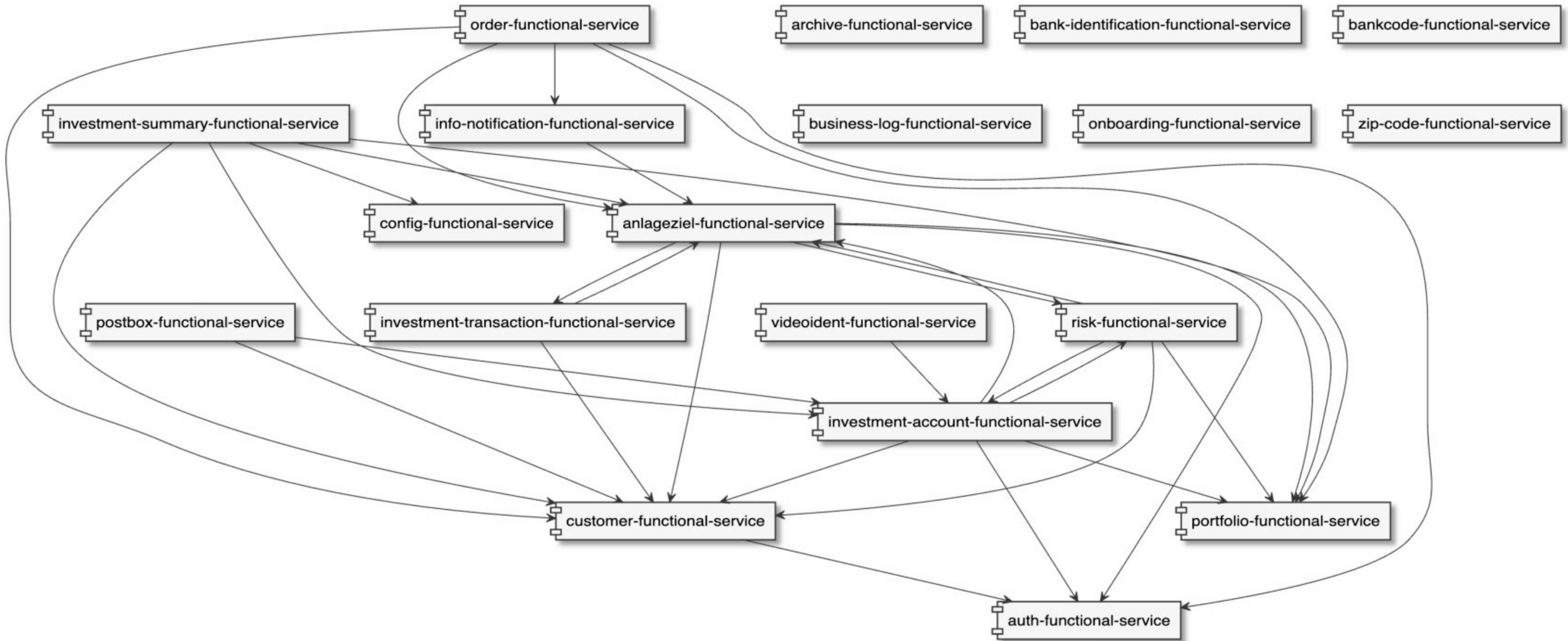
- Viele neue Features
- White-Labeling für über 300 Banken
- 5 verschiedene Produkte
- neue Vertriebskanäle
- Mobile-App
- Berater-Portal

- Skalierung auf 5 Scrum-Teams

Komplexität um ein Vielfaches höher

COFINPRO

Service-Abhängigkeiten



COFINPRO

Die Organisation

Integration-Team

Teams

Reaktion

- Einführung eines „Integration-Teams“
- hat die Architektur-Verantwortung
- keine festen Zuständigkeiten der Teams

Ergebnis

- Integration-Team überlastet
- Jeder muss alles kennen
- Teams demotiviert

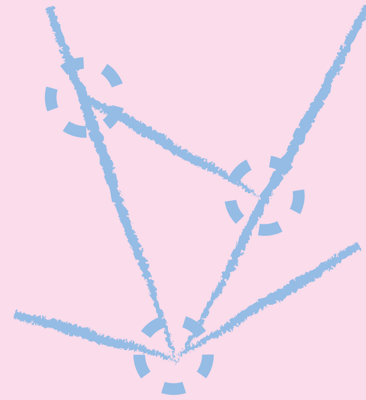
Die Herausforderung

- Anwendung groß und komplex
- neue Mitarbeiter brauchen lang für die Einarbeitung
- Anpassungen nur schwer möglich
- Fehler wirken sich auf andere Anwendungsteile aus
- Viel Aufwand für manuelle Tests
- Releases dauern zu lang

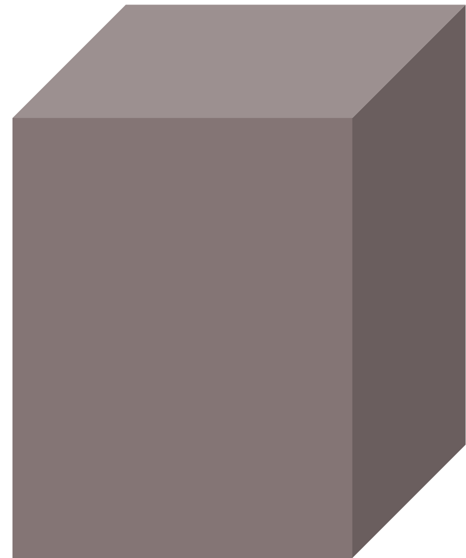
Wir haben einen Distributed-Monolith!

COFINPRO

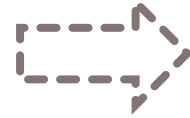
Alles ist schlimm! Und jetzt?

