Praktikum Woche 1

Microservices

Msc. Informatik

06. April 2021

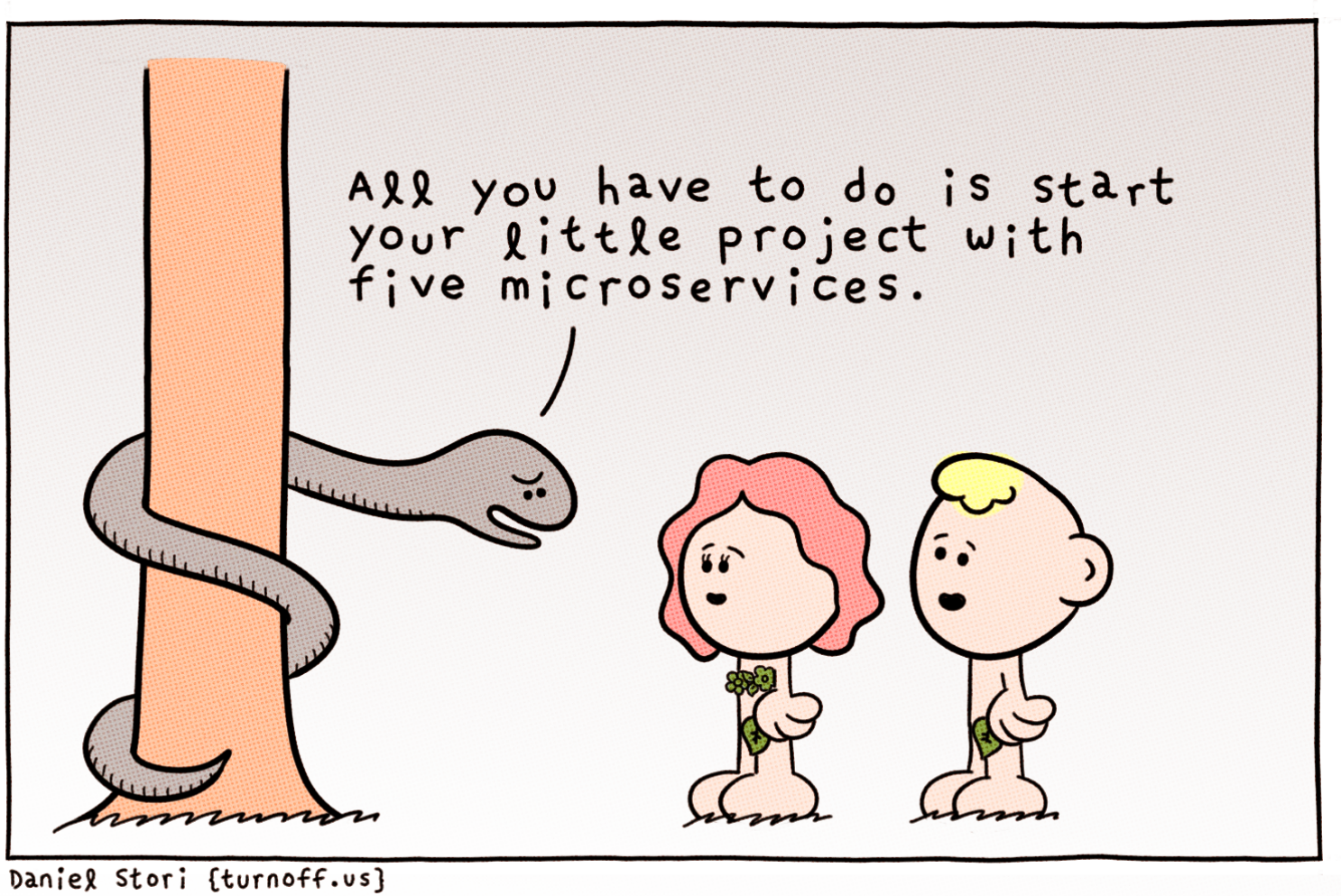

Abbildung: Starting Microservices

Entwicklung von Services

Bei Microservices bietet sich agile Entwicklung an:

- Fokus auf die Etappe, Blick auf's Weite
- Inkremente sind überschaubar
- Wichtig, schnell eine Basis zu schaffen

Generelle Tipps:

- Domänendiagramm gibt guten Überblick über das System
- Immer deployen und smoke testen
- Build Tools verwenden
- Manchmal "from scratch" testen
- namespaces nutzen

Entwicklung von Services

Die Qual der Wahl:

- Welche Sprache?
- Welches Framework?
- Welche zusätzlichen Libraries?

Worauf wichtig ist zu achten:

- Einfach testbar?
- Build Tools vorhanden und/oder ausgereift?
- Weite Verbreitung und hilsbereite Community?

