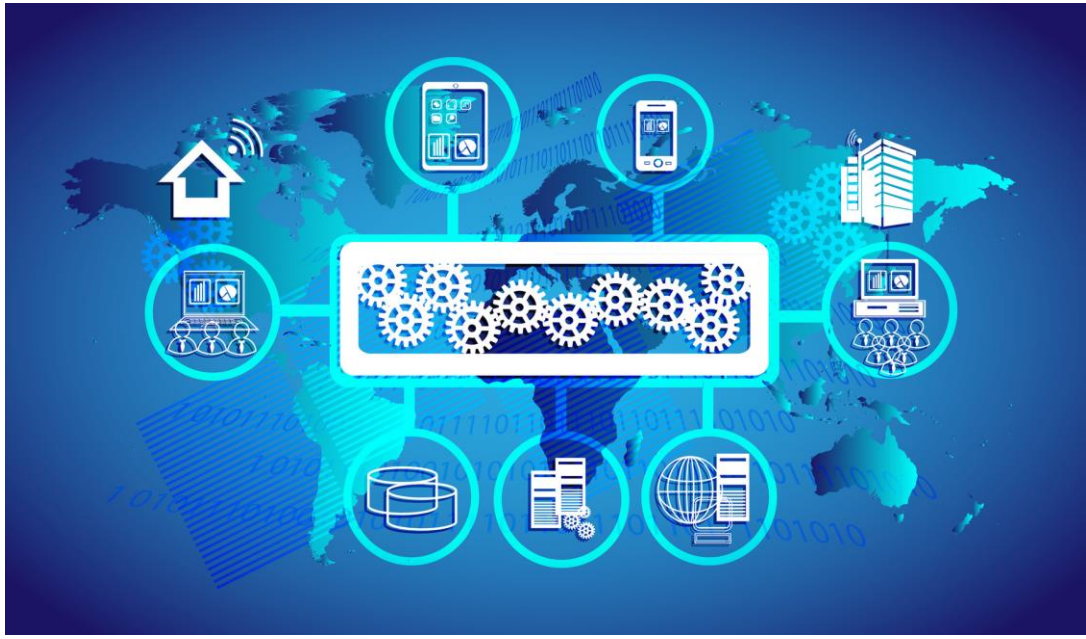
doubleSlash

Sebastian Lohr, Isabel Huber, Stefan Träger | 07.07.2022

Der Weg zum guten API Design



Wer wir sind

Sebastian Lohr
Software Developer

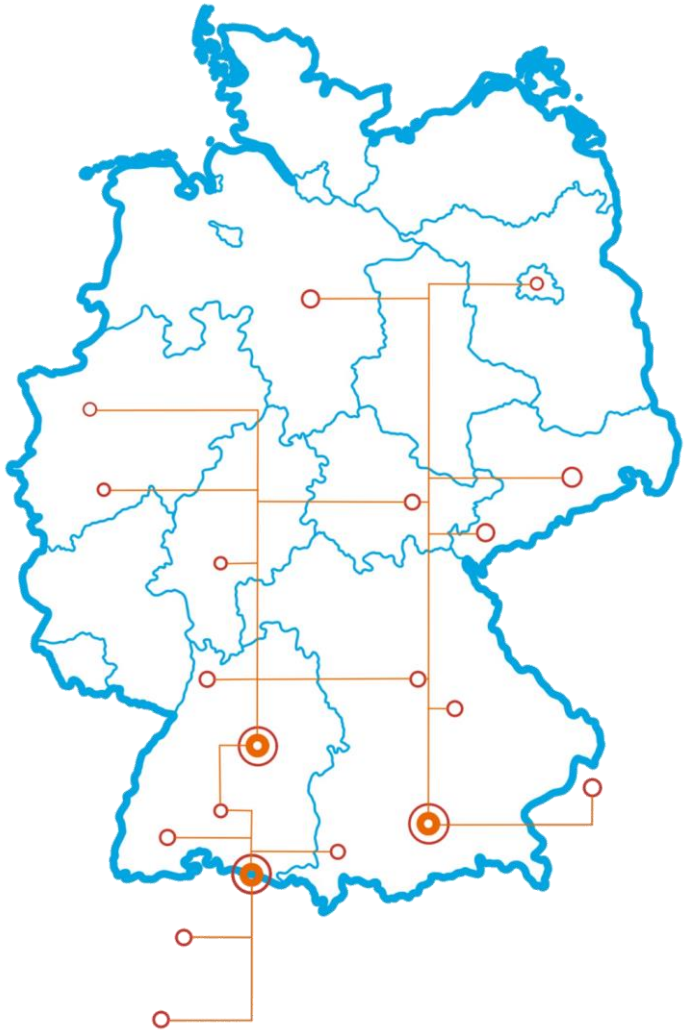
Isabel Huber
Software Developer

Stefan Träger
Software Developer

Kurz zu doubleSlash

// Ganzheitlicher Partner für innovative Softwareprodukte

// >20 Jahre Erfahrung
260 Mitarbeiter
26 Mio. € Umsatz



// Standorte in Friedrichshafen, München und Stuttgart

// ISO 9001
ISO 27001
TISAX

Das machen wir

Connected Things

Überwachen, Analysieren und Warten von IoT-Geräten.

Connected Mobility

Umsetzen von intelligenten und vernetzten Mobilitätslösungen.

Data Driven Services

Integrieren, Veredeln, Analysieren und Visualisieren von Daten.

Subscription Management

Verwalten und Monetarisieren digitaler und physischer Produkte.

Software Creation Factory

Design, Entwicklung und Betrieb über den gesamten Software-Produktlebenszyklus hinweg.

Wir alle kennen es …

Irreführende Returncodes:

PUT https://{hostname}:{port}/cli/application/propValue
Accept: application/json

IBM Request: Query String statt JSON Body

HTTP 401 Unauthorized
Message: "Use the Json template to set this property.“

Nicht REST konforme APIs:

GET /getPosts

POST /updatePosts

… fehlerhafte APIs

Falsche Verwendung von HTTP Methoden

POST vs. PUT

GET mit Request Body

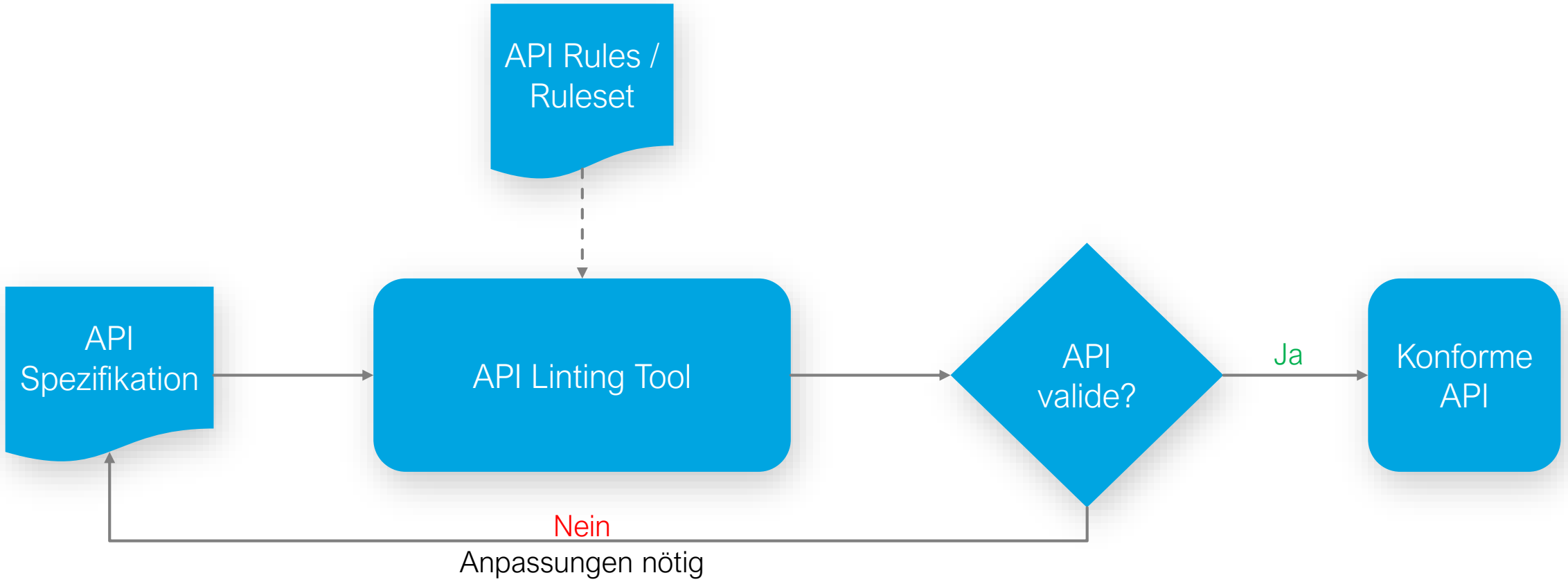
Irreführende fachliche Fehlermeldungen

Endpunkt um (mehrere) Entitäten zu speichern

Fehlermeldung, dass alle Entitäten falsch sind. Es werden aber alle bis auf die falsche gespeichert

> Senden aller Entitäten inkl. der korrigierten:
> Es kommt ein Fehler, da die bereits gespeicherten Fehler verursachen.
> → Einzelnes Speichern der Entitäten

Was uns hier helfen kann …



Was euch in unserem Vortrag erwarten wird…

- Vergleich von API Linting Tools
- Definition von API Guidelines
- Integration eines API Linters in den Buildprozess
    - Definition eines Rulesets
    - Automatisierte Validierung des Rulesets
    - „Schmankerl“

Vergleich

Funktionale Anforderungen

- Linter in gängige Build Pipelines integrierbar
- Regelset erweiterbar
- Programmatische Erweiterung des Regelsets möglich
- Anpassung der Fehlermeldungen im Regelset möglich
- Umgang des Linters mit gängigen Varianten der API Dokumentationen möglich
- Gutes Subset an vordefinierten Regeln vorhanden

Nicht Funktionale Anforderungen

- Keine Lizenzkosten
- Repository min. 1 Commit im Monat
- Dokumentation
- Keine externen infrastrukturellen Abhängigkeiten bei Ausführung

we Speccy


Zally


Spectral

Speccy

- Entwickelt von Wework
- Bietet ein CLI Tool
- Yaml Dateien zur Regelset Erstellung
- OpenApi v3 Validierung

Anforderungen

- ✓ Pipeline Integration
- ✓ Erweiterbares Regelset
- – Regelset programmatisch erweiterbar
- ✓ Fehlermeldungen anpassbar
- – Umgang mit gängigen API Dokumentationen
- ✗ Vordefinierte Regeln
- ✓ Keine Lizenzkosten
- ✗ Min. 1 Commit im Monat
- – Dokumentation
- ✓ Externe Abhängigkeiten

Zally

- Entwickelt von Zalando
- Bietet ein CLI Tool, eine Web UI und eine RESTful API
- Zentralisierter Ansatz – Validierungsserver notwendig
- Unterstützt OpenApi v2 und v3
- Validierung gegen Zalando REST Guidelines

Anforderungen

- ✓ Pipeline Integration
- ▬ Erweiterbares Regelset
- ✓ Regelset programmatisch erweiterbar
- ✓ Fehlermeldungen anpassbar
- ✓ Umgang mit gängigen API Dokumentationen
- ✓ Vordefinierte Regeln
- ✓ Keine Lizenzkosten
- ✓ Min. 1 Commit im Monat
- ▬ Dokumentation
- ▬ Externe Abhängigkeiten

