Softwareentwurf und Anwendungen verteilter Systeme

Digital Product Design and Development B.A.

Semester 3

Hochschule für Gestaltung Schwäbisch Gmünd

Dozent: Yannick Schiele

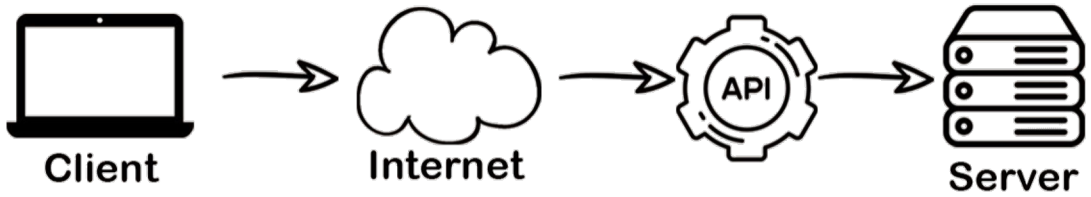
H f G

Hochschule für Gestaltung
Schwäbisch Gmünd

Ergebnisse der Übung

Sensorwerte (als JSON) an Webserver übertragen und darstellen

REST APIs



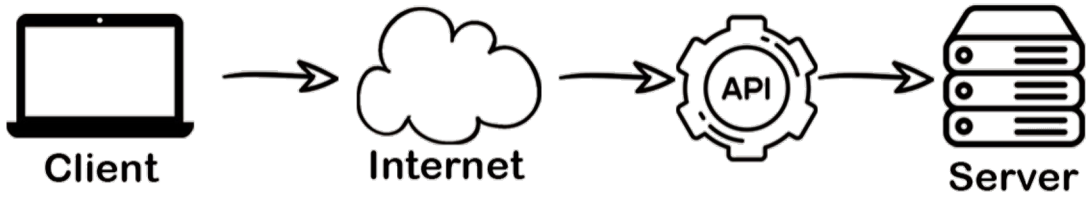
https://www.datacenter-insider.de/was-ist-ein-application-programming-interface-api-a-735797/

APIs

Eine Application-Programming-Interface ist eine Schnittstelle, die es zwei Programmen ermöglicht, miteinander zu kommunizieren

Deutsche Übersetzung: „Schnittstelle zur Anwendungsprogrammierung“ oder „Programmierschnittstelle“

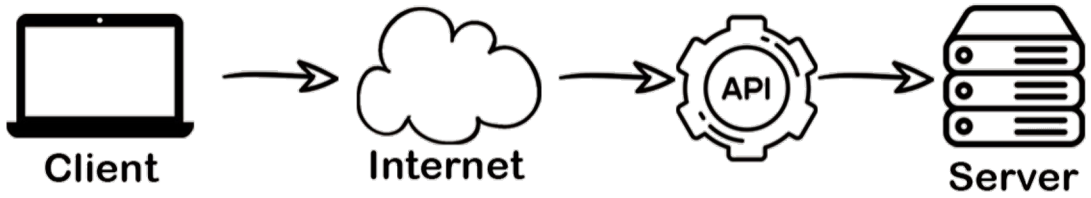
Eine API ist ein Programmteil, das die Verbindung eines Programms zu anderen Programmen oder Systemen auf Quelltext-Ebene ermöglicht



https://www.datacenter-insider.de/was-ist-ein-application-programming-interface-api-a-735797/

APIs im Einsatz

- Bspw. Datenimport aus einem isolierten, entfernten System, wie der Abruf von Inhalten eines externen Daten-Anbieters wie zum Beispiel von Wetterdaten
- APIs kommen auch für die Fernsteuerung von Systemen zum Einsatz (Aktor)
- erlaubt so den Zugriff auf Hardwarekomponenten, Datenbanken, einzelne Programmfunktionen und weitere Elemente
- Bereitstellung idealerweise gemeinsam mit einer ausführlichen Dokumentation der einzelnen Funktionen, der genauen Syntax und der möglichen Parameter



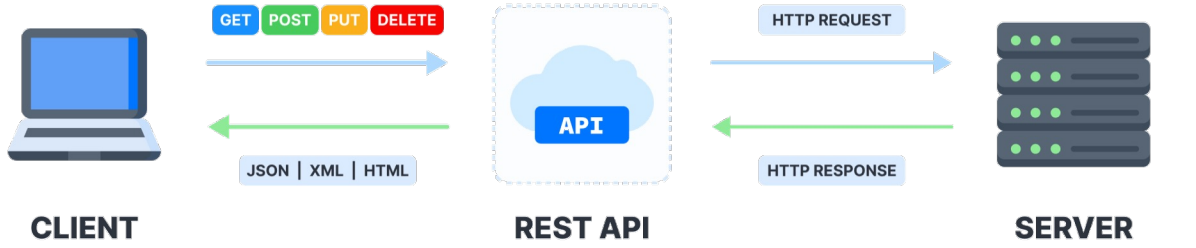
https://www.datacenter-insider.de/was-ist-ein-application-programming-interface-api-a-735797/

Wie eine API funktioniert

Regeln, die erklären, wie Computer oder Anwendungen miteinander kommunizieren

anhand von API-Spezifikation dokumentiert

sitzen zwischen einer Anwendung und dem Webserver und fungieren als Zwischenschicht, die den Datentransfer zwischen Systemen verarbeitet

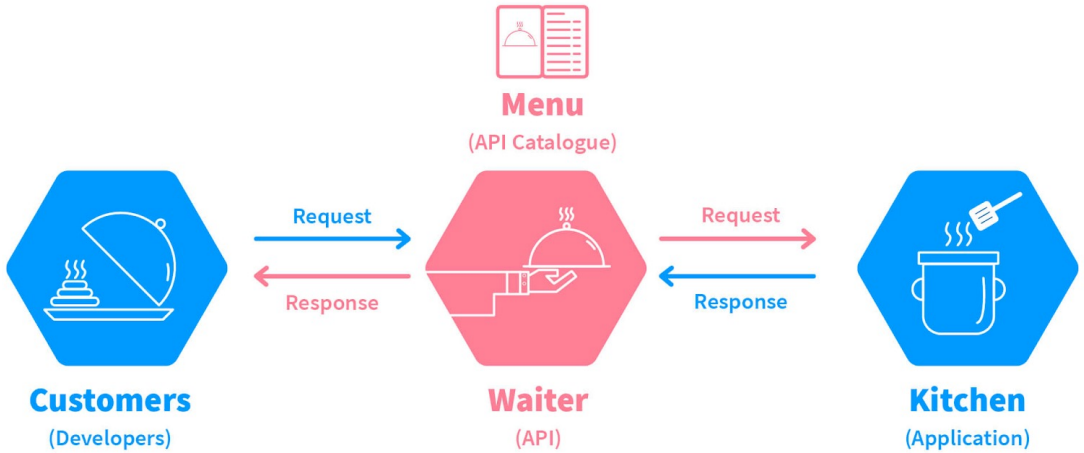


https://www.datacenter-insider.de/was-ist-ein-application-programming-interface-api-a-735797/

Wie eine API funktioniert

1. Eine Client-Anwendung initiiert einen API-Anfrage, die über den Uniform Resource Identifier (URI) an den Webserver weitergeleitet wird
2. Nach dem Empfang einer gültigen Anfrage ruft die API das externe Programm oder den Webserver auf
3. Der Server sendet eine Antwort mit den angeforderten Informationen an die API
4. Die API überträgt die Daten an die ursprünglich anfragende Anwendung

Waiter API Vergleich



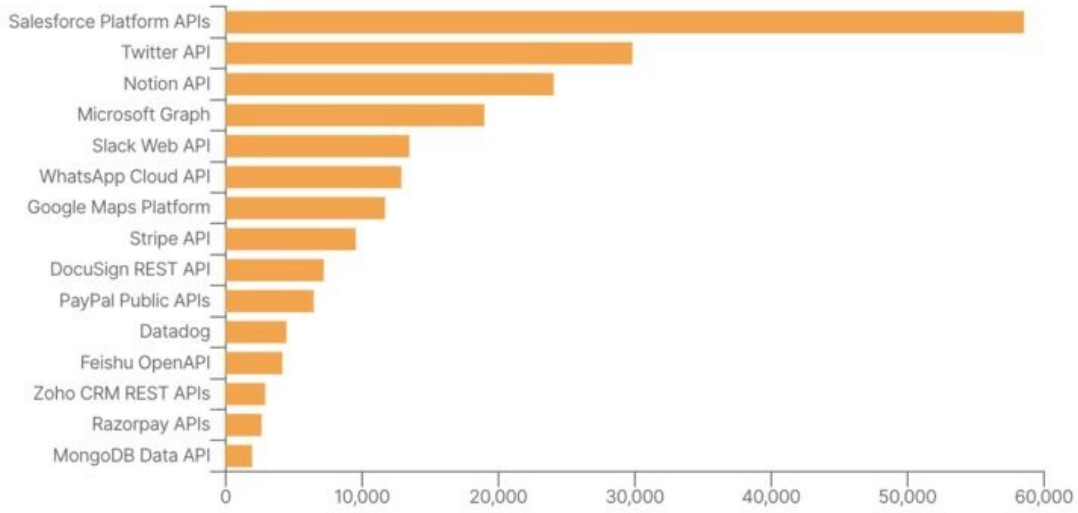
https://www.altexsoft.com/media/2021/03/word-image.png

Vorteile von APIs

- Verbesserte Zusammenarbeit:
  - APIs ermöglichen die Integration, so dass diese Plattformen und Anwendungen nahtlos miteinander kommunizieren können
- Leichtere Innovation:
  - APIs bieten Flexibilität und ermöglichen es, Verbindungen mit neuen Komponenten herzustellen und neue Dienste für das bestehende System anzubieten
- Monetarisierung von Daten:
  - Wenn eine API jedoch Zugang zu wertvollen digitalen Ressourcen gewährt, kann sie durch den Verkauf des Zugangs monetarisiert werden
- Zusätzliche Sicherheit:
  - Die Datensicherheit kann durch die Verwendung von Authentifizierungs-Tokens, Signaturen und Transport Layer Security (TLS)-Verschlüsselung verbessert werden.