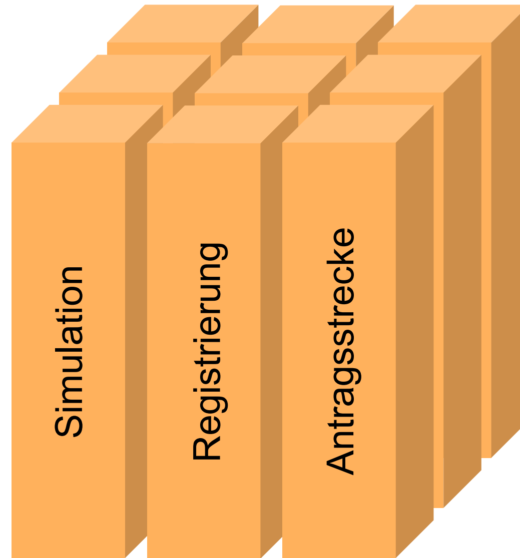


Self-Contained Systems

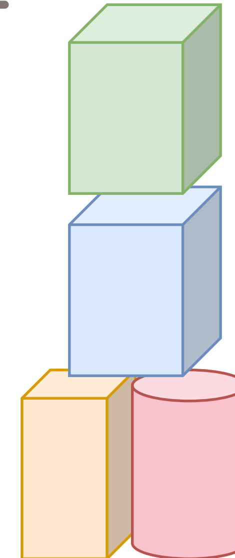


Distributed Monolith





Self-Contained Systems



Frontend

Business-Layer

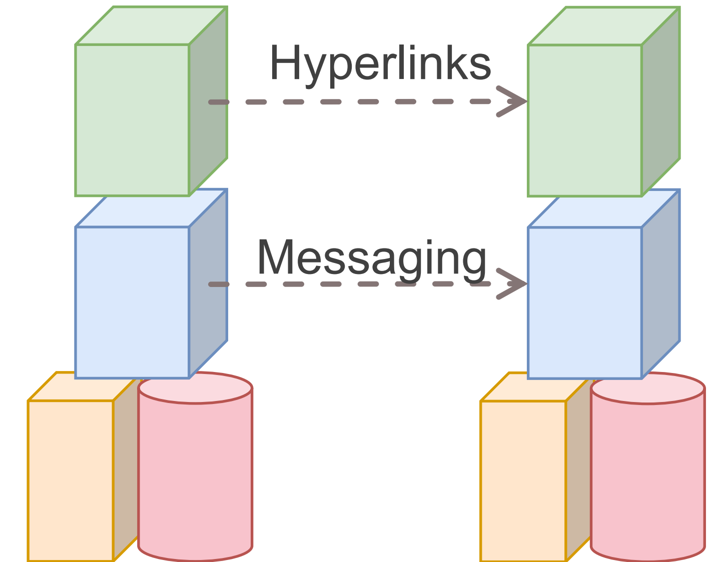
Integration-Layer

SCS oder „Vertical“

COFINPRO

Ein Self-Contained System

- ist eine **unabhängige** Web-Anwendung
- gehört zu **einem** Team
- kommuniziert meistens **asynchron**
- beinhaltet Frontend, Logik, und **Daten**
- benutzt keinen (kaum) geteilter Code oder Komponenten
- kann **allein** released/deployed werden



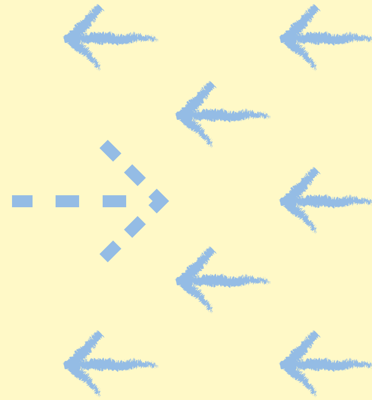
https://scs-architecture.org

Self-Contained Systems

Was wollen wir erreichen

- weniger komplexe Anwendungen
- unabhängiger arbeiten – weniger Overhead
- mehr Verantwortung in den Teams

Klingt toll! Wie kommen wir dahin?

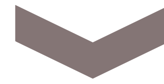


COFINPRO

Anwendungsschnitt

Schnitt sollte fachlich sein

- sonst keine Unabhängigkeit / Entkopplung
- nach „Bounded Context“ (DDD)



Nach Produkt / Bereich / Kundentyp

- geeignet für Produkte / Bereiche mit wenig Gemeinsamkeiten

Nach fachlichen Prozess-Schritten

- geeignet für Prozess-Schritte mit wenig Gemeinsamkeiten

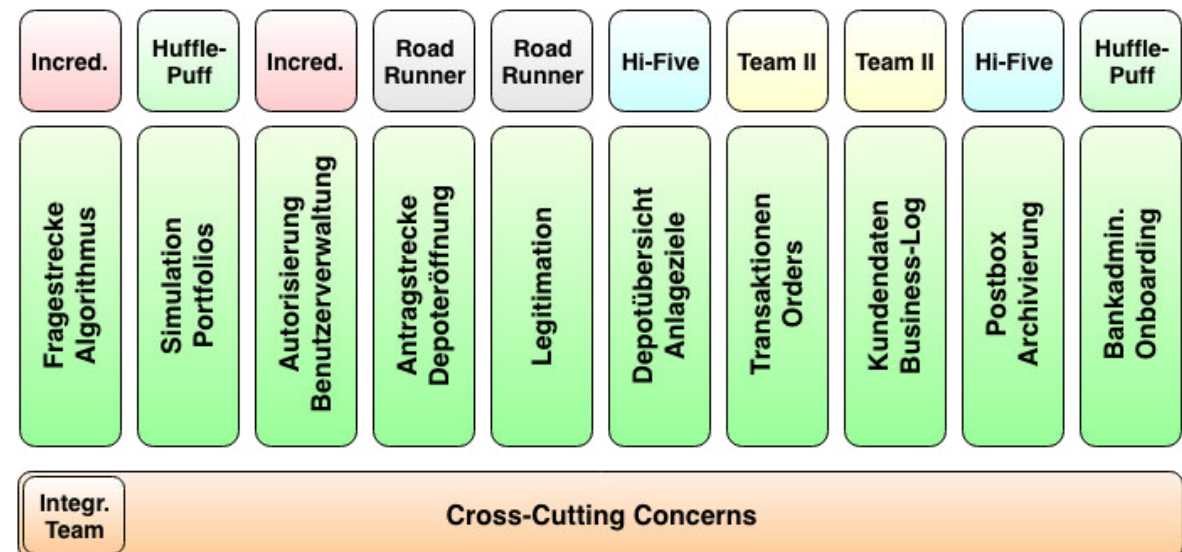
Zerteilung in 10 Verticals nach fachlichen Prozess-Schritten

Aufteilung der Verticals

COFINPRO

Verticals auf Teams verteilen

- Verticals auf Teams verteilt
  - Teams haben mitentschieden
  - vorhandenes Know-How als wichtigster Faktor
- manche Verticals größer als andere: Balance finden
- Cross-Cutting Concerns: beim Integration-Team geblieben



Erfahrung

- Immer wieder Feinjustierung nötig
- aber weniger als gedacht

COFINPRO

Bestandsaufnahme mit den Teams

Ein Workshop pro Team

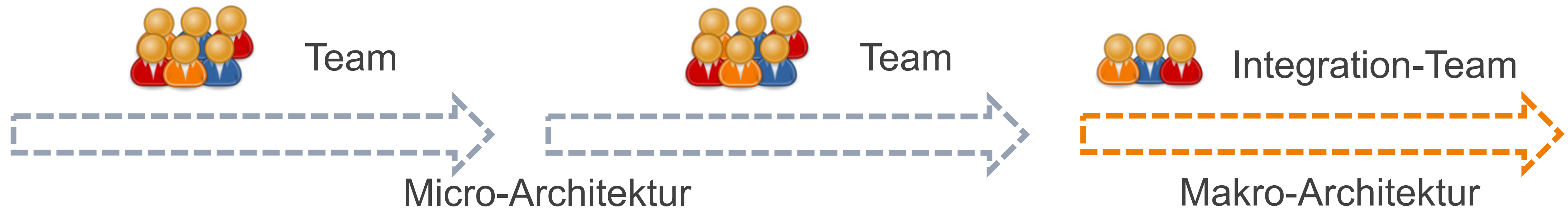
- ganzes Team + Integration-Team
- Analyse und Abgrenzung des Verticals
- Ermittlung der Abhängigkeiten

