2.11 Fehlerbehandlung, Logging, Retry und Rate Limits

Automatisierte Workflows müssen auch dann kontrolliert reagieren, wenn ein Schritt fehlschlägt.

Mögliche Fehlerquellen:

- Zielsystem nicht erreichbar
- ungültige Zugangsdaten
- fehlende Berechtigung
- falsche Eingabedaten
- abgelaufenes Token
- Rate Limit erreicht
- Timeout
- fehlerhafte API-Response
- unbekannter Systemfehler

> Ein professioneller Workflow behandelt nicht nur den Erfolgsfall, sondern auch Fehler, Ausnahmen und Teilerfolge.

Lernziele

Nach dieser Seite solltest du erklären können:

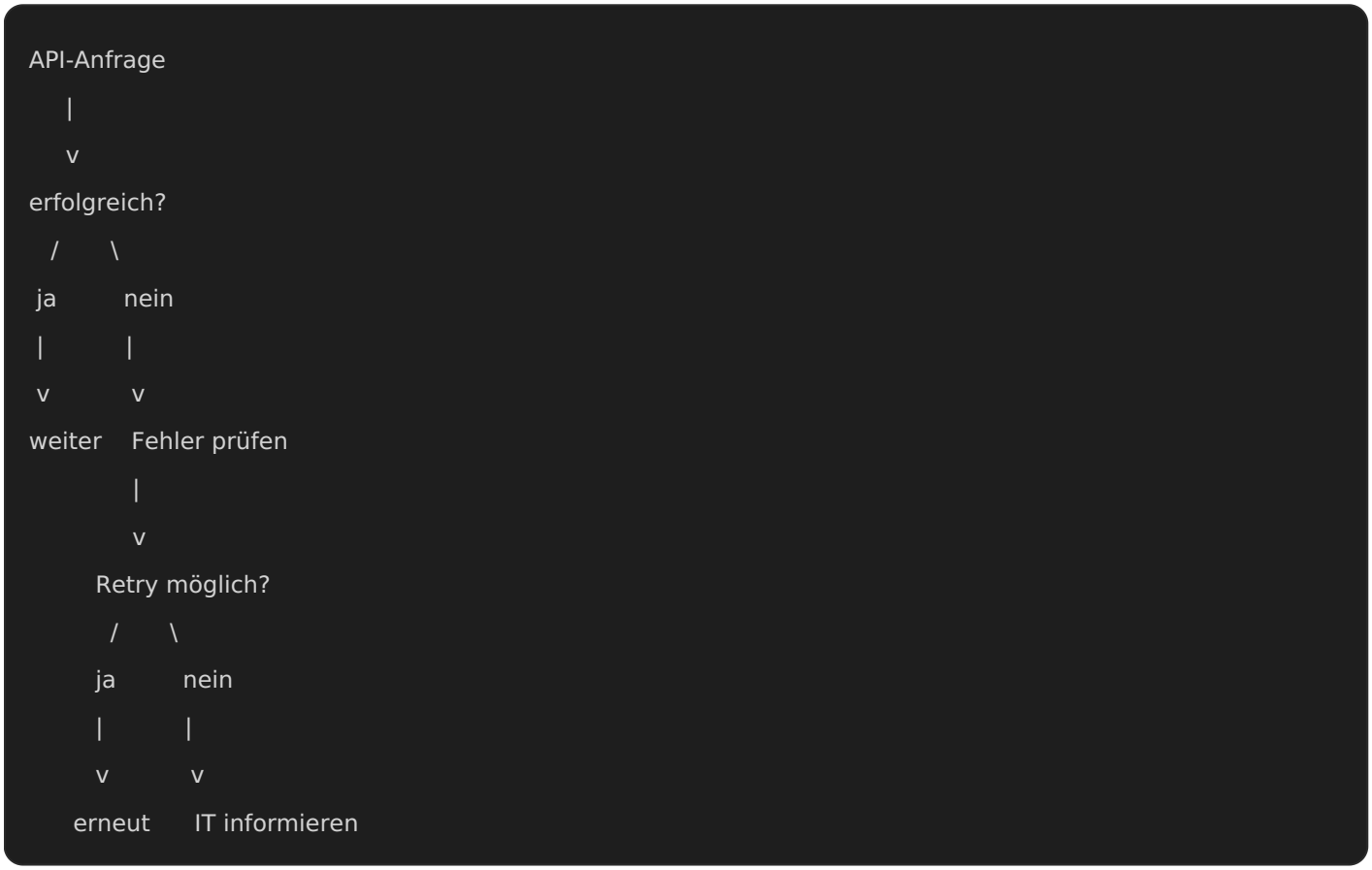
- was Fehlerbehandlung bedeutet
- welche Fehlerarten unterschieden werden
- was Logging ist
- wann ein Retry sinnvoll ist
- was Backoff bedeutet
- wie Rate Limits behandelt werden
- wie teilweise erfolgreiche Abläufe dokumentiert werden
- wann eine manuelle Bearbeitung notwendig ist

