Verteilte Systeme Master Labor

Sommer 2020 — Aufgabenstellung (Version 02.04.2020)

Im Zuge der **Gesamtaufgabe**, die gegebene monolithische Web-Anwendung nach dem Microservice Architekturstil umzustrukturieren, sind **vier** verschiedene **Teilaufgaben** zu erbringen, deren Ziele und Anforderungen im Folgenden beschrieben werden.

Alle Aufgaben sind mit **terminierten Projektphasen** und einem angestrebten Ergebnis (**Deliverable**) verbunden. Die Aufgaben können (und sollen) in **Gruppen** bearbeitet werden.

Die **Einreichung** von Deliverables erfolgt pro Gruppe zum Ende der Aufgabenphase. **Termine** sind bindend mit **Anwesenheit** und Beitrag aller Gruppenmitglieder. Die **Form** der Einreichung hängt von den Deliverables ab und ist in den Aufgaben beschrieben.

Aufgabe 1 — Analyse der Legacy Anwendung

Abgabe: 29.04. (zusammen mit Aufgabe 2)

In der ersten Aufgabe soll der existierende Web Shop installiert, getestet und systematisch analysiert werden.

Teilaufgaben

Aufgabe 1.1 Clonen Sie das Github Projekt des Web Shop (https://github.com/mavogel/hska-vis-legacy/) und erzeugen Sie ein lauffähiges System.

Aufgabe 1.2 Starten Sie den Web Shop und testen Sie seine Funktionalität.

Aufgabe 1.3 Analysieren sie den Sourcecode des Web Shops und dokumentieren sie das Ergebnis in einem geeigneten Modell. Die Art der Modellierung ist weitgehend Ihnen überlassen, es sollten jedoch Struktur und Verhalten des Systems wiedergegeben werden. Typischerweise erfolgt dies über geeignete UML Teildiagramme wie z.B. Klassen- und Aktivitätsdiagramme. Alternativ können Sie je nach Vorwissen und Vorliebe auch eine andere Art der Modellierung wählen (ggf. Absprache) und durch Architekturskizzen und Text ergänzen.

Deliverable

Am Ende der Analysephase soll ein (kurzes) **Dokument** (3-5 Seiten) mit Modellartefakten zur Dokumentation des Web Shops und seiner momentanen Implementierung vorliegen. Bitte laden Sie eine Kopie des PDF im ILIAS hoch.

Das **Deckblatt** soll die Namen der **Gruppenmitglieder** mit Matrikelnummer und Email enthalten. Die Einreichung erfolgt pro Gruppe mit einer kurzen **Vorstellung des Ergebnisses**.

Aufgabe 2 — Microservice Architekturentwurf

Abgabe: 29.04. (zusammen mit Aufgabe 1)

In der zweiten Aufgabe sollen **Entwürfe** einer Microservice-basierten **Zielarchitektur** und einer entsprechenden **REST API** erstellt werden.

Teilaufgaben

Aufgabe 2.1 Erstellen Sie auf Basis der Analyse aus Aufgabe 1 einen alternativen Entwurf der Web Shop Architektur.

(a) Identifizieren Sie zunächst Funktionen und Datenstrukturen, die sich zur Bereitstellung durch Services eignen und gruppieren Sie diese zu autonomen generischen **Core-Services**.
(b) Untersuchen Sie den Einsatz generischer Fassaden zur kombinierten Nutzung von Core Services und erstellen Sie mindestens einen **Composite Service**.
(c) Entwerfen sie Nun **API-Services**, die für die eShop Web Anwendung optimiert sind. API-Services sollen als Adapter dienen und den eShop von generischen Microservices entkoppeln.

Aufgabe 2.2 Konkretisieren Sie alle in Aufgabe 2.1 gefundenen API-Services als REST-basierte Schnittstellen und spezifizieren Sie diese mithilfe einer API Sprache Ihrer Wahl.

Deliverable

Deliverable 2.1 Der alternative Architekturentwurf soll zu einer Menge von Core-, Composite- und API-Services führen. Hierzu soll eine Architekturskizze der Service Komponenten und Abhängigkeiten entstehen. Optional können Modelle des internen Aufbaus von Core Service Komponenten aus den Modellen der Aufgabe 1 abgeleitet werden.

Deliverable 2.2 Der Entwurf der REST APIs soll zu einer API Spezifikation in einer der vorgestellten Spezifikationssprachen (Swagger, RAML u.a.) führen.

Beide Teile sollen zu einem **Dokument** zusammengefasst werden (für die API ist eine geeignete Repräsentation je nach verwendeter Spezifikationstechnik zu wählen). Das **Deckblatt** soll die Namen der **Gruppenmitglieder** mit Matrikelnummer und Email enthalten. Bitte laden Sie eine Kopie des **PDF** und die **Sourcen** der API-Spezifikation als **Datei** im ILIAS hoch. Die Einreichung erfolgt pro Gruppe mit einer kurzen **Vorstellung des Ergebnisses.**

Aufgabe 3 — Microservice Implementierung

Abgabe: 27.05.

In der dritten Aufgabe soll die Microservice-basierte Architektur des Web Shop Backends mit dem Spring Framework umgesetzt werden. Das Ziel ist zunächst die **Entwicklung aller Microservices** und deren **Integration** mit grundlegenden **Netflix OSS Middleware Services**.