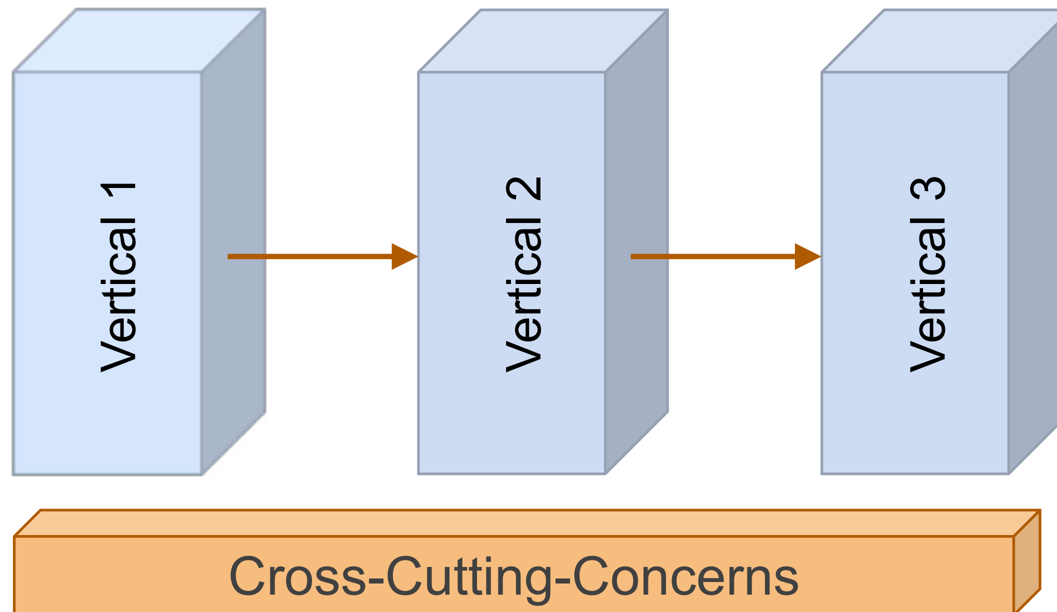
Die Erfahrung

- sehr oberflächliches Know-How in den Teams
- viele Abhängigkeiten
- fachliche Abgrenzung schwierig: Teilnahme eines POs?

COFINPRO

Makro- und Micro-Architektur





Das Ziel:

- Verantwortung verteilen
- spezialisierte Teams
- Autonomie der Teams
- Teams durch Eigen-verantwortung motivieren

COFINPRO

Verteilung der Verantwortung

Makro-Architektur

Micro-Architektur

Int.-Team entscheidet

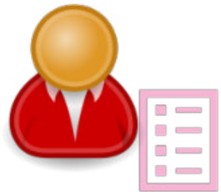
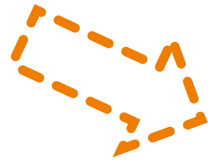
Int.-Team wird informiert und empfiehlt

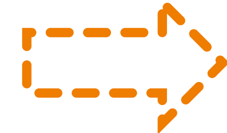
Team entscheidet selbst

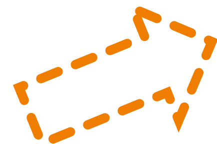
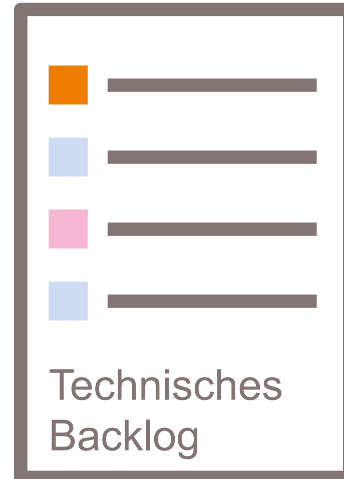
Ziel: weniger orange, mehr blau

COFINPRO

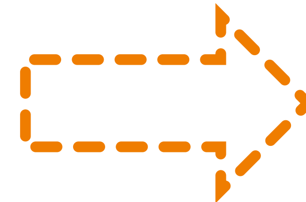
Technische Refactorings



Nexus
Refinement



Herausforderung: Konzeption

Umdenken bei der Konzeption von Stories:

- 1 User-Story ► 1 Vertical
- PO(s) mit „Produkt“-Ausrichtung
- vom Int.-Team „abhängig“

- Braucht Zeit…
- Inzwischen spezialisierte PO(s) nach Prozess-Schritt

COFINPRO

Herausforderung: Planung

Ungleichmäßige Team-Auslastung:

1. NICHT machen
   (balanciert planen)
2. Entwickler wechseln temporär das Team
3. Stories zwischen Teams schieben
   (Merge-Requests an das „Owner“-Team)

COFINPRO

Zähmen des Service-Schwarms: technische Änderungen

COFINPRO

Verticals entkoppeln

Abhängigkeiten

- viel Kommunikation zwischen Verticals
- manche Services mit Fachlichkeit aus zwei Verticals
- viel geteilter Code (insbesondere im Frontend)
- große Frontends – erstrecken sich über mehrere Verticals

COFINPRO

Ansätze zur Entkopplung

Abhängigkeiten entfernen

- Abspaltung von Services
- Code-Duplizierung
- Daten-Replizierung
- Zusammenfügen von Services

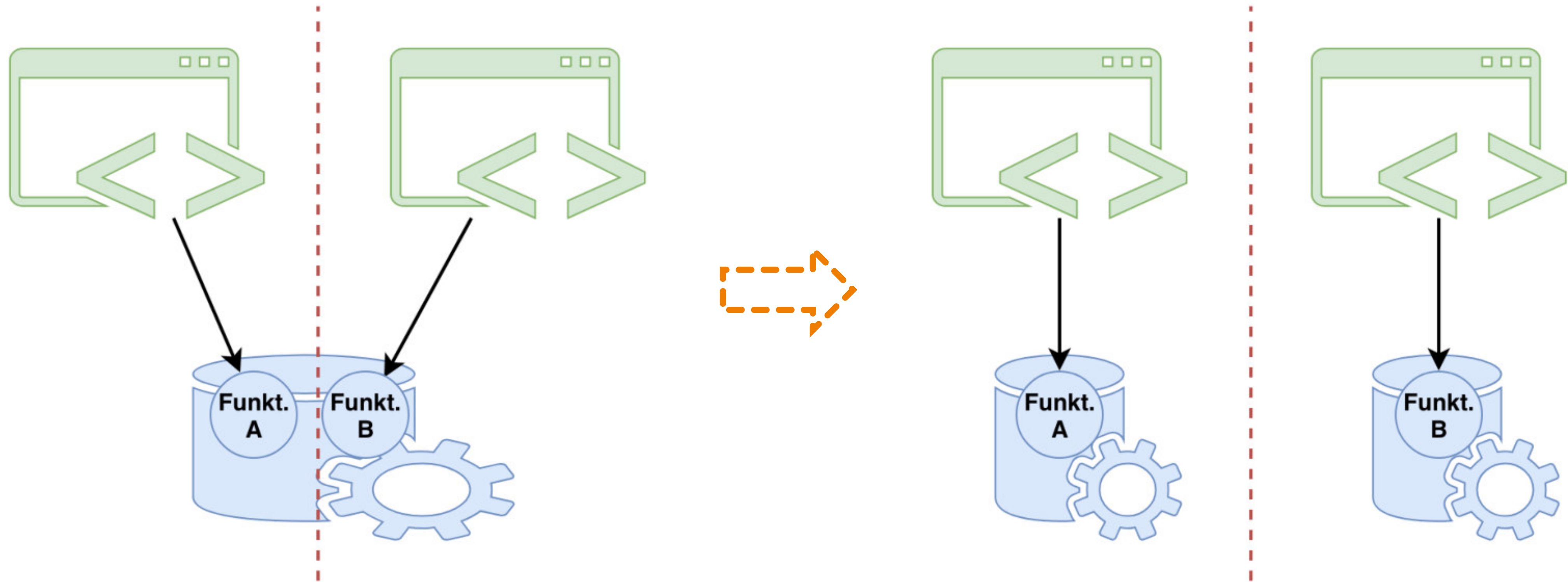
Abhängigkeiten pflegen

- Messaging
- spezialisierte APIs
- APIs rückwärtskompatibel
- Schnittstellen-Tests (CDCT)