Fehlerbehandlung

Fehlerbehandlung bedeutet:

> Ein System erkennt einen Fehler und führt daraufhin einen festgelegten Ablauf aus.

Beispiel:



Ohne Fehlerbehandlung kann ein Workflow unbemerkt abbrechen oder falsche Ergebnisse erzeugen.

Fehlerarten

Fehler können in verschiedene Gruppen eingeteilt werden.

Fehlerart	Beispiel
Eingabefehler	Pflichtfeld fehlt
Authentifizierungsfehler	Token ungültig
Berechtigungsfehler	Aktion nicht erlaubt
Netzwerkfehler	Server nicht erreichbar
Serverfehler	Zielsystem meldet 500
Logikfehler	falsche Bedingung
Datenfehler	falscher Datentyp
Zeitüberschreitung	API antwortet zu langsam
Rate Limit	zu viele Anfragen
Teilfehler	einige Schritte erfolgreich, andere nicht

Eingabefehler

Ein Eingabefehler entsteht, wenn Daten fehlen oder ungültig sind.

Beispiele:

- E-Mail-Adresse fehlt
- Datum besitzt falsches Format
- Zahl wird als Text übertragen
- Abteilung ist unbekannt
- Pflichtfeld ist leer

Beispiel:

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

Mögliche Behandlung:

1. Verarbeitung stoppen
2. fehlerhaftes Feld dokumentieren
3. zuständige Person informieren
4. keine API-Anfrage senden

> Ungültige Eingabedaten sollten möglichst vor dem API-Aufruf erkannt werden.

Authentifizierungsfehler

Typische Statuscodes:

Code	Bedeutung
401	Authentifizierung fehlt oder ist ungültig
403	Authentifizierung gültig, aber Berechtigung fehlt

Mögliche Ursachen:

- Token abgelaufen
- API-Key falsch
- Authorization-Header fehlt

- Scope fehlt
- Rolle besitzt zu wenige Rechte
- Credential wurde widerrufen

Diese Fehler sollten normalerweise nicht unverändert wiederholt werden.

Netzwerk- und Serverfehler

Mögliche Fehler:

- DNS-Auflösung schlägt fehl
- Verbindung wird abgelehnt
- Timeout
- Server antwortet mit 500
- Gateway meldet 502
- Dienst ist mit 503 nicht verfügbar
- Gateway meldet 504

Diese Fehler können vorübergehend sein.

Ein begrenzter Retry kann daher sinnvoll sein.

