Microservices reloaded

What about the frontend?

Stephan Rauh

Mission

Mit unserer Leidenschaft für neue Technologien und unserem Anspruch an herausragende Beratung sind wir bei unseren Kunden der **Motor der digitalen Transformation**.

» Wir entwickeln **überraschend mehr Möglichkeiten!**

1. Microservices in a nutshell
2. Transclusion
3. Design considerations

Microservices

- What?
- Why?
- When?

Microservices: What? Why? How? When? When Not?

- Less organisational/architectural friction
  - use Conway's Law to your benefit
- Separate deployments for separate services
- Allows for a heterogenous system,
- … which in turn leads to long-term evolvability

Inspired by https://gustafnk.github.io/microservice-websites/ published under an MIT licence

Martin Fowler’s Definition of Microservices

The microservice architectural style is an approach to developing a single application

- as a suite of small services,
- each running in its own process
- and communicating with lightweight mechanisms,
  - often an HTTP resource API.
- These services are built around business capabilities and independently deployable by fully automated deployment machinery.
- May be written in different programming languages
- and use different data storage technologies.

What About the Frontend?

- Most developers just ignore the frontend
- Martin Fowler's article about microservices explicitly included the UI (but nobody remembers)
- Let's coin a new term:

An Self Contained System, aka SCS is a "team-sized" microservice including its UI.

Or call it portlet, if you've got a Java background.

What Developers and Architects Make of it

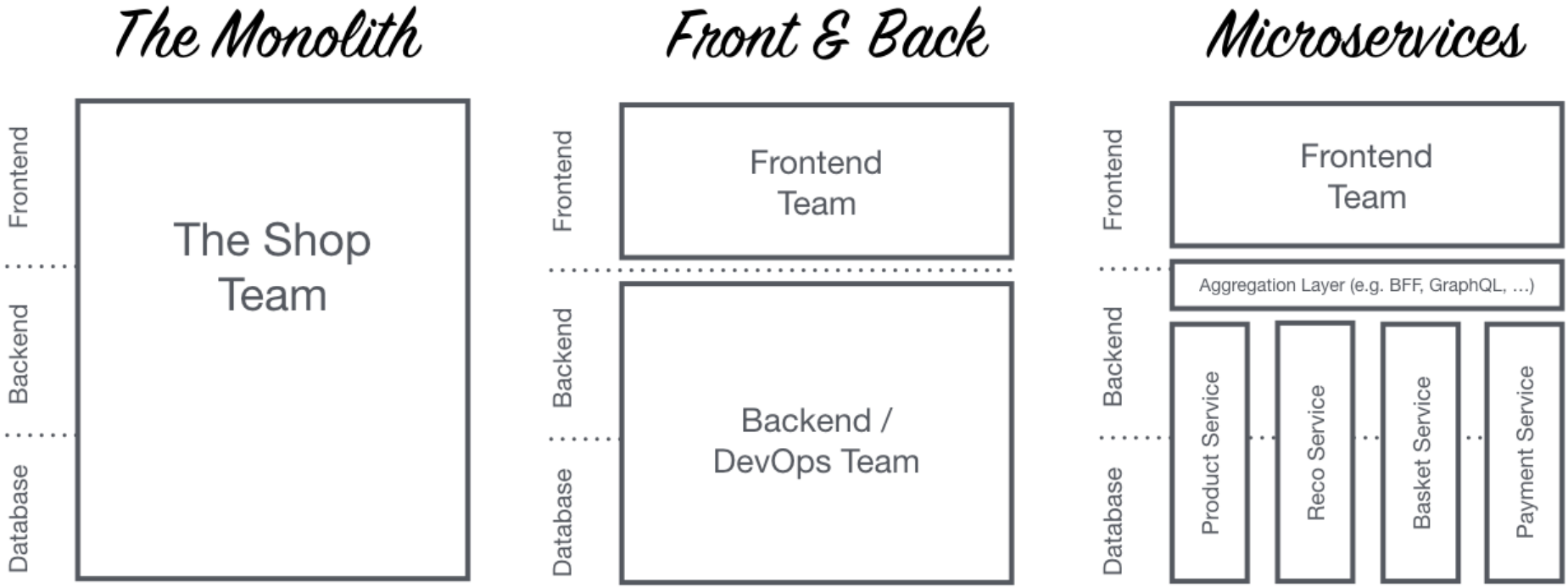
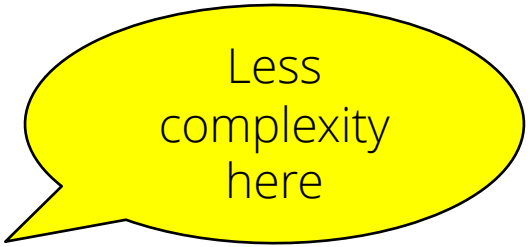


