

Wer darf das *freigeben*?

Jemand will etwas bestellen: Ein Monitor für 89 Euro geht sofort raus, ein Messestand für 4.500 Euro zur Geschäftsführung. Fehlt der Betrag, gibt es eine Rückfrage.



Der Fall: Bestellanforderung freigeben

- **Unter 100 Euro:** sofort freigeben, die Bestellung geht raus.
- **100 bis 1.000 Euro:** die Teamleitung entscheidet.
- **Über 1.000 Euro:** die Geschäftsführung entscheidet.
- **Kategorie IT:** zusätzlich eine IT-Prüfung, egal welcher Betrag.
- **Betrag fehlt oder ist Text:** Rückfrage an die Person, die bestellt hat.

Merk dir Jeder Weg endet mit einer Handlung: freigeben, weitergeleitet oder rückgefragt.
Keine Anforderung darf verschwinden.

Sechs Testfälle, bevor du baust

FALL	BETRAG	KATEGORIE	ERWARTETER WEG	IT-PRÜFUNG
B-01 Monitor	89	IT	Sofort freigegeben	ja
B-02 Bürostuhl	100	Büro	Teamleitung	nein
B-03 Laptop	1000	IT	Teamleitung	ja
B-04 Messestand	4500	Marketing	Geschäftsführung	nein
B-05 Kaffee	leer (Text)	Büro	Rückfrage	keine
B-06 Software	„ca. 300“ (Text)	IT	Rückfrage	keine

Diese Tabelle ist dein Testplan. Du baust nicht drauflos, du baust gegen die Tabelle.

Eine Anforderung ist ein Item

```
{ "nummer": "B-01", "artikel": "Monitor", "betrag": 89, "kategorie": "IT" }
```

- Links vom Doppelpunkt steht der Feldname, rechts der Wert.
- Text steht in Anführungszeichen. Eine Zahl nicht.
- `betrag` muss eine Zahl sein. Sonst kann niemand vergleichen, ob 89 unter 100 liegt.

Schritt 1: Die Anforderung

Manual Trigger, umbenannt in „Manuell starten“. Dahinter ein Edit Fields „Anforderung“ mit vier Feldern:

NAME	TYP	WERT
nummer	String	B-01
artikel	String	Monitor
betrag	Number	89
kategorie	String	IT

Hier trägst du später jeden Testfall ein. Ein Lauf, ein Fall.

Merk dir Der Typ Number bei **betrag** ist Absicht.

Expressions: drei Teile in geschweiften Klammern

Alles zwischen `{{` und `}}` rechnet n8n aus, statt es als Text zu nehmen. Jede Formel hat drei Teile:

TEIL	FRAGE	BEISPIEL
Woher	aus welchem Node?	<code>\$json</code> = vom Node direkt davor, <code>\${'Anforderung'}.item.json</code> = vom Node „Anforderung“
Welches Feld	Punkt, dann Feldname	<code>.betrag</code> , <code>.kategorie</code>
Was damit	vergleichen, rechnen, zusammensetzen	<code>< 100</code> , <code>=== 'IT'</code> , <code>+ ' Euro'</code>

Zwei Formeln zum Ausprobieren

Hinter die Anforderung ein Edit Fields „Probe“ mit zwei Feldern, Schalter auf **Expression**:

```
{{ $json.betrag < 100 }}
```

```
{{ $('Anforderung').item.json.kategorie === 'IT' ? 'ja' : 'nein' }}
```

- Die erste ergibt `true` oder `false`. Die zweite ergibt `ja` oder `nein`: Fragezeichen heißt dann, Doppelpunkt heißt sonst.
- Die Vorschau unter dem Feld zeigt sofort das Ergebnis. `[undefined]` heißt: Das Feld gibt es so nicht. Bei falschem Node-Namen meldet n8n einen Fehler.
- Lieber klicken als tippen? Der [Expression-Baukasten](#) setzt die drei Teile zusammen.

