2.15 Prüfungsvorbereitung

Diese Seite fasst die wichtigsten Inhalte zu SaaS-Systemen, APIs und Prozessautomatisierung prüfungsorientiert zusammen.

Im Mittelpunkt stehen:

- zentrale Fachbegriffe
- typische Prüfungsfragen
- praktische Fehlersuche
- sichere Automatisierung
- kurze Fallbeispiele
- wichtige Merksätze

Lernziele

Nach dieser Seite solltest du:

- SaaS-, CRM- und Projektmanagement-Systeme unterscheiden
- eine API verständlich erklären
- Request und Response beschreiben
- REST-Methoden und CRUD zuordnen
- wichtige HTTP-Statuscodes erkennen
- JSON lesen und Fehler finden
- Authentifizierung und Autorisierung unterscheiden
- Webhooks und Polling vergleichen
- einen n8n-Workflow erklären
- Sicherheits- und Datenschutzregeln anwenden
- Schnittstellenfehler systematisch eingrenzen

Wichtige Grundbegriffe

Begriff	Bedeutung
SaaS	Software wird als Onlinedienst bereitgestellt
CRM	Verwaltung von Kunden und Vertriebsprozessen
Projektmanagement-System	Verwaltung von Aufgaben, Terminen und Projekten
Kollaborationssystem	digitale Kommunikation und Zusammenarbeit
API	Schnittstelle für die Kommunikation zwischen Programmen
Endpoint	konkrete Adresse einer API-Funktion
Request	Anfrage des Clients

Begriff	Bedeutung
Response	Antwort des Servers
JSON	textbasiertes Format für strukturierte Daten
Webhook	aktive Meldung eines Ereignisses
Polling	regelmäßige Abfrage nach Änderungen
Workflow	festgelegte Folge automatisierter Schritte
Trigger	startet einen Workflow
Credential	gespeicherte Zugangsdaten
Logging	Protokollierung von Ereignissen und Fehlern

SaaS, CRM und Projektmanagement

Was beschreibt SaaS?

SaaS beschreibt die Bereitstellungsform einer Software.

Die Anwendung läuft normalerweise beim Anbieter und wird über das Internet verwendet.

Beispiele:

- Google Workspace
- Slack
- Asana
- Zoho CRM
- Microsoft 365

Was beschreibt CRM?

CRM beschreibt den Einsatzzweck einer Software.

Ein CRM-System verwaltet:

- Kunden
- Interessenten
- Ansprechpartner
- Gespräche
- Angebote
- Verträge
- Verkaufschancen

Was beschreibt ein Projektmanagement-System?

Es verwaltet:

- Projekte
- Aufgaben
- Zuständigkeiten
- Termine
- Prioritäten
- Bearbeitungsstände

Merksatz:

> SaaS beschreibt die Bereitstellung. CRM und Projektmanagement beschreiben den Einsatzzweck.

API-Grundlagen

API bedeutet:

> Application Programming Interface

Auf Deutsch:

> Programmierschnittstelle

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

Grundablauf:

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

Beispiel:

```
n8n
  |
  | Benutzer anlegen
  v
SaaS-System
  |
  | Benutzer wurde erstellt
  v
n8n
```

Bestandteile eines Requests

Ein Request kann enthalten:

- HTTP-Methode
- Endpoint
- Header
- Query-Parameter
- Path-Parameter
- Request-Body
- Authentifizierungsdaten

Beispiel:

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

Authorization: Bearer ***
Content-Type: application/json

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

HTTP-Methoden und CRUD

HTTP-Methode	Aufgabe	CRUD
GET	Daten lesen	Read

HTTP-Methode	Aufgabe	CRUD
POST	Daten erstellen	Create
PUT	Ressource vollständig ersetzen	Update
PATCH	einzelne Werte ändern	Update
DELETE	Ressource löschen	Delete

Merksatz:

> GET liest, POST erstellt, PUT ersetzt, PATCH ändert teilweise und DELETE löscht.

PUT und PATCH

PUT

Ersetzt normalerweise den vollständigen Datensatz.

```
PUT /users/15

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

PATCH

Verändert nur einzelne Felder.