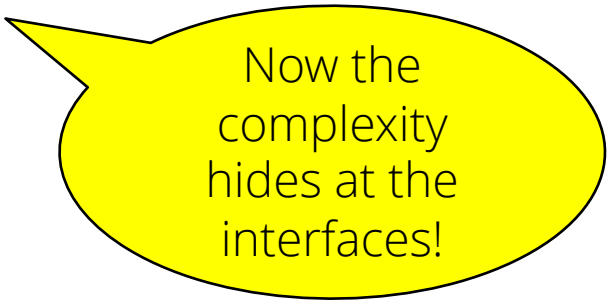
Image source: https://micro-frontends.org/ published under an MIT license

The Majestic Monolith

Key Benefits of Microservices

- Domain teams can work independently
- Microservices can be deployed independently
- Avoid constructing silos around technological capabilities
- Loose coupling
  - encouraged by slow and unreliable interfaces
  - typically REST calls via TCP/IP





Transclusion

Interacting tiles

tiles

HTML links

Transclusion Strategies

Sensordaten live erleben: Big Data im Kontext der Industrie 4.0 – Teil 3

Posted 9th Mai 2019

Kapitel 3: Visualisierungsansicht Die Daten, die aus IoT Core und Lambda-Function in den Elasticsearch Service (ES-Service) übergehen, werden automatisch einem Index zugeteilt. Die Instanzen des ES-Service bieten eine gewisse Speicherkapazität, die es ermöglicht, bereits mehrere Wochen zurückliegende Daten mit Kibana darzustellen. Kibana benötigt als ES-Service-Plugin

Sensordaten live erleben: Big Data im Kontext der Industrie 4.0 – Teil 2

Posted 8th Mai 2019

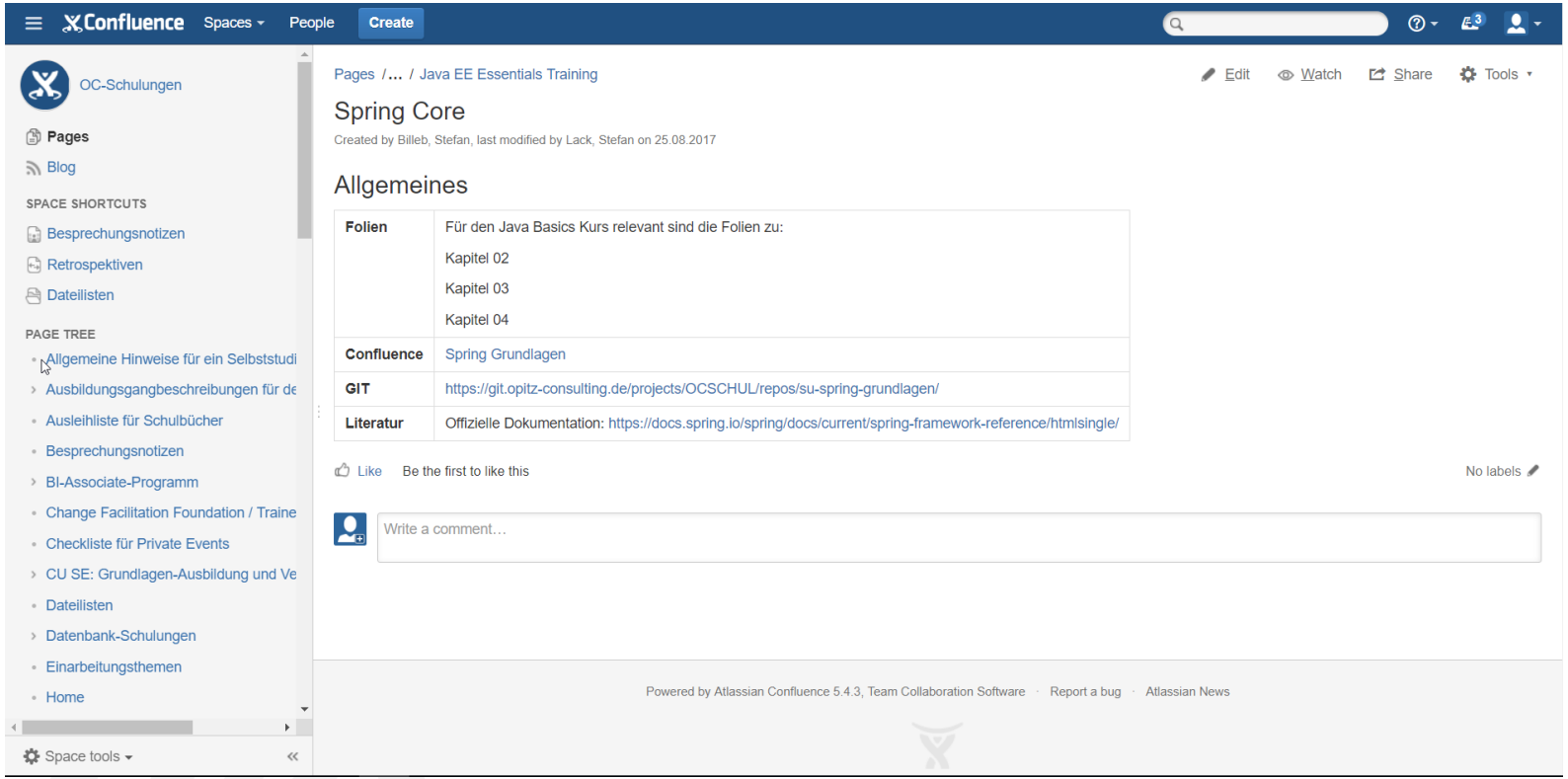
Kapitel 2: Verarbeitung der Daten Aufnahme über den IoT Core Wie bereits in Kapitel 1 erwähnt, erhalten wir alle eingehenden Maschinen-Daten im JSON-Format über das MQTT-Protokoll. Damit diese reibungslos in die Cloud-Architektur aufgenommen werden können, fungiert zu Beginn der IoT Core als sogenannter Broker, der in unserem Fall alle Nachrichten eines bestimmten

```
Blog about Dig
Transformation
Big Data, Clou
Infrastructure
Software Devel
BPM & Integrat
Powered by OPI
CONSULTING
```