Most popular APIs

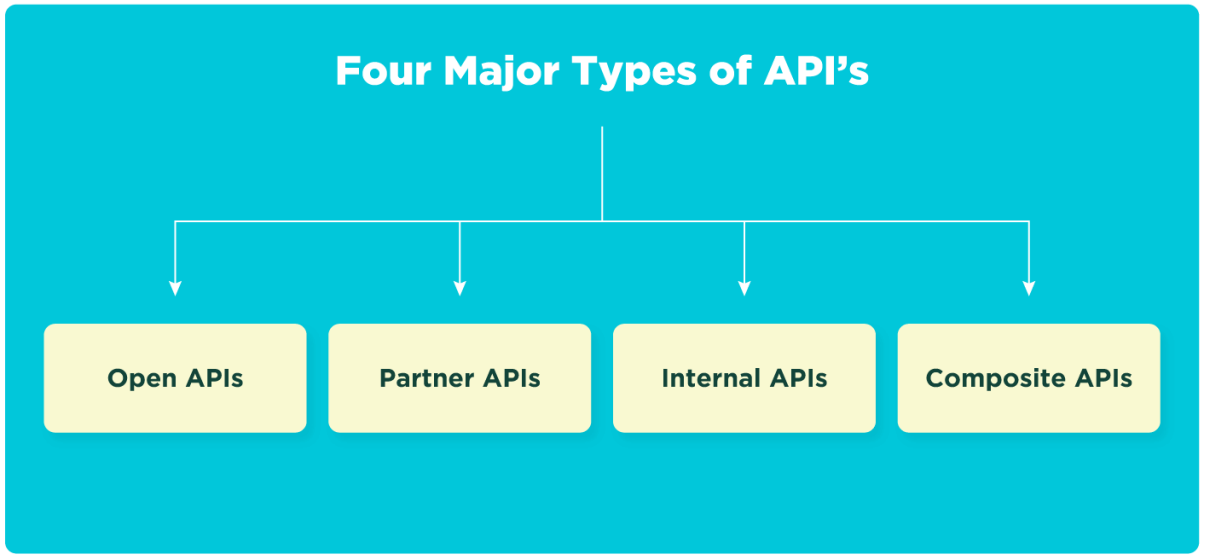
We looked across the Postman Public API Network, the world's largest public API hub, to see which API collections were forked most over the past year. Here's what we found:



https://blog.postman.com/wp-content/uploads/2022/08/image2-2.png

Allgemeine API-Beispiele

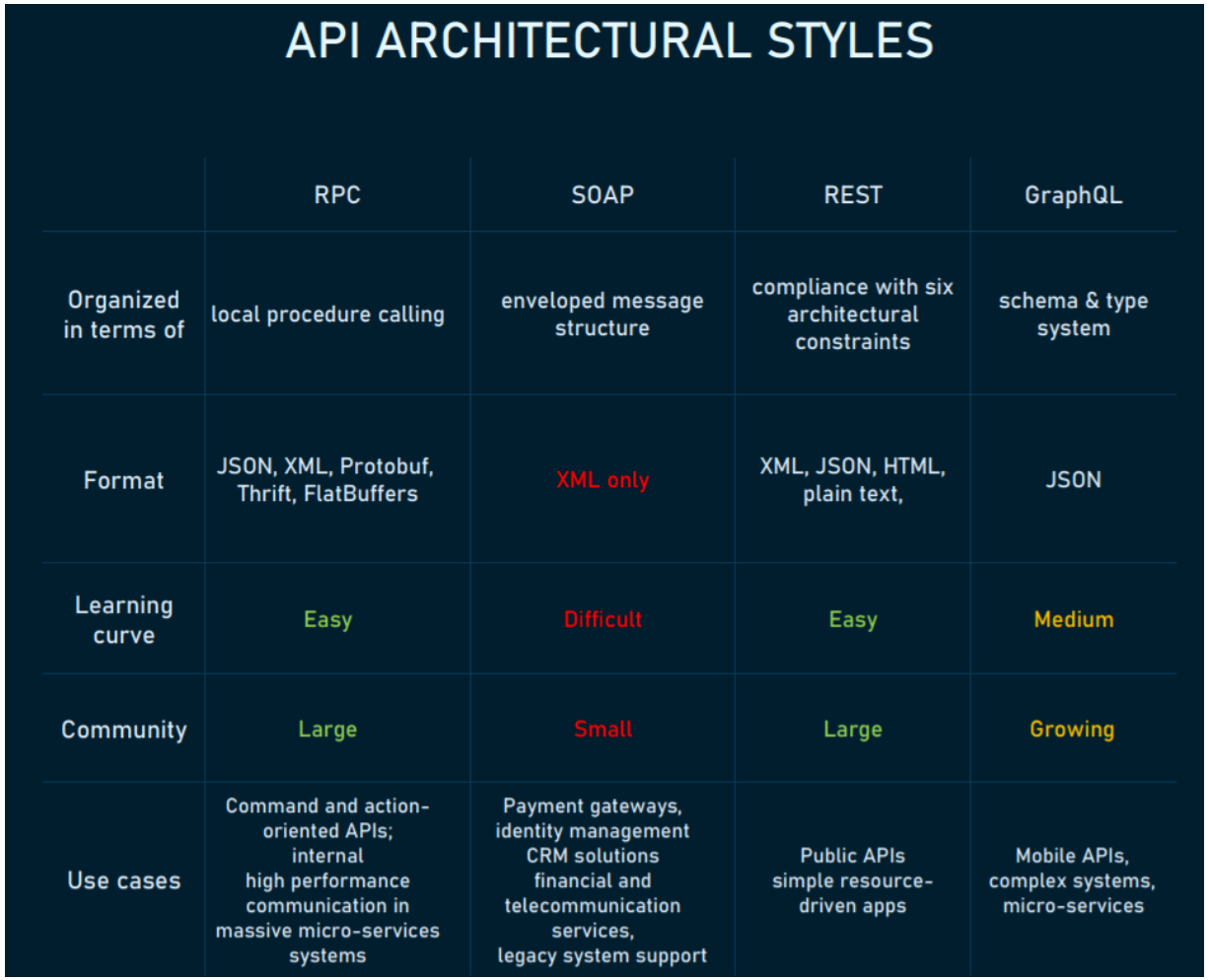
- Universelle Anmeldungen:
  - „Anmelden mit Facebook-, Twitter- oder Google“
- Zahlungsabwicklung von Drittanbietern:
  - "Mit PayPal bezahlen",
- Google Maps:
- Twitter:
  - Jeder Tweet enthält den Autor, eine ID, eine Nachricht, eine Zeitstempels und Geolokalisierungs-Metadaten



https://jelvix.com/wp-content/uploads/2021/02/types-of-api.png

Arten von APIs

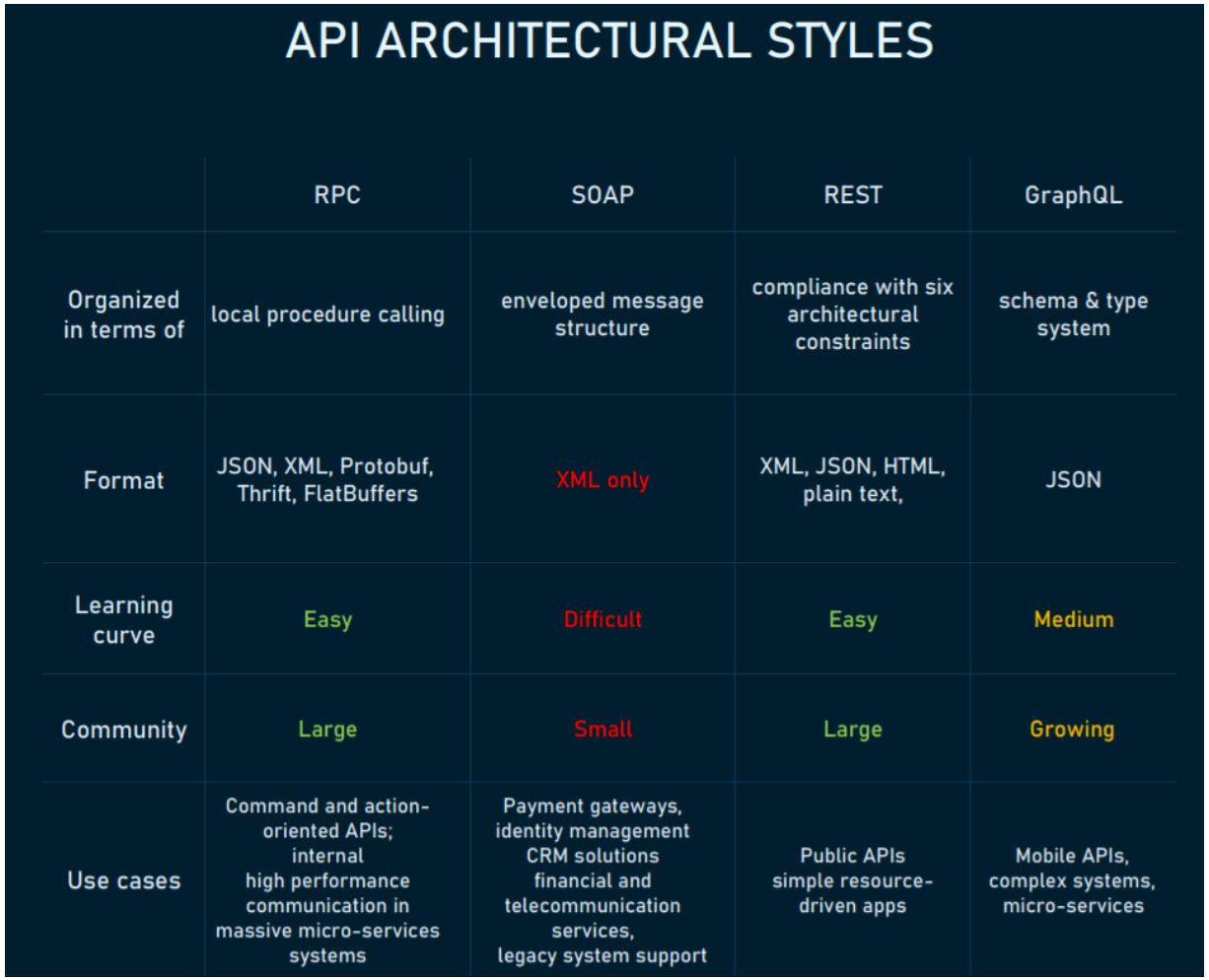
- Open APIs sind quelloffene Schnittstellen zur Anwendungsprogrammierung
- Partner-APIs werden von oder für strategische Partner bereitgestellt. In der Regel können Entwickler auf diese APIs im Self-Service zugreifen.
- Internal APIs bleiben für externe Nutzer verborgen. Diese privaten APIs sind für Benutzer (außerhalb des Unternehmens) nicht verfügbar
- Composite APIs kombinieren mehrere Daten- oder Service-APIs. Diese Dienste ermöglichen es Entwicklern, mit einem einzigen Aufruf auf mehrere Endpunkte zuzugreifen



https://www.mertech.com/hubfs/protocol%20comparison.webp

Arten von API-Protokollen

- SOAP (Simple Object Access Protocol)
  - ein auf XML basierendes API-Protokoll, mit dem Benutzer Daten über SMTP und HTTP senden und empfangen können
  - Mit SOAP-APIs ist es einfacher, Informationen zwischen Anwendungen auszutauschen, die in unterschiedlichen Umgebungen laufen
- XML-RPC
  - ein Protokoll, das ein bestimmtes XML-Format für die Datenübertragung verwendet,
  - XML-RPC ist älter als SOAP, aber einfacher und relativ leichtgewichtig, da es nur ein Minimum an Bandbreite benötigt