Schritt 2: If prüft, ob der Betrag eine Zahl ist

If „Betrag gültig?“ hinter die Anforderung:

TEIL	WERT
linker Wert (Expression)	<code>{{ typeof \$('Anforderung').item.json.betrag === 'number' }}</code>
Operator	Boolean, is true
true-Ausgang	weiter zur Verteilung
false-Ausgang	Edit Fields „Rückfrage Betrag fehlt“: <code>status</code> = Rückfrage, <code>naechster_schritt</code> = Person nach dem Betrag fragen

Ein If stellt eine Ja-Nein-Frage. Jedes Item landet auf genau einem der zwei Ausgänge.

Zahl oder Text? 300 ist nicht „ca. 300“

WERT IM ITEM	TYP	WAS DER IF SAGT
300	Zahl	true , weiter
"300"	Text	false , Rückfrage
"ca. 300"	Text	false , Rückfrage
"" (leer)	Text	false , Rückfrage

- Formulare liefern fast immer Text, auch bei Zahlen.
- Ohne Prüfung bricht ein Zahlenvergleich mit „Wrong type“ ab.
- Mit Prüfung gibt es eine Rückfrage statt eines roten Nodes.

Schritt 3: Switch verteilt nach Betrag

Switch „Nach Betrag verteilen“ am true-Ausgang des If.

Linker Wert in jeder Regel: `{{ $('Anforderung').item.json.betrag }}`

REGEL	BEDINGUNG	AUSGANG
1	Number, is less than 100	Sofort
2	Number, is less than or equal to 1000	Teamleitung
Fallback	keine Regel passt, also über 1.000	Geschäftsführung

Von oben nach unten: 89 bleibt bei Regel 1 hängen, 100 fällt durch zu Regel 2.

Achtung Ohne Fallback verschwinden Items, die zu keiner Regel passen. Still, ohne Fehler.

Filter: der Baustein, der wegwirft

IF	SWITCH	FILTER
zwei Ausgänge	viele Ausgänge plus Fallback	ein Ausgang
nichts geht verloren, wenn beide Ausgänge weiterführen	nichts geht verloren, wenn der Fallback da ist	der Rest wird verworfen
„Ja oder nein?“	„Wohin damit?“	„Brauche ich das überhaupt?“

Filter ist richtig, wenn du den Rest wirklich nie wieder brauchst. Eine Bestellanforderung darf nie einfach verschwinden.

Schritt 4: Drei Ergebnisse, und die Falle mit den Feldern

An jeden Switch-Ausgang ein Edit Fields, von oben nach unten: Sofort, Teamleitung, Geschäftsführung.

NODE-NAME	status	naechster_schritt
Sofort freigegeben	freigegeben	Bestellung auslösen
An Teamleitung	wartet auf Teamleitung	Freigabe anfragen
An Geschäftsführung	wartet auf Geschäftsführung	Freigabe anfragen

Achtung In allen drei **Include Other Input Fields** einschalten. Sonst ist die Nummer weg, und niemand weiß mehr, welche Anforderung freigegeben wurde.

Schritt 5: Die IT-Prüfung als Expression

In jedem der drei Ergebnisse ein drittes Feld `it_pruefung` (String), Schalter auf Expression:

```
{{ $('Anforderung').item.json.kategorie === 'IT' ? 'ja' : 'nein' }}
```

TEIL	BEDEUTUNG
<code>\$('Anforderung').item.json.kategorie</code>	die Kategorie aus der Anforderung
<code>=== 'IT' ?</code>	wenn genau IT
<code>'ja' : 'nein'</code>	dann ja, sonst nein

Den Node beim Namen nennen, nicht `$json`. Die Vorschau zeigt den echten Wert.

