URL-Aufbau

`https://api.example.com:443/users/123?sort=name&limit=10`

Bestandteil	Beispiel	Beschreibung
Schema	`https://`	Protokoll
Host	`api.example.com`	Domain oder IP
Port	`:443`	Optional (http=80, https=443)
Pfad	`/users/123`	Ressource
Query	`?sort=name&limit=10`	Parameter

HTTP-Methoden

Methode	Aktion	Idempotent
GET	Abrufen	
POST	Erstellen	
PUT	Ersetzen	
PATCH	Ändern	/
DELETE	Löschen	

HTTP-Statuscodes

2xx — Erfolg; 3xx — Umleitung; 4xx — Client-Fehler; 5xx — Server-Fehler

Code	Name	Bedeutung
200	OK	Anfrage erfolgreich, Inhalt in Response
201	Created	Ressource wurde erstellt (oft mit `Location`-Header)
204	No Content	Erfolgreich, kein Inhalt zurückgegeben (z.B. DELETE)
301	Moved Permanently	URL dauerhaft geändert, neue URL im `Location`-Header
302	Found	URL temporär geändert
304	Not Modified	Ressource unverändert, Cache ist aktuell
400	Bad Request	Ungültige Anfrage (z.B. fehlerhafter JSON)
401	Unauthorized	Nicht authentifiziert (kein oder ungültiges Token)
403	Forbidden	Authentifiziert, aber keine Berechtigung
404	Not Found	Ressource existiert nicht
405	Method Not Allowed	HTTP-Methode für diesen Endpunkt nicht erlaubt
422	Unprocessable Entity	Valides Format, aber Inhalt ungültig (Validierungsfehler)
429	Too Many Requests	Rate Limit überschritten
500	Internal Server Error	Unbekannter Fehler im Server
502	Bad Gateway	Upstream-Server hat ungültige Antwort geliefert
503	Service Unavailable	Server nicht verfügbar (Überlastung, Wartung)

Header

Header	Richtung	Funktion	Beispiel
`Content-Type`	Req / Res	Format der Daten	`application/json`
`Accept`	Request	Gewünschtes Format	`application/json`
`Authorization`	Request	Authentifizierung	`Bearer eyJhbG...`
`Host`	Request	Ziel-Server	`api.example.com`
`Cache-Control`	Response	Caching-Regeln	`max-age=3600`
`ETag`	Response	Versions-Fingerabdruck	`"33a64df5"`
`Location`	Response	URI bei 201 / 3xx	`/messages/42`

Weiterführende Links

Hoppscotch: https://hoppscotch.io

JWT Debugger: https://jwt.io

HTTP-Statuscodes: https://developer.mozilla.org/de/docs/Web/HTTP/Status

OpenAPI-Spec: https://spec.openapis.org/oas/latest.html

cURL

Flag	Wirkung
`-i`	Response-Header anzeigen
`-v`	Verbose (kompletter Request/Response)
`-X POST`	HTTP-Methode setzen
`-H "Key: Value"`	Header setzen (mehrfach möglich)
`-d '...'`	Request-Body setzen
`--json '...'`	Kurzform: setzt `Content-Type` + Body

Beispiel: POST mit Authentifizierung

```
curl https://rest.fschmalzel.de/api/v1/messages \
  -X POST \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1..." \
  --json '{"title": "Hallo", "content": "Welt"}'
```

JWT — Registered Claims

Claim	Bedeutung	Beispiel
`iss`	Aussteller (Issuer)	`example.com`
`sub`	Nutzer-ID (Subject)	`user_42`
`aud`	Zielgruppe (Audience)	`api.example.com`
`exp`	Ablaufzeit (Expiration)	`1516239022`
`nbf`	Gültig ab (Not Before)	`1516236022`
`iat`	Ausstelldatum (Issued At)	`1516236022`
`jti`	Token-ID (JWT ID)	`abc123`

`exp`, `nbf`, `iat` = Unix-Timestamps (Sekunden seit 01.01.1970)

Beispiel Request & Response

Request

```
POST /api/v1/messages HTTP/1.1
Host: rest.fschmalzel.de
Content-Type: application/json
Authorization: Bearer eyJhbGci...

{
  "title": "Hallo",
  "content": "Welt"
}
```

Response

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /api/v1/messages/42

{
  "id": 42,
  "title": "Hallo",
  "content": "Welt",
  "author": "max",
  "created_at": "2025-06-01T10:00:00Z"
}
```