Transclusion Strategies

Image source: https://pxhere.com/en/photo/1573451 published under an CC0 Public Domain license

Transclusion Strategies



Challenges of Loosely Coupled UIs

- Cross-app style guide
- Even across team boundaries!

Beware: UI team may become a bottleneck

Image source: https://livingstyleguide.org/ / https://github.com/livingstyleguide/livingstyleguide/tree/master/website published under an MIT license

Documents-to-Applications Continuum

Which kind of application do you have?

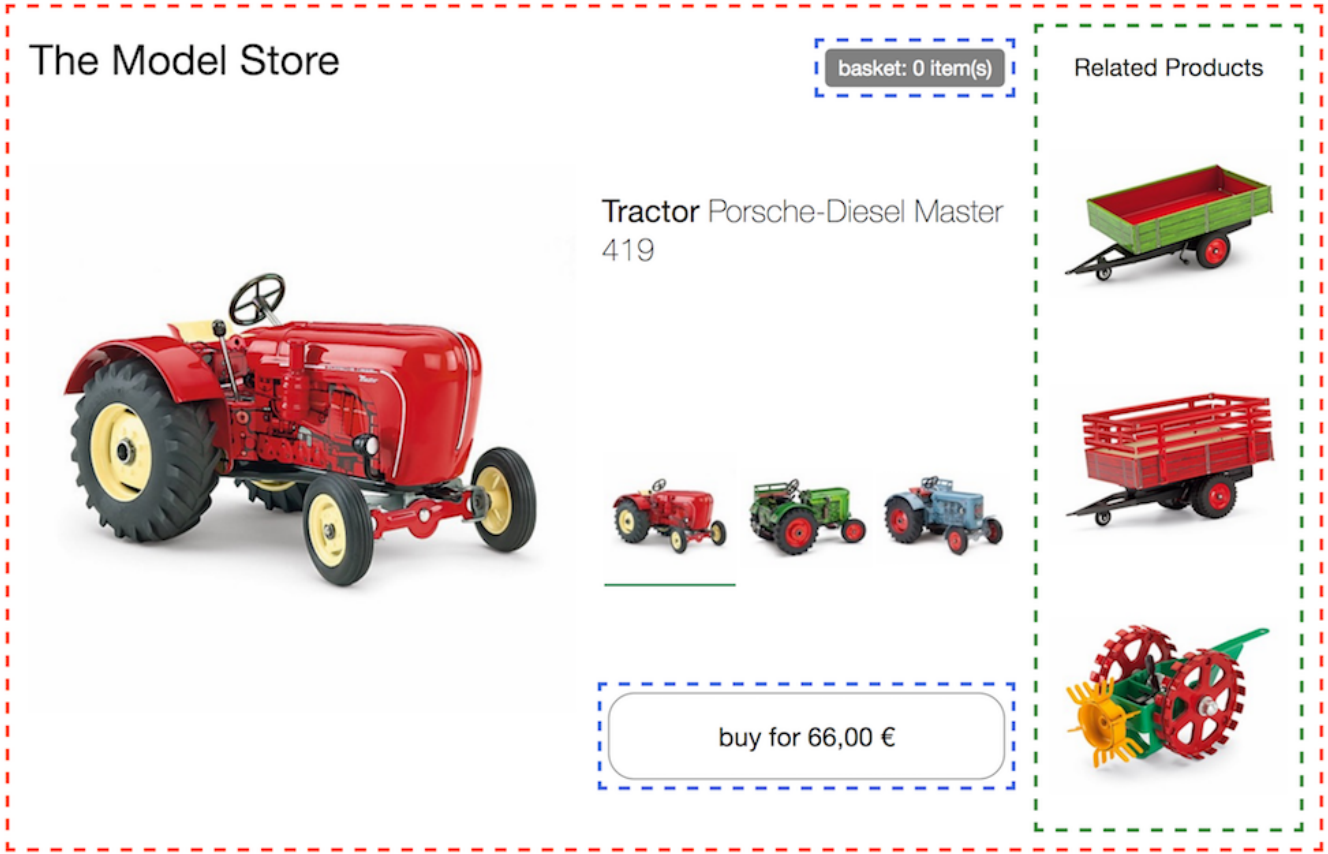
Image source: https://2018.ar.al/notes/the-documents-to-applications-continuum/ pulished under a CC BY-SA 4.0 licence