Teilaufgaben

Aufgabe 3.1 Zur Implementierung der in Aufgabe 2.2 spezifizierten REST-Schnittstellen soll pro Microservice ein Spring Boot Projekt entstehen, mit dem jeder Service als eigenständiger Prozess gestartet und zugegriffen werden kann.

(a) Core Microservices können auf den entsprechenden Komponenten der monolithischen Anwendung aufbauen oder neu implementiert werden (was empfehlenswert ist).
(b) Composite Microservices sollen als REST-Clients der Core Microservices realisiert werden. Die Bindung kann (zunächst noch) statisch erfolgen, indem die URLs der Microservices fest kodiert werden. Je nach gewählter Architektur können optional auch andere Formen der Kommunikation gewählt werden (z.B. Messaging/Event-Mechanismen).
(c) Geteilte Komponenten können in einem eigenständigen Projekt realisiert werden, dass dann als Abhängigkeit in die Microservices eingebunden wird.

Aufgabe 3.2 Die Web Shop Microservices sollen nun auf Basis von Netflix OSS Middleware Mechanismen wie folgt integriert werden:

(a) Als Basis der Infrastruktur soll ein Eureka Server als Spring Boot Anwendung realisiert werden. Alle implementierten Microservices sollen so erweitert werden, dass sie sich als Discovery Clients registrieren.
(b) Composite Microservices sollen zu Ribbon-Clients erweitert werden, die die verwendeten Core Microservices dynamisch Binden. Der Zugriff auf die Core Microservices soll per Hystrix gesichert werden.
(c) Es soll ein Zuul Edge Server als Spring Boot Anwendung realisiert werden. Der Edge Server soll eine öffentliche API bereitstellen und Client Aufrufe auf die implementierten Core- und Composite Microservices weiterleiten.
(d) Es soll ein Hystrix Dashboard als Spring Boot Anwendung (oder als Teil einer geeigneten anderen Komponente) realisiert werden. Auf dem Dashboard sollen die Aktivitäten des Edge Servers und der Hystrix-gesicherten Komponenten beobachtet werden.

Aufgabe 3.3 Alle eigenständigen Spring Boot Prozesse sollen in dedizierten **Docker Containern** bereitgestellt werden (sowohl Microservices als auch Middleware Services). Die Container sollen mittels **Docker Compose** in Beziehung gesetzt und gemeinsam gesteuert werden.

Deliverables

Erstellen Sie ein Dokument mit einer Skizze der implementierten Systemarchitektur (sowohl Microservices als auch Middleware Services). Das **Deckblatt** soll die Namen der **Gruppenmitglieder** mit Matrikelnummer und Email enthalten. Bitte laden Sie eine Kopie des **PDF** und die **Sourcen** der Projekte als **Datei** im ILIAS hoch.

Die Einreichung erfolgt pro Gruppe mit einer kurzen **Demonstration des Ergebnisses.** Dabei ist der **Zugriff auf einzelne Microservices** zu zeigen. Zudem sollen die **Funktionen der Netflix Komponenten** demonstriert werden (Routing, Monitoring, Load Balancing, Circuit Breaking).

Aufgabe 4 — Web Shop Implementierung

Abgabe: 01.07.

In der vierten und letzten Aufgabe soll die Struts-Anwendung des **Web Shops als REST-Client** an den in Aufgabe 3 erstellten Edge Server angebunden werden. Dabei soll eine **Autorisierung der Web Shop Anwendung mittels OAuth** vorgenommen werden. Konkret soll eine Lösung auf Basis von OAuth 2.0 realisiert werden.

Teilaufgaben

Aufgabe 4.1 Implementieren Sie einen *Authorisierungsserver* mit OAuth2 und nutzen z.B. den *ProductService* als *Resourceserver*. Testen Sie den Ablauf (Anfordern eines Tokens und Auslesen der Resource) mit Hilfe eines generischen HTTP Clients wie curl oder Postman. Binden Sie den *Edge-server* ein, d.h. alle Anfragen gehen über den Edge-Server.

Aufgabe 4.2 Der Login soll so angepasst werden, dass damit eine Autorisierung der Web Anwendung gegenüber der Microservice Infrastruktur erfolgt. Schreiben Sie die *LoginAction* des Webshops so um, dass der *Authorisierungsserver* bzw. *Userservice* mittels eines Restaufrufs benutzt werden kann und testen Sie den Zugriff.

Aufgabe 4.3 Passen Sie die Restaufrufe im Webshop an die Anforderungen für den Aufruf mit OAuthRestTemplate an.

Aufgabe 4.4 Als letzter Teil ist noch das Registrieren umzusetzen.

Deliverables

Erstellen Sie ein Dokument mit einer kurzen Beschreibung des implementierten OAuth-Mechanismus und einer Skizze von dessen Komponenten. Das **Deckblatt** soll die Namen der **Gruppenmitglieder** mit Matrikelnummer und Email enthalten. Bitte laden Sie eine Kopie des **PDF** und die **Sourcen** der Projekte als **Datei** im ILIAS hoch.

Die Einreichung erfolgt pro Gruppe mit einer kurzen **Demonstration der Ergebnisse.** Dabei sollen die **Funktionen des Web Shops** und insbesondere dessen **Autorisierung** gezeigt werden (es sollte demonstriert werden, dass der Zugriff eines nicht autorisierten Clients fehlschlägt).