Verteilte Systeme: Portfolio-Aufgaben

Studienjahr 2023 / 2024

Wichtige Hinweise

Umfang des Portfolios

Die Portfolioprüfung beinhaltet die beiden Vorlesungen „Webprogrammierung“ und „Verteilte Systeme“ aus dem dritten und vierten Semester. Das Prüfungsportfolio setzt sich deshalb aus folgenden Bestandteilen zusammen, die entsprechend der Anzahl Vorlesungsstunden wie folgt verrechnet werden:

Artefakt	Gewichtung	Bearbeitung
3. Semester: Webprogrammierung		
Lernkontrollen in Moodle	2	Einzelleistung
Aufgabenblatt mit Programmierprojekt	4	Einzelleistung
4. Semester: Verteilte Systeme		
Leistungsnachweis der Übungsaufgaben	1	Einzelleistung
Aufgabenblatt mit Einzelaufgaben	3	Einzelleistung

Bearbeitung in Eigenleistung

Alle Aufgaben sind in Eigenleistung zu bearbeiten. Gruppenarbeit ist nicht zulässig. Die Lösungen müssen soweit wie möglich einen individuellen Lösungsweg erkennen lassen. Zwar ist es durchaus erwünscht, das Internet und andere Quellen als Hilfestellung zu nutzen. Die Inhalte müssen aber dennoch selbst ausgearbeitet werden. Abschreiben von Kommilitonen (auch mit Umformulierungen) gilt als Täuschungsversuch!

Fremde Quellen und KI-Sprachassistenten

Fremde Quellen müssen durch Zitate gekennzeichnet werden. Einfache Fußnoten mit Quellenangabe (z.B. der URL einer Webseite) reichen dafür aus. Ein Literaturverzeichnis muss nicht erstellt werden und Webseiten müssen auch nicht gesichert werden.

Eine besondere Regel gilt jedoch für KI-Sprachassistenten wie ChatGPT: Richtig eingesetzt können sie sehr nützlich sein, weshalb die Verwendung selbstverständlich erlaubt ist. Es muss allerdings der komplette Chatverlauf exportiert und abgegeben werden. In der Fußnote kann dann zum Beispiel mit „ChatGPT zu Responsive Webdesign (s. Anhang)“ auf das angehängte Dokument verwiesen werden.

Maximal 25% Quellcode dürfen von fremden Quellen übernommen oder mit KI-Assistenten generiert werden. Nicht selbst geschriebener Quellcode muss durch Kommentare klar gekennzeichnet werden, in denen der verwendete KI-Assistent und Prompt bzw. die URL der Quelle enthalten ist.

Abgabe

Die Abgabe erfolgt vor Beginn der Klausurphase per Upload einer ZIP-Datei in Moodle. Quellcodes müssen, auch wenn Sie online verfügbar sind, ebenfalls in Moodle hochgeladen werden. Denken Sie daran, das Verzeichnis `node_modules` vor dem Erstellen der ZIP-Datei zu löschen!

Textdokumente und Office-Dateien müssen im PDF-Format abgegeben werden. Gehen Sie bitte nicht davon aus, dass die Lehrenden dieselben Softwarepakete wie Sie installiert haben.

Aufgabenstellung

Aufgabe 1: Architektur verteilter Systeme (5 + 5 + 5 + 5 = 20 Punkte)

Im Vorlesungsskript wurde der Artikel „Your Coffee Shop Doesn't Use Two-Phase Commit“[1] vorgestellt, auf welchen sich die folgende Aufgabe bezieht.

a) Zeichnen Sie ein UML-Sequenzdiagramm[2], das von der Bestellung bis zum Erhalt eines Kaffees alle Interaktionen zwischen Kund*in, Kassierer*in und Barista zeigt. Kennzeichnen Sie dabei die Nachrichten entsprechend der UML-Notation, ob es sich um synchrone oder asynchrone Nachrichten handelt. Beantworten Sie anschließend folgende Fragen anhand des Diagramms:

- Welche Teile des Prozesses laufen synchron und welche asynchron?
- Warum ist der Prozess aus Kundensicht halb-synchron?
- Werden die Kaffees immer in der Reihenfolge ihrer Bestellung gemacht?
- Wenn nein, wie wird sichergestellt, dass jede/r Kund*in den richtigen Kaffee erhält?
- Und wie wird sichergestellt, dass jeder Kaffee vor der Abholung bezahlt wurde?

b) Zeichnen Sie ein UML-Sequenzdiagramm, bei dem alle Schritte von der Bestellung bis zum Erhalt des Kaffees nach dem synchronen Request/Repsonse-Verfahren ablaufen und zeigen Sie daran, warum bei diesem Verfahren weniger Kund*innen bedient können und somit weniger Umsatz erzielt werden kann.

c) In der halb-synchronen Variante können die Kassierer*innen und Baristas unabhängig voneinander horizontal skaliert werden. Warum ist das so und warum ist das vorteilhaft? Beschreiben Sie ein Szenario, bei dem eine ungleiche Anzahl an Kassierer*innen und Baristas sinnvoll ist.

d) In der Überschrift wird das Zwei-Phasen-Commit-Protokoll genannt, das ebenfalls eine synchrone Kommunikation zwischen allen Teilnehmer*innen vorsieht. Im Artikel selbst wird das Protokoll jedoch gar nicht behandelt. Beantworten Sie dennoch folgende Fragen dazu:

- Wie müsste man sich dessen Einsatz in einem Coffee Shop grob vorstellen (keine Details)?
- Warum handelt es sich bei dem Beispiel stattdessen um ein optimistisches Transaktionsmodell?