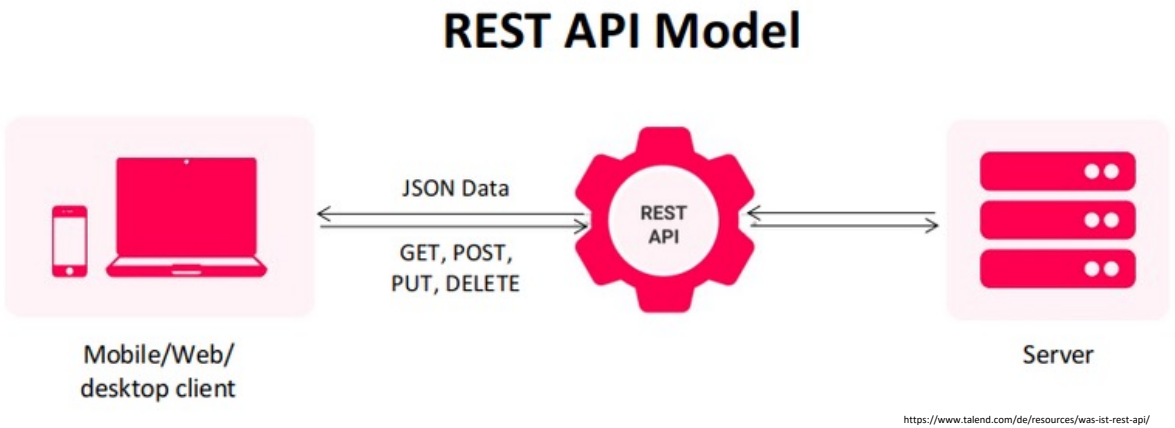


https://www.mertech.com/hubfs/protocol%20comparison.webp

Arten von API-Protokollen

- JSON-RPC
  - ähnliches Protokoll wie XML-RPC, da es sich bei beiden um Remote Procedure Calls (RPCs) handelt
  - aber dieses Protokoll verwendet JSON anstelle des XML-Formats zur Datenübertragung

- REST (Representational State Transfer)
  - eine Reihe von Grundsätzen der Web-API-Architektur, was bedeutet, dass es keine offiziellen Standards gibt.
  - Um eine REST-API zu sein, muss die Schnittstelle bestimmte architektonische Einschränkungen einhalten



https://www.talend.com/de/resources/was-ist-rest-api/

REST APIs

- REST- (Representational State Transfer) hat sich über die Jahre zur vielseitigsten und nützlichsten Webservice-API entwickelt
- ist eine Schnittstelle mit der zwei Computersysteme Informationen auf sichere Weise über das Internet austauschen können
- unterstützen diesen Informationsaustausch, da sie sicheren, zuverlässigen und effizienten Software-Kommunikationsstandards folgen

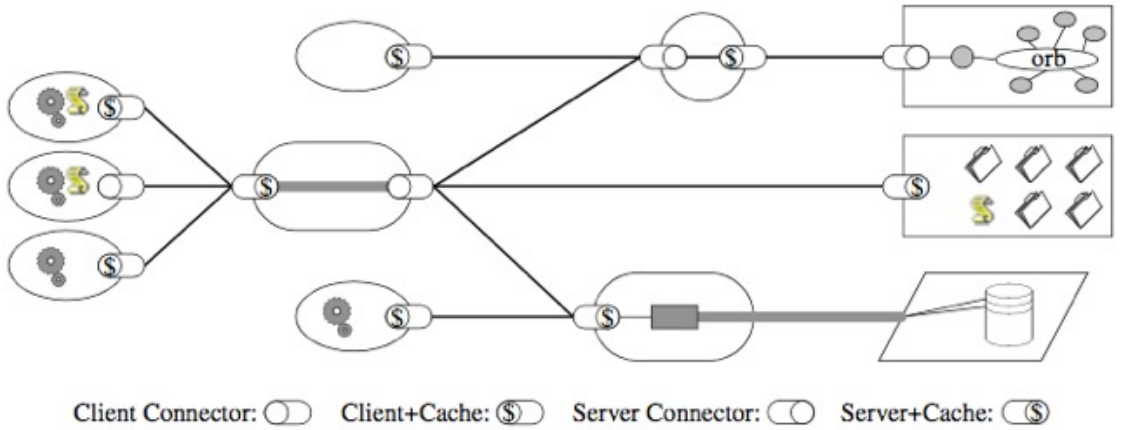
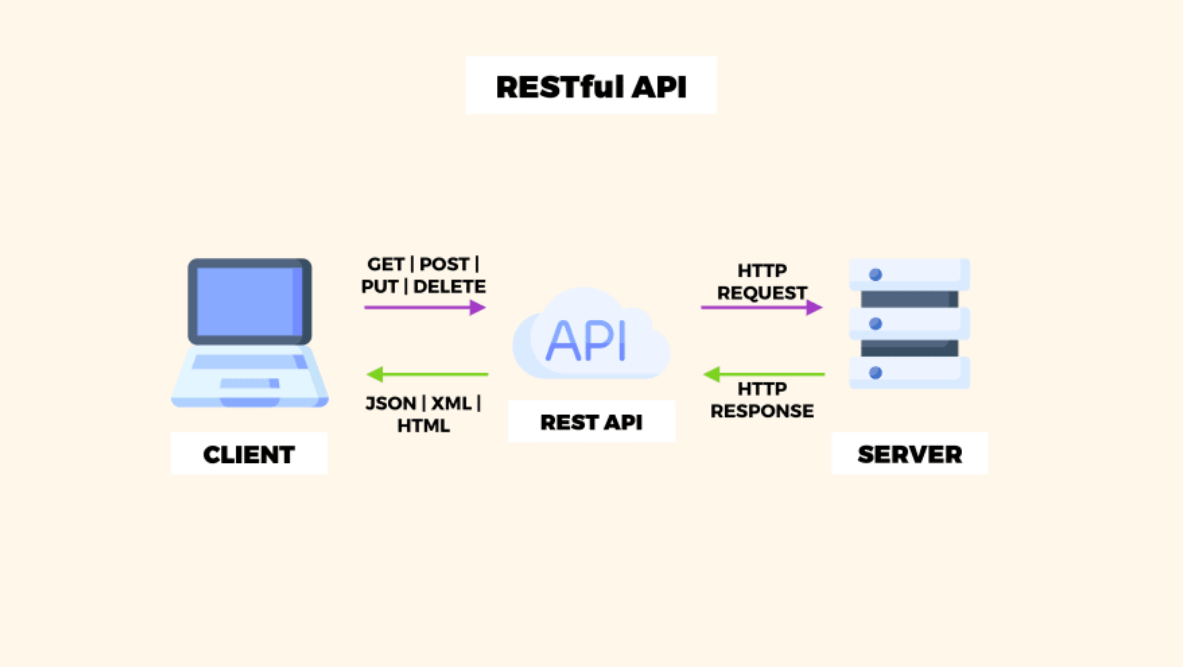


Figure 5-8. REST https://home.cs.colorado.edu/~kena/classes/7818/f08/lectures/lecture_9_fielding_disserta.pdf

REST APIs

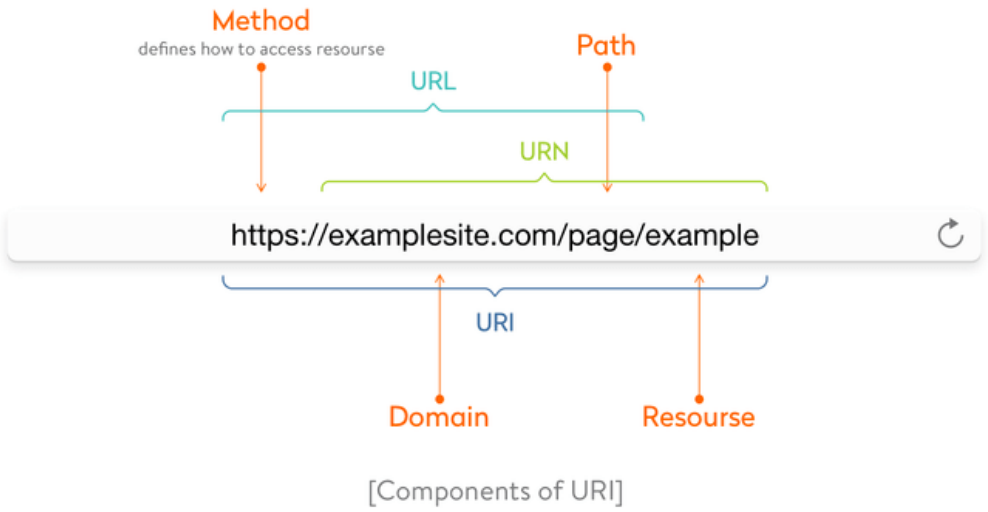
- Bis zum Jahr 2000 war SOAP, die am weitesten verbreitete Plattform für Client-Server-Interaktionen
- SOAP hatte aber zwei Nachteile:
  - Erstens mussten sich die Nutzer bei der Interaktion mit dem Server an strenge Regeln halten
  - Zweitens basierte das Protokoll auf dem XML-Format
- Der US-amerikanische Informatiker Roy Fielding entwickelte im Rahmen seiner Doktorarbeit aus dem Jahr 2000 eine Alternative zu SOAP namens „REST“
- Die REST-Schnittstelle hebt sich durch ihre größere Flexibilität, eine höhere Geschwindigkeit und eine geringere Bandbreite ab
- Heute ist REST eine der gängigsten APIs und wird von den meisten großen Plattformen wie Amazon, Facebook, Twitter und Google verwendet



https://www.talend.com/de/resources/was-ist-rest-api/

Architektonische Bedingungen der REST-API

1. Einheitliche Schnittstelle
2. Client-Server-Architektur
3. Zustandslosigkeit (Statelessness)
4. Schichtsystem
5. Zwischenspeicher
6. Code-on-demand