Spectral

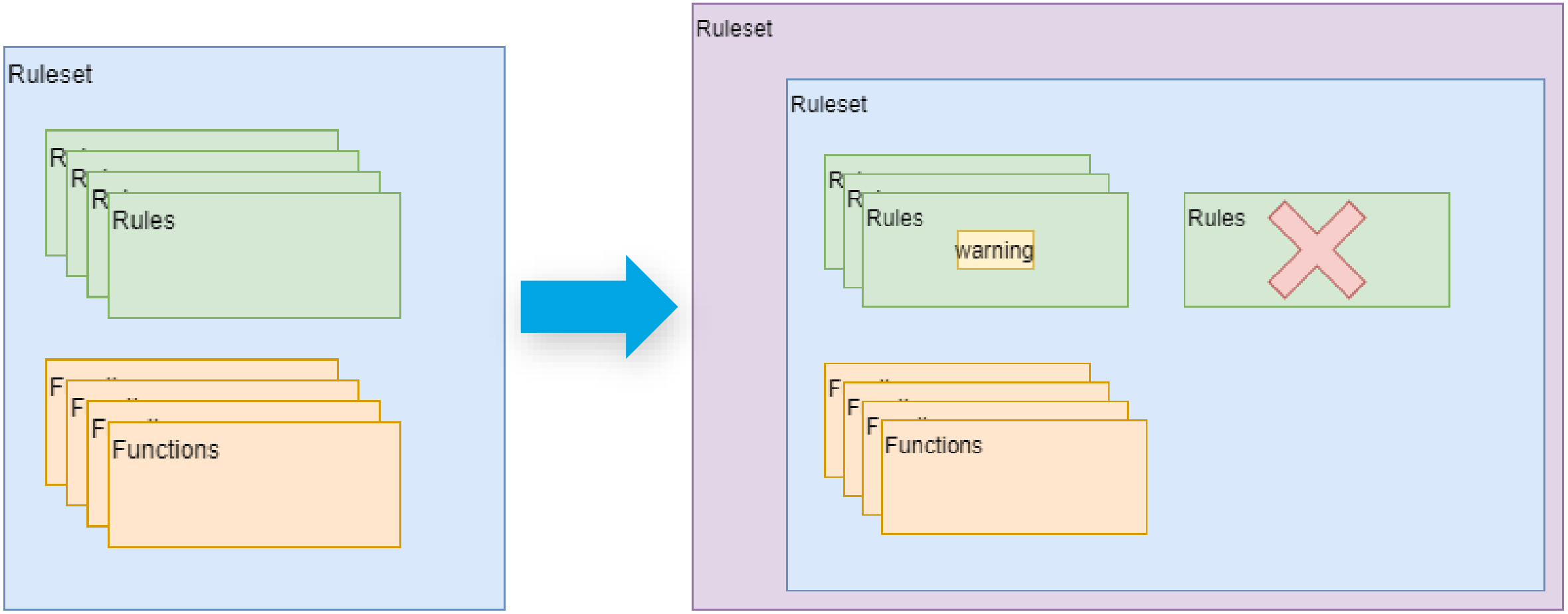
- Entwickelt von Stoplight
- Bietet ein CLI Tool, eine VS Code Extension und eine JavaScript API
- Standardregelsets für
  - OpenAPI v2 und v3
  - AsyncAPI
- Dezentraler Ansatz
- Yaml Dateien zur Regelset Erstellung

