2.3 API-Grundlagen – Client, Server, Endpoint, Request und Response

Eine API ermöglicht es verschiedenen Programmen, Daten auszutauschen oder Funktionen eines anderen Systems aufzurufen.

API bedeutet:

> Application Programming Interface

Auf Deutsch:

> Programmierschnittstelle

APIs werden beispielsweise verwendet, um:

- Benutzerkonten anzulegen
- Kundendaten abzurufen
- Aufgaben zu erstellen
- Nachrichten zu versenden
- Dateien hochzuladen
- Statusinformationen auszulesen
- automatisierte Workflows auszuführen

Lernziele

Nach dieser Seite solltest du erklären können:

- was eine API ist
- was Client und Server bedeuten
- was ein Endpoint ist
- wie Request und Response zusammenhängen
- wozu Header und Body verwendet werden
- was Path- und Query-Parameter sind
- wie eine typische API-Kommunikation abläuft
- wie API-Fehler systematisch eingegrenzt werden

Benutzeroberfläche und API

Ein Mensch verwendet meistens eine grafische Benutzeroberfläche.

Beispiel:

```
Benutzer
  |
  v
Webbrowser
  |
  v
Schaltfläche „Benutzer anlegen“
```

Ein Programm verwendet dagegen häufig eine API.

Beispiel:

```
n8n
  |
  | API-Anfrage
  v
Google Workspace
  |
  | API-Antwort
  v
n8n
```

Die API ermöglicht den Zugriff auf bestimmte Funktionen, ohne dass ein Mensch jede Aktion manuell über die Benutzeroberfläche ausführen muss.

> Eine Benutzeroberfläche ist für Menschen gedacht. Eine API ist für die Kommunikation zwischen Programmen gedacht.

Client und Server

Bei einer API-Kommunikation gibt es normalerweise einen Client und einen Server.

Client

Der Client sendet eine Anfrage.

Beispiele:

- n8n
- Webbrowser
- Smartphone-App
- PowerShell-Skript
- Verwaltungssoftware
- anderes SaaS-System

Server

Der Server verarbeitet die Anfrage und sendet eine Antwort zurück.

Beispiele:

- CRM-System
- Google Workspace
- Slack
- Projektmanagement-System
- Webserver
- Datenbankdienst

Grundablauf:

```
Client
  |
  | Request
  v
Server
  |
  | Response
  v
Client
```

> Der Client stellt eine Anfrage. Der Server verarbeitet sie und antwortet.

Request

Ein Request ist eine Anfrage an einen Server.

Ein Request kann beispielsweise bedeuten:

- Daten abrufen
- neue Daten anlegen
- vorhandene Daten ändern
- Daten löschen
- eine bestimmte Aktion ausführen

Beispiel:

> Rufe den Benutzer mit der ID 15 ab.

Ein Request enthält häufig:

- HTTP-Methode
- URL beziehungsweise Endpoint
- Header
- Parameter
- Body
- Authentifizierungsinformationen

Response

Eine Response ist die Antwort des Servers.

Sie enthält häufig:

- HTTP-Statuscode
- Header
- Daten
- Erfolgsbestätigung
- Fehlermeldung

Beispiel:

> Benutzer wurde gefunden.

Oder:

> Benutzer wurde nicht gefunden.

Grundprinzip:

Bestandteil	Bedeutung
Request	Anfrage des Clients
Response	Antwort des Servers
Client	sendet den Request
Server	verarbeitet den Request

Endpoint

Ein Endpoint ist eine konkrete Adresse innerhalb einer API.

Beispiel:

```
https://api.example.com/users
```

Dieser Endpoint könnte für Benutzer zuständig sein.

Weitere Beispiele:

```
https://api.example.com/projects

https://api.example.com/customers

https://api.example.com/messages
```

Ein einzelner Benutzer könnte über seine ID angesprochen werden:

```
https://api.example.com/users/15
```

> Ein Endpoint ist eine konkrete API-Adresse für eine Ressource oder Funktion.