[Components of URI]

https://www.talend.com/de/resources/was-ist-rest-api/

Architektonische Bedingungen der REST-API

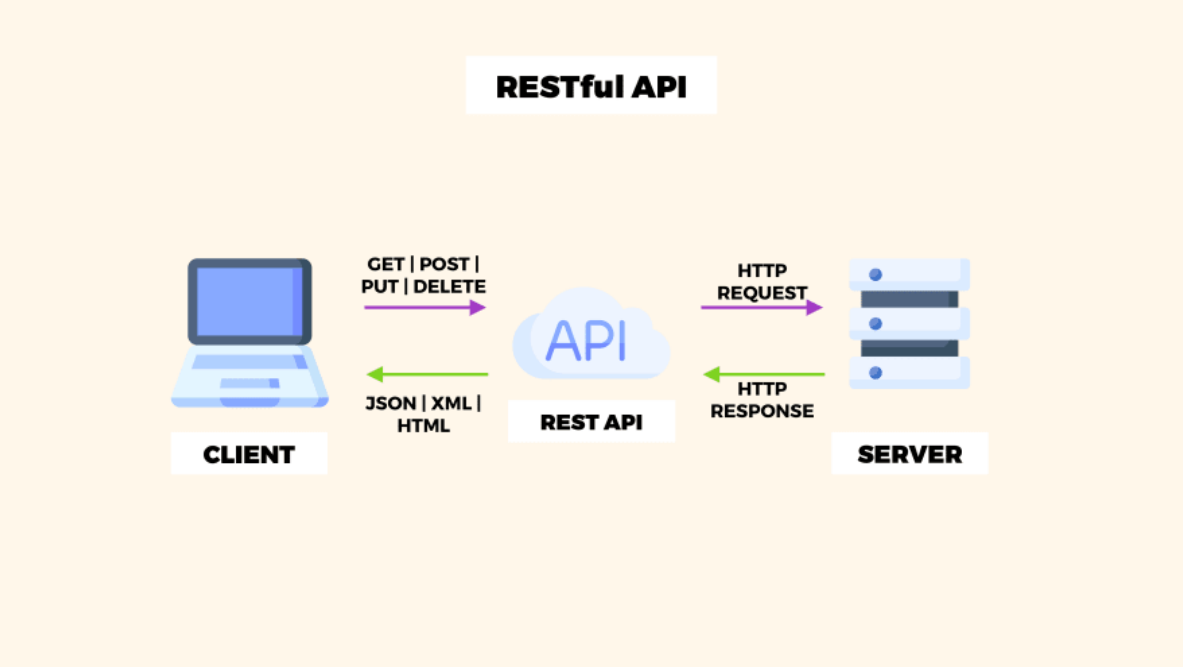
1. Einheitliche Schnittstelle

   bedeutet, dass jeder REST-Client den Server auf die gleiche Weise aufrufen kann– unabhängig ob JavaScript-Code oder mobile Anwendung

   Die Verwendung einer URI (Unique Resource Identifier), die oft eine URL wie zum Beispiel „https://api.twitter.com/“ ist, macht dies möglich

   Anfragen und Antworten müssen autark sein – der Client sendet alle Informationen, die der Server benötigt, um die Anfrage zu verarbeiten.

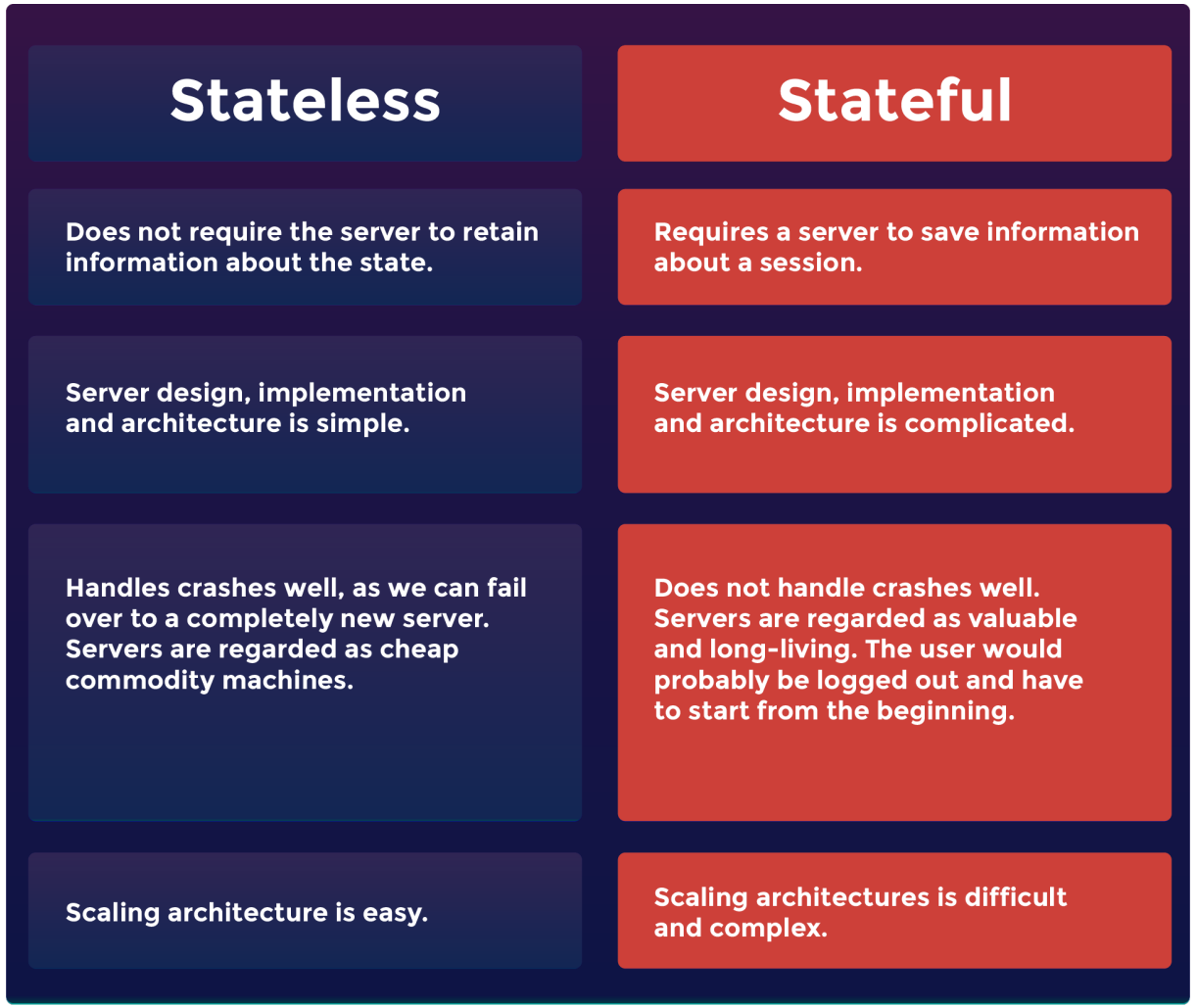
   Der Server antwortet mit einer Information darüber, die angeforderte Ressource zu ändern, zu löschen oder auf sie zuzugreifen



https://www.talend.com/de/resources/was-ist-rest-api/

Architektonische Bedingungen der REST-API

2. Client-Server-Architektur
   - In der REST-Architektur sind die Client- und Server-Komponenten klar definiert und voneinander getrennt
   - Dieser Aufbau ermöglicht es, dass Änderungen an beiden Komponenten unabhängig voneinander vorgenommen werden können, ohne dass dies Auswirkungen auf die jeweils andere Komponente hat
   - Die meisten Web-Clients und -Server sind auf diese Weise konzipiert

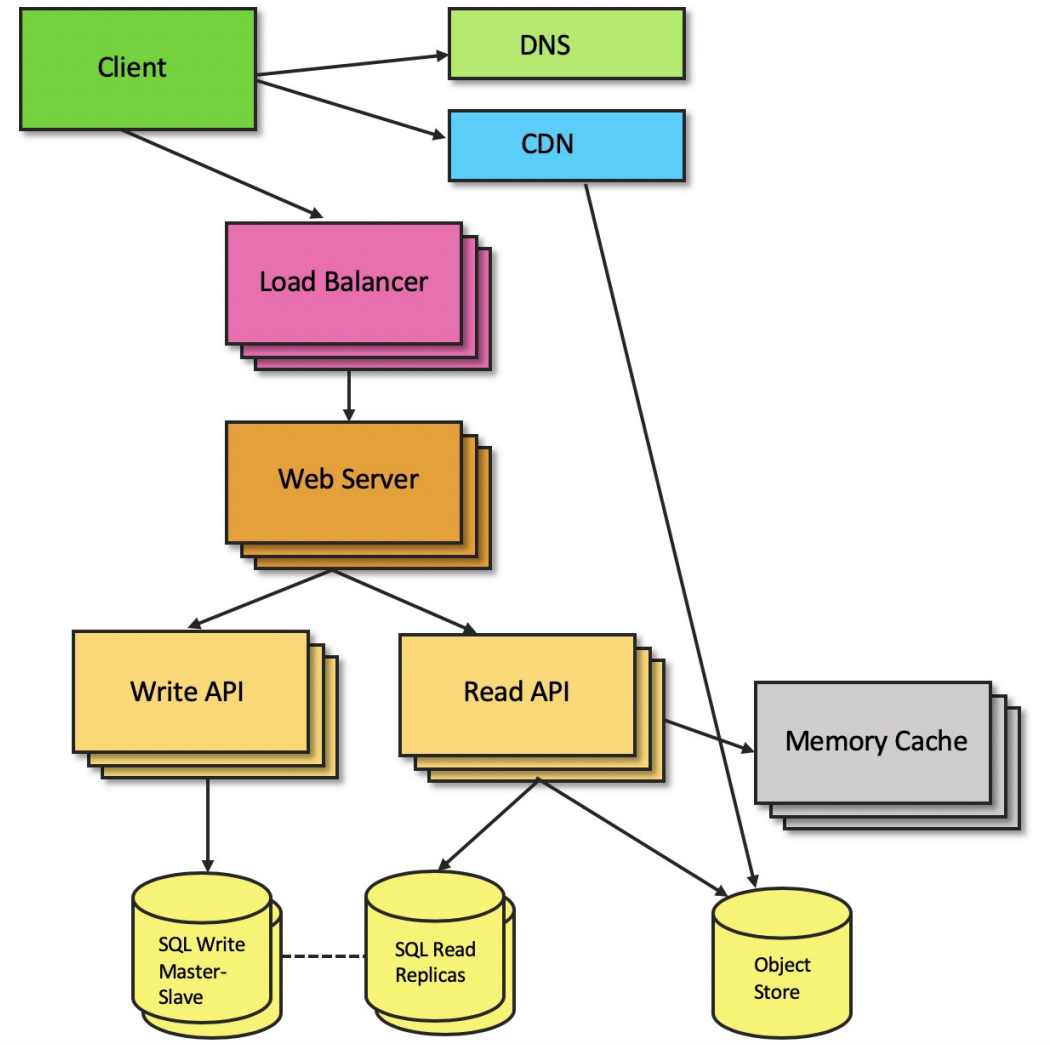


Stateless	Stateful
Does not require the server to retain information about the state.	Requires a server to save information about a session.
Server design, implementation and architecture is simple.	Server design, implementation and architecture is complicated.
Handles crashes well, as we can fail over to a completely new server. Servers are regarded as cheap commodity machines.	Does not handle crashes well. Servers are regarded as valuable and long-living. The user would probably be logged out and have to start from the beginning.
Scaling architecture is easy.	Scaling architectures is difficult and complex.