COFINPRO

Abhängigkeiten entfernen

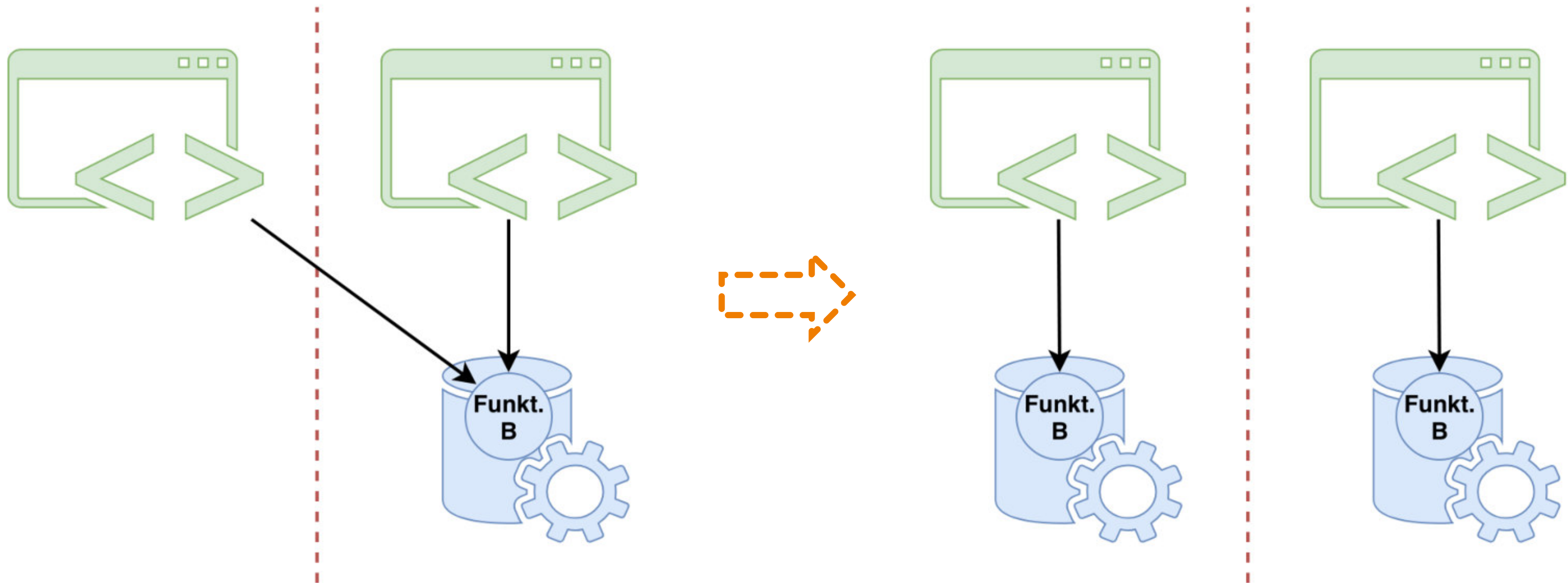
Abspaltung von Services



COFINPRO

Abhängigkeiten entfernen

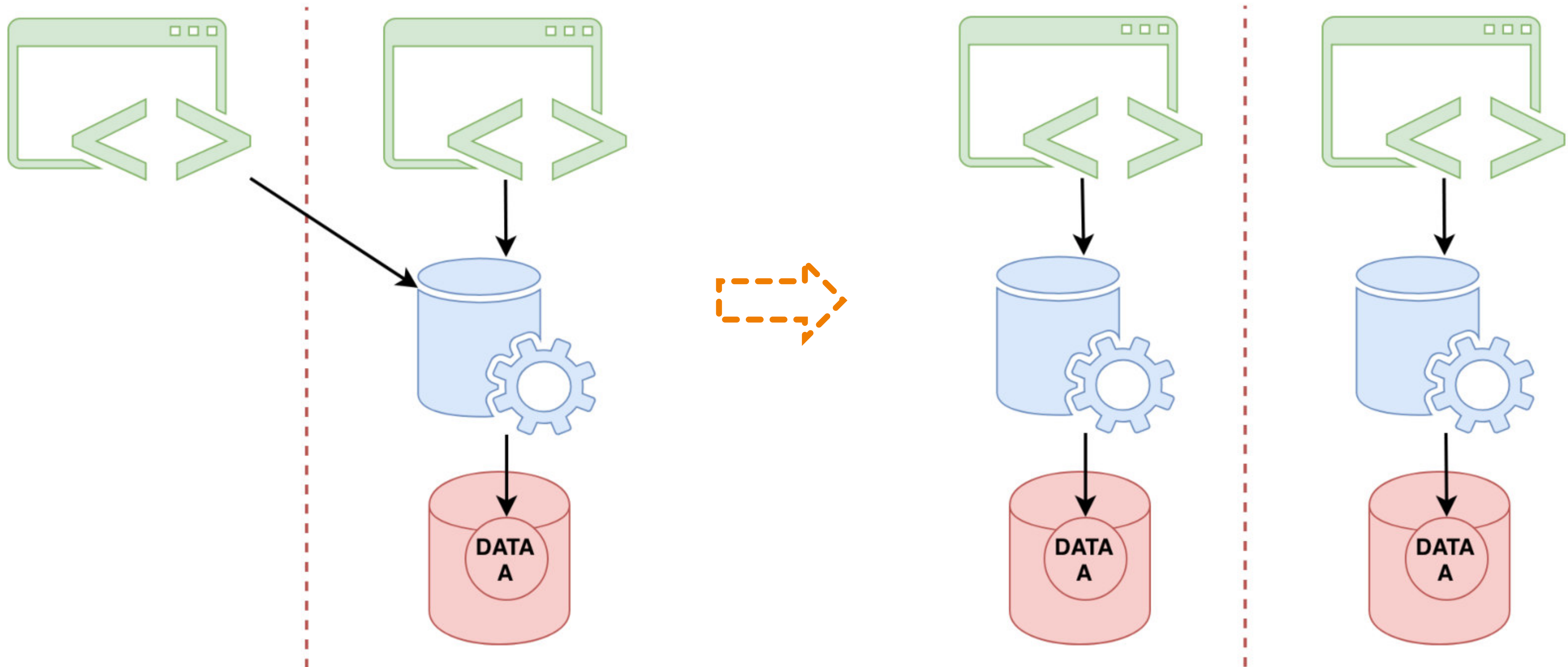
Code-Duplizierung



COFINPRO

Abhängigkeiten entfernen

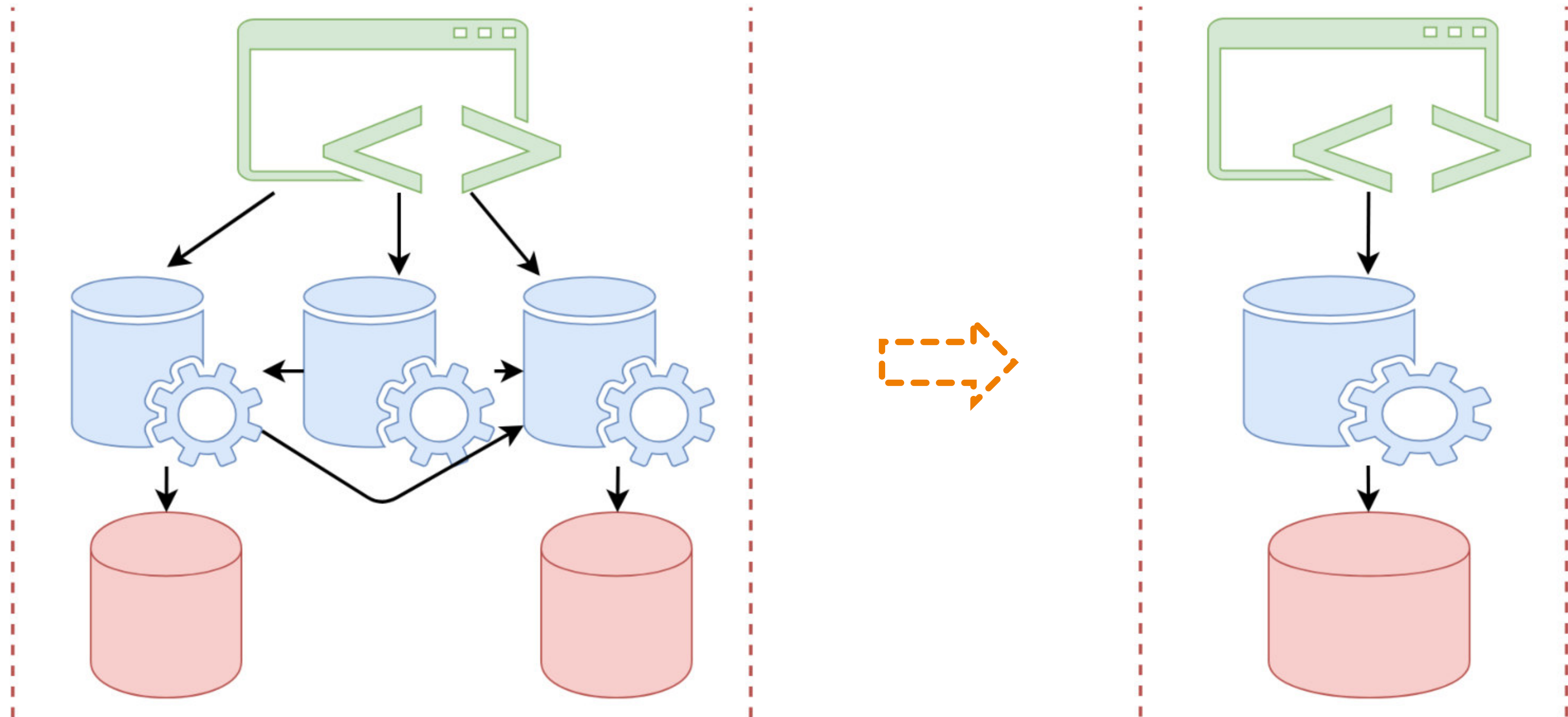
Daten-Replizierung



