A competitive food retail architecture with microservices

A five year journey with microservices

Sebastian Gauder
Software Architect eCommerce
@rattakresch

11.02.2019 ObjektForum Köln

REWE digital


Our history

Details REWE GROUP

Turnover
>57 bn

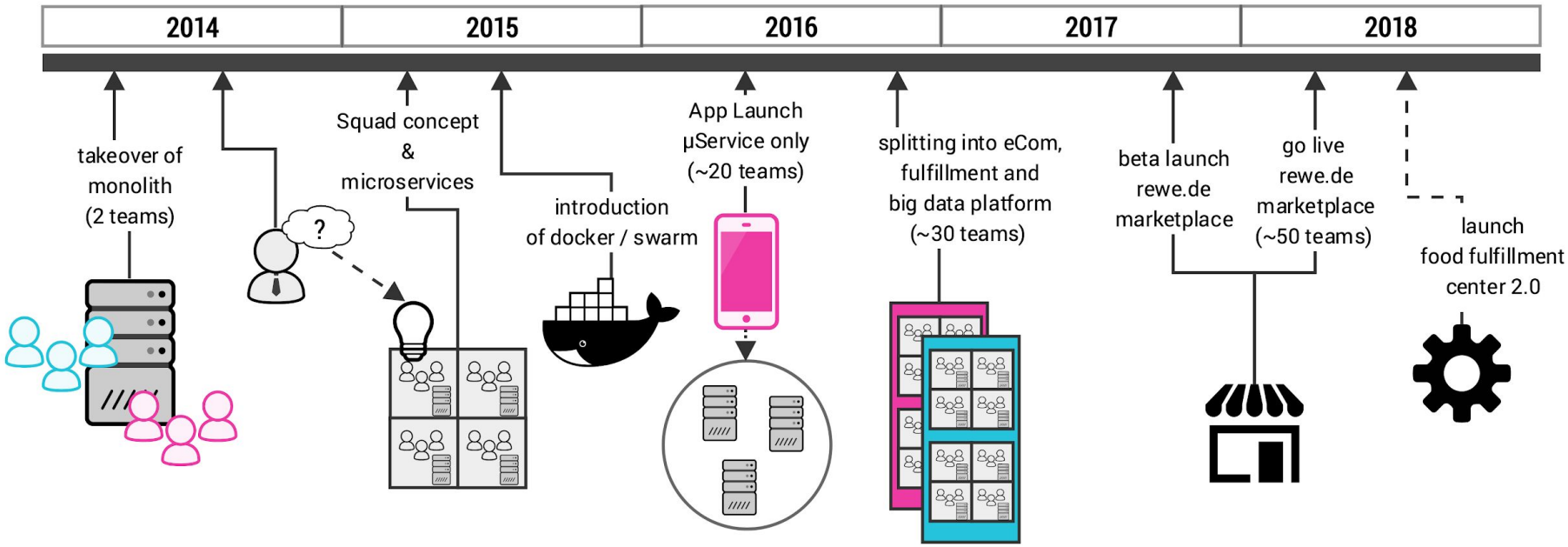
Employees
>345.000

Shops
>15.000

