# REST API

Was ist eine REST API?

# Definition

- Eine serverseitige REST-API besteht aus einem oder mehreren öffentlich zugänglichen Endpunkten für ein definiertes Request-Response-Nachrichtensystem, das in der Regel in JSON oder XML mit Hilfe eines HTTP-basierten Webservers ausgedrückt wird.

# Endpunkte

Endpunkte sind wichtige Aspekte bei der Interaktion mit serverseitigen Web-APIs, da sie angeben, wo sich Ressourcen befinden, auf die zugegriffen werden kann.

Normalerweise erfolgt der Zugriff über eine URI, an die HTTP-Anfragen gesendet werden und von der somit eine Antwort erwartet wird. Web-APIs können öffentlich oder privat sein, wobei letzteres einen Access-Token erfordert

http://example.com/api/events

# statische Endpunkte

Endpunkte müssen statisch sein, da sonst das korrekte Funktionieren von Software, die mit der API interagiert, nicht gewährleistet werden kann. Ändert sich der Ort einer Ressource (und damit der Endpunkt), dann wird die zuvor geschriebene Software nicht mehr funktionieren, da die benötigte Ressource nicht mehr an derselben Stelle zu finden ist.

Um eine Versionierung zu gewährleisten, gibt es häufig verschiedene Versionen einer API.

# REST

Eine REST-API (auch bekannt als RESTful API) ist eine API (Application Programming Interface) oder Web-API, die den Beschränkungen der REST-Architektur unterliegt und Interaktionen mit RESTful Webservices ermöglicht.

REST steht für „Representational State Transfer“ und wurde vom US-amerikanischen Informatiker Roy Fielding entwickelt.

# REST

Bei REST handelt es sich um eine Sammlung von Architekturbeschränkungen und nicht um ein Protokoll oder einen Standard. API-Entwickler können REST auf vielfältige Weise implementieren.

**Die wichtigsten REST Kriterien:**

- Client/Server-Architektur, die Anforderungen per HTTP verwaltet
- Zustandslose Client/Server-Kommunikation
- Cachingfähige Daten
- einheitliche Schnittstelle zwischen Komponenten

# Header und Parameter

Header und Parameter sind wichtige Komponenten in den HTTP-Methoden einer RESTful API. Sie enthalten wichtige Daten wie ID-Informationen, Authorisierung, den URI, Caching usw.

Es gibt Request- und Response-Header, die jeweils eigene HTTP-Verbindungsinformationen  und Statuscodes aufweisen.

Beispielsweise sendet folgender URI als Paramter noch ein Limit und Offset mit

http://example.com/api/events?limit=20&offset=300

# CRUD-Operationen

Überblick über CRUD-Operationen:

Create (POST)

Retrieve (GET)

Update (PUT/PATCH)

Delete (DELETE)

# CREATE (HTTP POST)

Erstellen neuer Ressourcen mittels POST-Anfragen.

Beispiel URL: POST /api/items/

HTTP-Response-Code:

201 Created - Erfolgreiches Erstellen einer Ressource.

# READ (GET)

Abrufen von Informationen über Ressourcen mittels GET-Anfragen.

Beispiel URLs:

GET /api/items/,

GET /api/items/{id}/

HTTP-Statuscodes:

200 OK - Erfolgreiches Abrufen der Ressource(n).

404 Not Found - Ressource nicht gefunden (bei spezifischer ID).

401 UNAUTHORIZED- Zugriff auf Ressource ist nicht authorisiert

403 FORBIDDEN - Es fehlen spezifische Berechtigungen

# UPDATE (PUT/PATCH)

bestehende Ressourcen aktualisieren

## Beispiel URL:

Für vollständige Aktualisierung: PUT /api/items/{id}/

Für teilweise Aktualisierung: PATCH /api/items/{id}/

## HTTP-Responsecodes:

200 OK - Ressource erfolgreich aktualisiert

404 Not Found - Ressource nicht gefunden.

Unterschiede zwischen PUT und PATCH:

PUT erfordert die vollständige Datenmenge der Ressource.

PATCH erlaubt die Aktualisierung mit nur den geänderten Daten.

# Löschen (Delete)

DELETE /api/items/{id}/

HTTP-Statuscodes:

204 No Content - Ressource erfolgreich gelöscht.

404 Not Found - Ressource nicht gefunden.