Documents-to-Applications Continuum

Interacting tiles

tiles

HTML links

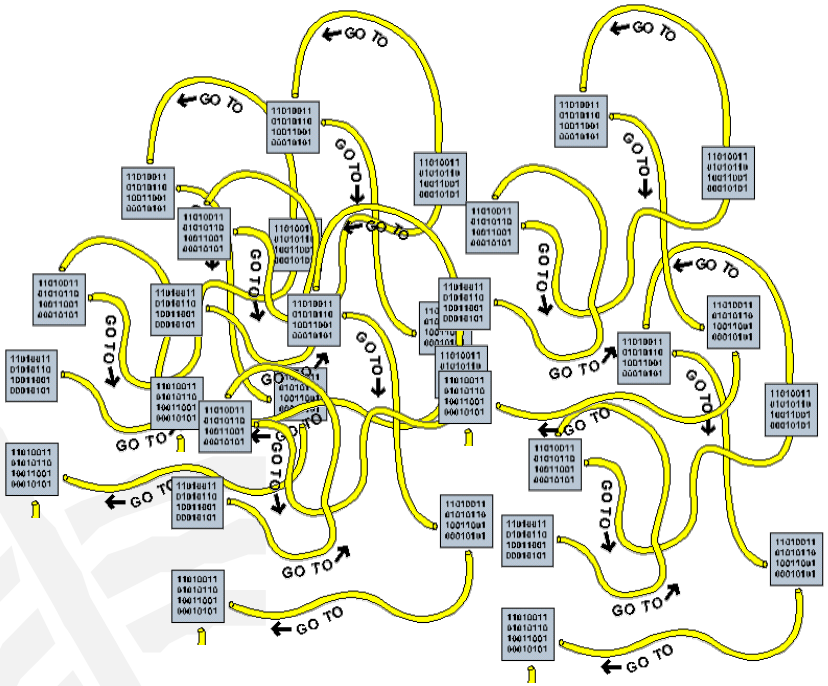


Team Product Team Checkout Team Inspire

Image source: https://micro-frontends.org/ published under an MIT license

Spontaneous Developer Reactions

We cannot do that! We must talk the customer out of it!