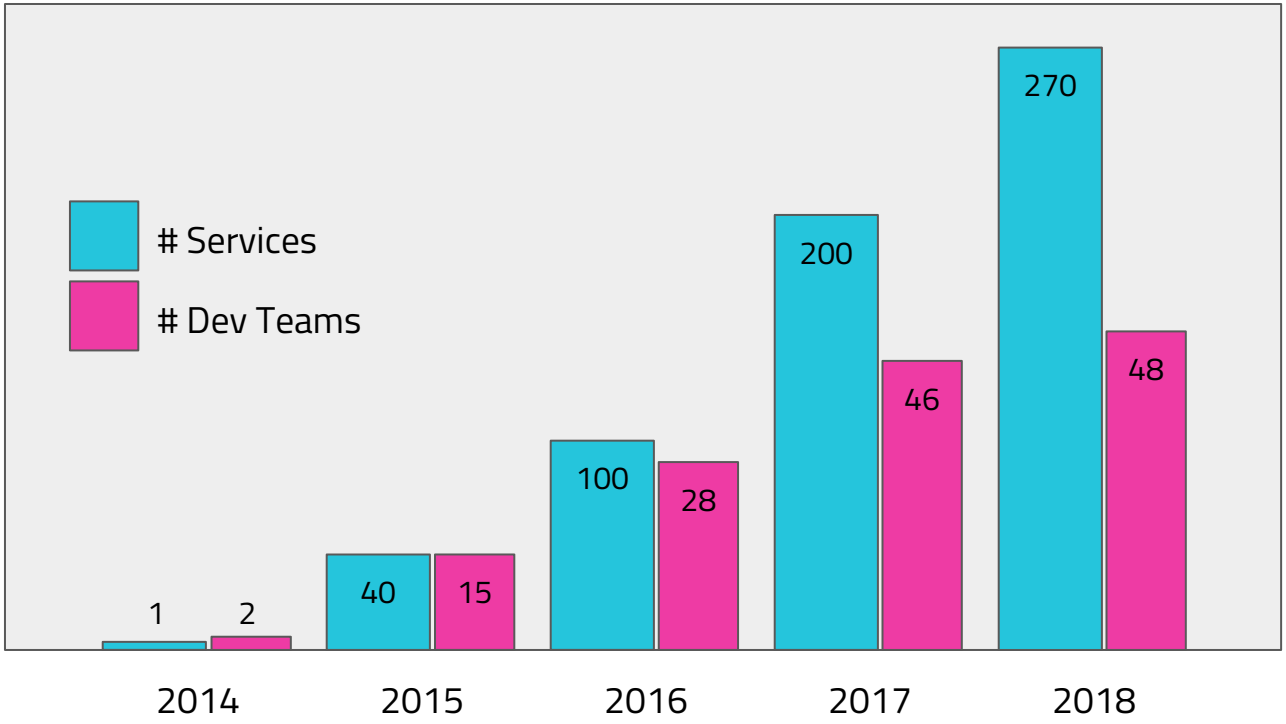
Inudstries
Food Retail, Tourism, DIY

History
>90 years

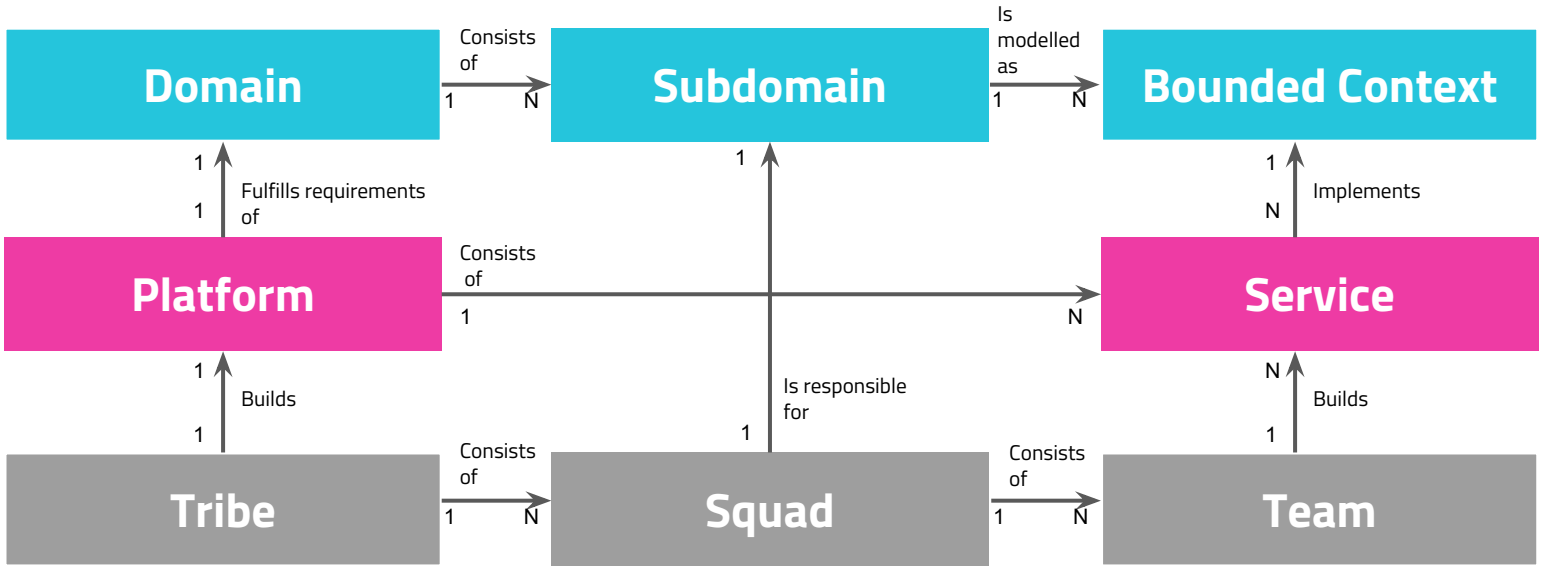
Our history (Spoiler alert)



Our history


Services
Dev Teams
1
2
40
15
100
28
200
46
270
48
2014
2015
2016
2017
2018

Organization and Architecture at Rewe Digital




How to scale monoliths?