Dauerhafte und vorübergehende Fehler

Fehler	Typische Einordnung
400 Bad Request	dauerhaft, Request korrigieren
401 Unauthorized	dauerhaft, Zugangsdaten prüfen
403 Forbidden	dauerhaft, Berechtigung prüfen
404 Not Found	meistens dauerhaft, Endpoint oder ID prüfen
409 Conflict	Zustand oder vorhandenen Datensatz prüfen
429 Too Many Requests	vorübergehend
500 Internal Server Error	möglicherweise vorübergehend
502 Bad Gateway	häufig vorübergehend
503 Service Unavailable	häufig vorübergehend
504 Gateway Timeout	häufig vorübergehend

Merksatz:

> Fehlerhafte Daten müssen korrigiert werden. Vorübergehende Systemfehler können erneut versucht werden.

Retry

Retry bedeutet:

> Eine fehlgeschlagene Aktion wird erneut ausgeführt.

Retry ist sinnvoll bei:

- kurzfristigem Netzwerkfehler
- `429 Too Many Requests`
- `502 Bad Gateway`
- `503 Service Unavailable`
- `504 Gateway Timeout`

Retry ist normalerweise nicht sinnvoll bei:

- ungültigem JSON
- fehlendem Pflichtfeld
- falschem Token
- fehlender Berechtigung
- falschem Endpoint

Begrenzte Wiederholungen

Ein Workflow darf eine fehlgeschlagene Aktion nicht unbegrenzt wiederholen.

Beispiel:

1. erster Versuch sofort
2. zweiter Versuch nach 5 Sekunden
3. dritter Versuch nach 15 Sekunden
4. vierter Versuch nach 30 Sekunden
5. anschließend Fehler melden

Vorteile:

- keine Endlosschleife
- Zielsystem wird nicht zusätzlich überlastet
- Fehler bleibt nachvollziehbar

- manuelle Bearbeitung kann beginnen

Backoff

Backoff bedeutet:

> Die Wartezeit zwischen Wiederholungen wird schrittweise erhöht.

Beispiel:

Versuch	Wartezeit
1	sofort
2	5 Sekunden
3	15 Sekunden
4	30 Sekunden
5	60 Sekunden

Ein exponentieller Backoff erhöht die Wartezeit besonders deutlich.

Beispiel:

```
2 Sekunden
4 Sekunden
8 Sekunden
16 Sekunden
32 Sekunden
```

Rate Limit

Ein Rate Limit begrenzt die Anzahl erlaubter API-Anfragen.

Beispiel:

> Maximal 100 Anfragen pro Minute.

Wird das Limit überschritten, antwortet die API häufig mit:

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

Mögliche Response:

```
Retry-After: 60
```

Bedeutung:

> Vor dem nächsten Versuch 60 Sekunden warten.

Rate Limits vermeiden

Mögliche Maßnahmen:

- Anfragen zeitlich verteilen
- mehrere Datensätze gesammelt übertragen
- nur notwendige Daten abrufen
- Ergebnisse zwischenspeichern
- Polling-Intervall vergrößern
- Wiederholungen begrenzen
- `Retry-After` auswerten
- Batch-Verarbeitung verwenden

Beispiel:

Statt 1.000 Einzelanfragen:

```
GET /users/1
GET /users/2
GET /users/3
```

besser, wenn unterstützt:

```
GET /users?limit=100
```

Timeout

Ein Timeout tritt auf, wenn eine Antwort nicht rechtzeitig empfangen wird.

Mögliche Ursachen:

- Zielsystem reagiert langsam
- Netzwerk ist gestört
- Datenbankabfrage dauert zu lange
- Workflow verarbeitet zu viele Daten
- Timeout-Wert ist zu niedrig

Bei einem Timeout ist nicht immer klar, ob die Aktion im Zielsystem bereits ausgeführt wurde.

Beispiel:

1. n8n sendet `POST /users`
2. Zielsystem legt den Benutzer an
3. Response erreicht n8n nicht rechtzeitig
4. n8n meldet Timeout
5. Workflow wiederholt den POST-Request
6. Benutzer könnte doppelt angelegt werden

Vor einem Retry sollte daher geprüft werden, ob die Ressource bereits existiert.

