INNOQ Technology Day / 05.06.2024

Von Microservices zum Modulithen

Erfahrungen aus einem Experiment

INNOQ

TORSTEN MANDRY
Senior Consultant

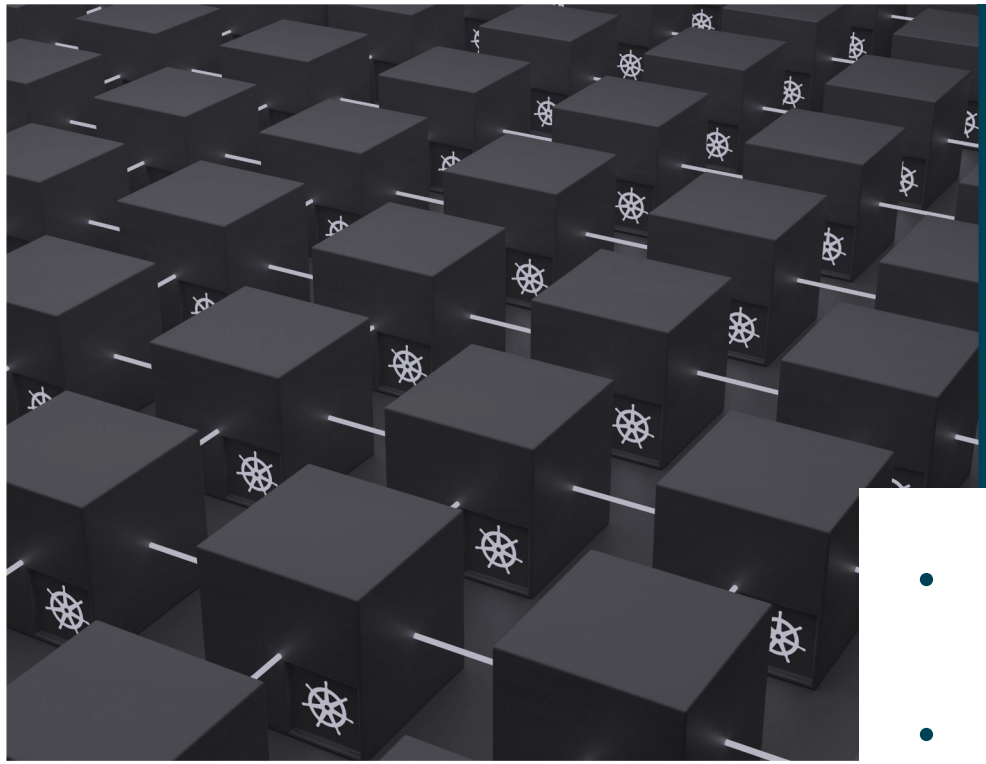
„Eine Lösung ist dann gut, wenn ich sie auch in ein paar Monaten noch verstehe“

Torsten Mandry
Senior Consultant @ INNOQ

Softwareentwickler, Architekt, Berater

Java, Web, Domain-driven Design, Clean Code, Clean Architecture, automatisiertes Testen


Von Microservices zum Modulithen?
Warum?



(Micro)services sind „der Standard“

- wenn mehrere Bounded Contexts
- starke Entkopplung
- unabhängige Entwicklung/Deployment

Foto von Growtika auf Unsplash

Microservices bringen Komplexität mit sich

- **Infrastruktur**
- **Deployment**
- **Schnittstellen**
- **„Microservice Premium“ (Fowler)**

https://martinfowler.com/bliki/MicroservicePremium.html

Foto von Ben Tofan auf Unsplash

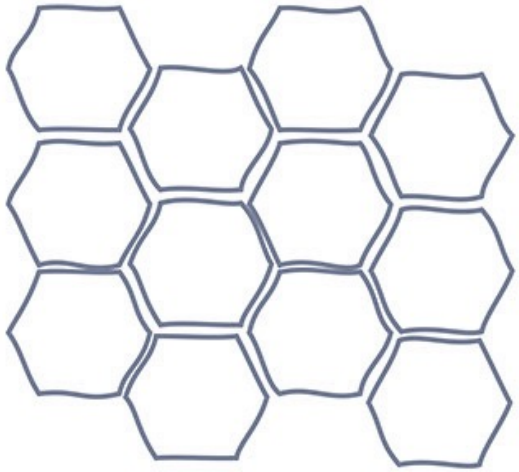
Häufige Frage: „Wäre mit einem Monolithen alles einfacher?“

Im Kontext **eines Entwicklungsteams,** welches **mehrere Microservices** betreut

