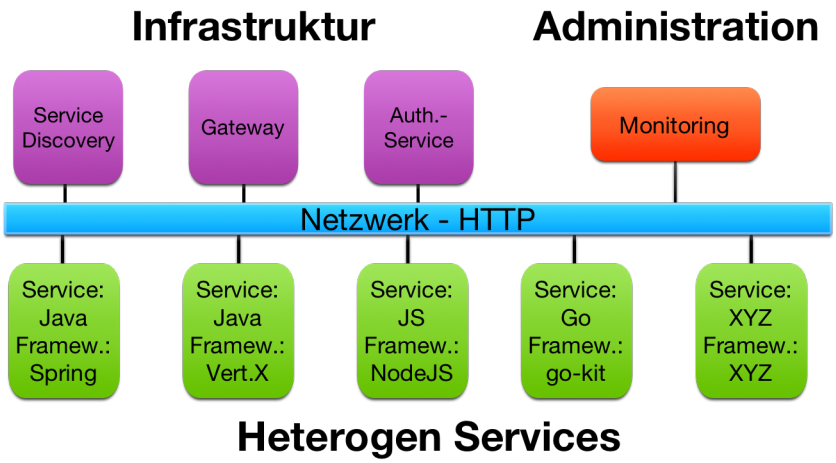
Proposal: Polyglotte und heterogene Microservices

René Zarwel

Bei der Microservice Architektur wird komplexe Anwendungssoftware mit kleinen, unabhängigen Services komponiert, welche über eine sprachunabhängige Programmierschnittstelle (z.B. Http/RESTful) kommunizieren. Diese Services sind klein, weitgehend entkoppelt und erledigen eine kleine Aufgabe. Dadurch wird ein modularer Aufbau ermöglicht und setzt innerhalb der Services den Fokus auf eine kleine leichter zu lösende Teilaufgabe. So kann sich das Entwicklerteam eines Services bei der Wahl der Programmiersprache und dem Framework an den eigenen Fähigkeiten und der zu lösenden Teilaufgabe konzentrieren. Durch diese freie Entwicklung kann der Service komplett unabhängig von den anderen Services in die Produktion eingebracht werden. Doch ist die Technologie-Entscheidung wirklich komplett unabhängig? Und passen bestimmte Frameworks besser auf eine gegebene Aufgabe als andere, obwohl sie die gleichen Funktionen bieten?

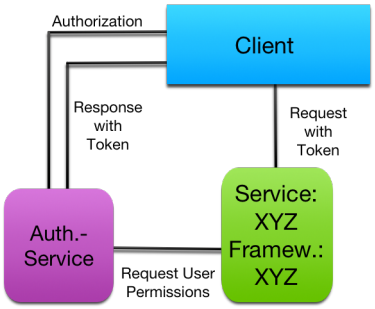
In der Praxis ergeben sich Anforderungen an die verwendeten Sprachen und Frameworks, um ein effizientes gemeinsames Ganzes schaffen zu können. Zum einen muss sich der Service in eine bestehende Infrastruktur und Platform integrieren lassen. Cloud Foundry ist dabei der weit verbreitetste Standard für Platform-as-a-Service. Um eine neue Funktion anbieten zu können, muss sich der Service auf dieser Platform einbringen lassen. So erhält er auch Zugriff auf wichtige Infrastruktur Elemente, wie der Service Registry. Dies ist essentiell in einer Microservice Architektur, da sich die Services dynamisch anpassen und der Client trotzdem ständig Zugriff auf die Funktionen haben muss.



Des Weiteren sollten die verwendeten Frameworks auch Schnittstellen für Monitoring und Metriken mitbringen. Gerade bei vielen Services kann es schnell unübersichtlich werden und ein manuelles Monitoring jedes einzelnen Services wird schier unmöglich. So müssen zentral und automatisch in einem Service alle Status-Informationen aller Services gesammelt werden können, um z.B. Fehler zu erkennen oder dynamische Anpassungen durchzuführen.

Neben den Anforderungen an die Integration eines Services müssen auch Kriterien der Sicherheit erfüllt werden. Gerade Services mit Anbindung an eine Datenbank besitzen meist zu schützende Inhalte. So muss eine Authentifizierung und Autorisierung am Service eingerichtet werden können, um den Service

vor Missbrauch schützen zu können. Damit sich nicht jeder Service eigenständig um eine Nutzer- und Rechteverwaltung kümmern muss, hat sich eine zentrale Lösung bewährt. Der RFC Standard OAuth2 bietet hierfür die beste Grundlage. Die API eines Services darf somit nur von Clients mit gültigem Token verwendet werden können. Und entsprechende Rechte, die z.B. an der Rechteverwaltung abgefragt werden, sollten den Zugriff limitieren.



Sollte mehrere Sprachen und Frameworks diese Anforderungen erfüllen, kann trotzdem ein Kandidat geeigneter für die Umsetzung sein als ein Anderer. So kann der Overhead eines Frameworks massiv die Performance des Services beeinträchtigen. Aber auch die Wartbarkeit spielt ein wesentliche Rolle. Eine Sprache in Kombination mit einem bestimmten Framework kann für die Umsetzung einer Aufgabe, z.B. aufgrund fehlender oder schwer zu konfigurierender Funktionen, schnell mehr als doppelt so viele Programmartefakte hervorbringen als eine andere. In Zeiten von schnellen Release Zyklen und Flexibilität spielt dies immer mehr eine Rolle.

Im Rahmen dieser Bachelor Arbeit sollen eine Auswahl an aktuellen Frameworks mit den zugehörigen Sprachen auf die Integrations- und Serviceanforderungen untersucht werden. Da nicht jeder Microservice die gleichen Aufgabentypen umsetzt, soll zuerst eine Kategorisierung der Services gefunden werden, um den Anforderungen Gewichte zuzuordnen. Um somit eine Empfehlung für den Einsatz der Sprachen und Frameworks aussprechen zu können.