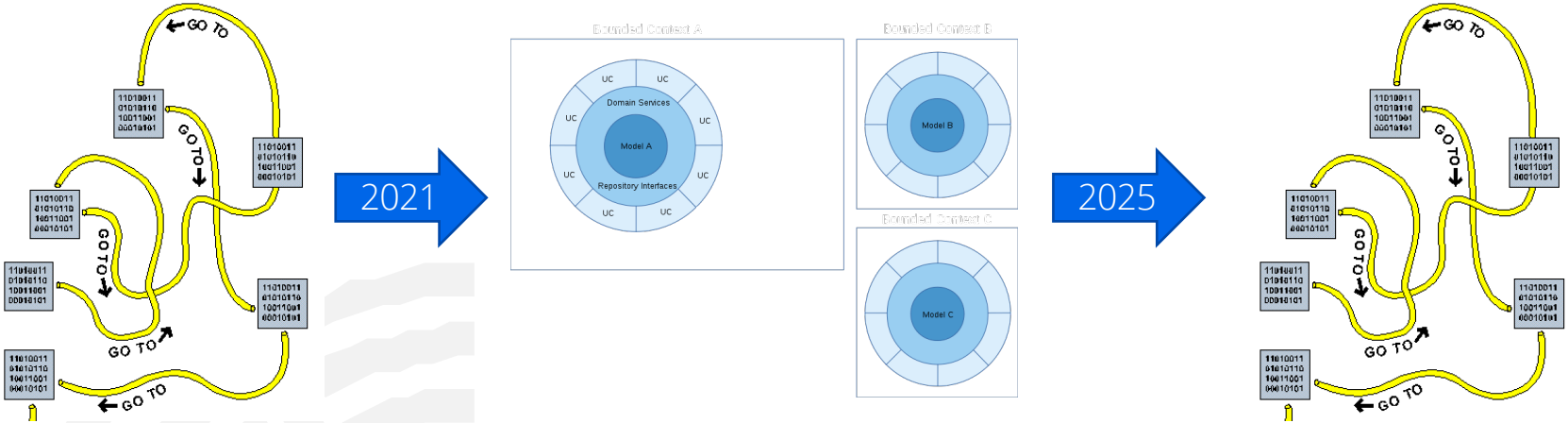
Sometime is a sensible requirement

Our job is to implement it in a maintainable way

https://commons.wikimedia.org/wiki/File:Spagetti_code_structural_graphic.GIF
https://commons.wikimedia.org/wiki/File:Clean_Architecture_%2B_DDD,_full_application.svg

The Real Challenge

- How to implement „portlets“ in the first place?
- How to avoid the mistakes of Java portlets?
- How to avoid future maintanence hell?



- https://commons.wikimedia.org/wiki/File:Spagetti_code_structural_graphic.GIF
  https://commons.wikimedia.org/wiki/File:Clean_Architecture_%2B_DDD,_full_application.svg

Design Considerations

One Microservice per Page?

- No! Don't be shy!

- Experience from the past:
  - Java portal server + portlets
  - That's how to do it all wrong

- Lessons learned:
  - Inter-portlet communication comes at a price
  - Keep the interfaces small

  → tiny portlets are useless

  → Portlets should cover a domain or a use-case

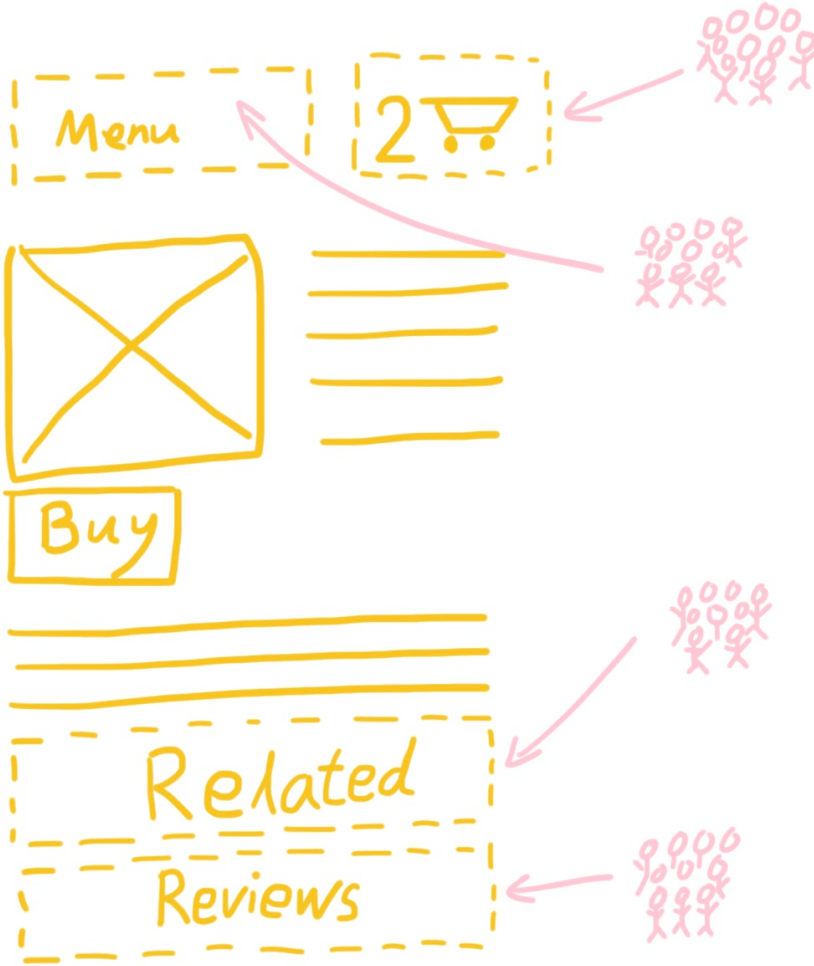


Image source: https://gustafnk.github.io/microservice-websites/ published under an MIT licence

Sketch of the Application

Menu
- Product catalog
- Shopping cart
- Account settings
- ...