https://www.talend.com/de/resources/was-ist-rest-api/

Architektonische Bedingungen der REST-API

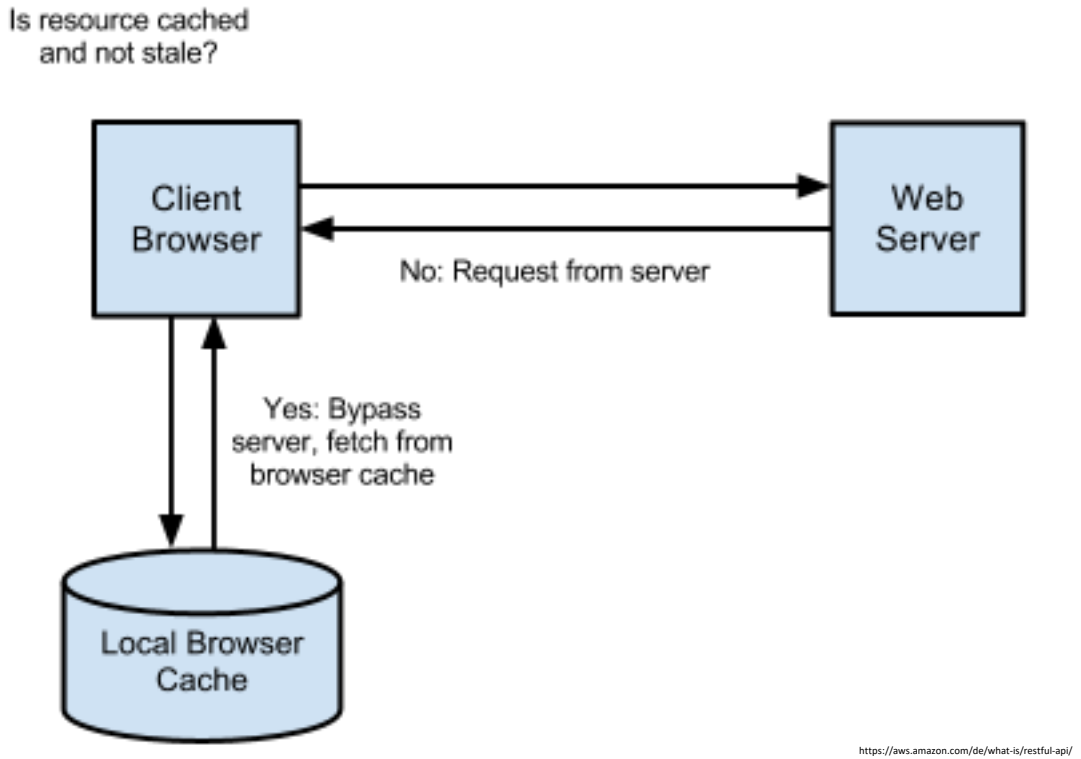
3. Zustandslosigkeit (Statelessness)
   - bedeutet, dass der Server den Anwendungszustand des Clients nicht in einer Datenbank abspeichern muss
   - in der Anfrage müssen alle notwendigen Informationen für den Datenaustausch enthalten sein
   - bedeutet auch, dass ausschließlich der Client dafür zuständig ist, den Anwendungszustand aufrecht zu erhalten
   - Server wird entlastet, da er keine Overhead-Daten über Client-Aufrufe speichern muss, um künftige Anfragen zu bearbeiten
   - Die Skalierbarkeit ist in einer zustandslosen Architektur einfacher.
   - kann jedoch passieren, dass der Client die Server mehrmals kontaktieren muss, um Daten zu erhalten, was die Leistung beeinträchtigt.



https://www.google.com/url?sa=i&url=https%3A%2F%2Fdanmartensen.svbtle.com%2Fexploring-rest-api-architecture&psig=AOvVaw09hYY8HkWKGkMXMfY0X9wv&ust=1667745208934000&source=images&cd=vfe&ved=0CA0QjRxqFwoTCKiq8JihI_sCFQAAAAAdAAAAABAN

Architektonische Bedingungen der REST-API

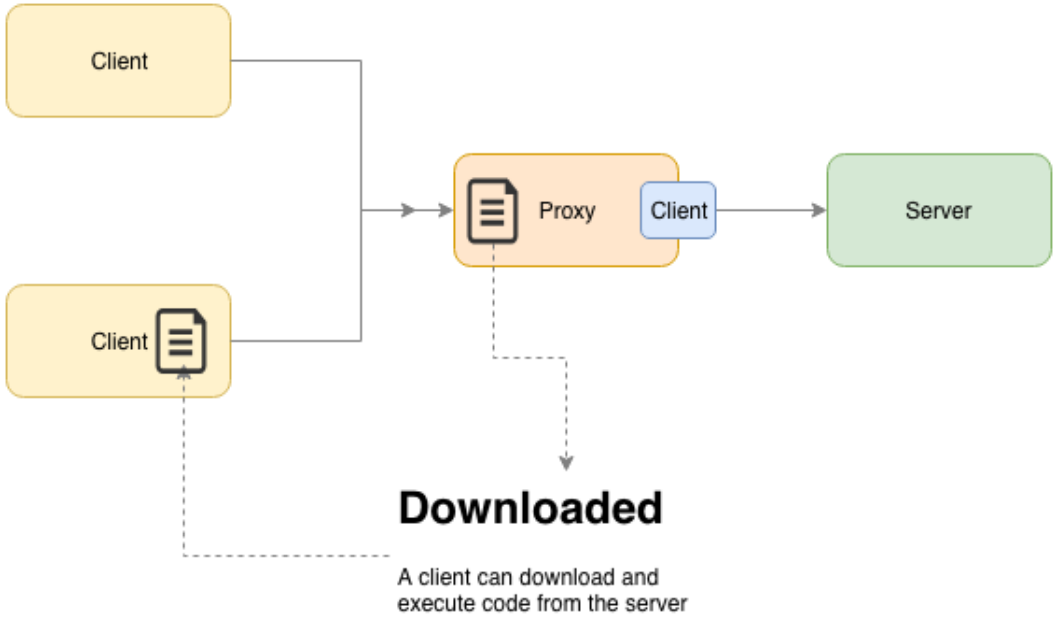
4. Schichtsystem
   - beschreibt die vollständige Trennung von Client und Server
   - Wenn ein Client eine Anfrage über eine Server-Endpunkt-URL stellt, muss er nicht genau wissen, welcher Server seine Anfrage bearbeitet
   - Es kann verschiedene Server geben, die alle eine eigene Aufgabe haben, wie z. B. Load Balancing, Caching usw.
   - Der Client bekommt von all diesen Aktionen im Hintergrund nichts mit.
   - Die Schichtung verleiht der REST-API zusätzliche Sicherheit, da Angriffe innerhalb einzelner Schichten isoliert und eingedämmt werden können.



https://aws.amazon.com/de/what-is/restful-api/

Architektonische Bedingungen der REST-API

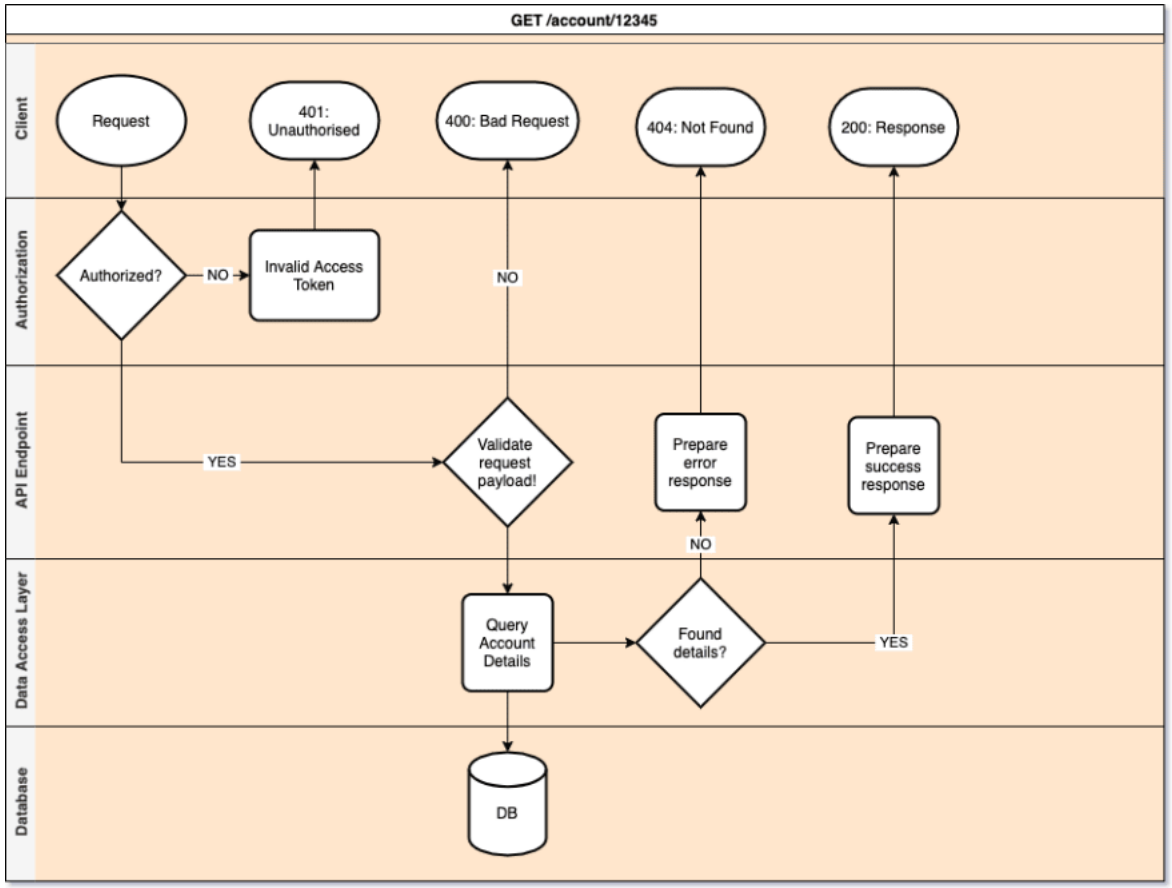
5. Zwischenspeicher
   - RESTful-Webservices unterstützen das Caching.
   - Das ist der Prozess des Zwischenspeicherns einiger Antworten auf dem Client oder auf einem Vermittler, um die Reaktionszeit des Servers zu verbessern
   - RESTful-Webservices kontrollieren das Caching mithilfe von API-Responses , die sie als cachebar definieren