COFINPRO

Abhängigkeiten entfernen

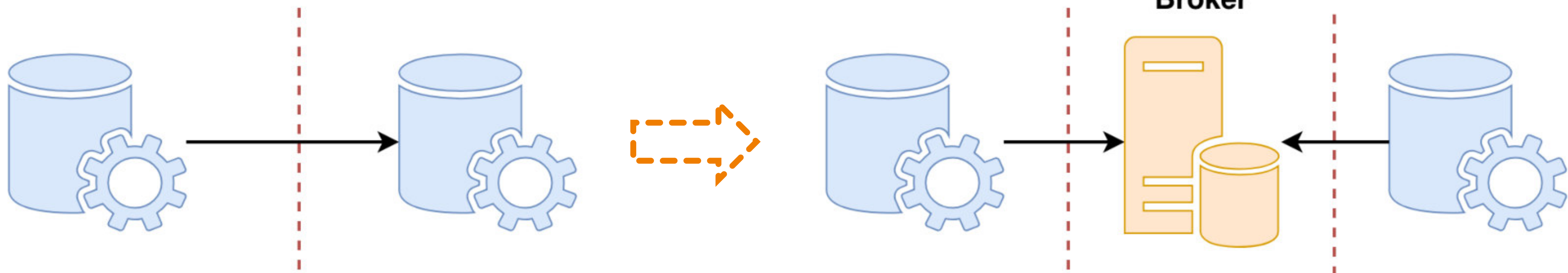
Zusammenfügen von Services



COFINPRO

Abhängigkeiten pflegen

Messaging



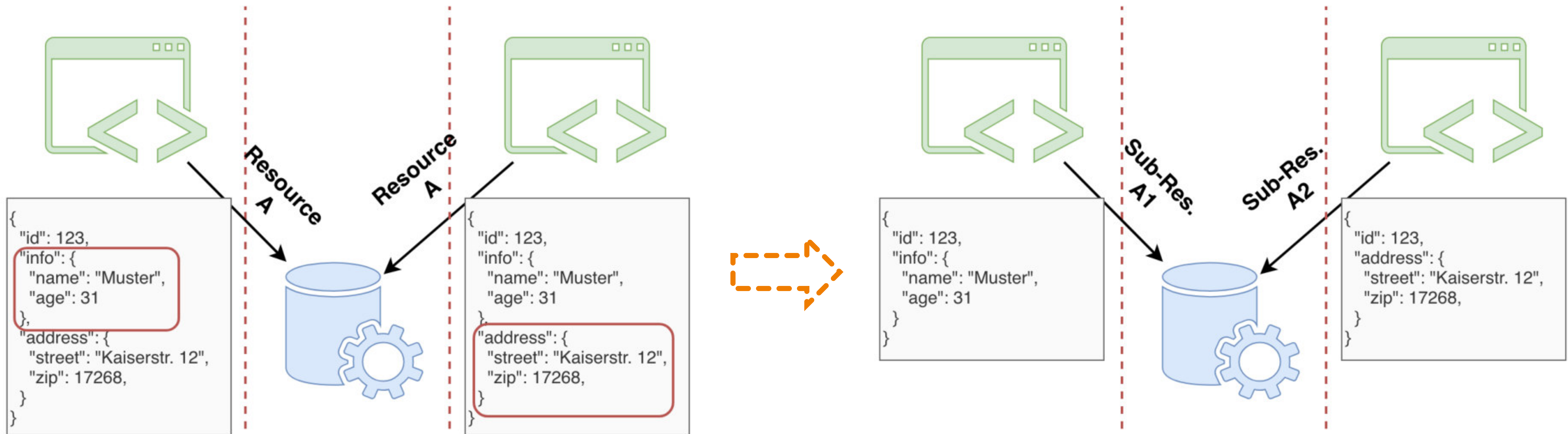
- Request-Response-Pattern
- einfaches Fehlerhandling
- Gegenseite muss verfügbar sein