Anforderungen

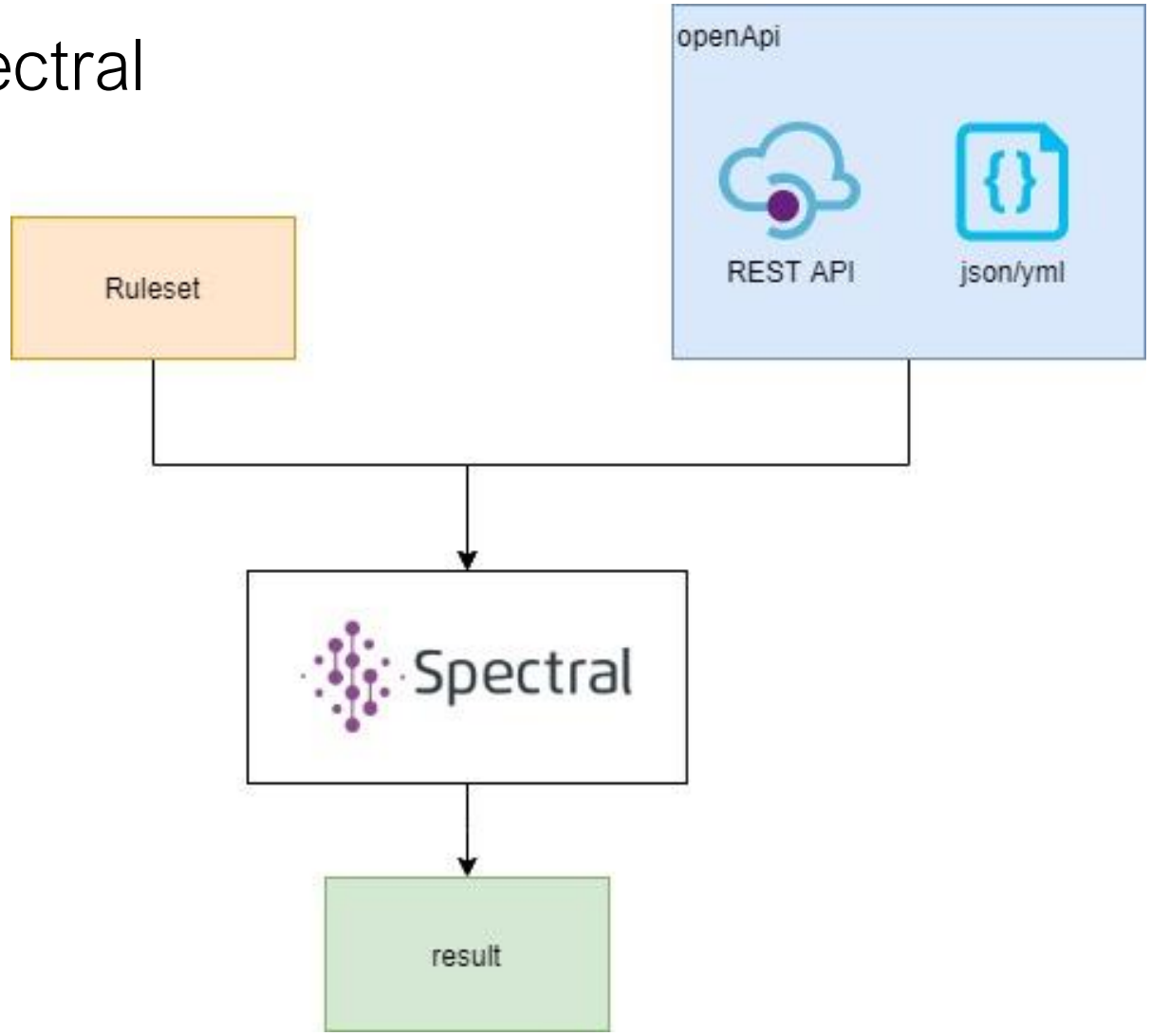
- ✓ Pipeline Integration
- ✓ Erweiterbares Regelset
- ✓ Regelset programmatisch erweiterbar
- ✓ Fehlermeldungen anpassbar
- ✓ Umgang mit gängigen API Dokumentationen
- ▬ Vordefinierte Regeln
- ✓ Keine Lizenzkosten
- ✓ Min. 1 Commit im Monat
- ✓ Dokumentation
- ✓ Externe Abhängigkeiten

Ergebnis

Aufbau Spectral Rulesets



Ablauf Spectral



Demo

Wrap-Up

- Zentraler – Dezentraler Ansatz
- Verbesserung der API
- Favorit Spectral

- GitHub Links
  - https://doubleslashde.github.io/API-Design-Guidelines
  - https://doubleslashde.github.io/API-Design-Guidelines/REST_API_Guidelines.pdf
  - https://github.com/doubleSlashde/API-Design-Guidelines

Fragen?

Für weitere Fragen

Sebastian Lohr
Sebastian.Lohr@doubleslash.de

Isabel Huber
Isabel.Huber@doubleslash.de

Stefan Träger
Stefan.Traeger@doubleslash.de