Idempotenz bei Retry

Ein Retry darf keine doppelten Ressourcen erzeugen.

Mögliche Schutzmaßnahmen:

- eindeutige Vorgangs-ID verwenden
- vor POST zunächst mit GET suchen
- Idempotency-Key verwenden
- Event-ID speichern
- bereits verarbeitete Datensätze markieren

Beispiel:

```
Request-ID: EMP-105
```

Vor dem Anlegen:

```
GET /users?employeeId=EMP-105
```

Nur wenn kein Benutzer vorhanden ist:

```
POST /users
```

Logging

Logging bedeutet:

> Ereignisse, Ergebnisse und Fehler werden protokolliert.

Ein Log sollte beantworten:

- Was ist passiert?
- Wann ist es passiert?
- Welcher Workflow war betroffen?
- Welche Node ist fehlgeschlagen?
- Welcher Statuscode wurde empfangen?
- Welche Ressource war betroffen?
- Welche Gegenmaßnahme wurde ausgeführt?

Sinnvolle Log-Daten

Information	Beispiel
Zeitpunkt	18.07.2026 10:15 Uhr
Workflow	Mitarbeiter-Onboarding
Node	Google-Benutzer anlegen
Aktion	`POST /users`
Statuscode	403
Fehlermeldung	fehlende Berechtigung
Vorgangs-ID	EMP-105
Versuch	1 von 3
Ergebnis	Administrator informiert

Sensible Daten im Log

Nicht vollständig protokollieren:

- Passwörter
- API-Keys

- Bearer-Tokens
- Refresh Tokens
- Client Secrets
- vertrauliche Personaldaten
- vollständige Zahlungsdaten

Unsicher:

```
Authorization: Bearer abc123vollstaendig
```

Sicherer:

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

Oder:

```
Token-Ende: ...7F3A
```

> Logs müssen bei der Fehlersuche helfen, ohne neue Sicherheitsrisiken zu erzeugen.

Log-Level

Logs können nach Wichtigkeit eingeteilt werden.

Level	Bedeutung
Debug	detaillierte technische Informationen
Info	normaler Ablauf
Warning	ungewöhnliches, aber noch beherrschbares Ereignis
Error	Aktion ist fehlgeschlagen
Critical	schwerwiegender Ausfall oder Sicherheitsvorfall

Beispiele:

```
INFO: Benutzerprüfung gestartet

WARNING: API-Antwort dauerte länger als erwartet
```

ERROR: Benutzer konnte nicht angelegt werden

CRITICAL: mehrere zentrale Workflows ausgefallen

Error Workflow in n8n

Ein Error Workflow kann bei einer fehlgeschlagenen Ausführung automatisch gestartet werden.

Mögliche Aktionen:

- Slack-Nachricht senden
- E-Mail versenden
- Ticket erstellen
- Fehler in Datenbank speichern
- verantwortliche Person informieren
- Vorgang zur manuellen Bearbeitung markieren

Beispielmeldung:

Workflow: Mitarbeiter-Onboarding

Node: Benutzer anlegen

Status: fehlgeschlagen

Fehler: 403 Forbidden

Vorgangs-ID: EMP-105

Zeitpunkt: 18.07.2026 10:15 Uhr

Teilerfolg

Ein Workflow kann teilweise erfolgreich sein.

Beispiel:

Schritt	Ergebnis
Benutzerkonto anlegen	erfolgreich
Google-Gruppe zuweisen	erfolgreich
Slack-Zugang erstellen	fehlgeschlagen
Asana-Aufgabe erstellen	nicht ausgeführt

Der Workflow darf nicht melden:

> Vorgang vollständig erfolgreich.

Besser:

> Benutzerkonto und Gruppe wurden eingerichtet. Die Slack-Einrichtung ist fehlgeschlagen. Die Asana-Aufgabe wurde nicht erstellt.