Design Goals

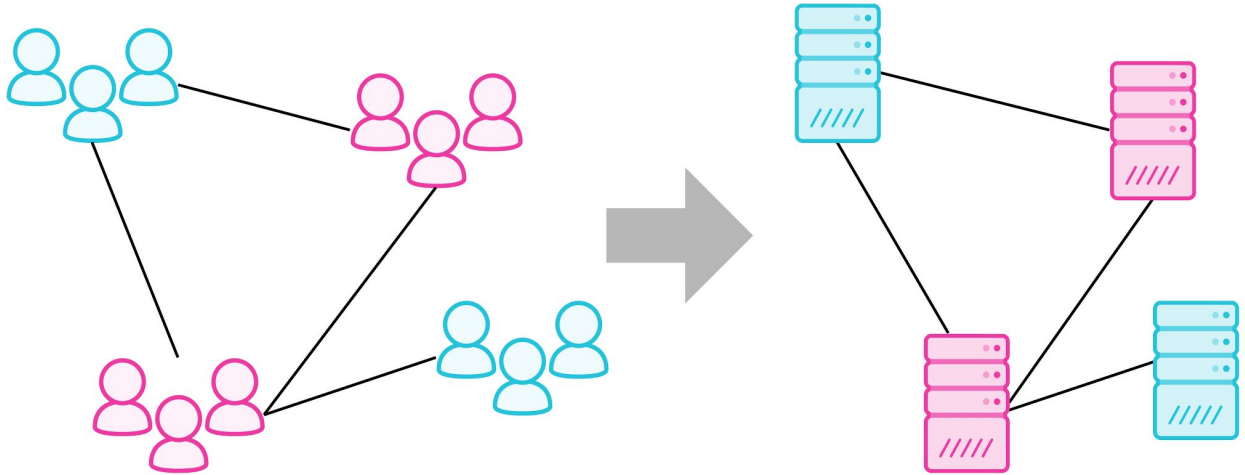
Decentralization

Vertical Boundaries

Conway's law

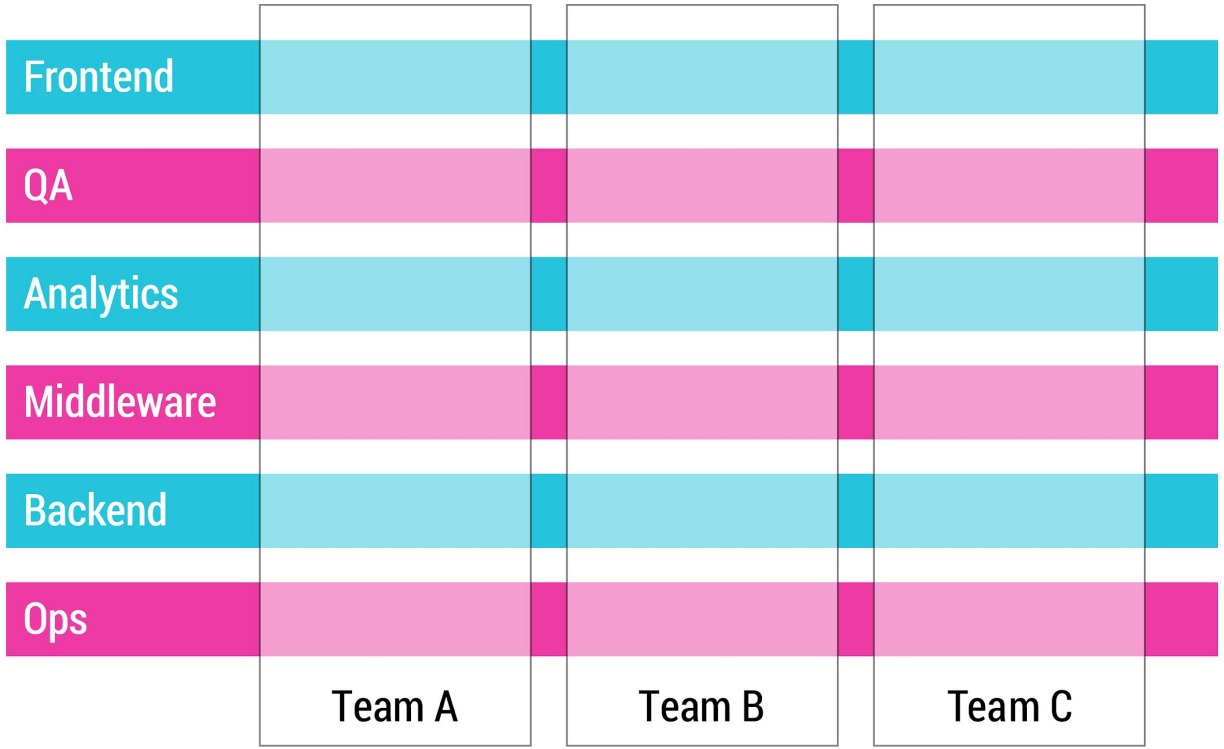
"Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."

Melvin Conway (1967)



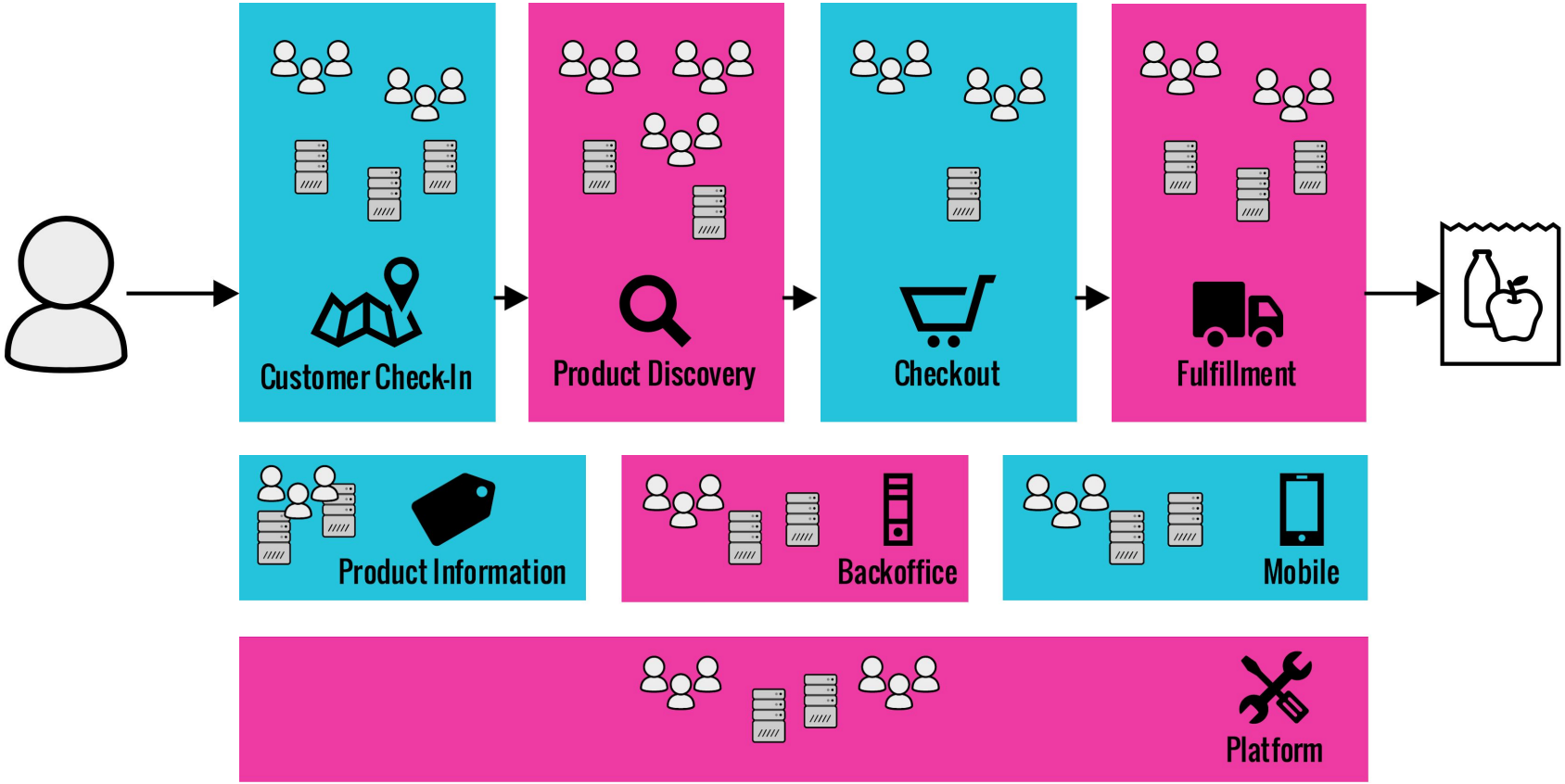
How to organize teams to build a scalable system?

