Hauptprojekt

Alexander Piehl
alexander.piehl@haw-hamburg.de

Hamburg University of Applied Sciences,
Dept. Computer Science,
Berliner Tor 7
20099 Hamburg, Germany

1 Einleitung

2 REST

REST ist ein Architekturstil für verteilte Systeme. Die Abkürzung REST steht für Representational State Transfer [CK09]. Erstmals wurde REST 2000 von Roy Fielding vorgestellt [KLC13]. Die Rest-Architektur wird häufig für Client-Server Anwendungen verwendet. Dabei ist REST zurzeit sehr belie bbei der Entwicklung von Webservices, da REST Webservices wohl nicht nur leichter zu implementieren sind, sondern auch einfacher zu skalieren sind. [CK09]. Unter Anderem aus diesen Gründen stellten Google, Facebook und Yahoo ihre Services von SOAP auf REST um [Rod08, NCH+14].

REST basiert dabei auf Resource Oriented Architecture, kurz ROA [CK09]. Dies bedeutet, dass jede wichtige Information als Ressource zur Verfügung stehen muss [PR11]. Die Zugänglichkeit zu der Ressource muss über eine eindeutige URI gegeben sein. Die Ressourcen sollen zusätzlich über verschiedene Methoden manipuliert werden können. Es müssen mindestens die sogenannten CRUD-Operatoren zur Verfügung stehen. CRUD steht für Create, Read, Update und Delete und beschreibt die grundsätzlichen Daten Operationen. Bei Rest werden dafür die standardisierten HTTP-Methoden verwendet, welche im Standard RFC 2616 definiert wurden sind [KLC13]. In der Tabelle 1 werden die Beziehung zwischen den CRUD Operatoren und den HTTP Operatoren dargestellt.

CRUD-Operation	HTTP-Methode
Create	POST
Read	GET
Update	PUT
Delete	DELETE

Tabelle 1. Beziehung CRUD und HTTP Operatoren [RVG10]

Neben den CRUD-Operatoren können noch weitere HTTP-Methoden zur Verfügung stehen, wie z.B. HEAD und OPTIONS [PR11].

Die jeweiligen Aufrufe müssen Zustandslos erfolgen [RVG10, PR11, KLC13]. Im Detail heißt dies, dass der Webservices keine Informationen über den Zustand seiner einzelnen Clients speichert. Sollten Informationen über den Zustand notwendig sein, müssen die Clients die Informationen mitgeben. Dahingehend ist es auch mit REST möglich kompliziertere Programmzustände abzubilden [PR11].

Die jeweiligen Nachrichten können in verschiedene Formate vorliegen [RVG10]. Sehr häufig werden XML oder JSON oder beide Formate für die Nachrichten verwendet. Eine Vorschrift existiert nicht.

Zusammenfassend lässt sich REST in vier Grundprinzipien zusammenfassen [PR11]:

- **Addressability:** Jede wichtige Informationen muss als Ressource vorliegen und über eine eindeutige URI erreichbar sein.
- **Connectedness:** Die Repräsentation der Ressourcen ist getrennt von den Ressourcen. Dies bedeutet Ressourcen können in verschiedenen Formate vorliegen, wie JSON und XML.
- **Uniform Interface:** Auf die Ressourcen wird nur über standardisierten HTTP-Methoden zugegriffen.
- **Statelessnes:** Jede Kommunikation erfolgt Zustandslos.

Webservices, welche die vier Grundprinzipien einhalten, bekommen häufig den Beinamen RESTful [PR11]. Der Begriff RESTful ist jedoch nicht eindeutig definiert.

2.1 Besonderheiten beim Testen

3 Schnittstellen Tests

3.1 Consumer Driven Contract Test

- Erklärung
- Erläuterung zu der Verbindung mit Microservice
- Normale Implementierung
- Einbinden in Mars
- Fazit

4 Fazit

Literaturverzeichnis

[CK09] Sujit Kumar Chakrabarti and Prashant Kumar. Test-the-rest: An approach to testing restful web-services. In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:*, pages 302–308. IEEE, 2009.

[FB15] Tobias Fertig and Peter Braun. Model-driven testing of restful apis. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1497–1502. ACM, 2015.

[KLC13] Chia Hung Kao, Chun Cheng Lin, and Juei-Nan Chen. Performance testing framework for rest-based web applications. In *2013 13th International Conference on Quality Software*, pages 349–354. IEEE, 2013.

[NCH+14] Alvaro Navas, Pedro Capelastegui, Francisco Huertas, Pablo Alonso-Rodriguez, and Juan C Dueñas. Rest service testing based on inferred xml schemas. *Network Protocols and Algorithms*, 6(2):6–21, 2014.

[PR11] Ivan Porres and Irum Rauf. Modeling behavioral restful web service interfaces in uml. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1598–1605. ACM, 2011.

[Rod08] Alex Rodriguez. Restful web services: The basics. *IBM developerWorks*, 2008.

[RVG10] Hassan Reza and David Van Gilst. A framework for testing restful web services. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 216–221. IEEE, 2010.