Optimistische Transaktionen laufen häufiger auf Fehler, die typischerweise mit einem „Write-Off“, „Retry“ oder einer „Compensating Action“ behandelt werden. Erklären Sie die Begriffe anhand selbstgewählter Beispiele aus Ihrem beruflichen oder privaten Alltag.

Aufgabe 2: REST-Webservices (5 + 5 + 10 = 20 Punkte)

a) Entwerfen Sie das Datenmodell für einen einfachen Microservice mit drei voneinander abhängigen Datenbankentitäten. Als Vorlage können Sie sich dabei an der Aufgabe „Wir eröffnen ein Kino“ aus dem Vorlesungsskript orientieren. Beschreiben Sie zunächst den Anwendungsfall in wenigen Worten und dokumentieren Sie das Datenmodell in einer geeigneten Form beispielsweise als E/R-Diagramm.

b) Zeigen Sie nun, wie die Datenbankentitäten als Collections und Ressourcen auf einen REST-Webservice abgebildet werden können. Dokumentieren Sie hierfür die Endpunkte in tabellarischer Form, wobei zu jedem Endpunkt die URL, HTTP-Verb, die ausgelöste Aktion sowie das JSON-Format der Anfrage/Antwort beschrieben werden sollten. Achten Sie darauf, dass die REST-Prinzipien vollständig eingehalten werden. Als Vorlage können Sie sich hierfür an der Aufgabe „Kleiderkreisel“ orientieren.

c) Programmieren Sie den eben beschriebenen Webservice, wie in der Vorlesung gezeigt, mit Node.js, **`Express`**[3] und einer Bibliothek wie **`lowdb`**[4] zur Speicherung der Daten.

1 Vgl. IEEE Software, Volume: 22, Issue: 2, Pages 64–66. Der Artikel kann auf der Webseite online gelesen werden.
2 Vgl. https://www.ionos.de/digitalguide/websites/web-entwicklung/sequenzdiagramme-mit-uml-erstellen/ zur Notation
3 http://expressjs.com/
4 https://www.npmjs.com/package/lowdb

Aufgabe 3: Asynchroner Nachrichtenaustausch (15 + 5 = 20 Punkte)

a) Schreiben Sie zwei Node.js-Programme, die beide die Bibliothek **`kafkajs`**[5] verwenden, um im JSON-Format kodierte Datensätze über den Kafka-Server **`zimolong.eu:9092`** auszutauschen. Den Aufbau des Datensatzes können Sie selbst wählen; er muss aber mindestens zwei Felder enthalten, von denen eines den Typ **`number`** haben muss. Beispiele könnten sein:

- **Bücher:** ISBN (Zahl), Autor, Titel
- **Lieder:** Interpret, Titel, Spielzeit in Sekunden

Das erste Programm soll **`Sender.js`** heißen und einen Datensatz auf das Kafka-Topic schreiben. Die einzelnen Werte sollen unter Verwendung einer geeigneten Bibliothek (z.B. **`readline-sync`**[6]) von der Tastatur eingelesen werden.

Das zweite Programm soll **`Empfaenger.js`** heißen und die Datensätze vom Topic lesen, deserialisieren und die Werte der einzelnen Felder auf die Konsole schreiben.

Der Name des Topics sowie das Kafka-Objekt mit der Verbindungskonfiguration sollen in einer Datei names **`Gemeinsam.js`** definiert sein, um doppelte Quellcode-Fragmente zu vermeiden. Der in der Datei hinterlegte Topic-Name soll hierbei nach folgendem Muster gebildet werden:

`<kurs>.<nachname><vorname>.Aufgabenblatt`

Ein konkretes Beispiel wäre: **`WWI22B3.MustermanMax.Aufgabenblatt`**

b) Auf der Apache Kafka Webseite wird unter „Use Cases“ der folgende Anwendungsfall beschrieben:[7]

„Kafka can serve as a kind of external commit-log for a distributed system. The log helps replicate data between nodes and acts as a re-syncing mechanism for failed nodes to restore their data.“

Zeichnen Sie eine Skizze mit drei verteilten Knoten, die dieses Verfahren umsetzen, um ihre Daten untereinander zu replizieren. Erklären Sie daran den typischen Ablauf bei Änderung eines Werts durch einen der Knoten, wobei einer der beiden anderen Knoten temporär nicht erreichbar ist, zu einem späteren Zeitpunkt aber wieder online ist. Erörtern Sie anschließend, ob durch das Verfahren auch das Lost-Update-Problem gelöst werden kann.

Aufgabe 4: Wahlvertiefungen (10 Punkte)

In dieser Aufgabe geht es darum, ein über die Vorlesungsinhalte hinausgehendes Vertiefungsthema zu recherchieren und anhand eines realen Fallbeispiels zu beschreiben. Suchen Sie sich hierfür eines der folgenden Themen aus und recherchieren Sie ein Fallbeispiel dazu. Formulieren Sie anschließend auf ca. einer DIN A4 Seite (500 Wörter) sowohl eine möglichst vollständige Begriffsdefinition als auch das Fallbeispiel aus. Folgende Themen stehen zur Auswahl:

- OpenAPI
- API-Gateway
- OAuth
- Circuit Breaker Pattern
- MQTT

Alternativ können Sie auch ein selbstgewähltes Thema mit Bezug zu verteilten Systemen vertiefen. Sprechen Sie dieses aber bitte unbedingt zuvor ab.

5 https://www.npmjs.com/package/kafkajs
6 https://www.npmjs.com/package/readline-sync
7 https://kafka.apache.org/uses