Rollback

Rollback bedeutet:

> Bereits ausgeführte Änderungen werden bei einem Fehler zurückgenommen.

Beispiel:

1. Benutzerkonto wird angelegt.
2. wichtige Gruppenzuweisung schlägt fehl.
3. Workflow deaktiviert das neu angelegte Konto wieder.

Ein Rollback ist nicht immer sinnvoll oder technisch möglich.

Alternativen:

- Vorgang pausieren
- Teilerfolg dokumentieren
- manuelle Prüfung anfordern
- fehlgeschlagenen Schritt gezielt erneut ausführen

Manuelle Bearbeitung

Eine manuelle Bearbeitung ist sinnvoll, wenn:

- der Fehler nicht automatisch lösbar ist
- eine Entscheidung erforderlich ist
- Berechtigungen fehlen
- Daten widersprüchlich sind
- ein Sicherheitsvorfall möglich ist
- mehrere Systeme unterschiedliche Zustände besitzen
- ein Rollback riskant wäre

Der Workflow sollte dafür alle notwendigen Informationen bereitstellen.

Alerting

Alerting bedeutet:

> Zuständige Personen werden bei wichtigen Fehlern automatisch benachrichtigt.

Mögliche Kanäle:

- Slack
- E-Mail
- Ticketsystem
- Monitoring-System
- SMS oder Bereitschaftsdienst

Nicht jeder kleine Fehler benötigt sofort einen Alarm.

Sinnvolle Priorisierung:

Priorität	Beispiel
niedrig	einzelner unkritischer Datensatz fehlerhaft
mittel	wiederholter Workflow-Fehler
hoch	kompletter Onboarding-Prozess ausgefallen
kritisch	Sicherheitsvorfall oder zentraler Systemausfall

Fehler nicht verschlucken

Ein Workflow darf einen Fehler nicht einfach ignorieren und trotzdem Erfolg melden.

Schlechtes Verhalten:

```
API-Fehler
  |
  v
Workflow läuft ohne Prüfung weiter
  |
  v
Meldung „erfolgreich“
```

Besser:

```
API-Fehler
  |
  v
Status prüfen
  |
  v
Fehler dokumentieren
  |
  v
abhängige Schritte stoppen
  |
  v
IT informieren
```

Praxisbeispiel: Benutzer-Onboarding

Möglicher Ablauf:

```
Webhook empfangen
     |
     v
Daten validieren
     |
     v
Benutzer suchen
     |
     v
Benutzer anlegen
     |
     v
Statuscode prüfen
  /        \
 201        Fehler
  |          |
  v          v
weiter     Fehlerart prüfen
             |
```

Behandlung nach Statuscode

Status	Reaktion
200	Daten weiterverarbeiten
201	Erstellung bestätigen
400	Eingabedaten korrigieren
401	Credential prüfen
403	Rollen und Scopes prüfen
404	Endpoint oder ID prüfen
409	vorhandenen Datensatz prüfen
429	`Retry-After` beachten
500	Fehler protokollieren und begrenzt wiederholen
503	warten und erneut versuchen
504	Timeout und Zielsystem prüfen

Monitoring wichtiger Workflows

Neben einzelnen Fehlern sollte auch der Gesamtzustand überwacht werden.

Mögliche Prüfungen:

- letzter erfolgreicher Lauf
- Anzahl fehlgeschlagener Ausführungen
- durchschnittliche Laufzeit
- Anzahl der Retries
- ungewöhnlich viele Statuscodes 401 oder 403
- ungewöhnlich viele Rate-Limit-Fehler
- seit langer Zeit kein Trigger empfangen
- Workflow deaktiviert
- Credential läuft bald ab

Gute Fehlerbehandlung

Eine gute Fehlerbehandlung besitzt:

- klare Validierung
- eindeutige Fehlermeldungen
- begrenzte Retries
- passende Wartezeiten
- Schutz vor Doppelverarbeitung
- sichere Logs
- Benachrichtigung nach Priorität
- Behandlung von Teilerfolgen
- dokumentierte manuelle Schritte
- regelmäßiges Monitoring

Systematische Fehlersuche

Wenn ein Workflow fehlschlägt:

1. Wurde der Workflow gestartet?
2. Welche Node ist fehlgeschlagen?
3. Welche Eingangsdaten wurden verwendet?
4. Welcher Statuscode wurde empfangen?
5. Welche Fehlermeldung enthält die Response?
6. Ist der Fehler dauerhaft oder vorübergehend?
7. Ist ein Retry sinnvoll?
8. Wurde ein Rate Limit erreicht?
9. Besteht das Risiko einer doppelten Ausführung?
10. Welche Schritte waren bereits erfolgreich?
11. Muss ein Rollback erfolgen?
12. Wurde der Fehler korrekt protokolliert?
13. Wurde die zuständige Person informiert?
14. Kann der Vorgang manuell fortgesetzt werden?

Wichtige Begriffe

Begriff	Bedeutung
Fehlerbehandlung	festgelegter Umgang mit Fehlern
Retry	erneuter Ausführungsversuch
Backoff	steigende Wartezeit zwischen Versuchen
Rate Limit	Begrenzung der API-Anfragen
Timeout	Antwortzeit wurde überschritten

Begriff	Bedeutung
Logging	Protokollierung von Ereignissen
Log-Level	Einordnung der Wichtigkeit
Alerting	automatische Benachrichtigung
Teilerfolg	nur ein Teil des Ablaufs war erfolgreich
Rollback	Änderungen werden zurückgenommen
Idempotenz	Wiederholung erzeugt keine zusätzlichen Änderungen
Error Workflow	separater Ablauf für Fehler

Gesamtmerksatz

> Fehlerbehandlung erkennt Probleme und führt einen kontrollierten Fehlerpfad aus. Vorübergehende Fehler können mit begrenzten Retries und Backoff behandelt werden. Rate Limits müssen berücksichtigt und Logs sicher gespeichert werden. Teilerfolge, doppelte Ausführungen und manuelle Nacharbeiten müssen eindeutig dokumentiert werden.

Kontrollfragen

Was bedeutet Fehlerbehandlung?

Ein Fehler wird erkannt und nach festgelegten Regeln verarbeitet.

Wann ist ein Retry sinnvoll?

Bei vorübergehenden Netzwerk-, Gateway-, Server- oder Rate-Limit-Fehlern.

Wann ist ein Retry nicht sinnvoll?

Wenn Eingabedaten, Credentials, Berechtigungen oder Endpoints falsch sind.

Was bedeutet Backoff?

Die Wartezeit zwischen Wiederholungen wird schrittweise erhöht.

Was ist ein Rate Limit?

Eine Begrenzung der erlaubten API-Anfragen innerhalb eines Zeitraums.

Was bedeutet Logging?

Ereignisse, Abläufe und Fehler werden protokolliert.

Warum dürfen Tokens nicht vollständig im Log erscheinen?

Weil sie geheime Zugangsdaten darstellen.

Was ist ein Teilerfolg?

Ein Teil des Workflows war erfolgreich, während andere Schritte fehlgeschlagen sind.

Was bedeutet Rollback?

Bereits durchgeführte Änderungen werden zurückgenommen.

Warum muss vor einem Retry geprüft werden, ob eine Ressource bereits erstellt wurde?

Damit durch die Wiederholung keine doppelten Datensätze entstehen.

Quellen

- n8n-Dokumentation – Error Handling
- n8n-Dokumentation – Error Trigger
- n8n-Dokumentation – Executions
- MDN Web Docs – HTTP-Statuscodes
- MDN Web Docs – 429 Too Many Requests
- OWASP – Logging Cheat Sheet