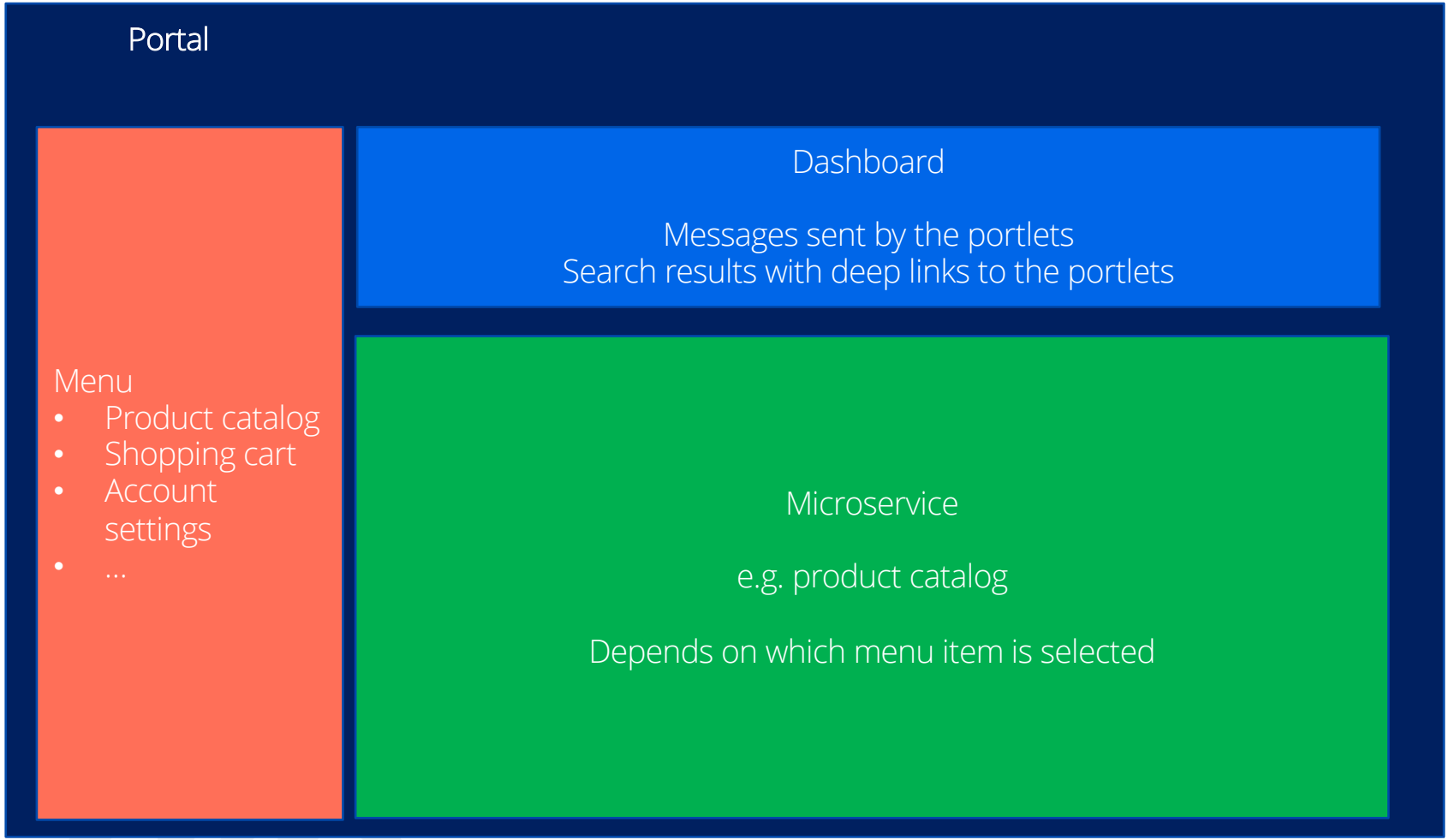
Dashboard

Messages sent by the portlets
Search results with deep links to the portlets

Microservice

e.g. product catalog

Depends on which menu item is selected


Portal
Menu
• Product catalog
• Shopping cart
• Account settings
• ...
Dashboard
Messages sent by the portlets
Search results with deep links to the portlets
Microservice
e.g. product catalog
Depends on which menu item is selected

Sketch of the Solution

- One „portal“ SCS
  - „portlet“ registry
  - Glue code to display them all on one page
  - Resilient to missing or broken „portlets“
- Multiple „portlet“ SCSs
  - Registering at the portal
  - Loosely coupled to other „portlets“
  - Communication:
    - Server-side REST
    - Client-Side DOM events plus LocalStorage