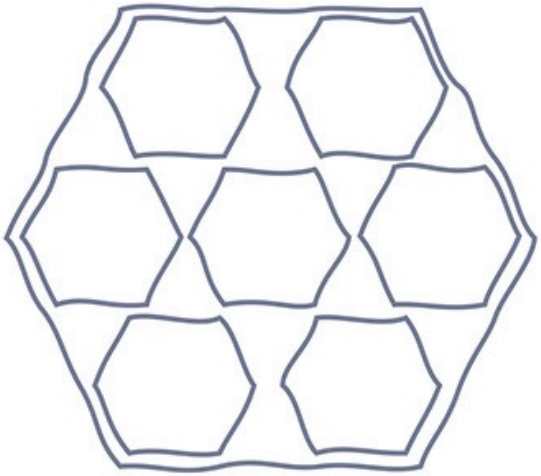
Foto von Dylan Gillis auf Unsplash

Modulith

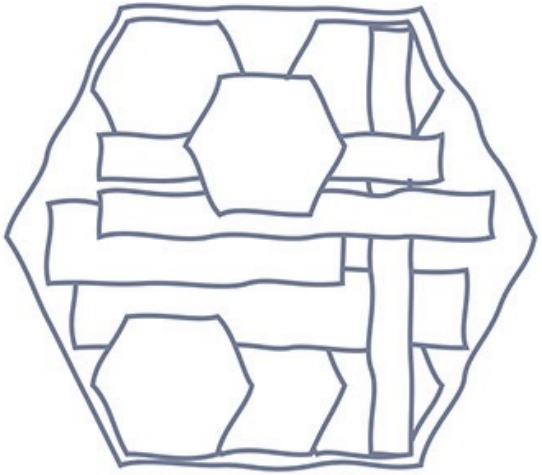
a. k. a. Deployment Monolith



Microservices-Architektur



Modulith-Architektur



Monolith-Architektur

Quelle: https://entwickler.de/software-architektur/microservices-oder-monolithen-beides

Wie würde ein solcher Modulith aussehen?

