Verteilte Systeme: Portfolio-Aufgaben

Studienjahr 2024 / 2025

Prüfungshinweise

Bearbeitung in Eigenleistung

Alle Aufgaben sind in Eigenleistung zu bearbeiten. Die Lösungen müssen einen individuellen Lösungsweg erkennen lassen. Zwar ist es erwünscht, fremde Quellen als Hilfestellung zu nutzen. Die Lösungen müssen aber dennoch selbst ausgearbeitet werden. Abschreiben (auch mit Umformulierungen) gilt als Täuschungsversuch.

Fremde Quellen und KI-Sprachassistenten

Fremde Quellen dürfen verwendet werden, müssen aber durch Kommentare im Quellcode mit einem kurzen Verweis auf die Quelle und die verwendeten Inhalte gekennzeichnet werden.

Eine besondere Regel gilt für KI-Sprachassistenten: Richtig eingesetzt können sie sehr nützlich sein, weshalb die Verwendung natürlich erlaubt ist, aber als Quellenangabe gekennzeichnet werden muss. Generierter Quellcode ist durch Angabe des verwendeten Prompts vom selbst geschriebenen Quellcode klar abzugrenzen. Insgesamt dürfen nicht mehr als 33% des Quellcodes generiert sein (ausgenommen Kommentare). KI-Chatverläufe müssen vollständig gesichert und dem Portfolio beigelegt werden. Hier ein paar Beispiele, was erlaubt ist:

- **Klärung konkreter Fragen:** *Wie lässt sich das HATEOAS-Prinzip für REST-Webservices umsetzen?*
- **Recherche eines Themas:** *Welche Möglichkeiten gibt es, in Node.js auf Datenbanken zuzugreifen?*
- **Verprobung von Annahmen:** *Stimmt es, dass Webservice-Aufrufe immer synchron erfolgen?*
- **Hilfestellung beim Programmieren:** *Warum funktioniert folgender Quellcode nicht?*
- **Generieren von Kommentaren:** *Schreibe einen JSDoc-Kommentar zu dieser Funktion.*
- **Hilfe bei der Fehlersuche:** *Was bedeutet der Fehler „Cannot read properties of undefined“?*

Grundsätzlich sind Sie angehalten, Chat-Assistenten als Werkzeug angemessen zu nutzen und die Antworten zu hinterfragen. Dann können sie Ihnen gute Dienste leisten. Stellen Sie deshalb möglichst viele Folgefragen und weißen Sie darauf hin, wenn etwas unklar ist oder eine Antwort nicht stimmig erscheint. Oftmals werden Sie sehr gute Impulse erhalten. Manchmal aber auch nur teilweise richtige, veraltete oder schlicht unsinnige Antworten. Bedenken Sie, dass Chat-Assistenten in Wirklichkeit „statistische Textgeneratoren“ sind und nicht wirklich „denken“ oder „etwas verstehen“ können. Das müssen Sie immer noch selbst tun. ☺ Nicht erlaubt sind deshalb:

- Die Aufgabenstellung hineinkopieren und hoffen, dass die Antwort schon stimmen wird.
- Die Antworten ohne eigene Überlegungen und Folgerecherche unverändert zu übernehmen.

Denn am Ende geht es nicht einfach darum, die richtige Antwort zu Papier zu bringen, sondern anhand der Aufgabenstellung etwas zu lernen. Die Nichteinhaltung dieser Vorgaben kann daher zu Punkteabzug oder Nichtbestehen der Prüfung führen.