Functional and vertical teams



How to **determine boundaries?**

Customer journey defines subdomains




How to guide developers?

Guarding rails for developers

Design Goals

- Vertical boundaries
- Decentralization

Architectural principles

- Collection of 9 basic ‘laws’
- Autonomy, Automation and Communication

Guides

- Practical manual for common tasks (RFC 2119)
- e.g. Eventing, REST, Authentication

MUST
SHOULD
COULD

autonomy

Autonomy principles

Deploy independently

Ensure that the services can and are deployed by the teams themselves.

Isolate failure

Make the services as resilient as possible.

Hide implementation Details

Different verticals must not share state. Verticals hide their implementation details.

Encapsulate Data Storage

For any data resource exactly one service is responsible. If possible, the data supply should proceed asynchronously in the background

Challenges

Autonomous Teams

- challenge for product owners as they have to rethink their features
  - they might have to reduce the scope
  - split the original feature into smaller ones for each team / bounded context

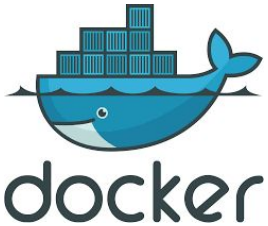
Data-integration pattern

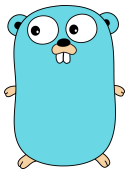
- Change your service behavior depending on the data you have to display
- Enable others to deliver functionality later

