

Inhaltsverzeichnis

1	Anforderungen, Ziele und Rahmenbedingungen	2
1.1	Empirische Untersuchung	2
1.2	Anforderungen	2
1.3	Ziele	2
1.3.1	Strategische Ziele	2
1.3.2	Taktische Ziele	3
1.3.3	Operative Ziele	3
1.4	Funktionsumfang des Prototyps	4

1 Anforderungen, Ziele und Rahmenbedingungen

1.1 Empirische Untersuchung

1.2 Anforderungen

1.3 Ziele

Im Folgenden werden die konkreten Ziele des Systems hierarchisch angeordnet. Zunächst werden die strategischen, langfristigen Ziele festgelegt, aus denen sich anschließend die taktischen und operativen Ziele ableiten. Die Ziele sind in der jeweiligen Kategorie nach ihrer Priorität angeordnet. Bei allen Zielen wird davon ausgegangen dass eine API Spezifikation vorliegt, auf Basis derer die Aufgaben vorgenommen werden können.

1.3.1 Strategische Ziele

1. **Optimierung des REST-API Entwicklungsprozesses.**

Durch zumindest teilweise Automatisierung von verschiedenen Aufgaben im Entwicklungsprozess (Testing & Dokumentation) soll Zeit eingespart werden.

2. **Erhöhte Softwarequalität.**

Die Vereinheitlichung des Prozesses und auch die automatisierte Generierung von Artefakten sollte zu einer höheren Softwarequalität und einfacheren Wartung führen.

3. **Vereinheitlichung des REST-API Entwicklungsprozesses.**

Mit der Unterstützung des Entwicklungsprozesses durch das System sollen Grenzen und Vorgaben gesetzt werden, in denen sich die Entwickler bewegen. Das Design der REST APIs folgt einem einheitlicherem Konzept, wodurch Nutzer nicht bei jeder API mit neuen Designentscheidungen konfrontiert sind. Generierte Artefakte sind konsistent, und ihre Nutzung muss nur einmal erlernt werden.

1.3.2 Taktische Ziele

1. **Abstraktes Datenmodell.**

Das Datenmodell des Systems sollte so konzipiert sein, dass mehrere Spezifikationsformate überführt werden können.

2. **Erweiterbarkeit des Systems.**

Das System sollte so modular konzipiert und implementiert sein, dass Teile des Systems erweiterbar sind. Damit könnten beispielsweise neue Parser (zur Unterstützung von neuen Spezifikationsformaten) oder Factories (z.B. zur Generierung von Testcases in einer anderen Sprache) in das System eingebunden werden.

3. **Generierung von Testcases.**

Es sollen automatisch sinnvolle Testcases generiert werden, um die Funktionalität der API zu testen. Diese sollten dabei unterstützend wirken. Das Ziel ist es nicht, die Entwickler komplett vom Schreiben von Tests zu entbinden; vielmehr sollen einfache Tests zwar komplett automatisiert werden, bei solchen Tests, die noch manuelle Ergänzungen der Entwickler benötigen, soll das Systems allerdings nur hilfreiche Vorarbeit leisten.

4. **Generierung von Dokumentationsartefakten.**

Zur Verbesserung der Dokumentation der API sollen automatisch kollaborationsunterstützende Artefakte generiert werden.

5. **Überprüfung auf Best Practices.**

Es sollte (optional) auf die Befolgung von Best Practices beim Design von REST-APIs geprüft werden sollen (Linting).

1.3.3 Operative Ziele

1. **Generierung von JSON Schemas.**

Aus dem eingelesenen Metamodell sollen JSON Schemas generiert werden, welche unter anderem für Testcases verwendet werden.

2. **Generierung von nominalen Testfalld Definitionen.**

Zur Überprüfung der Funktionalität der API, sollen Testcases generiert

werden, die jeden Endpunkt der API mit den in der Spezifikation definierten Parametern und Attributen aufrufen, und auf valide Antworten prüfen.

3. Generierung von fehlerhaften Testfalld Definitionen.

Es sollen nicht nur Testfälle generiert werden, die auf Funktionalität der API in Einhaltung aller in der Spezifikation definierten Einschränkungen prüfen, sondern auch solche, die auf korrekte Fehlerbehandlung prüfen. Dies kann z.B. in Form einer Anfrage mit falschen Attributen oder Attributwerten geschehen.

4. Generierung einer API Dokumentation.

Es soll eine responsive, interaktive und leicht anpassbare API-Dokumentation generiert werden, welche die Benutzung der API verdeutlicht.

5. Generierung und Synchronisation von Postman Collections.

Als weiteres kollaborationsunterstützendes Artefakt sollen automatisch Postman Collections generiert werden, und diese automatisch über die Postman API bei allen Entwicklern synchronisiert werden.

6. Einfache Einbindung in Projekte.

Das System sollte ohne großen Aufwand in neue oder bestehende Projekte eingebunden werden können.

7. Partielle Überschreibung von Testcases bei Neugenerierung.

Insbesondere bei den Testcases, die noch weitere manuelle Ergänzungen der Entwickler benötigen, sollen bei Neugenerierung nicht die Ergänzungen überschrieben werden.

1.4 Funktionsumfang des Prototyps