Abgabe

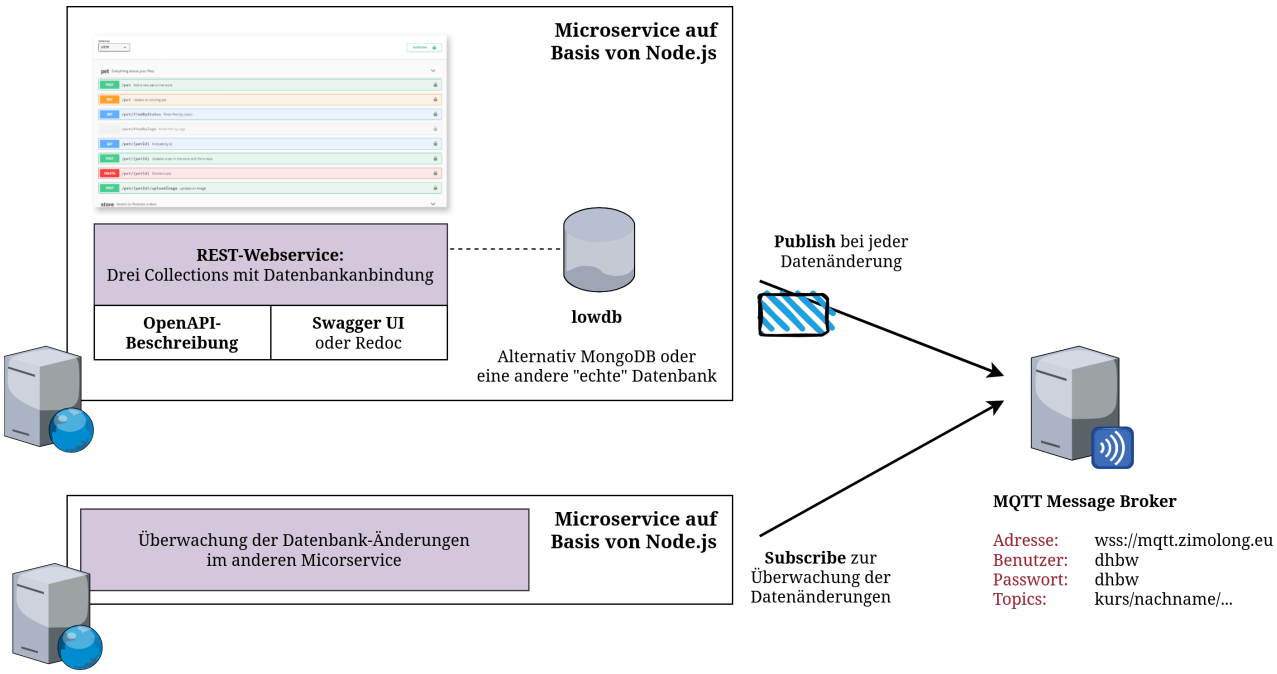
Die Abgabe erfolgt zum in der Vorlesung genannten Stichtag per Upload in Moodle. Quellcodes müssen, auch wenn Sie online verfügbar sind, ebenfalls in Moodle hochgeladen werden. Denken Sie daran, das Verzeichnis `node_modules` vor dem Erstellen der ZIP-Datei zu löschen! Ein Git-Repository wird nicht benötigt (kann aber angelegt werden, wenn Sie wollen).

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

Bitte beachten Sie, dass die Aufgabe eine **Einzelleistung** ist, die mit der Gruppenarbeit des letzten Semesters verrechnet wird. Da die Portfolionote überwiegend personenbezogen gebildet werden muss, muss hier jede*r Studierende eine eigene Lösung erarbeiten und abgeben.

Auch dieses Jahr wollen wir das theoretische Wissen aus den Vorlesungen wieder in einem praktischen Programmierprojekt anwenden. Dabei soll eine **verteilte Anwendung** mit einer **Microservice-Architektur**, wie in der folgenden Architekturskizze dargestellt, realisiert werden. Wir konzentrieren uns ausschließlich auf das Server-Backend – also die fachlichen Anwendungsfunktionen ohne Benutzeroberfläche (für die in einem echten Projekt beispielsweise Webanwendungen oder Mobile Apps in Frage kämen). Das fachliche Anwendungsszenario können Sie selbst bestimmen.



Folgende Anforderungen soll die Anwendung erfüllen:

Datenmodell: Der große Microservice soll ein Datenmodell aus einer beliebigen Anwendungsdomäne mit drei zusammenhängenden Entitäten besitzen. Das Datenmodell soll in der README-Datei des Projekts in geeigneter Form dokumentiert werden, beispielsweise als E/R-Diagramm mit draw.io oder drawdb.vercel.app. Die Zeichnung reicht aus, muss das Datenmodell aber vollständig beschreiben.