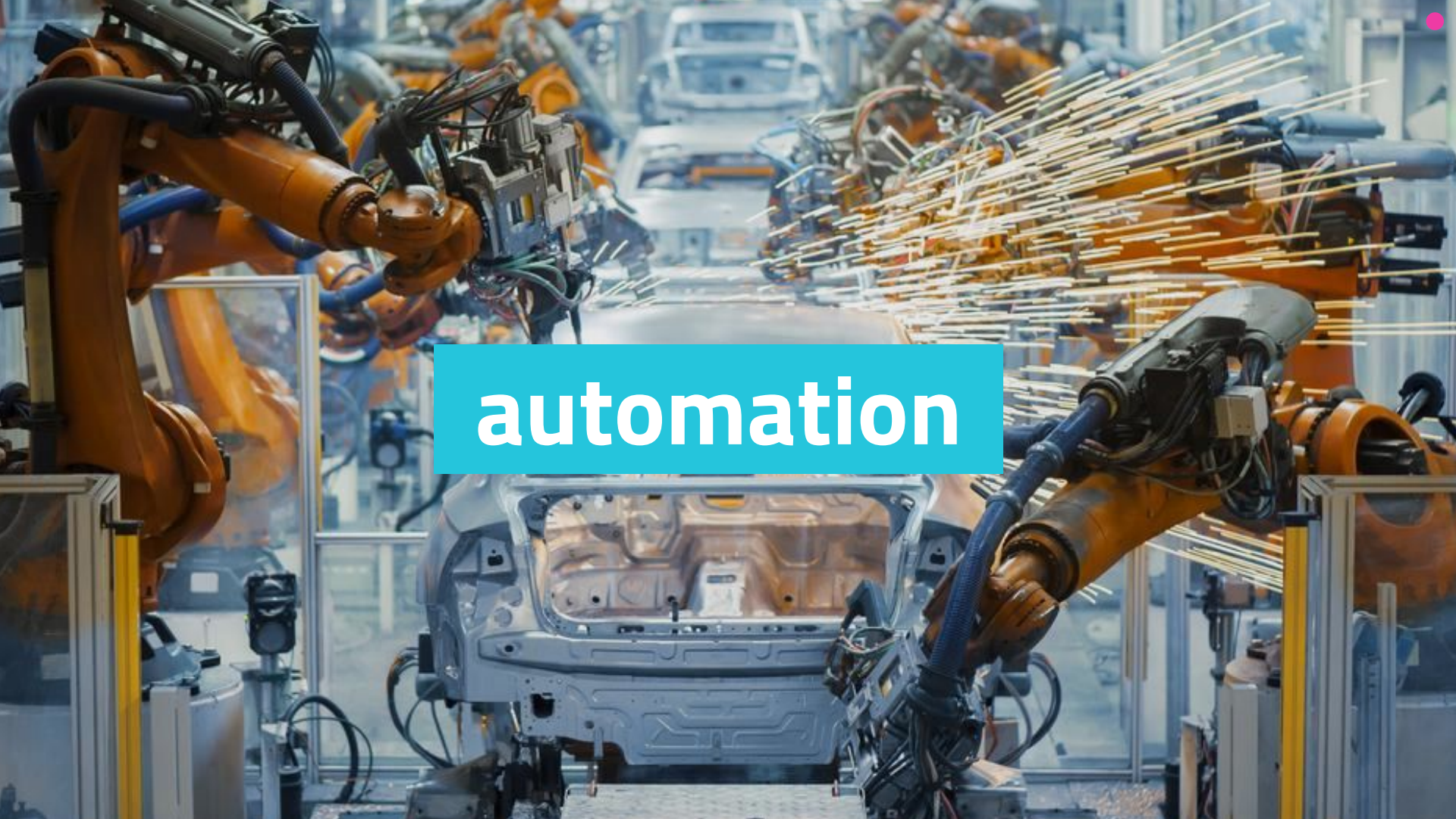
Evolving Implementations

- Start with a fitting but simple solution and get better by every team implementing it

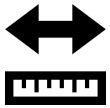
Autonomy opens your tech stacks






automation

Automation principles



Be Scalable

Services are scaled horizontally

Embrace a Culture of Automation

Test, deployment and operations are completely automated

Be Highly Observable

Use of semantic monitoring to see if the whole platform works correctly.


communication

Communication principles

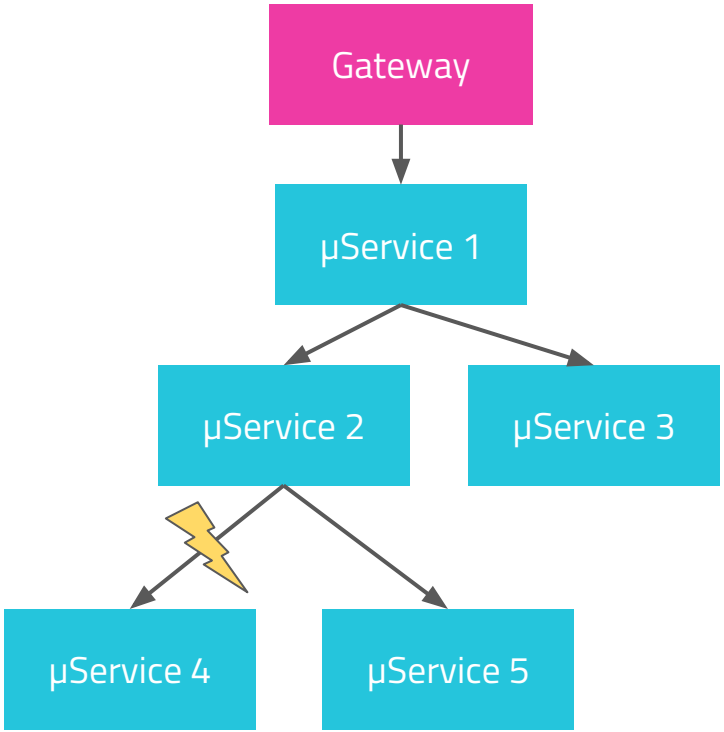
Follow REST Principles

The API of a service follows the RESTful paradigm (REST maturity level 2)

Standardize Service communication

Inter-service communication is standardized and if possible asynchronous

Problems in HTTP/REST-only architectures



Things that help:

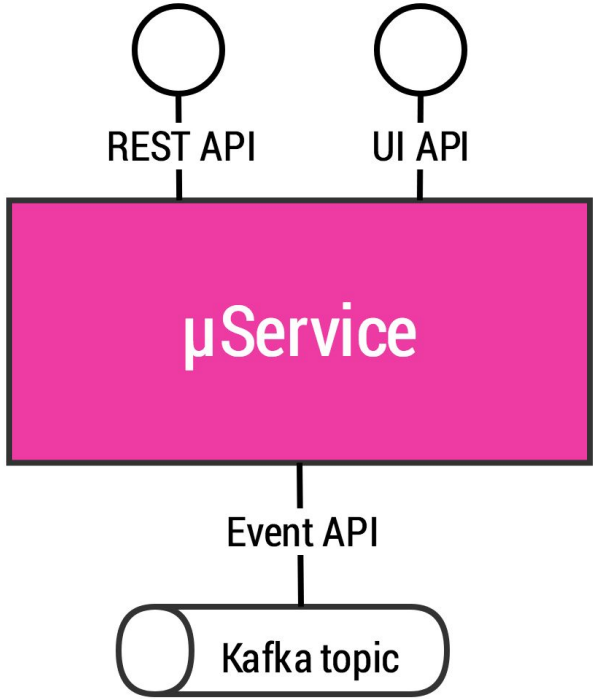
- Timeouts
- Fallbacks
- Circuit Breakers
- **Eventing**