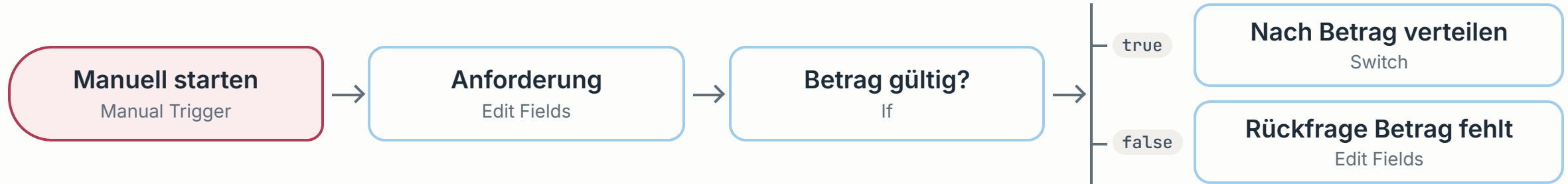
Warum ist die IT-Prüfung kein vierter Weg?

Ein Switch schickt jedes Item auf **genau einen** Ausgang. Ein vierter Ausgang „IT“ hieße: Der Laptop für 1.000 Euro geht zur IT und **nicht mehr** zur Teamleitung. Die Betragsregel wäre weg.

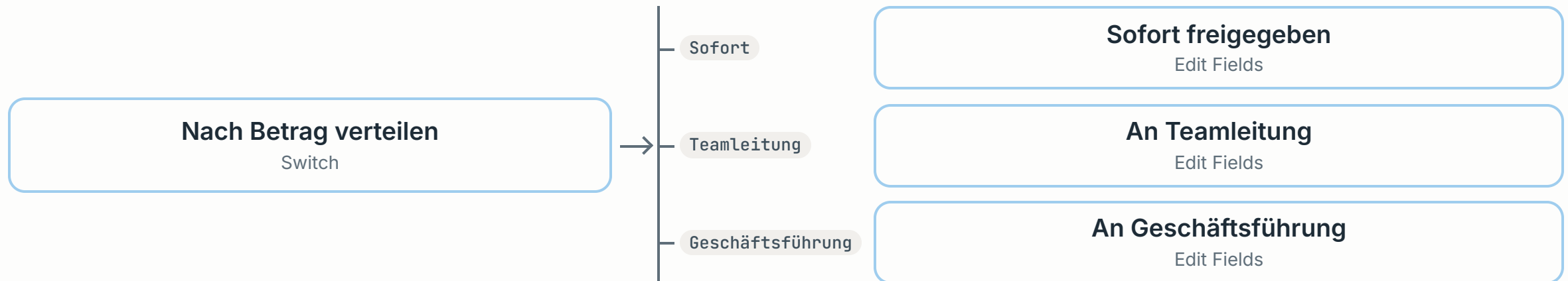
BAUART	WIE	WANN
Feld	<code>it_pruefung</code> in jedem Ergebnis	Die IT bekommt die Info mit, der Weg bleibt gleich.
Schritt davor	If „Kategorie IT?“ vor dem Switch, bei true ein Edit Fields „IT-Prüfung anfragen“, dann weiter zum Switch	Die IT soll wirklich zuerst dran sein, bevor verteilt wird.

Merk dir Trifft etwas **statt** der anderen zu, ist es ein Ausgang. Trifft es **zusätzlich** zu, ist es ein Feld oder ein Schritt davor.

Zum Nachbauen: der ganze Workflow



WEITER AM TRUE-AUSGANG: DERSELBE SWITCH



Zwei Läufe zum Start: B-01 endet bei „Sofort freigegeben“, B-04 bei „An Geschäftsführung“.

Grenzfälle: da entscheidet sich, ob es stimmt

FALL	WARUM ER WICHTIG IST
B-02, genau 100	„unter 100“ heißt kleiner, nicht kleiner gleich: Teamleitung, nicht Sofort
B-03, genau 1.000	„bis 1.000“ heißt kleiner gleich: noch Teamleitung
B-05, leerer Betrag	darf nicht abstürzen, muss zur Rückfrage
B-06, „ca. 300“	Text, muss zur Rückfrage
jedes Ergebnis	ist die <code>nummer</code> noch im Output?

Merk dir Ein Workflow, der nur mit dem Monitor getestet wurde, ist nicht getestet.