Persönliches Experiment

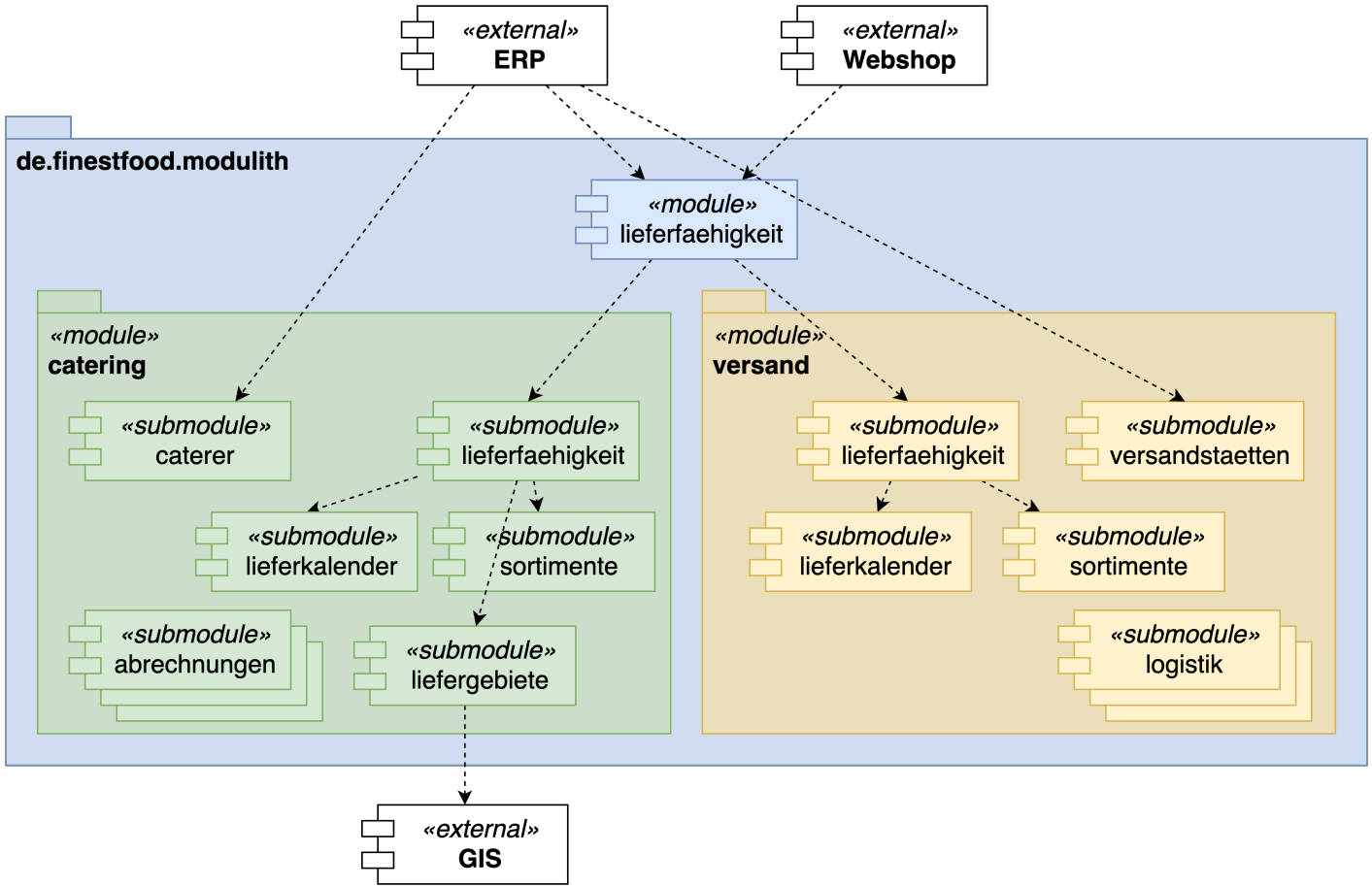
Foto von Ludovic Migneault auf Unsplash

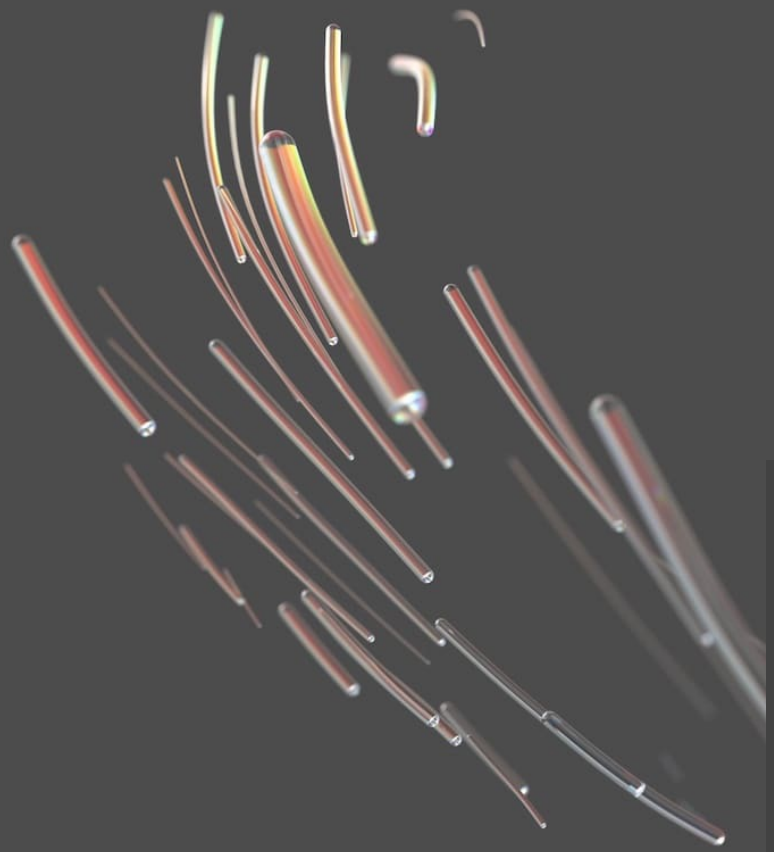
Zwei Prämissen

- **Module möglichst sauber getrennt**
- **Ansonsten so einfach wie möglich**

Foto von Dennis Zhang auf Unsplash

Kontext


«external»
ERP
«external»
Webshop
de.finestfood.modulith
«module»
lieferfaehigkeit
«module»
catering
«submodule»
caterer
«submodule»
lieferfaehigkeit
«submodule»
lieferkalender
«submodule»
sortimente
«submodule»
abrechnungen
«submodule»
liefergebiete
«module»
versand
«submodule»
lieferfaehigkeit
«submodule»
versandstaetten
«submodule»
lieferkalender
«submodule»
sortimente
«submodule»
logistik
«external»
GIS



Erwartete und unerwartete Fragen

Wie bilde ich Module ab?

Bei Microservices

- *separate Applikationen / Repositories*

Optionen für Modulith

- Java-Packages
- Maven-Module
- Java Module (JPMS)
- separate Libraries / Repositories

Beispiel Package Struktur

```
de.finestfood.modulith
      .catering
            .abrechnungen
            .caterer
            .lieferfaehigkeit
            ...
      . versand
            .lieferfaehigkeit
            .lieferkalender
            .logistik
            ...
```

Welche Module will ich haben?

Optionen

- Microservice = Modul
- weitere/kleinere Module
- ganz andere Module

Wie greifen die Module aufeinander zu?