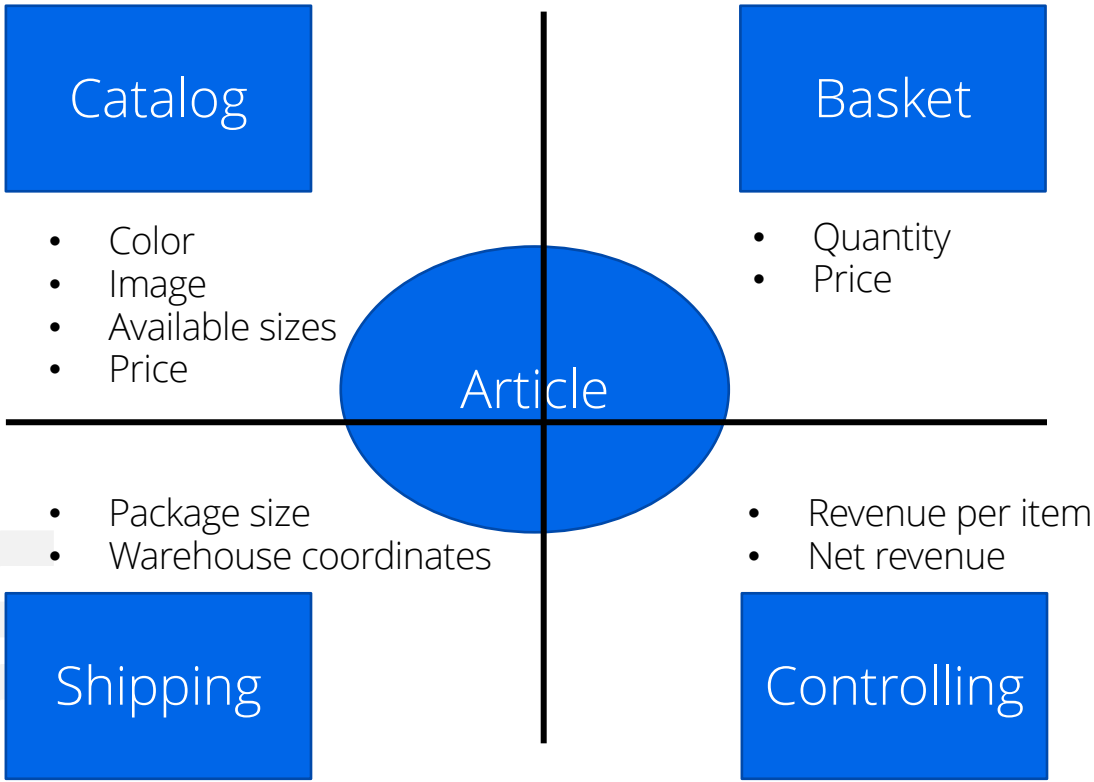
Domain-Driven Architecture

- You all know it
- ... 'nuff said!

Image source: tweet by Michael Plöd

Domain-Driven Surprises

- Sometimes entities belong to several domains
- With little to no shared information