- Publish-Subscribe Pattern
- eventually consistent
- Broker muss verfügbar sein

COFINPRO

Abhängigkeiten pflegen

Spezialisierte APIs



COFINPRO

Abhängigkeiten pflegen

APIs rückwärtskompatibel

- keine breaking changes
- alte Felder behalten (deprecated)
- API versionieren (vermeiden)

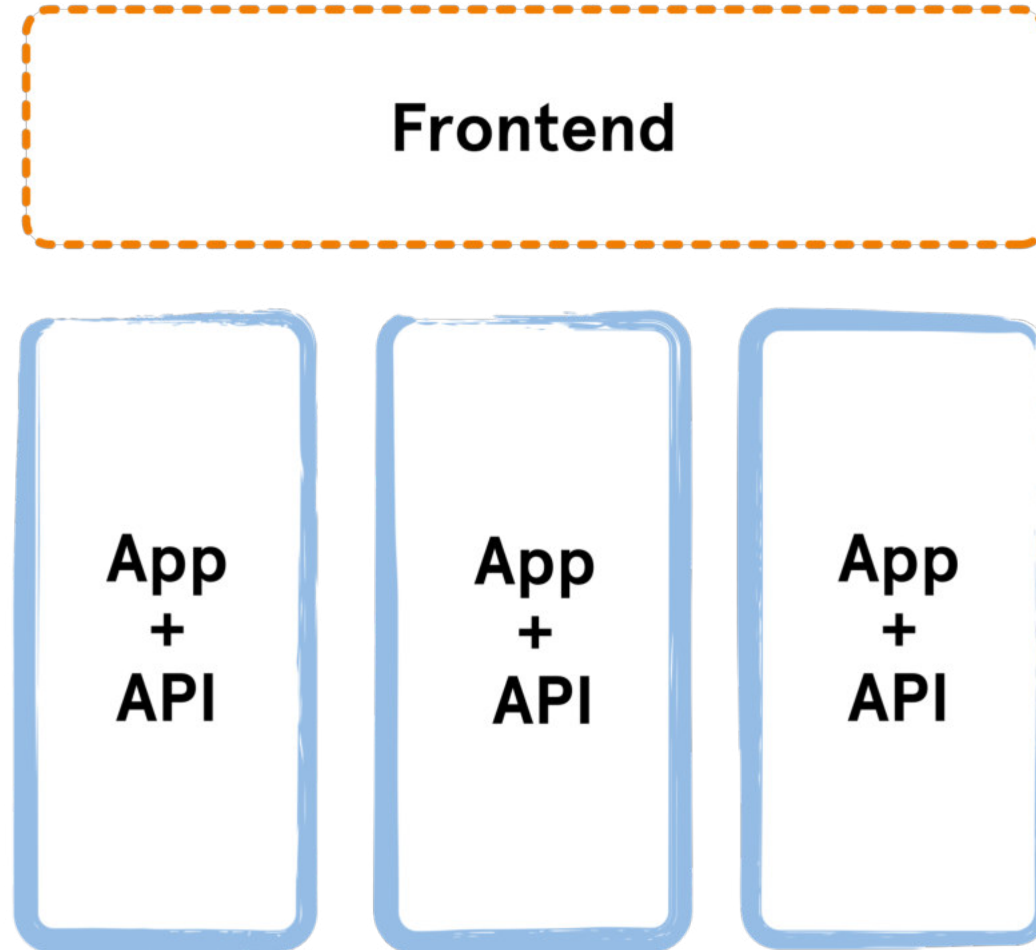
Schnittstellen-Tests

- Sicherstellen der API-Kompatibilität zwischen Versionen
- Consumer Driven Contract Tests (z.B. PACT)

Consumer	Consumer Version	Pact Published	Provider	Provider Version	Pact verified
zoo-app	0724701 master	about 12 hours ago (revision 1)	animal-service		
zoo-app	955b9ea master	2 days ago (revision 1)	animal-service	82b59ef master	1 day ago (number 2)
zoo-app	9632729 prod master	4 days ago (revision 1)	animal-service	a519731 prod master	3 days ago (number 1)