Bei Microservices

- *explizite Schnittstellen (API, Messaging)*

Optionen für Modulith

- interne Domain/Komponenten
- definierte Interfaces

Beispiel Interface

```
public interface VerfuegbareLiefertage {


  List<VerfuegbarerLiefertag> findFor(
          Lieferadresse lieferadresse,
          LocalDate datumVon,
          LocalDate datumBis);


}
```

Wie schütze ich mich vor ungewollten Zugriffen?

Bei Microservices

- *versehentliche Abhängigkeiten eher unwahrscheinlich*

Optionen für Modulith

- Visibility
- 4-Augen-Prinzip (Pairing, Reviews)
- Validierung (ArchUnit, jQAssistant)
- Andere Form der Trennung (z. B. Java Platform Modules)

Spring Modulith

Spring Modulith allows developers to build well-structured Spring Boot applications and guides developers in finding and working with application modules driven by the domain. It supports the verification of such modular arrangements, integration testing individual modules, observing the application's behavior on the module level and creating documentation snippets based on the arrangement created.

https://spring.io/projects/spring-modulith

Beispiel: Definition Application Module

```
@org.springframework.modulith.ApplicationModule(
  allowedDependencies = {
    "de.finestfood.modulith.catering.lieferfaehigkeit",
    "de.finestfood.modulith.versand.lieferfaehigkeit",
    "de.finestfood.modulith.support::geocoding"
  })
package de.finestfood.modulith.lieferfaehigkeit;
```

/src/main/java/de/finestfood/modulith/lieferfaehigkeit/package-info.java

Welche Datentypen werden ausgetauscht?

Bei Microservices

- *i. d. R. JSON = generisches DTO*

Optionen für Modulith

- Domain Typen
  - Value Types
  - Aggregates / Entities
- explizite Schnittstellenobjekte (DTOs)

Beispiel Schnittstelle

Bei Microservices

```
GET /verfuegbarkeit/liefertage?
    strasse=…&hausnummer=…& … &
    datumVon=…&datumBis=…

{
  “verfuegbareLiefertage“: [
    {
      “datum“: “…“,
      “verfuegbarBis“: “…“
    }
    ,
  ]
}
```

im Modulith

```
public interface VerfuegbareLiefertage {
  List<VerfuegbarerLiefertag> findFor(
    Adresse adresse,
    LocalDate datumVon,
    LocalDate datumBis);
}

record Adresse(
    String strasse, String hausnummer, …) {}

record VerfuegbarerLiefertag(
    LocalDate datum, Instant verfuegbarBis) {}
```

Was mache ich mit Datentypen, die in mehreren Modulen verwendet werden?

Bei Microservices

- *jeweils eigene Repräsentation*

Optionen im Modulith

- Repräsentation pro Modul
- Definition/Bereitstellung in einem Modul
- Shared Commons

… und viele weitere Frage-stellungen

- Onion-Architektur
- Persistenz / Datenbank
- Externe Schnittstellen
  (z. B. GIS)
- Querschnittsthemen
  (z. B. Logging, Error Handling, …)
- …


Learnings

Modulith ist einfacher als Micro-services

- Komplexität der Service-2-Service Kommunikation entfällt
- Weniger Repositories, Deployments und Infrastruktur
- Direkte Abhängigkeiten erleichtern Entwicklung
  - Find Usages
  - Refactoring
  - …

Modulith ist aber immer noch komplex

- Modularisierung ist nicht trivial
- muss explizit adressiert werden
- andere/ungewohnte Lösungsansätze
- große Code-Basis wird unübersichtlich

Gewohnheit macht blind

- Dinge werden als gegeben hingenommen
  - ohne sie zu hinterfragen
  - unbewusst
- Zum Beispiel
  - JSON = generische DTOs
  - dedizierte Typen und deren Mapping

Gefühl: „Separate Services sind sauberer“

- für Haupt-Module
  - Catering
  - Versand
- Explizitere Trennung
  - Vielleicht nur Gewohnheit?
- Unterschiedliche Qualitätsanforderungen
  - Priorität
  - Performance
  - Verfügbarkeit

Bestehende Microservice-Systeme migrieren?

- Aufwand vs. Nutzen?
- Einschränkung bzgl. Skalierung
- Wenn Microservice-Schnitt OK

  → lohnt sich vermutlich nicht

Neue Projekte als Modulith starten!

- Anpassungen des Modulschnitts = einfaches Refactoring
- Overhead für Modularisierung vs. reduzierter Aufwand an Schnittstellen
- Separate Services erst dann extrahieren wenn notwendig

Torsten Mandry
torsten.mandry@innoq.com

INNOQ
www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180E
63067 Offenbach

Kreuzstr. 16
80331 München

Wendenstraße 130
20573 Hamburg

Spichernstrasse 44
50672 Köln