HOCHSCHULE HEILBRONN

EXPOSÉ

ZUM THEMA

Vergleichende Untersuchung von GraphQL und REST, hinsichtlich Leistungsfähigkeit und Flexibilität

Erstellt von:
Robin Hefner
206488

1 Problemstellung

In der modernen Softwareentwicklung spielen APIs (Application Programming Interfaces) eine entscheidende Rolle bei der Integration und Kommunikation zwischen verschiedenen Diensten und Anwendungen. Traditionell wurde REST (Representational State Transfer) als Standard für die Erstellung und Nutzung von APIs verwendet. Mit der Einführung und zunehmenden Verbreitung von GraphQL, einer Abfragesprache für APIs, die von Facebook entwickelt wurde, stehen Entwickler nun vor der Wahl zwischen diesen beiden Ansätzen. Die Wahl zwischen REST und GraphQL hat signifikante Auswirkungen auf die Entwicklung und den Betrieb der Anwendung. Unternehmen müssen eine fundierte Entscheidung treffen, welche Technologie besser zu ihren Anforderungen, im Hinblick auf Leistungsfähigkeit und Flexibilität, passt.

2 Stand der Technik

REST ist eine Architektur, die 2000 zum ersten Mal in einer Dissertation von Roy Fielding beschrieben wurde. Sie integriert das HTTP-Protokoll und nutzt als zentrales Element die Ressourcenorientierung, bei der jede Entität als Ressource betrachtet und durch eine eindeutige URL identifiziert wird. Die CRUD-Operationen (Create, Read, Update, Delete) werden mithilfe der HTTP-Methoden (POST, GET, PUT, DELETE) repräsentiert. Ein besonderes Merkmal ist die Zustandslosigkeit, bei der jeder API-Aufruf alle notwendigen Informationen enthält, um diesen zu bearbeiten. Hierdurch wird die Client-Server-Interaktion vereinfacht. Zudem nutzt REST das in HTTP integrierte Caching um Antwortzeiten und Leistung zu verbessern. (Fielding; 2000) (Vadlamani et al.; 2021)

GraphQL wurde 2012 von Facebook für den internen Gebrauch entwickelt. 2015 wurde es als Open Source Projekt für die Allgemeinheit veröffentlicht. Das Herzstück von GraphQL stellen die client-getriebenen Abfragen dar, bei denen der Client die Struktur der Daten genau spezifizieren kann und nur die benötigten Daten erhält. Hierdurch werden überflüssige Datenübertragungen reduziert und effizientere Netzwerkaufrufe ermöglicht. Durch eine hierarchische Struktur der Abfragen, die die Graph-Struktur widerspiegelt, wird eine intuitive Datenmodellierung ermöglicht. Die starke Typisierung in GraphQL wird durch ein Schema definiert, dass die Typen der Daten spezifiziert und somit eine bessere Validierung und Dokumentation ermöglicht. Im Gegensatz zu REST, das für verschiedene Operationen mehrere Endpunkte benötigt, verwendet GraphQL einen einzigen Endpoint für alle API-Abfragen. (GraphQL Foundation; 2024) (Vadlamani et al.; 2021)

