2.8 Webhooks, Polling, Trigger und Zeitpläne

Automatisierte Workflows benötigen einen Auslöser.

Dieser Auslöser wird als Trigger bezeichnet.

Typische Trigger sind:

- eingehender Webhook
- regelmäßige Abfrage
- festgelegter Zeitplan
- Änderung in einem System
- neuer Datensatz
- neue E-Mail
- manuelle Ausführung

> Ein Trigger bestimmt, wann ein Workflow gestartet wird.

Lernziele

Nach dieser Seite solltest du erklären können:

- was ein Trigger ist
- wie ein Webhook funktioniert
- was Polling bedeutet
- wie sich Webhook und Polling unterscheiden
- was ein Zeitplan-Trigger ist
- welche Sicherheitsmaßnahmen bei Webhooks notwendig sind
- wie doppelte oder fehlende Ausführungen verhindert werden

Trigger

Ein Trigger ist ein Ereignis, das einen automatisierten Workflow startet.

Beispiele:

Trigger	Mögliche Aktion
neuer Mitarbeiter	Benutzerkonto anlegen
neuer CRM-Kunde	Projekt erstellen
neue Supportanfrage	IT benachrichtigen

Trigger	Mögliche Aktion
fehlgeschlagener Dienst	Alarm senden
täglicher Zeitpunkt	Bericht erzeugen
manuelle Ausführung	Testworkflow starten

Grundablauf:

```
Trigger
  |
  v
Daten prüfen
  |
  v
Aktionen ausführen
  |
  v
Ergebnis protokollieren
```

Webhook

Ein Webhook ist eine automatische Nachricht, die beim Eintritt eines Ereignisses an ein anderes System gesendet wird.

Beispiel:

> Im CRM wurde ein neuer Kunde angelegt.

Das CRM sendet die Kundendaten direkt an n8n.

```
CRM
 |
 | Webhook
 v
n8n
 |
 v
Workflow startet
```

Der Webhook enthält häufig JSON-Daten.

Beispiel:

```
{
  "event": "customer.created",
  "customerId": 125,
  "company": "Musterfirma GmbH"
}
```

Webhook-Endpoint

Damit ein System einen Webhook senden kann, benötigt es eine Zieladresse.

Beispiel:

```
https://automation.example.com/webhook/new-customer
```

Diese Adresse wird als Webhook-Endpoint oder Webhook-URL bezeichnet.

Ablauf:

1. n8n stellt einen Webhook-Endpoint bereit.
2. Die Webhook-URL wird im Quellsystem eingetragen.
3. Im Quellsystem tritt ein Ereignis ein.
4. Das Quellsystem sendet einen HTTP-Request.
5. n8n empfängt die Daten.
6. Der Workflow wird gestartet.

Webhook als HTTP-Request

Ein Webhook verwendet häufig die HTTP-Methode POST.

Beispiel:

```
POST /webhook/new-employee

Content-Type: application/json

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

```
  "department": "IT",
  "startDate": "2026-08-03"
}
```

Der Empfänger antwortet beispielsweise mit:

```
Status: 200 OK
```

Oder:

```
Status: 202 Accepted
```

Vorteile von Webhooks

- Ereignisse werden sofort übertragen
- kein regelmäßiges Nachfragen notwendig
- geringere Anzahl von API-Anfragen
- schnelle Reaktion
- gut für ereignisgesteuerte Workflows
- effizienter als häufiges Polling

Nachteile von Webhooks

- Quellsystem muss Webhooks unterstützen
- Zielsystem muss erreichbar sein
- verlorene Webhooks können unbemerkt bleiben
- doppelte Übertragungen sind möglich
- öffentliche Endpoints müssen abgesichert werden
- Fehlerbehandlung ist notwendig

> Ein Webhook kann mehrfach eintreffen oder vollständig ausbleiben. Der Workflow muss darauf vorbereitet sein.

Polling

Polling bedeutet:

> Ein System fragt regelmäßig bei einem anderen System nach neuen Daten.

Beispiel:

n8n fragt alle fünf Minuten beim CRM:

> „ Gibt es neue Kunden?

Ablauf:

```
n8n
 |
 | API-Abfrage
 v
CRM
 |
 | neue Daten vorhanden?
 v
Antwort an n8n
```

Beispiel:

```
GET /customers?createdAfter=2026-07-17T10:00:00
```

Polling-Intervall

Das Polling-Intervall legt fest, wie häufig eine Abfrage ausgeführt wird.

Beispiele:

- jede Minute
- alle fünf Minuten
- jede Stunde
- einmal täglich

Ein sehr kurzes Intervall sorgt für schnelle Reaktionen, erzeugt aber viele API-Anfragen.

Ein langes Intervall reduziert die Last, kann aber zu Verzögerungen führen.

Webhook und Polling im Vergleich

Merkmal	Webhook	Polling
Auslösung	Quellsystem sendet aktiv	Zielsystem fragt regelmäßig
Geschwindigkeit	meistens sofort	abhängig vom Intervall
API-Anfragen	nur bei Ereignissen	auch ohne neue Daten
Einrichtung	Webhook-Unterstützung notwendig	API-Abfrage ausreichend
Fehlerfall	Nachricht kann verloren gehen	nächster Abruf kann Daten finden
Rate Limits	meist geringer belastet	können schneller erreicht werden

Merksatz:

> Beim Webhook meldet das Quellsystem ein Ereignis. Beim Polling fragt das Zielsystem regelmäßig danach.

Wann ist ein Webhook sinnvoll?

Ein Webhook eignet sich besonders:

- bei zeitkritischen Ereignissen
- bei häufigen Änderungen
- wenn das Quellsystem Webhooks unterstützt
- wenn der Ziel-Endpoint zuverlässig erreichbar ist
- wenn sofort reagiert werden soll

Beispiele:

- neue Supportanfrage
- neuer Mitarbeiter
- eingegangene Zahlung
- geänderter Projektstatus
- fehlgeschlagener Prozess

Wann ist Polling sinnvoll?

Polling eignet sich besonders:

- wenn keine Webhooks unterstützt werden
- wenn Änderungen nicht sofort verarbeitet werden müssen
- wenn Daten regelmäßig gesammelt werden sollen

- wenn verlorene Ereignisse vermieden werden sollen
- wenn eine API nach Änderungszeitpunkt filtern kann

Beispiele:

- stündlicher Abruf neuer Rechnungen
- täglicher Statusabgleich
- regelmäßige Kontrolle von Benutzerkonten
- Synchronisierung von Projektdaten

Zeitplan-Trigger

Ein Zeitplan-Trigger startet einen Workflow zu einem festgelegten Zeitpunkt oder Intervall.

Beispiele:

- täglich um 08:00 Uhr
- montags um 09:00 Uhr
- alle 30 Minuten
- am ersten Tag jedes Monats
- einmal pro Nacht

Mögliche Aufgaben:

- Berichte erstellen
- Lizenzen kontrollieren
- Backups prüfen
- Benutzerkonten synchronisieren
- abgelaufene Zugänge erkennen
- Erinnerungen senden

Cron-Ausdruck

Zeitpläne können mit Cron-Ausdrücken beschrieben werden.

Beispiel:

```
0 8 * * 1-5
```

Bedeutung:

> Montag bis Freitag um 08:00 Uhr

Grundaufbau:

Stelle	Bedeutung
1	Minute
2	Stunde
3	Tag des Monats
4	Monat
5	Wochentag

Weitere Beispiele:

Cron-Ausdruck	Bedeutung
`0 8 * * *`	täglich um 08:00 Uhr
`0 9 * * 1`	montags um 09:00 Uhr
`*/15 * * * *`	alle 15 Minuten
`0 0 1 * *`	am ersten Tag des Monats

Die genaue Cron-Unterstützung kann je nach System unterschiedlich sein.

Manueller Trigger

Ein Workflow kann auch manuell gestartet werden.

Dies ist sinnvoll für:

- Tests
- einmalige Aufgaben
- kontrollierte Ausführungen
- Fehleranalysen
- administrative Tätigkeiten

Ein manueller Trigger sollte nicht unkontrolliert auf Produktivdaten angewendet werden.

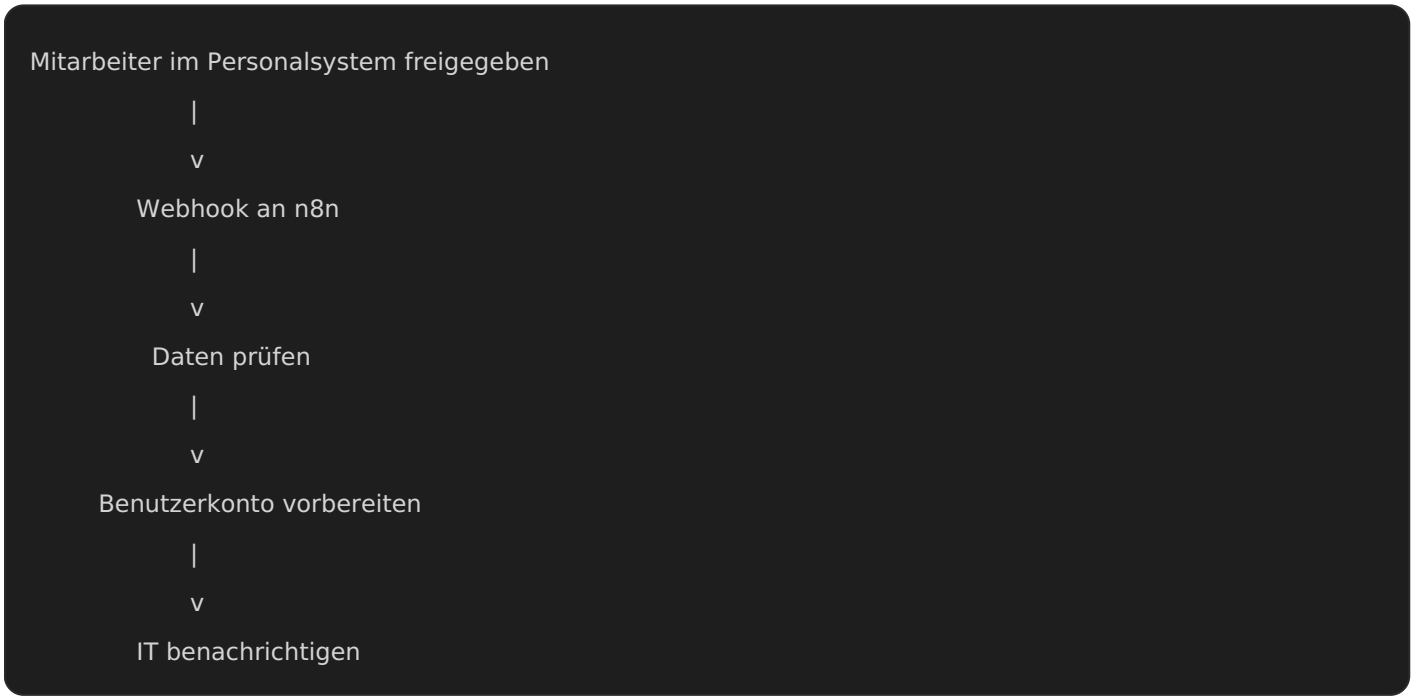
Vorher sollten geprüft werden:

- Eingabedaten
- Berechtigungen
- Zielsystem
- mögliche Änderungen
- Wiederholbarkeit

Ereignisgesteuerter Workflow

Ein ereignisgesteuerter Workflow reagiert auf ein bestimmtes Ereignis.

Beispiel:



Das Ereignis ist in diesem Fall:

> Mitarbeiter wurde freigegeben.

Webhook-Sicherheit

Webhook-Endpunkte können über das Internet erreichbar sein.

Deshalb müssen sie geschützt werden.

Mögliche Maßnahmen:

- geheime Webhook-URL
- API-Key
- Bearer-Token
- Signaturprüfung
- IP-Filter
- HTTPS
- Eingabedaten validieren
- Rate Limit
- Zeitstempel prüfen

- Ereignis-ID prüfen

Ein öffentlicher Webhook darf eingehende Daten niemals automatisch als vertrauenswürdig behandeln.

Webhook-Signatur

Ein Quellsystem kann eine Signatur mitsenden.

Beispiel:

```
X-Webhook-Signature: abc123
```

Der Empfänger berechnet selbst eine Signatur aus:

- Webhook-Inhalt
- gemeinsamem Geheimnis
- festgelegtem Verfahren

Stimmen beide Signaturen überein, wurde die Nachricht wahrscheinlich nicht verändert und stammt vom erwarteten Absender.

Replay-Angriff

Bei einem Replay-Angriff wird ein gültiger Webhook erneut gesendet.

Mögliche Folgen:

- Benutzer wird doppelt angelegt
- Zahlung wird doppelt verarbeitet
- Nachricht wird mehrfach versendet
- Aufgabe wird mehrfach erstellt

Schutzmaßnahmen:

- eindeutige Event-ID speichern
- Zeitstempel prüfen
- doppelte Ereignisse erkennen
- Idempotency-Key verwenden
- bereits verarbeitete Vorgänge ablehnen

Beispiel:

```
{
  "eventId": "evt-98452",
  "event": "employee.created"
}
```

Vor der Verarbeitung wird geprüft, ob `evt-98452` bereits verarbeitet wurde.

Doppelte Webhooks

Webhooks können mehrfach gesendet werden, wenn das Quellsystem keine erfolgreiche Antwort erhält.

Beispiel:

1. Quellsystem sendet Webhook.
2. n8n verarbeitet den Vorgang.
3. Die Antwort an das Quellsystem erreicht den Absender nicht.
4. Das Quellsystem sendet den Webhook erneut.

Der Workflow muss deshalb idempotent aufgebaut sein.

Möglicher Ablauf:

```
Webhook empfangen
      |
      v
Event-ID prüfen
      |
      v
bereits verarbeitet?
   /      \
 ja         nein
 |           |
 v           v
stoppen    verarbeiten
```

Verlorene Webhooks

Ein Webhook kann verloren gehen, wenn:

- der Zielserver nicht erreichbar ist

- DNS nicht funktioniert
- ein Timeout auftritt
- der Workflow deaktiviert ist
- der Endpoint falsch ist
- eine Firewall blockiert
- das Quellsystem keine Wiederholung durchführt

Mögliche Schutzmaßnahmen:

- Retry durch das Quellsystem
- Fehlerprotokolle
- Monitoring
- zusätzliche regelmäßige Synchronisierung
- fehlende Datensätze über Polling nachladen

Eine Kombination aus Webhook und gelegentlichem Polling kann besonders zuverlässig sein.

Webhook-Antwort

Der Empfänger sollte schnell eine passende HTTP-Response senden.

Beispiele:

Statuscode	Bedeutung
200	Webhook erfolgreich verarbeitet
202	Webhook angenommen, Verarbeitung folgt
400	Daten fehlerhaft
401	Authentifizierung fehlt
403	Zugriff nicht erlaubt
500	interner Fehler

Bei umfangreichen Workflows kann zuerst `202 Accepted` zurückgegeben und die eigentliche Verarbeitung anschließend durchgeführt werden.

Praxisbeispiel: IT-Onboarding

Ein Personalsystem sendet einen Webhook:

```
{
  "eventId": "evt-10025",
  "event": "employee.approved",
```

```
  "employee": {
    "name": "Max Mustermann",
    "department": "IT",
    "startDate": "2026-08-03"
  }
}
```

Möglicher Workflow:

1. Signatur prüfen
2. Event-ID prüfen
3. Pflichtfelder validieren
4. Startdatum kontrollieren
5. Benutzerkonto vorbereiten
6. Projektaufgabe erstellen
7. Slack-Nachricht senden
8. Ergebnis protokollieren
9. Event-ID als verarbeitet speichern

Praxisbeispiel: Polling

Ein SaaS-System unterstützt keine Webhooks.

n8n fragt alle 15 Minuten:

```
GET /employees?modifiedAfter=2026-07-17T12:00:00
```

Möglicher Ablauf:

1. letzten erfolgreichen Abrufzeitpunkt laden
2. neue oder geänderte Datensätze abrufen
3. Datensätze prüfen
4. jeden Datensatz verarbeiten
5. Ergebnisse protokollieren
6. neuen Abrufzeitpunkt speichern

Der letzte Abrufzeitpunkt darf erst gespeichert werden, wenn die Verarbeitung erfolgreich abgeschlossen wurde.

Zeitzonen

Zeitplan-Trigger müssen die richtige Zeitzone verwenden.

Beispiel:

```
Europe/Berlin
```

Probleme können entstehen durch:

- Sommerzeit
- Winterzeit
- unterschiedliche Serverzeitzonen
- UTC-Zeit
- internationale Standorte

Ein Workflow um 08:00 Uhr sollte eindeutig festlegen, welche Zeitzone gemeint ist.

Monitoring

Trigger und Workflows sollten überwacht werden.

Wichtige Kontrollen:

- letzter erfolgreicher Lauf
- letzte fehlgeschlagene Ausführung
- Anzahl empfangener Webhooks
- Dauer des Workflows
- wiederholte Ereignisse
- fehlende Daten
- Rate Limits
- ungewöhnlich viele Fehler

Beispiel:

> Wenn seit 24 Stunden kein erwarteter Webhook empfangen wurde, wird die IT informiert.

Systematische Fehlersuche

Wenn ein Workflow nicht startet:

1. Ist der Workflow aktiviert?
2. Ist der richtige Trigger konfiguriert?
3. Ist die Webhook-URL korrekt?
4. Ist der Endpoint erreichbar?

5. Wird die richtige HTTP-Methode verwendet?
6. Wird der Webhook im Quellsystem ausgelöst?
7. Sind Authentifizierung oder Signatur gültig?
8. Ist der Zeitplan korrekt?
9. Ist die richtige Zeitzone eingestellt?
10. Wurde ein Rate Limit erreicht?
11. Gibt es einen Fehler im Trigger-Log?
12. Wurde das Ereignis möglicherweise als Duplikat erkannt?

Wichtige Begriffe

Begriff	Bedeutung
Trigger	startet einen Workflow
Webhook	aktive Ereignismeldung eines Systems
Polling	regelmäßige Abfrage nach Änderungen
Zeitplan	festgelegter Ausführungszeitpunkt
Cron	Schreibweise für wiederkehrende Zeitpläne
Event-ID	eindeutige Kennung eines Ereignisses
Idempotenz	doppelte Ausführung erzeugt keine zusätzlichen Änderungen
Replay-Angriff	erneutes Senden einer gültigen Nachricht
Signatur	Nachweis von Herkunft und Unveränderheit
Retry	erneuter Übertragungsversuch
Zeitzone	regionale Grundlage einer Zeitangabe

Gesamtmerksatz

> Trigger starten automatisierte Workflows. Webhooks melden Ereignisse aktiv und meist sofort, während Polling regelmäßig nach neuen Daten fragt. Zeitplan-Trigger starten Workflows zu festgelegten Zeiten. Sichere und zuverlässige Automatisierungen müssen doppelte, verlorene und manipulierte Ereignisse berücksichtigen.

Kontrollfragen

Was ist ein Trigger?

Ein Ereignis oder Zeitpunkt, der einen Workflow startet.

Was ist ein Webhook?

Eine automatische HTTP-Nachricht, die beim Eintritt eines Ereignisses an ein anderes System gesendet wird.

Was bedeutet Polling?

Ein System fragt regelmäßig nach neuen oder geänderten Daten.

Was ist der wichtigste Unterschied zwischen Webhook und Polling?

Beim Webhook sendet das Quellsystem aktiv. Beim Polling fragt das Zielsystem regelmäßig nach.

Was ist ein Zeitplan-Trigger?

Ein Trigger, der einen Workflow zu einer festgelegten Zeit oder in einem bestimmten Intervall startet.

Warum muss eine Event-ID geprüft werden?

Damit ein Ereignis nicht mehrfach verarbeitet wird.

Warum müssen Webhook-Daten validiert werden?

Weil ein öffentlich erreichbarer Endpoint manipulierte oder fehlerhafte Daten empfangen kann.

Was ist ein Replay-Angriff?

Eine bereits gültige Nachricht wird erneut gesendet.

Warum ist die Zeitzone bei Zeitplänen wichtig?

Weil der Workflow sonst zu einer falschen Uhrzeit ausgeführt werden kann.

Wann kann eine Kombination aus Webhook und Polling sinnvoll sein?

Wenn Ereignisse sofort verarbeitet und möglicherweise verlorene Webhooks später durch eine Kontrollabfrage erkannt werden sollen.

Quellen

- MDN Web Docs – HTTP
- n8n-Dokumentation – Webhook Node
- n8n-Dokumentation – Schedule Trigger

- n8n-Dokumentation – Trigger Nodes
- OWASP – Webhook Security Guidelines