Ressource

Eine Ressource ist ein Objekt oder Datentyp, der über eine API verwaltet wird.

Beispiele:

- Benutzer

- Kunde
- Projekt
- Aufgabe
- Nachricht
- Datei
- Rechnung

Typische Zuordnung:

Ressource	Endpoint
Benutzer	`/users`
Projekte	`/projects`
Kunden	`/customers`
Aufgaben	`/tasks`
Dateien	`/files`

Aufbau einer API-Adresse

Beispiel:

```
https://api.example.com/v1/users/15
```

Die Adresse besteht aus mehreren Teilen:

Teil	Bedeutung
`https://`	verwendetes Protokoll
`api.example.com`	Server beziehungsweise Host
`/v1`	API-Version
`/users`	Ressource
`/15`	ID eines bestimmten Benutzers

Die API-Version ist wichtig, weil Anbieter ihre Schnittstellen weiterentwickeln können.

Beispiel:

```
/v1/users

/v2/users
```

Eine neue Version kann andere Funktionen oder Datenstrukturen besitzen.

HTTP-Methode

Die HTTP-Methode beschreibt, welche Aktion ausgeführt werden soll.

Typische Methoden:

Methode	Aufgabe
GET	Daten abrufen
POST	neue Daten anlegen
PUT	Daten vollständig ersetzen
PATCH	einzelne Daten ändern
DELETE	Daten löschen

Beispiel:

```
GET /users/15
```

Bedeutung:

> Rufe den Benutzer mit der ID 15 ab.

Beispiel:

```
DELETE /users/15
```

Bedeutung:

> Lösche den Benutzer mit der ID 15.

Die Methoden werden auf der nächsten Seite genauer behandelt.

Header

Header enthalten zusätzliche Informationen über den Request oder die Response.

Typische Header:

Header	Bedeutung
`Authorization`	enthält Zugangstoken
`Content-Type`	beschreibt das Datenformat
`Accept`	beschreibt das gewünschte Antwortformat
`User-Agent`	beschreibt den Client

Beispiel:

```
Authorization: Bearer abc123

Content-Type: application/json
```

Der erste Header übermittelt ein Zugangstoken.

Der zweite Header teilt dem Server mit, dass die gesendeten Daten im JSON-Format vorliegen.

> Header enthalten Steuerungs- und Zusatzinformationen.

Body

Der Body enthält die eigentlichen Nutzdaten eines Requests oder einer Response.

Beispiel für einen Request-Body:

```
{
  "name": "Max Mustermann",
  "department": "IT",
  "active": true
}
```

Diese Daten könnten verwendet werden, um einen Benutzer anzulegen.

Nicht jeder Request benötigt einen Body.

Beispiele:

- Ein GET-Request besitzt häufig keinen Body.

- Ein POST-Request enthält häufig neue Daten.
- Ein PATCH-Request enthält häufig die zu ändernden Werte.

> Der Header beschreibt die Anfrage. Der Body enthält häufig die eigentlichen Daten.

Path-Parameter

Ein Path-Parameter ist direkt Bestandteil des Endpoints.

Beispiel:

```
/users/15
```

Die Zahl `15` ist die ID eines bestimmten Benutzers.

Allgemeine Schreibweise:

```
/users/{id}
```

Weitere Beispiele:

```
/projects/25

/customers/107

/tasks/42
```

Path-Parameter werden häufig verwendet, um eine bestimmte Ressource eindeutig anzusprechen.

Query-Parameter

Query-Parameter stehen hinter einem Fragezeichen in der URL.

Beispiel:

```
/users?department=IT
```

Bedeutung:

> Rufe alle Benutzer aus der Abteilung IT ab.

Mehrere Query-Parameter werden häufig mit `&` getrennt.

Beispiel:

```
/users?department=IT&active=true
```

Bedeutung:

> Rufe alle aktiven Benutzer aus der Abteilung IT ab.

