

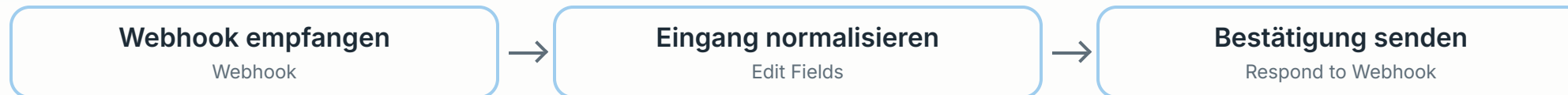
Daten, die du nicht selbst *tippst*

In Teil 3 hast du eine Frage ins Internet geschickt und eine Antwort bekommen. Jetzt kommt eine ganze Liste zurück, und danach drehst du die Richtung um. Keine neuen Formeln zum Selberschreiben. Nur klicken, ausführen, hinschauen.

FRAGEN: PRODUKTLISTE AUS DEM INTERNET



ZUGESTELLT BEKOMMEN: DAS POSTFACH



Zwei Richtungen: fragen und zugestellt bekommen

RICHTUNG	WER IST AKTIV?	NODE	BEISPIEL
n8n fragt	n8n	HTTP Request	Wetter abrufen, Produktliste holen
n8n bekommt zugestellt	das andere System	Webhook	Shop meldet Bestellung, Formular meldet Eingabe

Erst fragen, dann zugestellt bekommen. [Lektion 4 im Lernpfad](#) spielt beide Richtungen zum Anklicken durch.

Merk dir Beim Fragen ist n8n aktiv. Beim Zustellen ist es das andere System, und n8n wartet.

HTTP Request: n8n stellt eine Frage im Internet

FELD	WAS DU EINTRÄGST	HIER
Method	GET = fragen, POST = schicken	GET
URL	die Adresse	<code>https://dummyjson.com/products</code>
Send Query Parameters	Zusatzwünsche hinter dem Fragezeichen	an, <code>limit</code> = <code>30</code>
Authentication	Zugangsdaten	None

dummyjson.com ist ein Übungsdienst mit erfundenen Produkten. Kein Konto, keine Kosten.

Merk dir Will ein Dienst Zugangsdaten, kommen sie unter Authentication als Credential hinein. Nie in ein Textfeld.

Die Antwort lesen: ein Item, und eine Liste darin

```
{
  "products": [
    { "title": "Essence Mascara Lash Princess", "price": 9.99, "rating": 2.56, ... },
    ...
  ],
  "total": 194
}
```

- Geschweifte Klammer: ein Datensatz mit Feldern.
- Eckige Klammer: eine Liste. Hier stecken die 30 Produkte drin.
- Oben rechts im Output: **Schema, Table, JSON**. Schema zeigt nur die Feldnamen, das ist die entspannte Ansicht.

Split Out: aus einer Liste werden Items

- Im Menü: Plus, Suche „Split Out“, umbenennen in „Liste aufteilen“.
- **Fields To Split Out:** `products` . Tippen oder aus dem Schema links ins Feld ziehen.
- Vorher: 1 Item mit einer Liste von 30 Produkten.
- Nachher: 30 Items, jedes ein Produkt.

Merk dir Ab hier läuft jeder weitere Node dreißig Mal, einmal pro Produkt. Die Zahl oben rechts am Node springt von 1 auf 30.

Felder auswählen: nur behalten, was du brauchst

Edit Fields „Felder auswählen“, vier Felder, Werte per Drag-and-drop aus dem Schema:

NAME	TYP	WERT
produkt	String	{{ \$json.title }}
preis	Number	{{ \$json.price }}
bewertung	Number	{{ \$json.rating }}
kategorie	String	{{ \$json.category }}

Include Other Input Fields bleibt aus: Der Rest fliegt raus. Immer noch 30 Items.

Achtung preis und bewertung brauchen den Typ Number. Als String sortiert der Sort nach Zeichen, nicht nach Wert.

Filter, Sort, Limit: drei Werkzeuge, drei Klicks

NODE	NAME IM WORKFLOW	EINSTELLUNG	ITEMS
Filter	Bewertung ab 4?	bewertung, Number, is greater than or equal to, 4	von 30 auf etwa 13
Sort	Nach Preis sortieren	Type Simple, Field preis, Order Descending	bleibt
Limit	Nur die ersten 5	Max Items 5	auf 5

Merk dir Filter wirft weg, was nicht passt: kein zweiter Ausgang, kein roter Node. Die Zahl am Node zeigt, wie viele übrig sind.

Wann Filter und wann If: [Lektion 12](#).

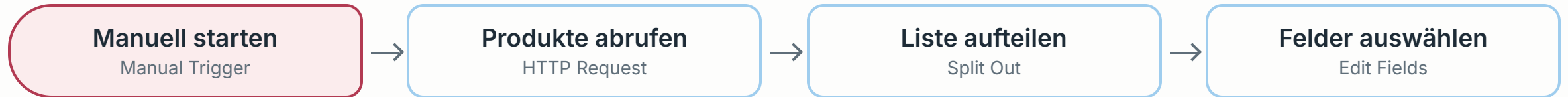
Aggregate: alles wieder in ein Item packen

- Im Menü: Plus, Suche „Aggregate“, umbenennen in „Alle in ein Item“.
- **Aggregate:** All Item Data (Into a Single List).
- **Put Output in Field:** produkte .
- Aus 5 Items wird 1 Item mit einer Liste produkte von 5 Produkten.

