The Microservices journey from a startup perspective

Susanne Kaiser  
CTO  
@suksr

Just Software  
@JustSocialApps
Each journey is different

“People try to copy Netflix, but they can only copy what they see. They copy the results, not the process.”

Adrian Cockcroft, AWS VP Cloud Architect, former Netflix Chief Cloud Architect
Our Transformation Process

- Identify candidates
- Decompose candidates
- Establish Microservices ecosystem
Transformation process

Start → End
Transformation process

Theory (straightforward)

Reality (evolutionary)
Our first version of JUST SOCIAL is out now.

JUST SOCIAL will present the world’s first collaboration with social sharing and social networking in one!

Today’s JUST SOCIAL features:

- Chat with everyone: Stay connected with the entire JUST SOCIAL community.
- Project management: Organize and prioritize tasks with ease.
- Collaboration tools: Work together with real-time updates and comments.
- Social sharing: Share your projects directly to social media.

Our team is excited to bring you this powerful tool. Join us today and see how JUST SOCIAL can transform your work and social life.
The beginning ... A monolith in every aspect

One team

Single Unit

One technology stack

One collaboration product
After an evolving while ...

Productivity suffered

Usability and UX suffered

New features released slowly
Separate Collaboration Apps

JUST PAGE
Social Network

JUST CONNECT
Real-time collaboration

JUST TASKS
Task Management

JUST DRIVE
Document Sharing
Small, autonomous teams
with well-defined responsibilities

- JUST PAGE
  Social Network

- JUST CONNECT
  Real-time collaboration

- JUST DRIVE
  Document Sharing

- JUST TASKS
  Task Management
In the long run ...
Microservices come with complexities

- Multiple independent services
- Slow, unreliable network
- Partitioned data

Operational complexity
Communication complexity
Complexity of eventual consistency
Challenges of transformation

- Different skills & tools required
- Core functionality is hard to untangle
- You still have to take care of your existing system
- Transformation takes longer than anticipated
Our Motivation

- Product and organizational/culture driven
- Enabling autonomous teams with well-defined responsibilities
- Develop and deploy independently to release changes quickly
How to start?
Transformation process

Identify candidates
Key concepts of modelling Microservices

Loose coupling between services

High cohesion within a service
Identify Bounded Contexts

Well defined business function
Bounded Contexts = Collaboration Apps
Transformation process

Decompose candidates
First approach as a co-existing service

Monolith

JUST DRIVE
JUST CONNECT
JUST PAGE
JUST LIST

DB Adapter

Message Broker

Web App
REST API
JUST DRIVE

Message Broker Adapter
DB Adapter

Message

Monolith
Hard work if you do all at once

- New UI
- New data structure
- More features
- Maintain & run current system
Split in steps – e.g. top down
Split in steps – e.g. top down
Split in steps – Step 1) Extracting Web App

Monolith

Web Client

Business Logic

DB Adapter

Browser

REST API

REST Client

Web App
Split in steps – Step 2) Extracting Business Logic
Split in steps – Step 3) Extracting Data Storage
Which one first?

- Easy to extract
- Changing frequently
- Different resource requirements
Stop feeding the monolith
How to handle Authz? Our authz context ...

Authorization based on domain object level

Each domain object has its own authorization handling
How to handle Authz? We started with...
How to handle Authz? But we missed a point ...
How to handle Authz? Leading to ...
How to handle Authz? Not decentralized!

+ Fast in-process calls for read access
+ No single point of failure
- Every MS has to implement authz logic
- For a change every MS has to be updated
- Duplication of all global authz data
- Verbose communication
- Tight coupling between services

=> Authorization is a cross-cutting concern

Decentralized
How to handle Authz? Centralized!

Prerequisite:
General rules applicable for every Microservice!
How to handle Authz?

+ One authz logic implementation
+ Change at one place
+ Explicit data sovereignty
+ No duplicated data
- Communication over network
- Single Point of Failure

Centralized
Whenever you encounter communication and implementation overhead leading to high coupling between the services the seam might be wrong.
Transformation process

Establish Microservices ecosystem
Microservices ecosystem

- CI/CD Pipeline
- Monitoring
- Log tracing
- Testing (incl. API)
- Central Configuration
- Design for Failure
- API-Gateway
- Service Discovery
- Load Balancing
- Dev Sandbox
Microservices ecosystem Tool examples f. Java

- Jenkins
- Prometheus & Grafana
- Spring Cloud Sleuth & Zipkin
- Spring Cloud Config
- Pact (CDC-Testing)
- Hystrix
- Zuul
- Eureka
- Ribbon
- Vagrant
Lessons learned

- Establishing Microservices ecosystem takes time and requires different skills & tools
- No explicit infrastructure team slows down the process
- Starting with decomposing big chunks frustrates
- Evaluate communication flow to identify wrong seams
- It takes far longer than originally anticipated
By starting small and decomposing in manageable steps and taking care of your ecosystem from the beginning the transformation process can be handled with even limited resources.
MADE IN
ST. PAULI
W/ LOVE
SWEAT & TEARS

*) Quarter of Hamburg, famous for its soccer club & entertainment district :)

... AND W/ MICROSERVICES!
THANK YOU!

Susanne Kaiser
CTO
@suksr

Just Software
@JustSocialApps