https://www.talend.com/de/resources/was-ist-rest-api/

Architektonische Bedingungen der REST-API

6. Code-on-demand
   - Server können vorübergehend die Client-Funktionalität erweitern, indem Software-Programmiercode an den Client übertragen wird
   - Wenn zum Beispiel ein Anmeldeformular auf einer Website ausgefüllt wird, hebt der Browser Fehler hervor, wie bspw. eine falsche Telefonnummer.
   - Der Browser kann das aufgrund des vom Server gesendeten Codes tun



https://dev.to/raevilman/rest-api-flowchart-with-swimlanes-36lc

Wie funktionieren RESTful-APIs?

Der Client nimmt mithilfe der API Kontakt zum Server auf, wenn eine Ressource erforderlich ist. Die API-Dokumentation erklärt, wie sie verwendet werden sollte.

Schritte für die Durchführung eines REST-API-Aufrufs:

- Der Client sendet eine Anforderung an den Server. Der Client folgt der API-Dokumentation, um die Anforderung so zu formatieren, dass der Server sie versteht.
- Der Server authentifiziert den Client und bestätigt, dass der Client berechtigt ist, diese Anforderung zu stellen.
- Der Server empfängt die Anforderung und verarbeitet sie
- Der Server sendet eine Antwort an den Client zurück. Die Antwort enthält Informationen, die dem Client mitteilen, ob die Anforderung erfolgreich war und die Informationen, die der Client angefordert hat.

https://www.sitepoint.com/rest-api/

RESTful API Request

Unique Resource Identifier

- Der Server identifiziert jede Ressource mit Unique Resource Identifiers
- Der Server führt die Ressourcen-Identifizierung mit dem Uniform Resource Locator (URL) durch
- Die URL (ähnlich Website Adresse) gibt den Pfad zur Ressource an
- Der URL wird als Endpunkt der Anforderung bezeichnet und gibt für den Server an, was der Client anfordert

https://www.sitepoint.com/rest-api/

RESTful API Request

Methode

RESTful-APIs basieren meist auf HTTP. Die HTTP-Methode teilt dem Server mit, was er mit der Ressource machen soll.

GET

> Clients verwenden GET zum Zugriff auf Ressourcen, die sich in der angegebenen URL auf dem Server befinden.

POST

> Clients verwenden POST zum Senden von Daten an den Server. Dazu gehört die Daten-Repräsentation mit der Anforderung.

https://www.sitepoint.com/rest-api/

RESTful API Request

Methode

RESTful-APIs basieren meist auf HTTP. Die HTTP-Methode teilt dem Server mit, was er mit der Ressource machen soll.

PUT

Clients verwenden PUT zum Aktualisieren von bestehenden Ressourcen auf dem Server.

DELETE

Die Clients verwenden die DELETE-Anforderung zum entfernen der Ressource.

https://www.sitepoint.com/rest-api/

RESTful API Request

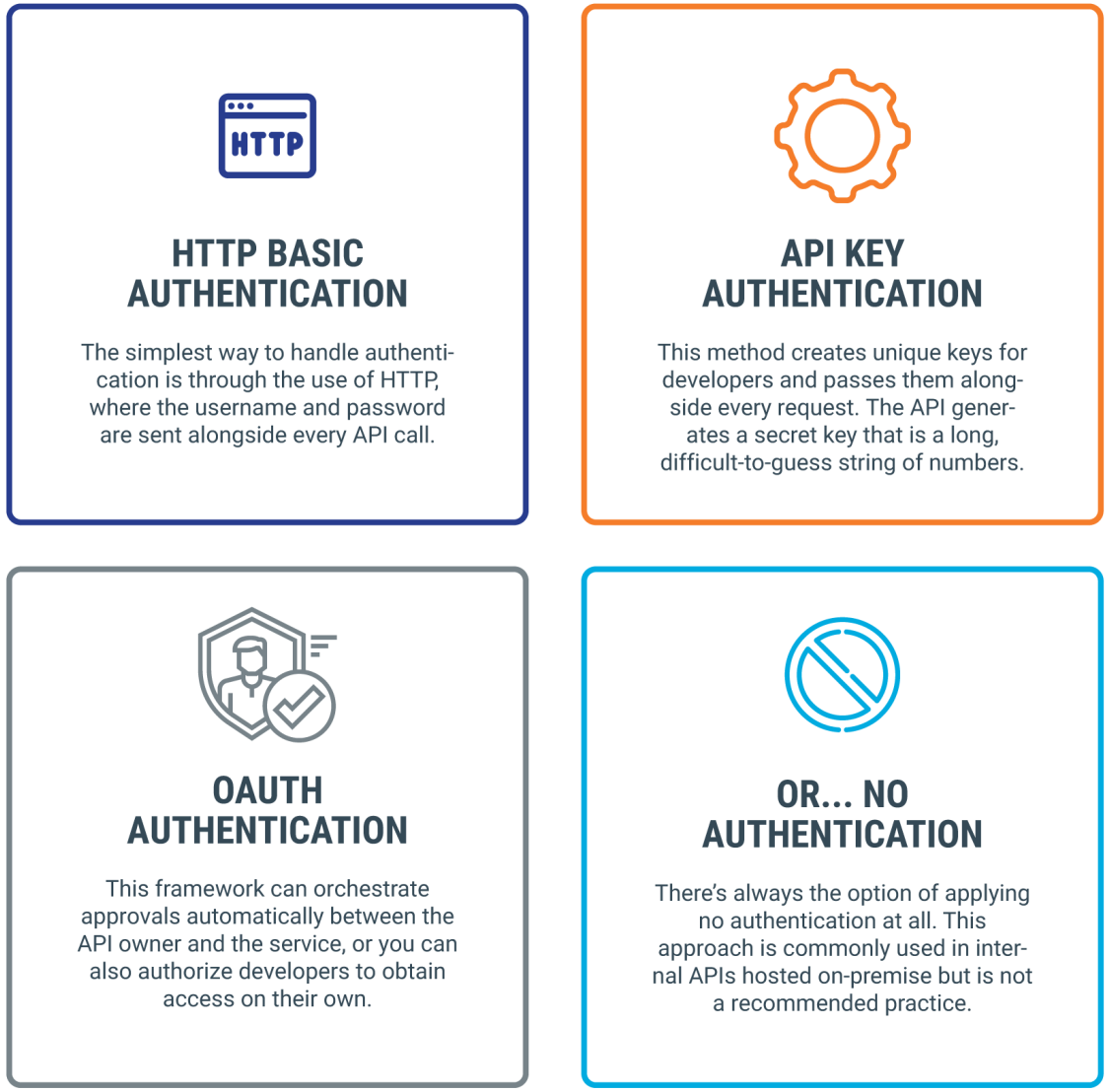
HTTP-Header

Die Kopfzeilen der Anforderung sind die Metadaten, die zwischen dem Client und dem Server ausgetauscht werden.

content-type gibt das Datenformat in dem Request und *accept* welche Inhaltstypen der Client verarbeiten kann, an

Vollständige Dokumentation aller Header Felder:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers?retiredLocale=de



https://www.3pillarglobal.com/insights/most-popular-api-authentication-methods/

RESTful-API-Authentifizierung

Ein RESTful-Webservice muss Anforderungen authentifizieren, bevor er eine Reaktion senden kann. Die Authentifizierung ist der Prozess der Verifizierung einer Identität.