Typische Anwendungen:

- Filtern
- Sortieren
- Suchen
- Seitennavigation
- Begrenzen der Ergebnisse

Path- und Query-Parameter im Vergleich

Parameterart	Beispiel	Aufgabe
Path-Parameter	`/users/15`	bestimmte Ressource auswählen
Query-Parameter	`/users?active=true`	Ergebnisse filtern oder sortieren

Merksatz:

> Path-Parameter bestimmen häufig, welche Ressource gemeint ist. Query-Parameter verändern oder begrenzen die Abfrage.

Beispiel eines vollständigen Requests

```
GET https://api.example.com/v1/users/15

Authorization: Bearer abc123
Accept: application/json
```

Bedeutung:

- `GET` ruft Daten ab.
- Der Endpoint verweist auf Benutzer 15.
- Das Token authentifiziert den Client.
- Die Antwort soll im JSON-Format zurückgegeben werden.

Beispiel einer erfolgreichen Response

```
Status: 200 OK

{
  "id": 15,
  "name": "Max Mustermann",
  "department": "IT",
  "active": true
}
```

Die Response enthält:

- Statuscode `200`
- Erfolgsstatus `OK`
- Benutzerdaten im JSON-Format

Beispiel einer fehlerhaften Response

```
Status: 404 Not Found

{
  "error": "User not found"
}
```

Bedeutung:

- Der Server wurde erreicht.

- Die Anfrage wurde verarbeitet.
- Der Benutzer wurde jedoch nicht gefunden.

Typischer API-Ablauf

1. Der Client erstellt einen Request.
2. Der Request wird an einen Endpoint gesendet.
3. Der Server prüft die Anfrage.
4. Die Authentifizierung wird kontrolliert.
5. Die Berechtigungen werden geprüft.
6. Der Server führt die gewünschte Aktion aus.
7. Der Server erstellt eine Response.
8. Der Client wertet Statuscode und Daten aus.
9. Das Ergebnis wird weiterverarbeitet oder protokolliert.

Darstellung:

```
Client
  |
  | Request
  v
API-Endpoint
  |
  v
Authentifizierung prüfen
  |
  v
Berechtigung prüfen
  |
  v
Aktion ausführen
  |
  v
Response erstellen
  |
  v
Client
```

API-Dokumentation

Eine API-Dokumentation beschreibt, wie eine Schnittstelle verwendet wird.

Sie enthält normalerweise:

- verfügbare Endpoints
- erlaubte HTTP-Methoden
- benötigte Parameter
- Datenformate
- Authentifizierungsverfahren
- Beispiel-Requests
- Beispiel-Responses
- Statuscodes
- Fehlermeldungen
- Rate Limits
- API-Versionen

Vor der Nutzung einer API sollte zuerst die offizielle Dokumentation geprüft werden.

> Eine API sollte nicht durch Vermutungen verwendet werden, sondern anhand ihrer Dokumentation.

Authentifizierung und Autorisierung

Authentifizierung beantwortet die Frage:

> Wer greift auf das System zu?

Beispiele:

- API-Key
- Bearer-Token
- OAuth 2.0
- Benutzername und Passwort

Autorisierung beantwortet die Frage:

> Was darf dieser Client tun?

Beispiel:

Ein API-Token darf Benutzer lesen, aber keine Benutzer löschen.

Begriff	Frage
Authentifizierung	Wer bist du?
Autorisierung	Was darfst du?

API-Schlüssel und Tokens

API-Zugangsdaten müssen sicher behandelt werden.

Sie dürfen nicht:

- öffentlich veröffentlicht werden
- direkt in Dokumentationen stehen
- unverschlüsselt weitergegeben werden
- in öffentlich zugänglichen Git-Repositories gespeichert werden
- in Screenshots sichtbar sein

Sicherer sind:

- geschützte Credential-Speicher
- Umgebungsvariablen
- Secret-Management-Systeme
- eingeschränkte Berechtigungen
- regelmäßiger Austausch der Tokens

Praxisbeispiel mit n8n

Ein n8n-Workflow soll einen Benutzer aus einem CRM-System abrufen.