Communication in microservice worlds

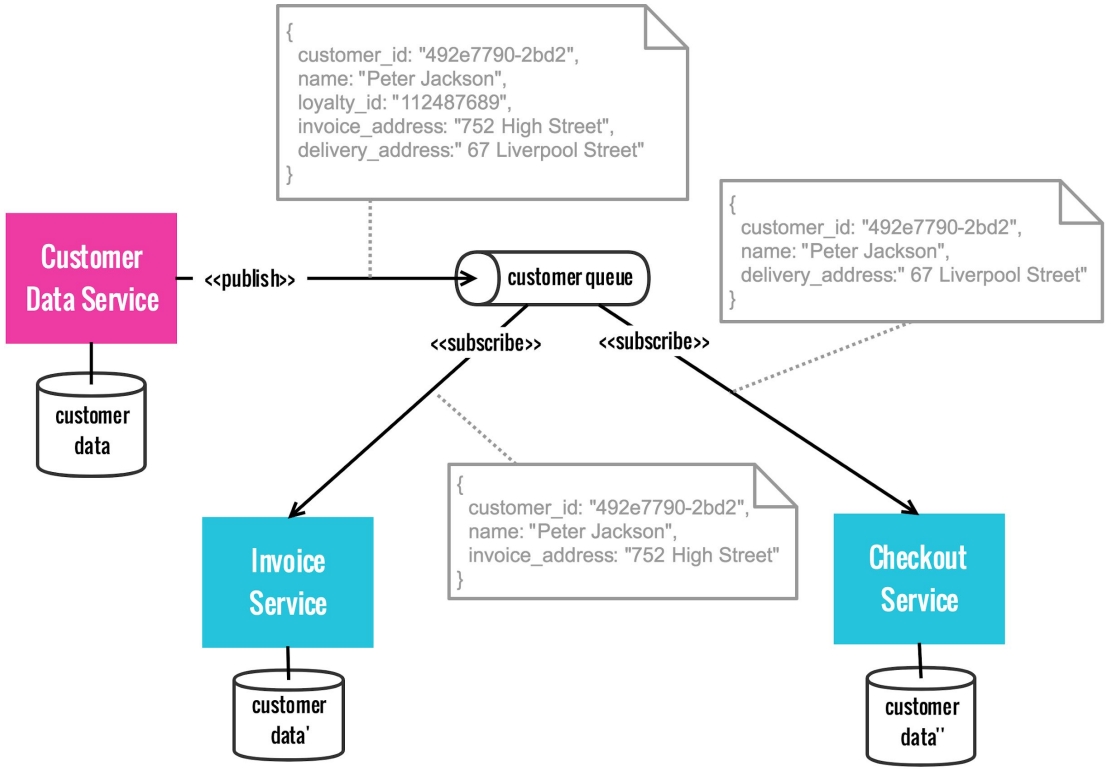
Having data is better than fetching data

Asynchronous > Synchronous

- Have as much needed data locally as possible -> resilience at request time
- Duplication of data is accepted and wanted
- Problem of eventual consistency



Eventing with Apache Kafka



- **Eventing != Messaging**
  - Publish events that already **happened**
  - One owning service per queue/topic
  - Autonomy at request time
- **Complete entities - not deltas**
  - Transactional completeness
  - Allows log compaction
- **Re-writing and Re-reading**
  - needed in case of additional data
  - useful in case of data loss

Lessons learned with microservice APIs

Eventing is an API as well

- Events have to behave like APIs and avoid breaking changes.

Writing APIs is hard

- Teams tend to write APIs for special clients (e.g. mobile apps)
- New clients often don’t match and require API changes

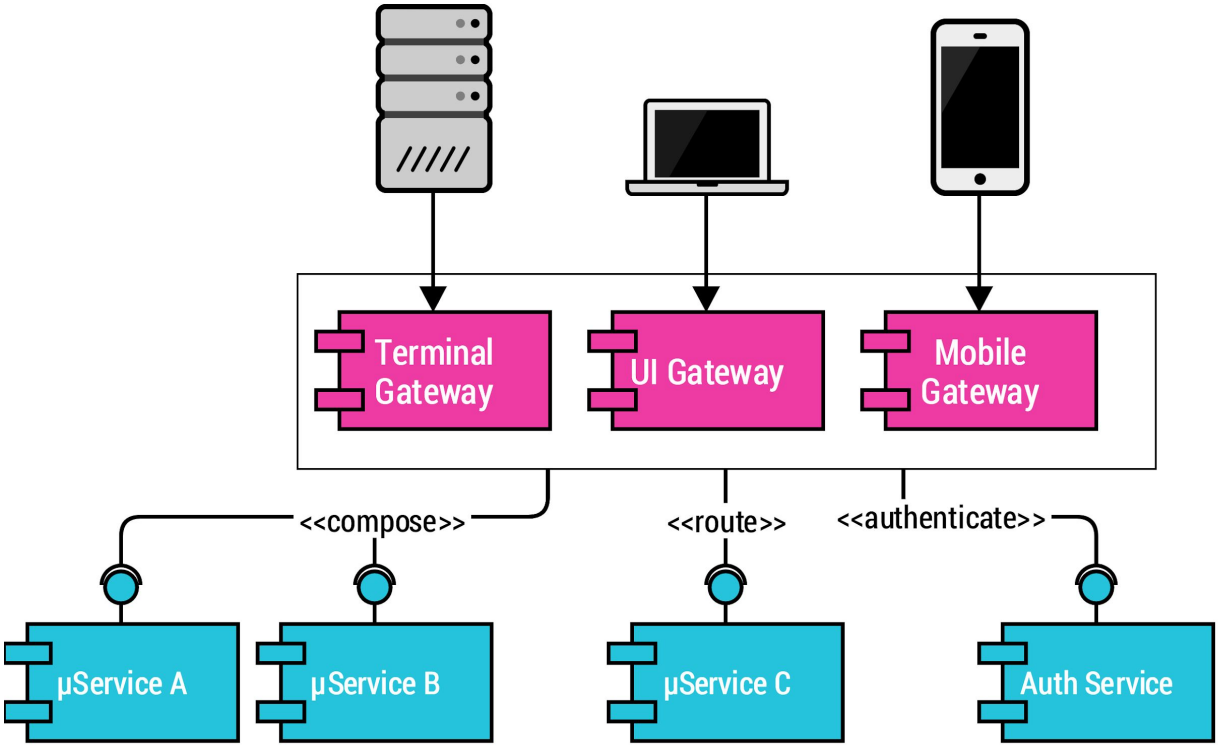
Breaking changes require a strict procedure