3 Forschungsfrage

Diese Arbeit soll auf den folgenden zwei Forschungsfragen basieren.
Erstens, wie unterscheiden sich GraphQL und REST hinsichtlich der Anfrage- und Antwortzeiten unter verschiedenen Lastbedingungen und Anfragekomplexitäten? Diese Frage zielt darauf ab, die Performance beider Systeme unter variablen Bedingungen zu vergleichen. Beispielsweise könnte untersucht werden, wie schnell eine API auf eine einfache Datenabfrage reagiert, im Vergleich zu einer komplexeren, die mehrere Abhängigkeiten involviert. Diese Untersuchung könnte Einblicke in die Effizienz der beiden Technologien bieten und somit als Entscheidungshilfe für Entwickler dienen, die die beste Lösung für ihre spezifischen Bedürfnisse auswählen möchten. Zweitens stellt sich die Frage, inwiefern bieten GraphQL und REST unterschiedliche Möglichkeiten zur Abfrageanpassung? Diese Frage beleuchtet die Flexibilität beider Systeme in Bezug auf die Individualisierung von Datenabfragen. Während REST traditionell durch feste Endpunkte gekennzeichnet ist, die jeweils eine bestimmte Datenstruktur zurückgeben, bietet GraphQL eine dynamischere Herangehensweise. Mit GraphQL können Clients genau die Daten anfordern, die sie benötigen, und keine zusätzlichen Informationen, was zu effizienteren Datenübertragungen führen kann. Diese Fähigkeit, Anfragen präzise anzupassen, könnte die Effizienz und Benutzerfreundlichkeit von Webanwendungen erheblich beeinflussen. Insgesamt bieten diese beiden Forschungsfragen einen umfassenden Rahmen, um die Vor- und Nachteile von GraphQL und REST eingehend zu untersuchen. Durch die Analyse der Antwortzeiten, sowie der Möglichkeiten der Abfrageanpassung können Entwickler fundierte Entscheidungen treffen und die für Ihre Anforderung optimale Technologie wählen.

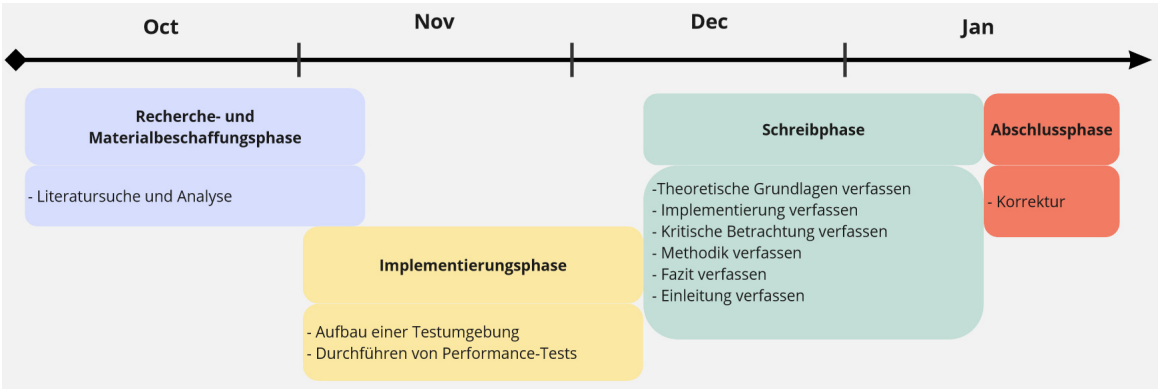
4 Methodik

Für die Untersuchung werden sowohl theoretische Analysen als auch empirische Experimente durchgeführt. Im Rahmen der theoretischen Analyse erfolgt eine umfassende Literaturrecherche und die Analyse bestehender Studien zur Leistungsfähigkeit und Flexibilität von REST und GraphQL. Die empirischen Experimente umfassen die Implementierung von Beispiel-APIs mit beiden Technologien sowie die Durchführung von Leistungstests. Die Leistungstests konzentrieren sich auf das Messen der Latenz bei verschiedenen Abfrageszenarien.

5 Vorläufige Gliederung

1. Abstract
2. Einleitung
3. Theoretische Grundlagen
    3.1 API-Grundlagen
    3.2 REST
    3.3 GraphQL
4. Implementierung
    4.1 Aufbauen einer Testumgebung
    4.2 Durchführen von Performance-Tests
5. Methodik
6. Ergebnisse
7. Diskussion
8. Fazit

6 Vorläufiger Zeitplan



Literatur

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. Online erhältlich unter `https://ics.uci.edu/~fielding/pubs/dissertation/top.htm`; abgerufen am 23. Juli 2024.

GraphQL Foundation (2024). What is the graphql foundation?, Website. Online erhältlich unter `https://graphql.org/community/foundation/`; abgerufen am 23. Juli 2024.

Vadlamani, S. L., Emdon, B., Arts, J. and Baysal, O. (2021). Can graphql replace rest? a study of their efficiency and viability. Online erhältlich unter `https://ieeexplore.ieee.org/abstract/document/9474834`; abgerufen am 23. Juli 2024.