Datenbankanbindung: Die Daten des großen Microservices sollen in einer Datenbank abgelegt werden. Um keinen echten Datenbankserver aufsetzen zu müssen, können Sie die Datenbank mit dem Paket lowdb simulieren. Wahlweise können Sie sich von Co-Pilot oder ChatGPT zeigen lassen, wie Sie eine echte Datenbank wie MongoDB ansprechen können. In diesem Fall legen Sie dem Quellcode ein Docker Compose File bei, mit dem die Datenbank für die Bewertung lokal gestartet werden kann.

REST-Webservice: Die Daten des großen Microservices sollen über einen REST-Webservice abrufbar und veränderbar sein. Der Microservice soll hierfür je Entität einen HTTP-Endpunkt bieten, der die üblichen CRUD-Operationen (Create, Read, Update und Delete) sowie eine Suche nach geeigneten Textattributen unterstützt. Als Datenformat können Sie JSON verwenden. Achten Sie darauf, dass der Webservice möglichst „restful“ ist (eindeutige URLs, richtige Verwendung der HTTP-Verben, Statuscodes und Header Fields). Hierfür gelten folgende Empfehlungen:

- **Laufzeitumgebung:** Node.js
- **Web-Framework:** Express

Achten Sie auf eine saubere Trennung von HTTP-Verarbeitung **(Controller)**, fachlicher Logik **(Service)** und Datenmodell **(Model)**, wie in der Vorlesung gezeigt. Dies entspricht im Grunde genommen einer Einhaltung der bekannten SOLID-Prinzipien in der objekt-orientierten Programmierung. Die Quellcodes aus der Vorlesung dürfen als Vorlage verwendet werden.

OpenAPI-Beschreibung: Der Webservice soll, um seine Nutzung zu vereinfachen, über eine OpenAPI-Spezifikation im YAML-Format beschrieben werden. Diese können Sie entweder mit der Webanwendung Apicurio Studio grafisch erstellen, händisch schreiben oder mit KI-Werkzeugen generieren lassen. Die Beschreibung muss den Webservice jedoch vollständig und exakt beschreiben.

Online-Dokumentation: Anhand der OpenAPI-Spezifikation soll auch eine menschenlesbare Dokumentation erstellt werden. Zum Glück kann dies – eine gute OpenAPI-Beschreibung vorausgesetzt – mit Paketen wie swagger-ui-express oder redoc komplett automatisch geschehen. Eines der beiden Pakete soll daher derart in den großen Microservice integriert werden, dass die Dokumentation über einen eigenen HTTP-Endpunkt online einsehbar ist.

Event-basierte Kommunikation: Microservices nutzen häufig ebenfalls REST-Webservices, um untereinander zu kommunizieren. Es gibt jedoch Anwendungsfälle, bei denen eine derartige synchrone Punkt-zu-Punkt-Verbindung unpraktisch ist. Häufig wird dann auf ein **asynchrones Publish/Subscribe-Modell** ausgewichen. Im Projekt soll hierfür das besonders leichtgewichtige MQTT-Protokoll zusammen mit dem unter der URL wss://mqtt.zimolong.eu erreichbaren MQTT-Server verwendet werden.

- Der große Microservice soll bei jeder Datenänderung eine Nachricht an den MQTT-Server schicken **(publish)**. Die Nachricht soll die REST-URL des veränderten Datensatzes sowie die Art der Änderung beinhalten. Ggf. kann es auch Sinn machen, den Datensatz selbst mitzuschicken.
- Je Entität sollten Sie die Nachricht an ein anderes Topic schicken, um den Empfängern die Möglichkeit zu geben, nur bestimmte Datenänderungen zu empfangen.
- Ein zweiter Microservice soll die Topics überwachen **(subscribe)** und beispielsweise einen Newsticker in der Form „Neue Bestellung eingegangen: *Daten der Bestellung …*“ auf der Konsole ausgeben. Weitere Geschäftslogik oder eine Datenbankanbindung müssen nicht implementiert werden.

Achten Sie darauf, die in der Abbildung gezeigten Zugangsdaten für den MQTT-Server zu verwenden. Um Überschneidungen mit anderen Studierenden zu vermeiden, sollte jedes Topic mit Ihrer Kursnummer und Ihrem Nachnamen beginnen. Für die Programmierung können Sie das mqtt-Paket für Node.js verwenden.