Entwicklung von Services

Der Weitblick auf das, was alles gebraucht wird:

- REST Framework
- Servlet Container
- Anbindung an Message Broker (kafka)
- Anbindung an Datenbank(en)
- Remote steuerbar
- Möglichkeit, API zu veröffentlichen

Von der Idee zu den Services

Am wichtigsten ist die Anforderungsanalyse

- Aus Use Cases folgt die Beschreibung der Domäne
- aus dem Domain Model können Bounded Contexts hergeleitet werden
- Microservices teilen dabei das Domain Model auf
- Interaktionen im Domain Model sind Events und Nachrichten

Service Scope eingrenzen

Wann hat ein Service einen guten Umfang?

- Wenn er Teil des Unternehmens/Geschäfts repräsentiert
- Wenn er Informationen, die er braucht, über einen Service-Hop bekommt
- Wenn er in kurzem Zeitraum entwickelt werden kann
- Wenn er nicht für jeden Use Case des Systems gebraucht wird

Das Projekt

Der Grundriss

Bei der zu entwerfenden Software handelt es sich um ein kleines Projektverwaltungssystem:

- Verwaltung von:
  - Projekten
  - Issues (Aufgaben)
  - Nutzern
- Nutzer sind Projekten zugeordnet
- Es gibt eine Verwaltung von Neuigkeiten

Rollensystem

Um die Anforderungen zumindest etwas einfacher zu gestalten, gibt es ein einfaches Rollensystem:

- Admin-Accounts können neue Nutzer anlegen
- Nur eingeloggte Nutzer können auf das Backend zugreifen
- Jeder angelegte Nutzer kann Projekte erstellen
- Projektersteller können Nutzer zu den Projekten hinzufügen
- Nutzer können in diesen Projekten Issues erstellen

Neuigkeiten

Es soll für jeden Nutzer eine "*News*"-Seite geben:

- Quasi eine Startseite nach dem Login
- Informationen zu Neuigkeiten im System:
  - Neue Issues in den Projekten des Nutzers
  - Neue Zugehörigkeiten zu Projekten
  - etc.

Bewusst recht weiche Anforderungen

Es soll die Möglichkeit bestehen, das Projekt auch nach eigenen Vorstellungen zu gestalten.

Wenn es also Ideen, Vorschläge oder Rückfragen gibt, können diese gerne besprochen werden.
Es ist zudem auch möglich, einen eigenen Vorschlag für ein System zu bringen. Das muss allerdings mit mir gut abgesprochen sein, sodass gewisse Grundanforderungen geklärt sind:

- Menge an benötigten Services
- Autorisierung
- Aggregate von Daten (entspricht den News)

Eigene Ideen müssen spätestens im nächsten Termin besprochen sein.

Arbeitsauftrag

Es soll ein Domain-Diagram für die Anwendung erstellt werden.

Hierbei sollen die einzelnen Entities aufgelistet und in Beziehung zueinander gebracht werden.
Ebenfalls sollen *Bounded Contexts* herauskristallisiert und kenntlich gemacht werden.

Exkurs in Domain-Driven Design

Ubiquitäre Sprache

> *Ubiquitous Language is the term Eric Evans uses in Domain Driven Design for the practice of building up a common, rigorous language between developers and users. This language should be based on the Domain Model used in the software - hence the need for it to be rigorous, since software doesn't cope well with ambiguity.*[1]
>
> – Martin Fowler

Es geht bei der ubiquitären Sprache um eine Optimierung der Kommunikation in allen Bereichen des Projekts. Mit dem **Domänenmodell** angefangen entsteht ein Glossar, der die Begrifflichkeiten definiert.

[1] https://martinfowler.com/bliki/UbiquitousLanguage.html

Der Designansatz (Auszug)

Visionsdokument (engl.: "*vision document*")

- Kurze Beschreibung der Vision bzw. Ziele des Projektes
- Bindeglied zwischen Auftraggeber und Entwicklerteam

Kontextübersicht (engl.: "*context map*")

- Übersicht über alle (Teil-)Modelle
- Darstellung der Grenzen einzelner Teile und der Kommunikation zwischen diesen

Kernfachlichkeit (engl.: "*core domain*")

- Übersicht über alle (Teil-)Modelle
- Darstellung der Grenzen einzelner Teile und der Kommunikation zwischen diesen

Bestandteile (Auszug)

Entität(en) (engl.: "*entities*")

- Objekte des Domänenmodells, definiert durch ihre Identität

Wertobjekt (engl.: "*value object*")

- Meist unveränderliche Objekte, definiert durch deren Eigenschaften

Aggregate (engl.: "*aggregates*")

- Zusammenschluss von u. A. Entitäten und / oder Wertobjekten

Assoziationen (engl.: "*associations*")

- Beziehungen zwischen Objekten des Domänenmodells

Fachliche Ereignisse (engl.: "*domain events*")

- In Objekten festgehaltene Geschehnisse, die Veränderungen von Objekten beschreiben können

Repositorien (engl.: "*repositories*")

- Abstrahierte Persistenzschicht der Objekte