Inhaltsverzeichnis

1 Einführung in die Thematik

1.1 REST

Representational State Transfer (REST) ist eine Abstraktion der architektonischen Elemente innerhalb eines verteilten Hypermediasystems. Es ist somit ein Softwarearchitekturstil, der die architektonischen Prinzipien, Eigenschaften und Einschränkungen für die Umsetzung von internetbasierten verteilten Systemen definiert (Fielding und Taylor, 2000: 86). REST basiert auf fünf Kernprinzipien (Tilkov u. a., 2015: 11):

- Ressourcen als Abstraktion von Informationen, identifiziert durch einen eindeutigen *resource identifier*
- Verknüpfungen / Hypermedia
- Standardmethoden
- Unterschiedliche Repräsentationen
- Statuslose Kommunikation

In Folgendem wird untersucht, welche maschinenlesbare Spezifikationsformate zur Beschreibung von auf HTTP basierenden REST APIs zur Verfügung stehen. Ziel dieser Untersuchung ist es, eine abstrakte Modellierung von REST APIs zu konzipieren, die verwendet werden kann, um eine API Spezifikation in eine interne Datenstruktur zu überführen, welche dann anschließend angereichert wird um automatisiert Artefakte wie Dokumentation oder Test-Cases zu generieren. Im Zuge dessen wird ebenfalls analysiert, welche Möglichkeiten in diesem Feld bereits existieren.

1.2 Spezifikationsformate

1.2.1 Contract First und Contract Last

Es kann im Wesentlichen zwischen zwei verschiedenen Ansätzen zur Definition des Vertrages, den eine API garantiert, unterschieden werden (Spichale, 2017: 272):

- *Contract-First*. Bei diesem Ansatz wird zunächst der Vertrag (also die Spezifikation der API) geschrieben, und anschließend die Implementierung. Vorteilhaft ist hier der größere Fokus auf dem Design der Schnittstelle. In frühen Iterationen der Spezifikation kann in Abstimmung mit den Konsumenten der API mit geringen Aufwand der Vertrag gefestigt werden, sodass später bei der Implementierung weniger Änderungen anfallen. Nachteil ist, dass Entwickler mehr mit den eigentlichen Spezifikationen arbeiten müssen, und diese nicht automatisch generieren können.

- *Contract-Last*. Bei diesem Ansatz existiert die Implementierung bereits, und die Beschreibung wird nachträglich erzeugt. Häufig können hier unterstützend Tools benutzt werden, welche die Spezifikation automatisch aus der Implementierung generieren, und der Entwickler muss meistens nur noch einige Annotationen anpassen. Da bei der automatischen Generierung Schemas in vielen Fällen dupliziert werden, kann dies zu mehr Aufwand führen, wenn manuelle Anpassungen vorgenommen werden müssen. Ebenso reduziert dies die Wiederverwendbarkeit von Spezifikationsteilen oder Schemas (Zhong und Yang, 2009: 1).

Bei beiden Ansätzen müssen Tests angelegt werden, die sicherstellen, dass die Implementierung nicht von der Beschreibung bzw. Dokumentation abweicht.

1.2.2 RAML

RAML (*RESTful API Modeling Language*) ist ein auf YAML basierende Spezifikationsformat. Es wurde insbesondere für den *Contract-First* Ansatz konzipiert, der Schwerpunkt liegt damit auf dem API-Design (Spichale, 2017: 277; Tilkov u. a., 2015: 165). Beschreibungen können in Markdown formatiert werden. Jedes RAML-Dokument beginnt mit der Angabe der RAML-Version und einigen allgemeinen Informationen zur API, wie Titel, URL und Version. Hier können ebenfalls Strukturierungshilfen wie Traits oder Datentypen definiert werden, welche dann im Rest des Dokumentes eingebunden werden können. Anschließend folgen die Beschreibungen der Ressourcen und Subressourcen. Zur Beschreibung von Requests und Responses können optional die

bereits definierten Datentypen oder auch JSON-Schema Definitionen verwendet werden. Über *Includes* kann die Spezifikation in mehrere Dateien aufgeteilt werden.
Folgendes Beispiel wurde aus der RAML-Spezifikation entnommen (*RAML Version 1.0 Spezifikation* o.D.).

```
#%RAML 1.0
title: GitHub API
version: v3
baseUri: https://api.github.com
mediaType: application/json
securitySchemes:
  oauth_2_0: !include securitySchemes/oauth_2_0.raml
traits:
securedBy: [ oauth_2_0 ]
title: API with Types
types:
  User:
    type: object
    properties:
      firstname: string
      lastname: string
      age: number
/users/{id}:
  get:
    responses:
      200:
        body:
          application/json:
            type: User
```

Literatur

Fielding, Roy T und Richard N Taylor (2000). *Architectural styles and the design of network-based software architectures*. Bd. 7. University of California, Irvine Irvine, USA.

RAML Version 1.0 Spezifikation (o.D.). Abgerufen am 21. November 2018. URL: https://github.com/raml-org/raml-spec/blob/master/versions/raml-10/raml-10.md.

Spichale, K. (2017). *API-Design: Praxishandbuch für Java- und Webservice-Entwickler*. dpunkt.verlag.

Tilkov, Stefan u. a. (2015). *REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web*. dpunkt. verlag.

Zhong, Youliang und Jian Yang (2009). „Contract-First design techniques for building enterprise web services“. In: *Web Services, 2009. ICWS 2009. IEEE International Conference on*. IEEE, S. 591–598.