```
PATCH /users/15

{
  "department": "Support"
}
```

Merksatz:

> PUT ersetzt vollständig. PATCH ändert teilweise.

Wichtige HTTP-Statuscodes

Code	Bedeutung
200	Anfrage erfolgreich
201	Ressource erstellt
204	erfolgreich ohne Response-Body
400	fehlerhafte Anfrage
401	Authentifizierung fehlt oder ist ungültig
403	Berechtigung fehlt
404	Ressource nicht gefunden
409	Konflikt mit vorhandenem Zustand
415	falsches Datenformat
422	fachlich ungültige Daten
429	zu viele Anfragen
500	interner Serverfehler
502	fehlerhafte Antwort eines nachgelagerten Dienstes
503	Dienst vorübergehend nicht verfügbar
504	Zeitüberschreitung am Gateway

---

401 und 403

Statuscode	Bedeutung
401	Identität konnte nicht bestätigt werden
403	Identität ist bekannt, aber die Aktion ist nicht erlaubt

Merksatz:

> 401 fragt: Wer bist du?
> 403 sagt: Du bist bekannt, darfst das aber nicht.

JSON-Grundlagen

JSON speichert Daten als Schlüssel-Wert-Paare.

Beispiel:

```
{
  "name": "Max Mustermann",
  "department": "IT",
  "active": true,
  "roles": [
    "support",
    "user"
  ]
}
```

Wichtige Regeln:

- Objekte verwenden `{ }`
- Arrays verwenden `[ ]`
- Schlüssel stehen in doppelten Anführungszeichen
- Textwerte stehen in doppelten Anführungszeichen
- Boolean-Werte sind `true` oder `false`
- mehrere Einträge werden durch Kommas getrennt
- nach dem letzten Eintrag steht kein Komma

JSON-Datentypen

Datentyp	Beispiel
String	`"Max Mustermann"`
Zahl	`15`
Boolean	`true`
Objekt	`{ "name": "Max" }`
Array	`["IT", "Support"]`
Null	`null`

Unterschied:

```
true
```

ist ein Boolean.

```
"true"
```

ist ein String.

JSON-Fehler erkennen

Fehlerhaft:

```
{
  name: 'Max Mustermann',
  "active": "true",
}
```

Fehler:

- Schlüssel ohne doppelte Anführungszeichen
- einfache Anführungszeichen
- Boolean als String
- zusätzliches Komma

Korrekt:

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

Authentifizierung und Autorisierung

Begriff	Frage
Authentifizierung	Wer greift zu?
Autorisierung	Was darf der Zugriff ausführen?

Mögliche Verfahren:

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

Beispiel:

```
Authorization: Bearer ***
```

OAuth 2.0

OAuth 2.0 ermöglicht einer Anwendung einen begrenzten Zugriff auf ein anderes System.

Wichtige Begriffe:

Begriff	Bedeutung
Access Token	wird für API-Anfragen verwendet
Refresh Token	fordert ein neues Access Token an
Scope	begrenzt die erlaubten Aktionen
Client ID	identifiziert die Anwendung
Client Secret	geheimer Nachweis der Anwendung

Merksatz:

> OAuth 2.0 ermöglicht kontrollierten Zugriff, ohne dauerhaft das Benutzerpasswort zu speichern.

Webhook und Polling

Verfahren	Funktionsweise
Webhook	Quellsystem meldet ein Ereignis aktiv
Polling	Zielsystem fragt regelmäßig nach Änderungen

Webhook:

```
CRM
 |
 | neuer Kunde
 v
n8n
```

Polling:

```
n8n
 |
 | Gibt es neue Kunden?
 v
CRM
```

Merksatz:

> Beim Webhook wird gesendet. Beim Polling wird nachgefragt.

---

n8n-Grundaufbau

Ein n8n-Workflow besteht aus Nodes.

Beispiel:

```
Webhook Trigger
      |
      v
Daten prüfen
      |
      v
Benutzer suchen
      |
      v
Benutzer vorhanden?
   /        \
 ja          nein
 |            |
 v            v
stoppen     Benutzer anlegen
                |
                v
          Nachricht senden
                |
                v
          Ergebnis protokollieren
```

Typische n8n-Nodes

Node	Aufgabe
Webhook	empfängt HTTP-Anfragen
Schedule Trigger	startet nach Zeitplan
HTTP Request	ruft eine API auf
Edit Fields	verändert Datenfelder
If	prüft eine Ja-Nein-Bedingung
Switch	unterscheidet mehrere Fälle
Code	verarbeitet Daten mit JavaScript
Error Trigger	startet einen Fehlerworkflow

Expressions in n8n

Wert auslesen:

```
{{$json.name}}
```

Text zusammensetzen:

```
{{$json.firstName + " " + $json.lastName}}
```

Standardwert verwenden:

```
{{$json.department || "Allgemein"}}
```

Zahl umwandeln:

```
{{Number($json.licenseCount)}}
```

Datenmapping

Quellsystem:

```
{
  "firstName": "Max",
```

```
  "lastName": "Mustermann",
  "mail": "max.mustermann@example.com"
}
```

Zielsystem:

```
{
  "givenName": "Max",
  "familyName": "Mustermann",
  "primaryEmail": "max.mustermann@example.com"
}
```

Quellfeld	Zielfeld
`firstName`	`givenName`
`lastName`	`familyName`
`mail`	`primaryEmail`

> Beim Datenmapping werden Felder unterschiedlicher Systeme einander zugeordnet.

Bedingungen

If

Geeignet für zwei Wege:

```
Benutzer aktiv?
  /      \
 ja        nein
 |          |
 v          v
weiter    stoppen
```

Switch

Geeignet für mehrere Fälle:

```
Abteilung
 /  |  \
IT  CRM  Finance
```

Fehlerbehandlung

Ein Workflow sollte Fehler erkennen und kontrolliert behandeln.

Beispiel:

```
API-Anfrage
    |
    v
erfolgreich?
  /     \
 ja       nein
 |         |
 v         v
weiter   Fehler prüfen
           |
           v
       Retry möglich?
         /     \
       ja       nein
       |         |
       v         v
     warten    IT informieren
```

Retry

Retry bedeutet:

> Eine fehlgeschlagene Aktion wird erneut versucht.

Sinnvoll bei:

- 429
- 502

- 503
- 504
- kurzfristigem Netzwerkfehler

Nicht unverändert sinnvoll bei:

- 400
- 401
- 403
- ungültigem JSON
- fehlenden Pflichtfeldern

Backoff

Backoff bedeutet:

> Die Wartezeit zwischen Wiederholungen wird erhöht.

Beispiel:

Versuch	Wartezeit
1	sofort
2	5 Sekunden
3	15 Sekunden
4	30 Sekunden

Rate Limit

Ein Rate Limit begrenzt die Anzahl erlaubter API-Anfragen.

Beispiel:

```
Status: 429 Too Many Requests

Retry-After: 60
```

Bedeutung:

> Vor dem nächsten Versuch 60 Sekunden warten.

Least Privilege

Least Privilege bedeutet:

> Benutzer, Anwendungen und Workflows erhalten nur die Rechte, die sie tatsächlich benötigen.

Beispiel:

Ein Workflow soll Nachrichten senden.

Benötigt:

```
messages.send
```

Nicht benötigt:

```
users.delete
administrators.write
```

Sicherheitsregeln

- HTTPS verwenden
- Secrets geschützt speichern
- keine Tokens in Logs schreiben
- nur notwendige Scopes vergeben
- Webhooks authentifizieren
- Eingabedaten validieren
- Test- und Produktivsysteme trennen
- kritische Aktionen freigeben lassen
- Service Accounts verwenden
- Audit-Logs kontrollieren

Onboarding und Offboarding

Onboarding

- Benutzerkonto anlegen
- Gruppen zuweisen
- Rollen vergeben
- Lizenzen aktivieren
- MFA vorbereiten
- Hardware-Aufgabe erstellen
- Ergebnis dokumentieren

Offboarding

- Sitzungen beenden
- Konto deaktivieren
- Tokens widerrufen
- Gruppen entfernen
- Daten übertragen
- Lizenzen freigeben
- Geräte zurückfordern
- Vorgang dokumentieren

Typische Prüfungsaufgabe 1

Ein n8n-Workflow sendet folgende Anfrage:

```
POST /users

{
 "name": "Max Mustermann"
}
```

Die API antwortet:

```
Status: 400 Bad Request

{
 "error": "Missing required field",
 "field": "email"
}
```

Lösung

Das Pflichtfeld `email` fehlt.

Der Request muss ergänzt werden:

```
{
  "name": "Max Mustermann",
  "email": "max.mustermann@example.com"
}
```

Typische Prüfungsaufgabe 2

Eine API antwortet mit:

```
Status: 403 Forbidden
```

Lösung

Die Authentifizierung war wahrscheinlich erfolgreich, aber die benötigte Berechtigung oder der Scope fehlt.

Zu prüfen sind:

- Rolle
- Gruppenmitgliedschaft
- API-Scope
- Administratorfreigabe
- Zugriff auf die Ressource

Typische Prüfungsaufgabe 3

Ein Webhook wird zweimal gesendet und zwei Benutzerkonten werden erstellt.

Ursache

Der Workflow besitzt keinen Schutz vor doppelter Verarbeitung.

Lösung

- Event-ID speichern
- Benutzer vor dem Anlegen suchen
- eindeutige Mitarbeiter-ID verwenden
- Idempotency-Key einsetzen

Typische Prüfungsaufgabe 4

Ein Workflow erhält:

```
{
 "active": "true"
}
```

Die API erwartet:

```
{
 "active": true
}
```

Lösung

Der empfangene Wert ist ein String.

Die API erwartet einen Boolean.

Typische Prüfungsaufgabe 5

Eine API antwortet mit:

```
Status: 429 Too Many Requests
```

Lösung

Das Rate Limit wurde erreicht.

Mögliche Maßnahmen:

- `Retry-After` beachten
- Wartezeit einbauen
- Anfragen begrenzen
- Batch-Verarbeitung verwenden
- Polling-Intervall erhöhen

Typische Prüfungsaufgabe 6

Der Workflow meldet einen Timeout nach einer POST-Anfrage.

Problem

Die Ressource könnte im Zielsystem trotzdem erstellt worden sein.

Lösung

Vor einem erneuten POST-Request prüfen, ob die Ressource bereits existiert.

Dadurch werden doppelte Datensätze vermieden.

Typische Prüfungsaufgabe 7

Ein Benutzer kann sich anmelden, aber keine Projekte verändern.

Lösung

Die Authentifizierung funktioniert.

Wahrscheinlich fehlt die Autorisierung für die gewünschte Aktion.

Zu prüfen sind:

- Rolle
- Projektberechtigung
- Gruppenmitgliedschaft
- API-Scope

Typische Prüfungsaufgabe 8

Ein Workflow startet nicht.

Prüfreihenfolge

1. Workflow aktiv?
2. richtiger Trigger?
3. Webhook-URL korrekt?
4. richtige HTTP-Methode?
5. Quellsystem hat gesendet?
6. Endpoint erreichbar?
7. Authentifizierung gültig?
8. Trigger-Log vorhanden?
9. Test- oder Produktiv-URL verwechselt?

Typische Prüfungsaufgabe 9

Ein API-Token steht direkt in einer Code Node.

Bewertung

Das ist unsicher.

Bessere Lösung

Das Token wird in der Credential-Verwaltung oder einem Secret-Management-System gespeichert.

Typische Prüfungsaufgabe 10

Ein Workflow überträgt neben Name und E-Mail auch Gehalt und private Adresse an Slack.

Bewertung

Die Datenübertragung verstößt gegen das Prinzip der Datenminimierung.

Nur die benötigten Daten sollten übertragen werden.

Systematische Fehlersuche

Empfohlene Reihenfolge:

1. Problem genau beschreiben
2. Trigger prüfen
3. Eingangsdaten prüfen
4. Netzwerk und Erreichbarkeit prüfen
5. Endpoint prüfen
6. HTTP-Methode prüfen
7. Authentifizierung prüfen
8. Autorisierung prüfen
9. Header kontrollieren
10. JSON-Body prüfen
11. Statuscode auswerten
12. Response-Body lesen
13. Logs prüfen
14. letzte Änderungen kontrollieren
15. Lösung mit Testdaten prüfen
16. Ergebnis dokumentieren

Prüfungsmerksätze

> SaaS beschreibt die Bereitstellung, nicht den Einsatzzweck.

> Der Client sendet einen Request, der Server liefert eine Response.

> GET liest, POST erstellt, PUT ersetzt, PATCH ändert und DELETE löscht.

> 401 bedeutet fehlende Authentifizierung, 403 bedeutet fehlende Berechtigung.

> JSON verwendet doppelte Anführungszeichen.

> Beim Webhook wird gesendet, beim Polling wird regelmäßig abgefragt.

> Erst Daten validieren, danach übertragen.

> Secrets gehören nicht in Code, Logs oder Dokumentationen.

> Ein Retry muss begrenzt und vor Doppelverarbeitung geschützt sein.

> Jede Automatisierung benötigt Fehlerbehandlung und Logging.

Selbstkontrolle

Du solltest ohne Hilfsmittel erklären können:

- Unterschied zwischen SaaS und CRM
- Aufbau eines API-Requests
- Bedeutung von Endpoint
- Unterschied zwischen PUT und PATCH
- Unterschied zwischen 401 und 403

- Aufbau eines JSON-Objekts
- Unterschied zwischen Webhook und Polling
- Aufgabe eines Triggers
- Nutzen einer If- und Switch-Node
- Bedeutung von Retry und Backoff
- Zweck von Least Privilege
- Ablauf eines sicheren Onboardings
- systematische Fehlersuche bei API-Fehlern

Gesamtmerksatz

> APIs verbinden Systeme über Requests und Responses. REST verwendet HTTP-Methoden zur Verarbeitung von Ressourcen. JSON überträgt strukturierte Daten. Webhooks und Trigger starten Workflows, während n8n Bedingungen, Datenmapping und Aktionen verbindet. Sichere Automatisierungen benötigen begrenzte Rechte, geschützte Secrets, Validierung, Logging und Fehlerbehandlung.

Quellen

- IETF – HTTP Semantics
- IETF – JSON Data Interchange Format
- IETF – OAuth 2.0
- MDN Web Docs – HTTP
- n8n-Dokumentation
- OWASP – Authorization Cheat Sheet
- OWASP – Secrets Management Cheat Sheet