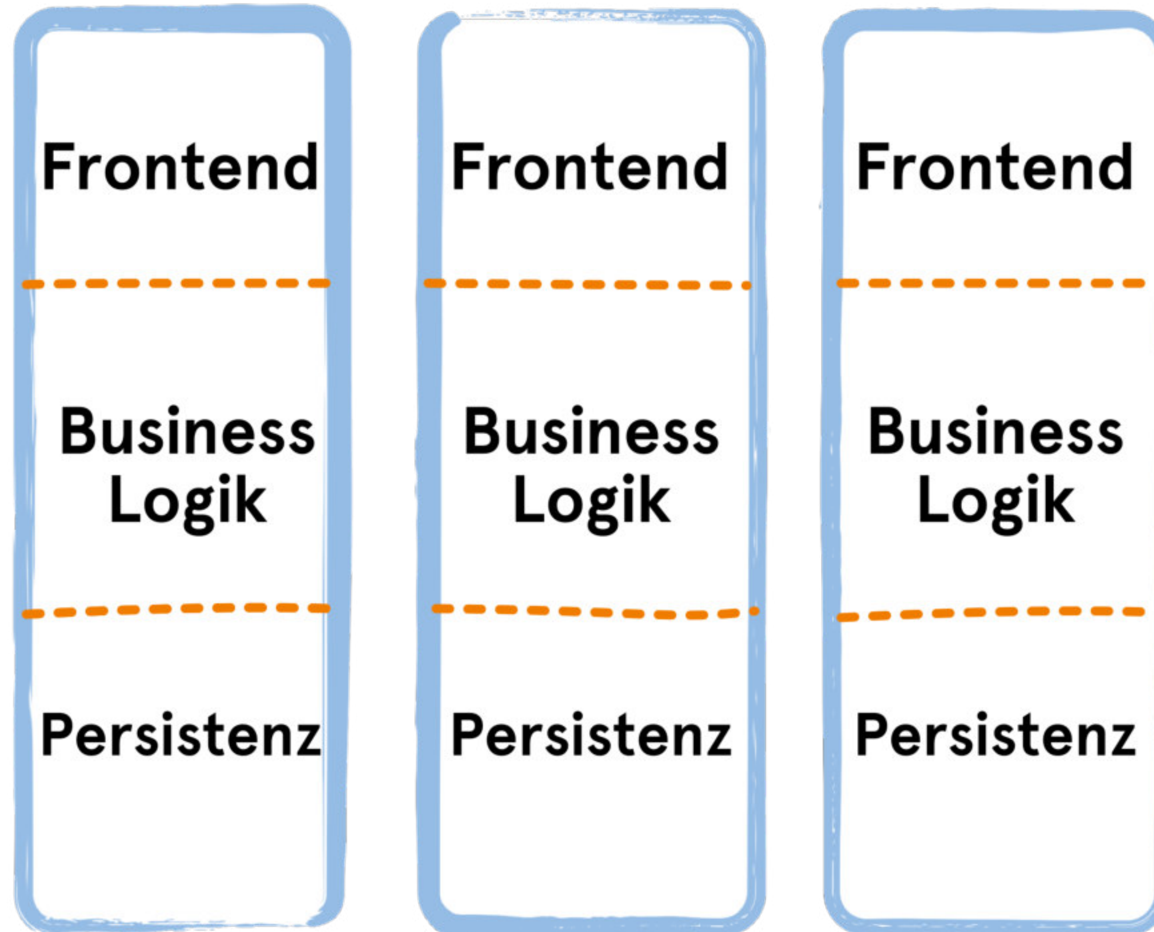
COFINPRO

Frontend-Modularisierung



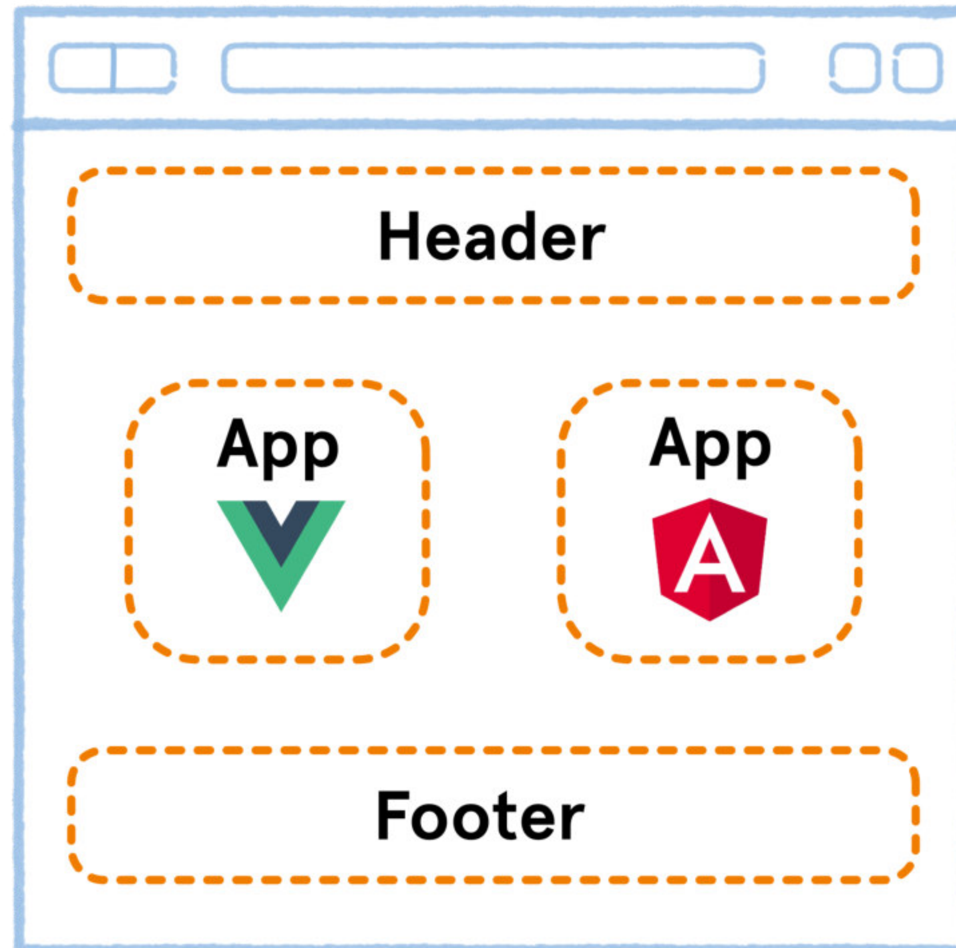
COFINPRO

Frontend-Modularisierung

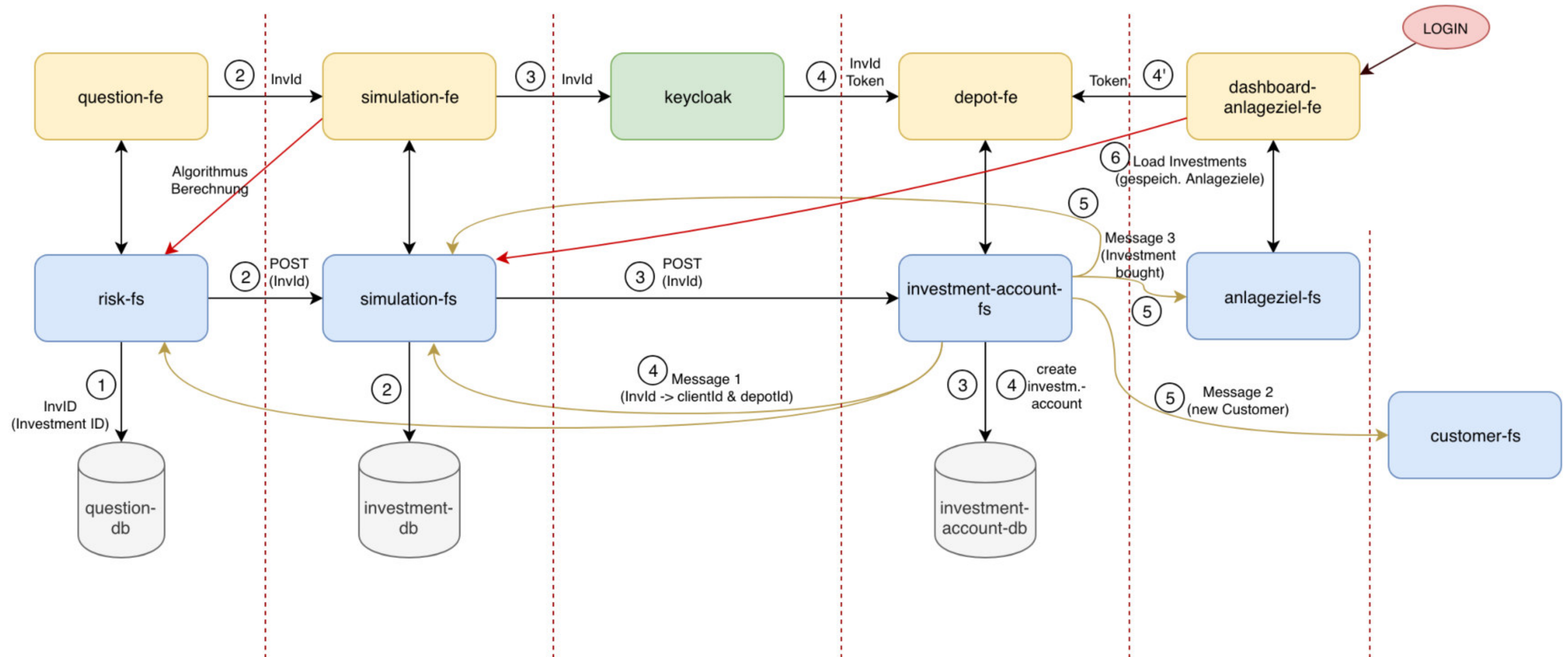


COFINPRO

Frontend-Modularisierung



Beispiel: ein Prozess komplett überdenken

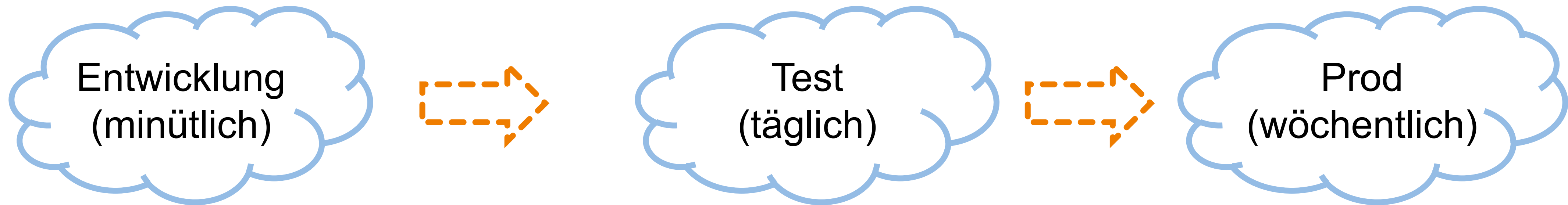


COFINPRO

Was ist noch zu tun

Betriebs- und Delivery-Prozesse

- Problem: stark regulierte Branche
- Regulatorik und DevOps passen nicht immer zusammen



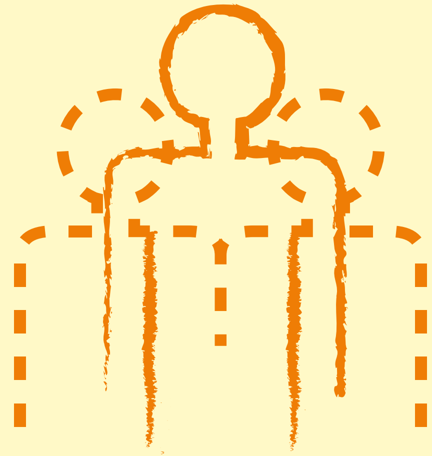
- Mehr automatisiertes Testen notwendig
- Testen der Schnittstellen (Contract Driven)
- Delivery-Pipeline-Verantwortung in den Teams

COFINPRO

Was ist noch zu tun?

- Klare Schnittstellen zwischen Verticals
  - Abhängigkeiten: wir haben noch zu viele
  - Alle problematischen schaffen wir vermutlich nie
- Verticals sind Cluster von Microservices
- Unabhängig Release-Zyklen
- Continuous Delivery
  - Nicht kompatibel mit den Release- und Freigabeprozessen

Was haben wir gelernt?



Was haben wir gelernt?