- Could be solved by new endpoints, new topics


How to access?

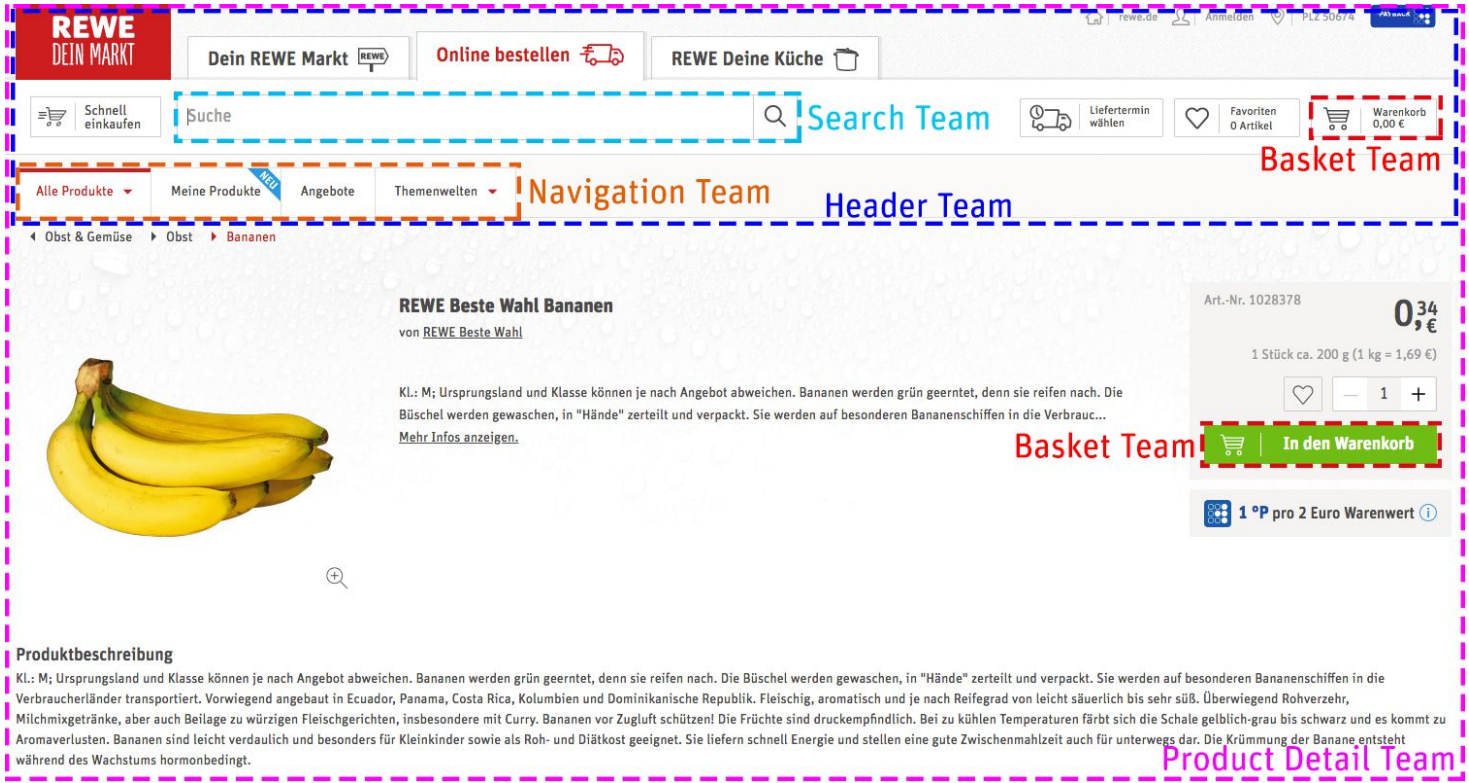
Usage of API gateways as entry points

How to access Microservices?