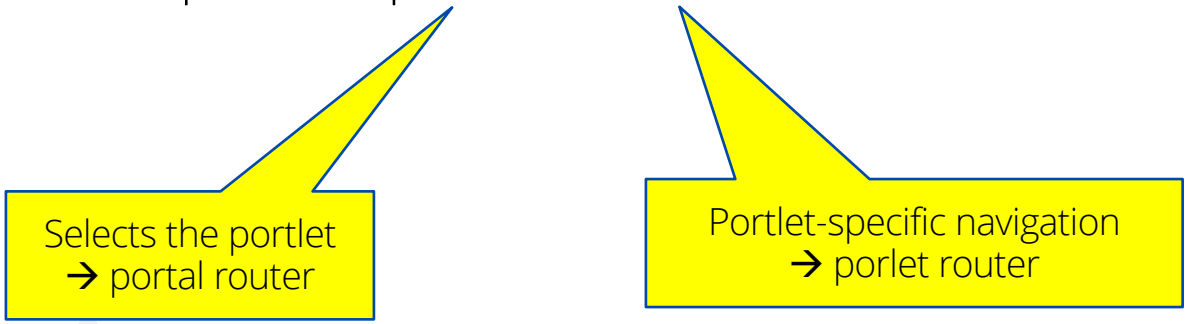


Deep Search

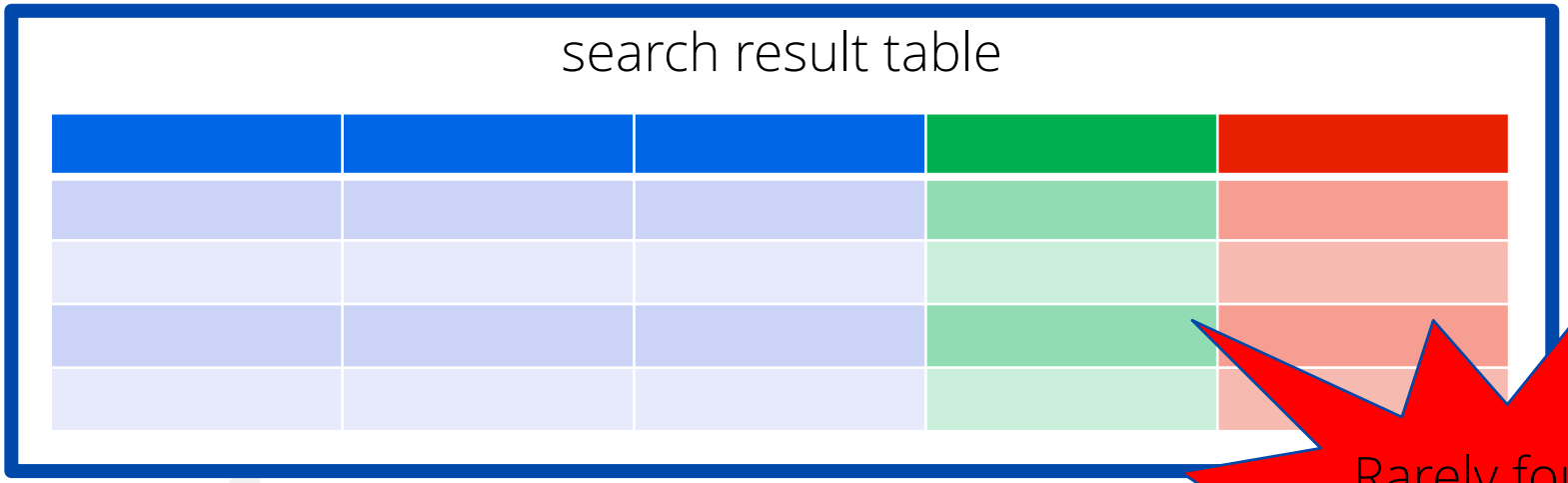
- Idea: portlets contain domain specific information
- Our search should find that, too

Deep Navigation

- Add additional navigation info to the URL
- http://example.com/product/324234



Adding Portlet Content to the Other Portlets



- Most teams shy away from that
  - Additional server roundtrips
  - Generic solutions are difficult
  - … but rewarding?

What About Transactions Across Services?

- You rarely need them!
- … if you‘re using useful domains boundaries

Implementing Micro-Frontends

- Server-side includes
- Client-Side includes
- Angular Microfrontends
- Alternative approaches

Server-Side Rendering

- Gustaf Nilsson Kotte has a great article on the topic

Client-Side Integration

- Allows for optimistic UIs
- Skeleton Screens
- Generally: more options to tweak the user experience

Angular Microfrontends

- We‘ve published two working sketches at our GitHub repository: https://github.com/opitzconsulting/ui-frontend-integration
- Monorepos containing one portal and three Microfrontends
- ng-micro-frontends-1 uses Angular 7
- ng8-ivy uses Angular 8
  - And the new Ivy compiler

Live Demo

Image source: https://www.flickr.com/photos/christiaancolen/20013034233 published under a CC BY-SA 2.0 license

Angular Microfrontends Implementation (1/2)

Challenges:

- Need to modify the Angular.json to create one single JS file
  - → ngx-build-plus by Manfred Steyer
  - → Angular.json on GitHub
  - → Package.json auf GitHub
- Remove the bootstrap code
  - The Angular runtime adds ~100 KB to each WebComponent
- Add the code registering the WebComponent
- Dynamically add the WebComponent to the portal page

Angular Microfrontends Implementation (2/2)

- Bundle server and client into a common folder
  - Later: common deployment articfact
- Create build skripts everybody can memorize
  - Old prototype used shell scripts
  - New prototype adds scripts to the package.json
- Find a way to develop efficiently
  - Compiling and starting the entire Microfrontend suite is slow and tedious
  - Solved in the old prototype, open task in the Ivy prototype

CSS Namespaces

- Every WebComponent uses the same CSS namespace
- Christof's build script fixes that:
  - view it on GitHub
  - Just add it to the **`postbuild`** script in the package.json
- Probably there's also a Webpack plugin doing the trick
  - but it's really simple!

Angular Microfrontends

Challenges:

- Both portal router and portlet router must react to URL changes
- Sometime the portal router doesn’t pick up the URL changes

Would you do it Again?

- Angular 8 supports creating WebComponents
  - Getting better each version
  - but it still feels clumsy
- Ivy is a huge improvement!
  - Plus, they promise to make Ivy even better with Angular 9
- Vue.js is reported to be much simpler
  - … but I didn't use it myself yet
  - Vue.js WebComponent need the Vue.js runtime, resulting in a memory penalty

→ evaluate both Angular and Vue.js before starting the project

→ Also have a look at Stencil and Svelte

Any Questions?

- It's your turn!

Let‘s make the web a place to be!

Stephan Rauh

OC architecture board

0172-205 59 66

Stephan.Rauh@opitz-consulting.com

@beyondjava

http://www.beyondjava.net

WWW.OPITZ-CONSULTING.COM

@OC_WIRE

OPITZCONSULTING

opitzconsulting

opitz-consulting-bcb8-1009116