Verteilte Softwaresysteme

Blatt 4

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 19.11.2019 07:39

Abgabe bis 13.01.2020, 08:00 per Pull-Request gegen den `master`-Branch.

Aufgabe — Multiplex-Kino

Die fristgerechte Abgabe einer Lösung für diese Aufgabe ist notwendige Voraussetzung für den Leistungsnachweis.

Diese Aufgabe können Sie wieder alleine oder zu zweit bearbeiten und abgeben. Über den GitHub-Classroom-Link https://classroom.github.com/g/tL91ST75 bekommen Sie ein leeres Repository und die Teams. Nachdem Sie auch in anderer Zusammensetzung als auf Blatt 3 arbeiten können, müssen Sie für Blatt 4 wieder ein neues Team erstellen.

Wenn Sie alleine arbeiten, dann gehen Sie jetzt nach dem Git Flow oder nach dem GitHub-Flow vor, zu zweit nach dem Git-Flow.

Die Starterrepositories sind leer. Binden Sie sie selbst, wie Sie es von den anderen Blättern kennen, an meinen Jenkins an.

Entwerfen Sie eine Verwaltung für ein Multiplex-Kino als verteilte Anwendung und Implementieren Sie einen Prototypen als *Proof of Concept*. Mit dem Proof of Concept sollen Sie im Wesentlichen zeigen, dass das Zusammenspiel der Services funktioniert. Es ist nicht notwendig als *Fleißarbeit* alles zu implementieren. Im Zweifelsfall fragen Sie.

Befüllen Sie Ihre Services mit konsistenten Beispieldaten:

- Mindestens 2 Kinosäle, die in Vorstellungen genutzt werden.
- Mindestens 4 Benutzer.
- Mindestens 4 Filme, die in Vorstellungen gezeigt werden.

- Mindestens 4 Vorstellungen.
- Mindestens 4 Reservierungen.

Stellen Sie mit Ihrer gesamten Anwendung eine "eventual consistency" sicher. Die Services haben ja immer nur IDs von den Daten anderer Services, analog zu Fremdschlüsseln in relationalen Datenbanksystemen. Wird beispielsweise ein Kinosaal gelöscht, weil er z.B. wegen einem Wasserschaden nicht benutzbar ist, müssen alle Vorstellungen darin gelöscht werden. Wenn eine Vorstellung gelöscht wird, müssen alle Reservierungen für diese Vorstellung gelöscht werden.

Setzen Sie mindestens folgende Szenarien um und führen Sie sie vor (bereiten Sie dafür ein Client-Programm, ein Shellscript oder ähnliches vor):

- Ein Kinosaal in dem Vorstellungen geplant sind, für die es Reservierungen gibt, wird gelöscht.
- Zwei Benutzer haben überlappende Reservierungsanfragen, die zunächst jede für sich möglich sind, und beide wollen die Reservierung durchführen (aber nur einer gewinnt ;-)).

Anmerkung: Die Konsistenz müssen die Services herstellen, nicht der Client!

Vorgaben

Die Verwaltung selbst besteht mindestens aus Services für die

- Kinosaalverwaltung
- Benutzerverwaltung
- Filmverwaltung
- Vorstellungsverwaltung
- Reservierungsverwaltung

Die Services werden durch Microservices unter Verwendung von Go Micro implementiert. Jeder Service ist ein eigenes ausführbares Programm und wird (über den Jenkins) als Docker-Container zur Verfügung gestellt.

Bestandteile der verwalteten Daten

- Kinosäle bestehen mindestens aus einem Namen, z.B. "Kino 1", und einer Repräsentation der vorhanden Sitzplätze. Sie können der Einfachheit halber rechteckige Kinosäle nutzen, d.h. durch eine Anzahl von Reihen und einer Anzahl von Sitzen pro Reihe ergeben sich alle Sitze. Kinosäle können jederzeit erzeugt und gelöscht werden.
- Benutzer haben mindestens einen Namen und eine *Verknüpfung* zu ihren Reservierungen, z.B. über IDs. Benutzer können nur gelöscht werden, wenn sie keine Reservierung im System haben.

- Filme haben mindestens einen Titel. Filme können jederzeit erzeugt und gelöscht werden.
- Eine Vorstellung verknüpft einen Film mit einem Kinosaal. Sie brauchen für den Prototyp keine Vorstellungstermine festzulegen. Vorstellungen können jederzeit erzeugt und gelöscht werden.
- Eine Reservierung verknüpft einen Benutzer mit einer Vorstellung. Eine Reservierung kann gleich mehrere Sitzplätze enthalten. Ein Reservierungsprozess erfolgt immer in zwei unabhängigen Schritten:

  1. Eine Reservierungsanfrage mit allen Infos, die als Ergebnis zurück liefert ob die Reservierung durchführbar ist, d.h. ob alle Sitzplätze frei bzw. überhaupt vorhanden sind. Ist sie durchführbar, so wird sie unter einer ID schon im Service gespeichert, aber noch nicht reserviert.
  2. Stimmt der User der Reservierung zu, wird sie durch einen erneuten Servicecall mit der ID tatsächlich durchgeführt.

Abzugebende Bestandteile

- Go-Code as usual.
- Anbindung an den Jenkins.
  - Ausführen der Unit-Tests auf dem Jenkins.
  - Erzeugen der Docker-Images auf dem Jenkins.
- Beschreibung der Kommunikation zwischen den Services in der Datei `Protocol.md` top-level im Repository.
- Anleitung wie die gesamte Anwendung zu starten ist und wie die Szenarien zu starten sind in der `README.md` top-level im Repository.
- `docker-compose`-Datei zum Starten der gesamten Anwendung.

Abnahme

Nach der Abgabe als Pull-Request erfolgt die Abnahme im Rahmen des Praktikums durch eine kurze Vorführung und Erklärung. Bereiten Sie dazu mindestens die beiden oben angegebenen Szenarien als Programme oder Scripts vor.