Merge: Wege wieder zusammenführen, wenn nötig

- Nach einer Weiche laufen die Wege meist getrennt zu Ende. Das ist in Ordnung.
- Merge brauchst du, wenn zwei Wege danach **denselben** Schritt brauchen, etwa eine Bestätigung an die Person, die bestellt hat.
- **Merge, Mode Append:** die Items beider Eingänge hintereinander.

Achtung Zwei Ausgänge in denselben Eingang eines Nodes stecken sieht aus wie ein Merge, ist aber keiner. Willst du Wege zusammenführen, nimm den Merge-Node.

KI darf vorschlagen, du prüfst

Auftrag an eine KI: „In n8n habe ich das Textfeld kategorie. Gib mir eine Expression, die Groß- und Kleinschreibung und äußere Leerzeichen ignoriert, damit auch ein kleines it mit Leerzeichen als IT zählt.“

Typischer Vorschlag: `{{ String($json.kategorie ?? '').trim().toUpperCase() }}`

PRÜFFRAGE

ANTWORT

Was macht er?

Text draus machen, Leerzeichen außen weg, alles groß

Was, wenn `kategorie` fehlt?

leerer Text statt Fehler

Was, wenn eine Zahl kommt?

wird still zu Text

Übernehmen?

ja, mit Node-Namen statt `$json`, dann die Testfälle laufen lassen

Dein Auftrag

- 1 **Nachbauen:** die ganze Kette, dann alle sechs Testfälle. Vor jedem Lauf die Erwartung aufschreiben, danach das Ergebnis.
- 2 **Abwandeln:** erst Drei-Punkte-Menü, Duplicate. Dann eine Stufe über 5.000 Euro für den Einkauf, Kategorie Reise immer zur Teamleitung und die IT-Prüfung als eigener Schritt vor dem Switch. Nach jeder Änderung alle Fälle erneut.
- 3 **Formular davor, wenn du magst:** auch hier erst Duplicate, denn Teil 6 braucht das Original. Form Trigger statt Manual Trigger, alle Felder Pflicht. Die Anforderung behält ihren Namen und holt ihre Werte aus dem Formular, `betrag` mit `Number()`. Nur der If prüft dann mit `Number.isFinite`.

Merk dir Neue Testfälle: Server, 12.000 Euro, IT › Einkauf. Messestand, genau 5.000 Euro, Marketing › Geschäftsführung. Bahnticket, 45 Euro, Reise › Teamleitung.

Was bleibt hängen?

1 Was passiert mit einer Anforderung über 1.000 Euro, wenn der Switch keinen Fallback hat?

Die Anforderung verschwindet still. Der Workflow bleibt grün, die Geschäftsführung erfährt nichts.

2 Ist 100 unter 100?

Nein. Der Bürostuhl für genau 100 Euro geht zur Teamleitung.

3 Warum fehlt nach Edit Fields plötzlich die Nummer?

Include Other Input Fields war aus. Dann gibt der Node nur die Felder aus, die er selbst anlegt.

4 Wann nimmst du Filter, wann If oder Switch?

Filter nur, wenn du den Rest nie wieder brauchst. Sonst If oder Switch mit Fallback.

Weiter im Kurs

LERNPFAD

Lektionen zum Nachklicken

5 · Mapping und Datentypen

10 · Items und Datenstruktur

11 · Expressions

12 · Flow-Logik

Lektion 10 bis 17 blendest du im Lernpfad mit dem Schalter „Fortgeschritten“ ein.

EXPRESSION-BAUKASTEN

Rezepte aus dem Kurs

WISSENSBOX

Die wichtigsten Dinge, in Bewegung

ÜBUNG 2

Item-Zahlen-Trainer

NÄCHSTER TEIL

Teil 6: Testen, Fehler, Übergabe

Erik



Du willst mehr als Selbstlernen? Dann lernst du n8n mit mir 1:1. Auf Anfrage bringe ich es auch Unternehmerinnen, Unternehmern und ganzen Teams bei. Erzähl mir im Erstgespräch, was du vorhast.

15 Minuten mit Erik >