Ablauf:

```
n8n
  |
  | GET /users/15
  v
CRM-API
  |
  | Response mit Benutzerdaten
  v
n8n
  |
  v
```

Daten prüfen und weiterverarbeiten

n8n benötigt dafür:

- Endpoint
- HTTP-Methode
- Authentifizierung
- mögliche Parameter
- erwartetes Antwortformat
- Fehlerbehandlung

Mögliche Fehlerquellen

Eine API-Anfrage kann aus verschiedenen Gründen fehlschlagen.

Typische Ursachen:

- falscher Endpoint
- falsche HTTP-Methode
- ungültiges Token
- fehlende Berechtigung
- falscher Header
- fehlerhafter JSON-Body
- fehlender Pflichtparameter
- falsche Benutzer-ID
- nicht erreichbarer Server
- veraltete API-Version
- Rate Limit erreicht

Systematische Fehlersuche

Wenn ein API-Aufruf nicht funktioniert:

1. Ist die URL korrekt?
2. Ist der Endpoint noch gültig?
3. Wird die richtige HTTP-Methode verwendet?
4. Ist das Token gültig?
5. Sind die benötigten Berechtigungen vorhanden?
6. Sind alle Pflichtparameter enthalten?
7. Ist der `Content-Type` korrekt?
8. Ist der JSON-Body gültig?
9. Welcher Statuscode wird zurückgegeben?
10. Welche Fehlermeldung enthält die Response?
11. Funktioniert die Anfrage mit Testdaten?
12. Wurde die API-Dokumentation geprüft?

Wichtige Begriffe

Begriff	Bedeutung
API	Schnittstelle zwischen Programmen
Client	sendet eine Anfrage
Server	verarbeitet die Anfrage
Request	Anfrage
Response	Antwort
Endpoint	konkrete API-Adresse
Ressource	verwaltetes Objekt, zum Beispiel Benutzer
HTTP-Methode	beschreibt die gewünschte Aktion
Header	zusätzliche Steuerungsinformationen
Body	eigentliche Nutzdaten
Path-Parameter	Teil des Endpoints
Query-Parameter	Filter oder Zusatzangabe in der URL
Token	Zugangsnachweis für eine API
API-Version	Entwicklungsstand einer Schnittstelle
API-Dokumentation	technische Beschreibung der Schnittstelle

Gesamtmerksatz

> Der Client sendet einen Request an einen API-Endpoint. Der Server prüft Authentifizierung, Berechtigungen und Daten, führt die gewünschte Aktion aus und sendet anschließend eine Response zurück.

Kontrollfragen

Was ist eine API?

Eine Programmierschnittstelle, über die Anwendungen Daten austauschen oder Funktionen aufrufen können.

Was ist ein Client?

Ein Programm oder System, das eine Anfrage sendet.

Was ist ein Server?

Ein System, das eine Anfrage verarbeitet und eine Antwort zurücksendet.

Was ist ein Endpoint?

Eine konkrete Adresse innerhalb einer API.

Was ist eine Ressource?

Ein Objekt, das über eine API verwaltet wird, beispielsweise ein Benutzer oder Projekt.

Was enthält ein Header?

Zusätzliche Informationen wie Datenformat, Authentifizierung oder gewünschtes Antwortformat.

Was enthält ein Body?

Die eigentlichen Nutzdaten eines Requests oder einer Response.

Was ist der Unterschied zwischen Path- und Query-Parameter?

Ein Path-Parameter identifiziert häufig eine bestimmte Ressource. Ein Query-Parameter filtert oder verändert die Abfrage.

Was ist der Unterschied zwischen Authentifizierung und Autorisierung?

Authentifizierung prüft die Identität. Autorisierung prüft die erlaubten Aktionen.

Quellen

- IETF – RFC 9110: HTTP Semantics
- MDN Web Docs – HTTP
- MDN Web Docs – HTTP Headers
- MDN Web Docs – URL Search Parameters
- n8n-Dokumentation – HTTP Request Node