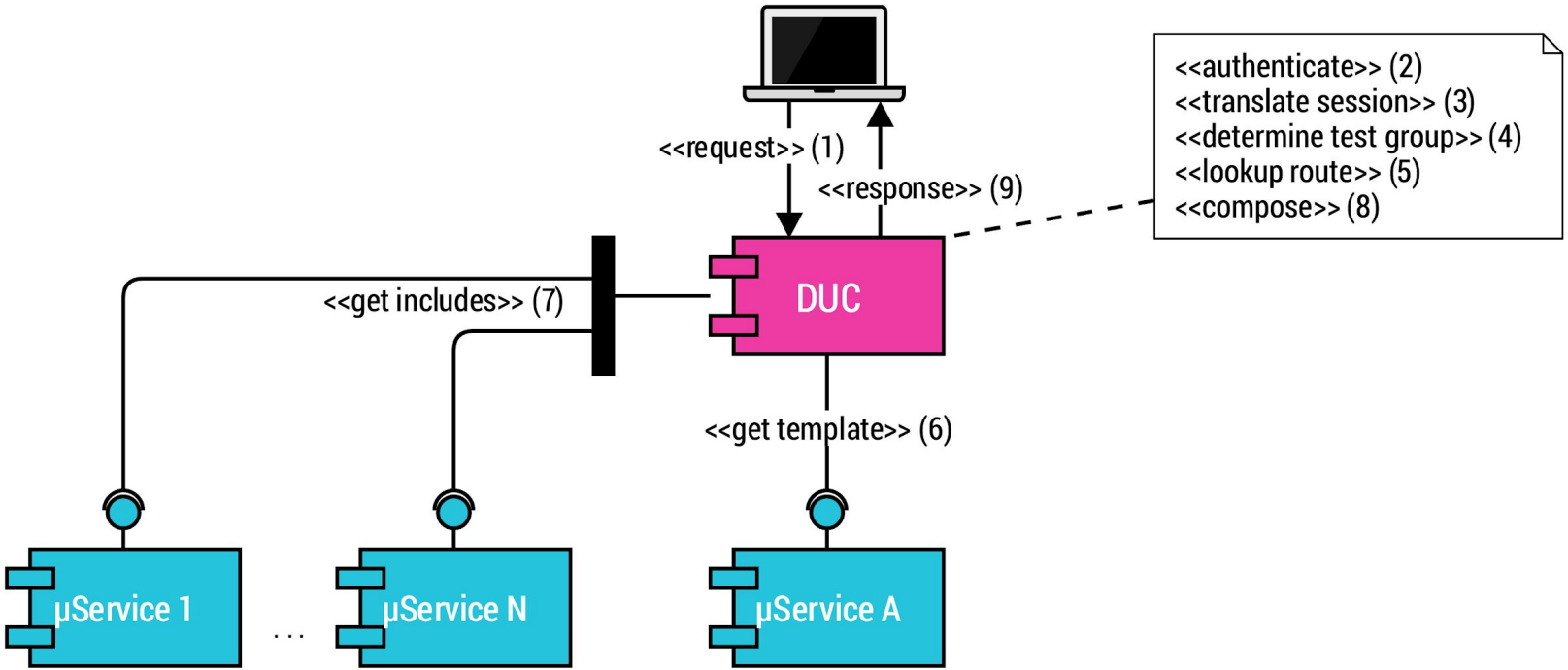
Dynamic frontend composition

How UIs in microservice architectures should be composed



Dynamic UI composition with DUC

Where does my UI come from?

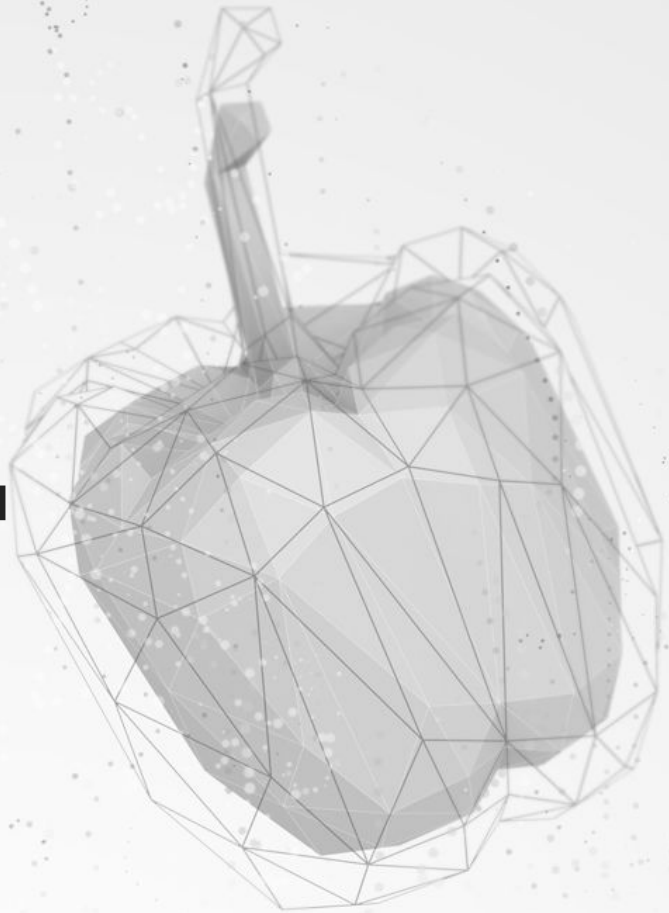


Git Project: rewe-digital/integration-patterns

https://git.io/vA2MY

Thank you

Questions ?



REWE digital

A competitive food retail architecture with microservices

A five year journey with microservices

Sebastian Gauder
Software Architect eCommerce
@rattakresch

11.02.2019 ObjektForum Köln

REWE digital