Was läuft besser?

- weniger Bugs
- klarere Prozesse
- schnellere Umsetzung
- motivierte Entwickler
- technische Probleme werden wieder angegangen

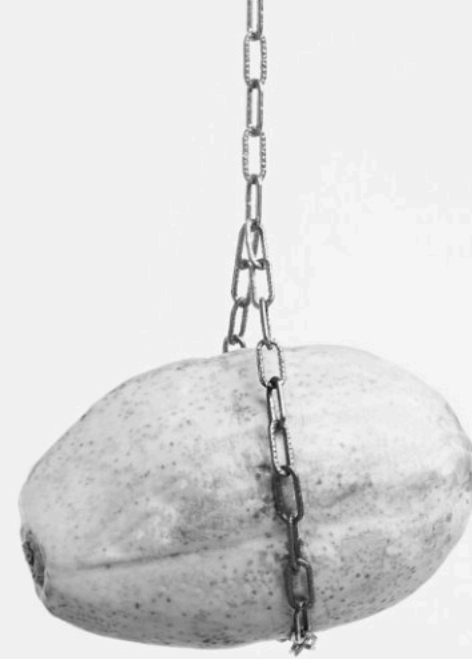
Was ist der Preis?

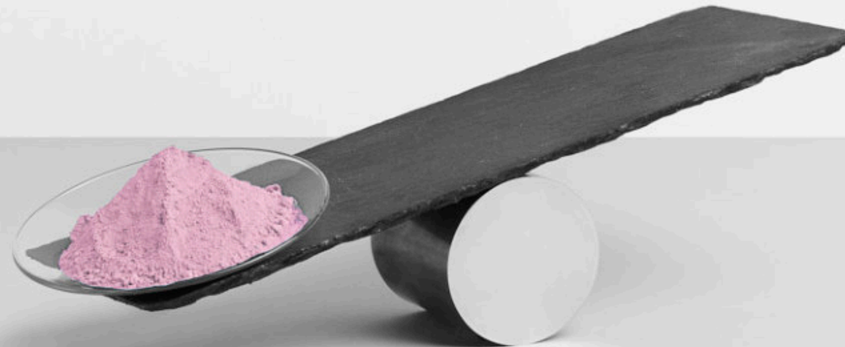
- weniger Konsistenz (Daten, Problemlösung)
- lokale Optimierungen: mein Vertical vs. die Plattform
- weniger Übersicht der Entwickler über die Gesamtplattform
- langsamer bei Anwendungs-übergreifenden Features

COFINPRO

Vielen Dank!

Fragen?



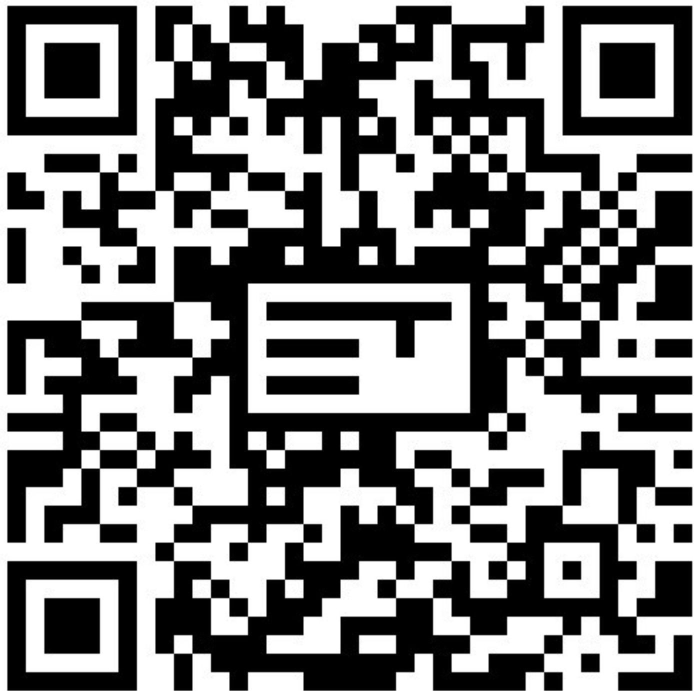


https://speakerdeck.com/mscholtz/vom-distributed-monolith-zu-self-contained-systems-ein-erfahrungsbericht

Bitte geben Sie uns jetzt Ihr Feedback!

Vom Distributed Monolith zu Self-Contained Systems: ein Erfahrungsbericht

Marcos Scholtz, Gregor Tudan



Nächste Vorträge in diesem Raum

11:45 Cloak-and-Dagger-Stories: Absichern von Anwendungen mit OAuth und Keycloak, *Gregor Tudan*

13:30 Testautomatisierung ohne Assertions, *Dr. Jeremias Rößler*

14:30 TDD demystified, *Tilmann Glaser, Peter Fichtner*