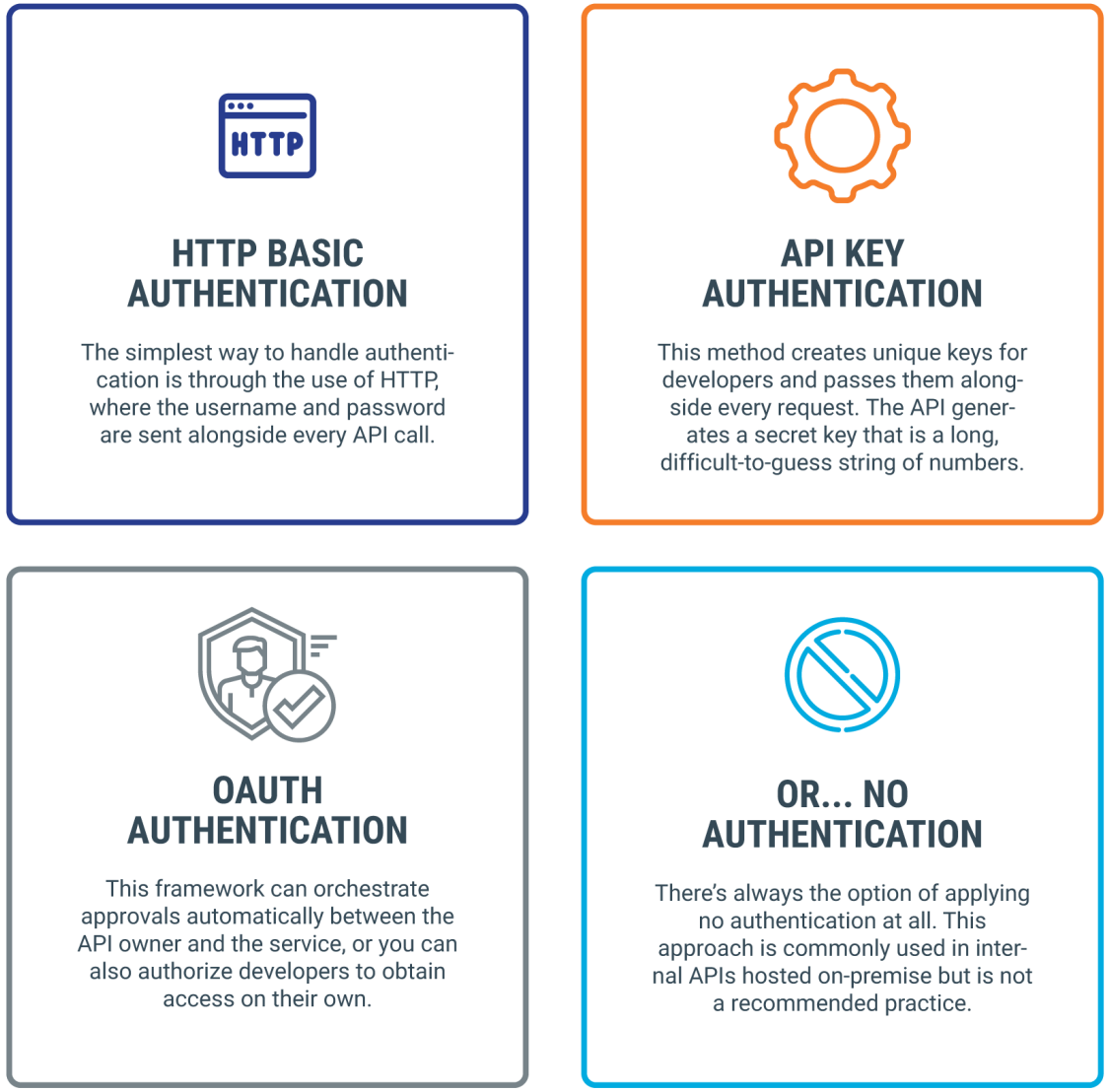
Die RESTful-API verfügt über vier übliche Authentifizierungs-Methoden:

Basis Authentifizierung

Bei der basic Authentifizierung sendet der Client den Benutzernamen und das Passwort in der Kopfzeile der Anforderung.

Bearer-Authentifizierung

Der Begriff Bearer-Authentifizierung bezieht sich auf den Prozess der Gewährleistung der Zugriffskontrolle für den Token-Bearer (Token-Inhaber). Das Bearer-Token ist üblicherweise eine verschlüsselte Zeichenfolge, die der Server als Antwort auf einer Anmeldeanforderung generiert. Der Client sendet das Token in den Kopfzeilen der Anforderung, um auf Ressourcen zugreifen zu können.



https://www.3pillarglobal.com/insights/most-popular-api-authentication-methods/

RESTful-API-Authentifizierung

API-Keys

Der Server weist einen eindeutigen generierten Wert für einen neuen Client zu. Jedes Mal, wenn der Client versucht, auf Ressourcen zuzugreifen, verwendet er den eindeutigen API-Schlüssel, um sich selbst zu verifizieren. API-Schlüssel sind weniger sicher, weil der Client den Schlüssel übertragen muss, was ihn gegenüber Netzwerk-Diebstahl anfällig macht.

OAuth

OAuth kombiniert Passwörter und Tokens, um einen äußerst sicheren Anmeldezugriff auf alle Systeme zu gewährleisten. Der Server fordert zunächst ein Passwort an und fordert anschließend ein zusätzliches Token an, um den Authorisierungsprozess abzuschließen. Das Token kann jederzeit und auch im Laufe der Zeit mit einem bestimmten Umfang und einer bestimmten Langlebigkeit überprüft werden

2- GET - Get item list - HTTP Response Code: **200**

```
HTTP/1.1 200
Pagination-Count: 100
Pagination-Page: 5
Pagination-Limit: 20
Content-Type: application/json

[
  {
    "id": 10,
    "name": "shirt",
    "color": "red",
    "price": "$123"
  },
  {
    "id": 11,
    "name": "coat",
    "color": "black",
    "price": "$2300"
  }
]
```

3- POST - Create a new item - HTTP Response Code: **201**

```
HTTP/1.1  201
Location: /v1/items/12
Content-Type: application/json

{
  "message": "The item was created successfully"
}
```

https://gist.github.com/igorjs/407ffc3126f6ef2a6fe8f918a0673b59

RESTful API Response

REST-Grundsätze erfordern, dass die Antwort die folgenden Hauptkomponente enthält:

Statuszeile

Die Statuszeile enthält den HTTP Statuscode, der den Erfolg oder den Fehlschlag der Kommunikation mitteilt

Text der Nachricht

Der Text der Antwort enthält die Repräsentation der Ressource. Der Server wählt ein Datenformat auf Grundlage von dem aus, was die Anfrage-Kopfzeilen enthalten.

Kopfzeilen

Die Antwort enthält auch Kopfzeilen oder Metadaten zur Antwort. Sie stellen mehr Kontext über die Antwort zur Verfügung und umfassen Informationen wie Server, Verschlüsselung und Art des Inhalts.

Vollständige Übersicht:

https://gist.github.com/igorjs/407ffc3126f6ef2a6fe8f918a0673b59

RESTful API Response

Übersicht der verwendeten Response Codes

GET

+ 200 OK
+ 400 Bad Request
+ 401 Unauthorized
+ 403 Forbidden
+ 404 Not Found
+ 500 Internal Server Error

POST

+ 201 Created
+ 400 Bad Request
+ 401 Unauthorized
+ 403 Forbidden
+ 409 Conflict
+ 500 Internal Server Error

PUT

+ 200 OK
+ 201 Created*
+ 204 No Content
+ 400 Bad Request
+ 401 Unauthorized
+ 403 Forbidden
+ 500 Internal Server Error

DELETE

+ 204 No Content
+ 400 Bad Request
+ 401 Unauthorized
+ 403 Forbidden
+ 500 Internal Server Error

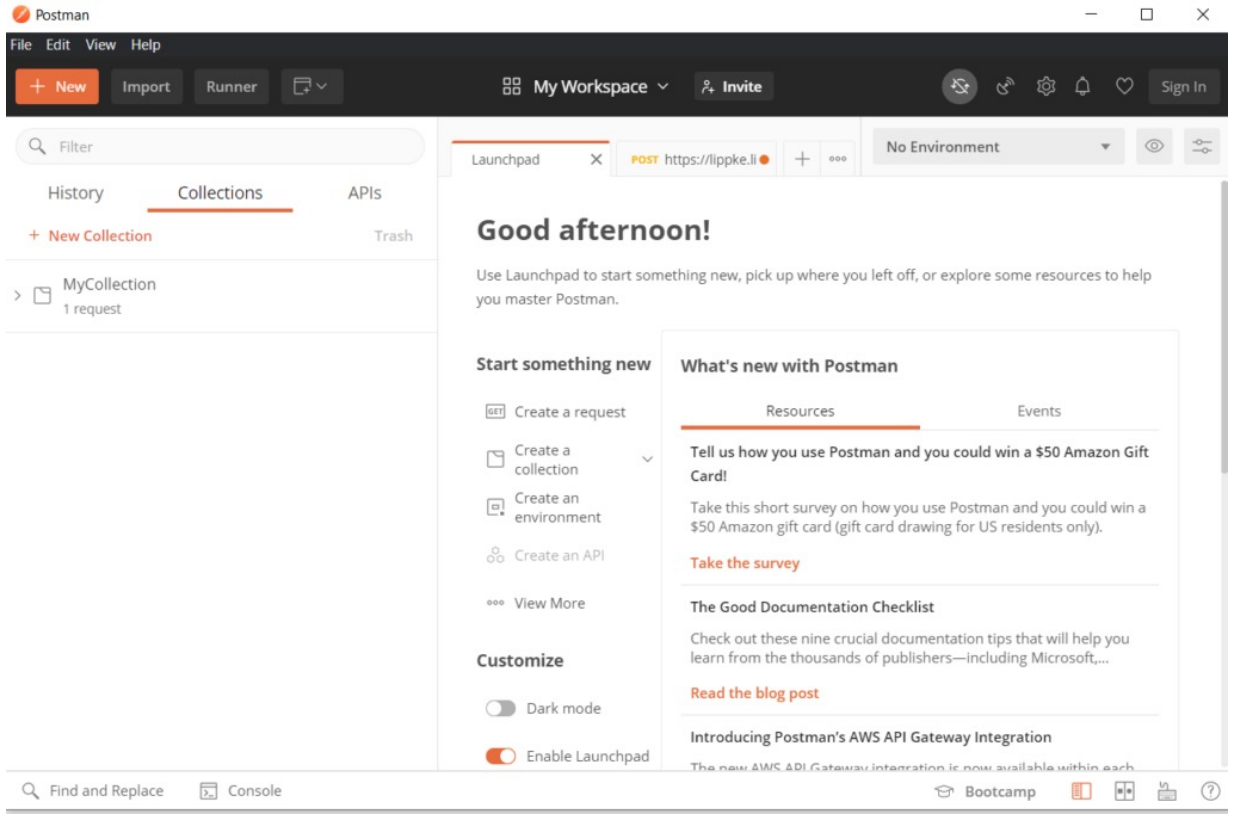
https://restfulapi.net/http-status-codes/

https://www.postman.com/downloads/

POSTMAN

- POSTMAN ist ein API-Client zum Entwickeln und Testen von APIs
- Sendet modifizierte HTTP-Anfragen an einen Server
- Parameter, Inhalte und Methoden können sehr einfach angepasst werden

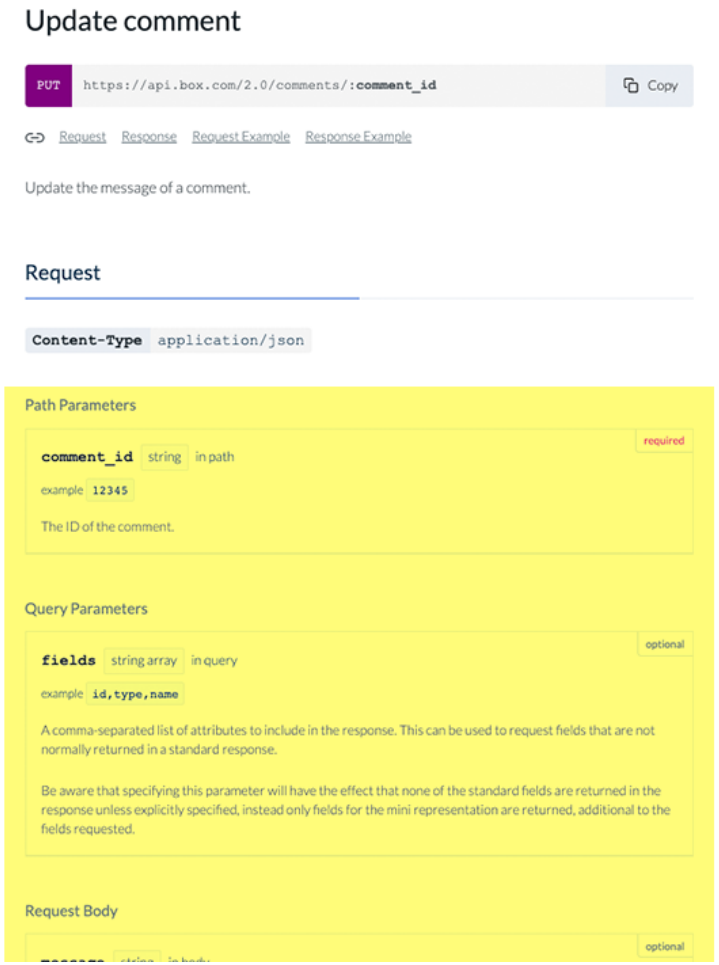
- Download unter:
  - https://www.postman.com/downloads/



https://www.postman.com/downloads/

POSTMAN

- Neue Collection hinzufügen
- Neuen Request erstellen:
  - GET http://dummy.restapiexample.com/api/v1/employees
  - Request & Header anschauen
- Response anschauen:
  - Headers & Body
  - Statistiken
- Weitere Beispiel APIs:
  - http://dummy.restapiexample.com
- Übung:
  - Alle Requests anlegen und ausführen



https://idratherbewriting.com/learnapidoc/docapis_doc_parameters.html

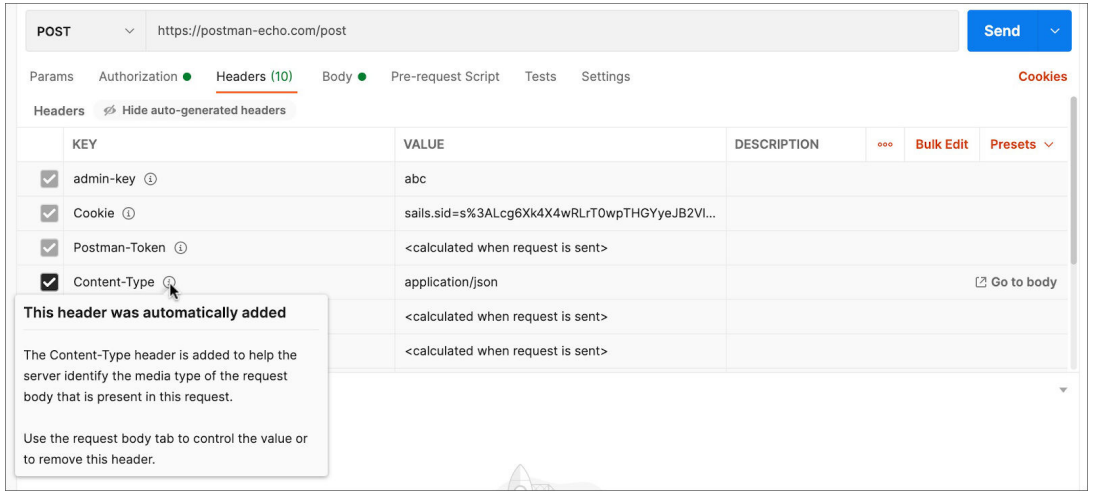
REST API Parameter

Parameter sind Optionen, die dem Endpunkt mitgegeben werden können um die Antwort zu beeinflussen.

Es gibt mehrere Arten von Parametern:

1. Header-Parameter
2. Path-Parameter
3. Query-Parameter

Request Bodies sind den Parametern sehr ähnlich, sind aber technisch gesehen keine Parameter

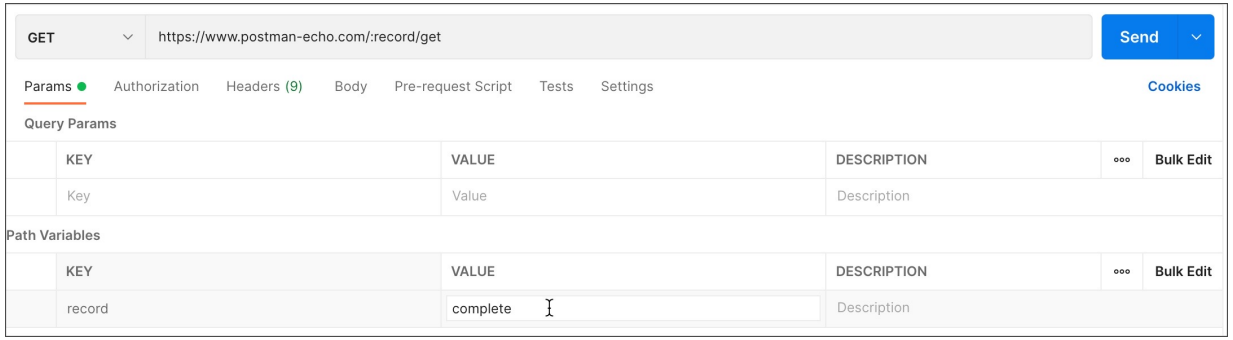


REST API Header Parameter

Header-Parameter werden in den Request-Header aufgenommen

Normalerweise enthält der Header nur Autorisierungsparameter, die für alle Endpunkte gleich sind

Wenn ein Endpunkt jedoch eindeutige Parameter für die Übergabe im Header benötigt, werden diese in der Parameterdokumentation für jeden Endpunkt dokumentiert

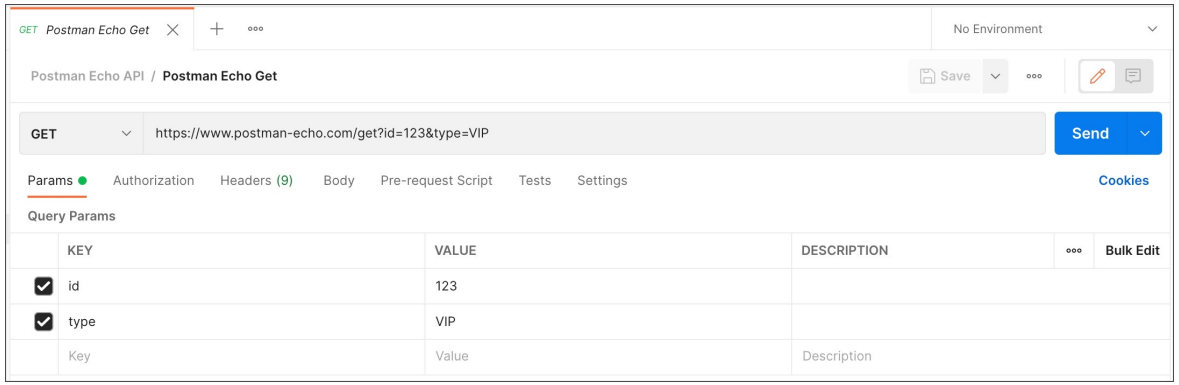


REST API Path Parameter

Path-Parameter sind Teil des Endpunkts selbst und sind nicht optional. Im folgenden Endpunkt sind `{user}` und `{bicycleId}` erforderliche Path Parameter:

```
/service/myresource/user/{user}/bicycles/{bicycleId}
```

Path-Parameter werden normalerweise mit geschweiften Klammern dargestellt, aber einige API-Dokumentationsstile stellen dem Wert einen Doppelpunkt voran (wie bei Postman)



REST API Query Parameter

Die Query-Parameter erscheinen nach einem „?“ im Endpunkt

Das Fragezeichen, gefolgt von den Parametern und ihren Werten, wird als "query string" bezeichnet

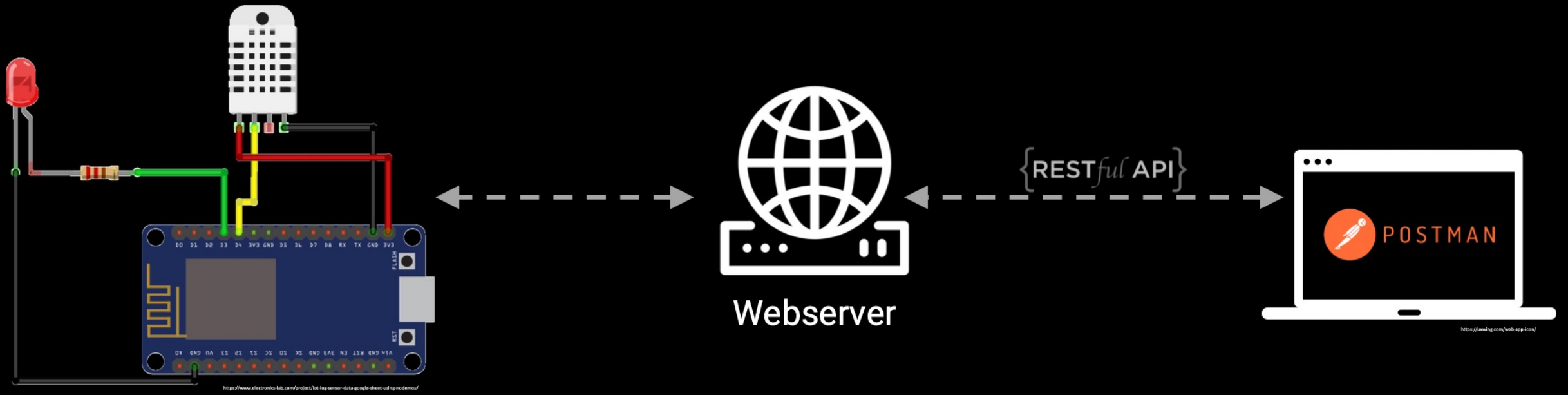
Im query string wird jeder Parameter direkt nacheinander aufgeführt, wobei sie durch ein „&“ voneinander getrennt werden

Die Reihenfolge der Parameter der Abfragezeichenfolge spielt keine Rolle

Programmierung REST Server

Auf dem ESP8266 & ESP32

ESP REST Server (1. Projekt ohne Sensor/Aktor)



Übung

JSON Sensorwerte über REST API abrufen

1. Projektabgabe

04.05 Präsentation der Ergebnisse