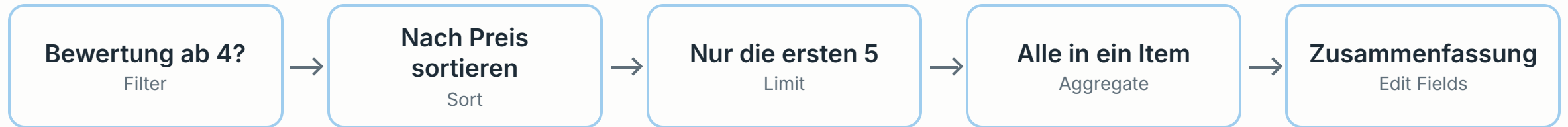
Merk dir Das Gegenteil von Split Out. Du brauchst es, wenn am Ende **eine** Nachricht rausgehen soll, nicht fünf.

Zum Nachbauen: die Produktliste

HOLEN UND AUSPACKEN



AUSWÄHLEN UND EINPACKEN



- Item-Zahlen von „Produkte abrufen“ bis „Zusammenfassung“: 1, 30, 30, etwa 13, 13, 5, 1, 1.
- „Zusammenfassung“ macht aus der Liste einen Satz. Felder und Formel stehen auf der nächsten Folie.

Aus der Liste einen Satz bauen

Edit Fields „Zusammenfassung“ mit drei Feldern. In `text` kommt diese Expression:

```
{{ $json.produkte.map(p => p.produkt + ' (' + p.preis + ' USD)').join(', ') }}
```

FELD	TYP, SCHALTER, WERT
titel	String, Fixed, Top-Produkte ab Bewertung 4
text	String, Expression, die Formel oben
anzahl	Number, Expression, {{ \$json.produkte.length }} , ergibt 5

Beide Expressions stehen als Rezept im [Expression-Baukasten](#), `text` dort nur mit den Namen.

Achtung Leerzeichen zählen mit: vor der Preisklammer, vor USD, nach dem Komma.

Die Zahl am Node ist deine Kontrolle

- Nach jedem Execute step steht oben rechts am Node die Zahl der Items.
- Von 1 auf 30: Split Out hat gearbeitet.
- Von 30 auf 13: Der Filter hat 17 weggeworfen.
- Von 13 auf 5: Das Limit hat gekürzt.
- Von 5 auf 1: Aggregate hat eingepackt.
- 0: Hier ist alles weg. Diesen Node öffnen, Input und Output vergleichen.

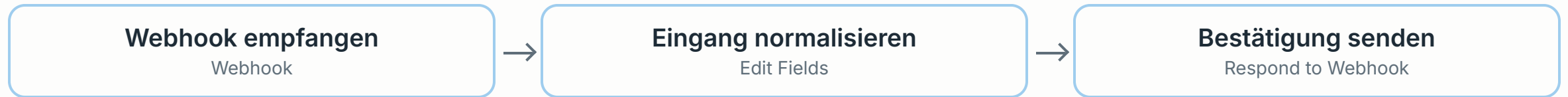
Merk dir Kein Node ist rot, und trotzdem kann das Ergebnis falsch sein. Die Zahl merkt es zuerst.

Webhook: ein anderes System klingelt bei dir

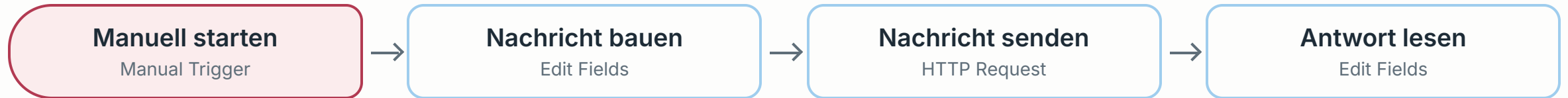
- **Webhook** ist ein Trigger. Im Menü: Add first step, „On webhook call“. n8n baut daraus eine Adresse.
- **Test URL:** gilt nur, solange du auf Execute workflow gedrückt hast und der Node „Listening“ zeigt, höchstens zwei Minuten lang. Eine Nachricht pro Klick.
- **Production URL:** dauerhaft, erst nach Publish.
- **HTTP Method POST:** Der Sender schickt Daten mit. Die landen im Output unter `body`.
- **Respond to Webhook:** die Antwort an den Sender, zum Beispiel „angenommen“.

Der Sender: HTTP Request mit POST

POSTFACH



SENDER



- 1 Postfach: Execute workflow. Der Webhook zeigt „Listening“.
- 2 Sender: Execute workflow, innerhalb von zwei Minuten. Die Nachricht geht an die Test URL.
- 3 Im Sender steht rechts die Antwort, im Postfach liegt die Nachricht unter `body`.

Postfach und Sender einstellen

NODE

EINSTELLUNG

Webhook empfangen	HTTP Method POST, Path <code>mein-postfach</code> , Respond: Using 'Respond to Webhook' Node
Eingang normalisieren	<code>absender</code> = <code>{{ \$json.body.absender }}</code> , <code>betreff</code> = <code>{{ \$json.body.betreff }}</code>
Bestätigung senden	Respond With JSON, Response Body <code>{ "status": "angenommen" }</code>
Nachricht bauen	<code>absender</code> Muster GmbH, <code>betreff</code> ein kurzer Satz, <code>vorgang</code> V-1001
Nachricht senden	POST, URL = Test URL des Postfachs, Send Body an, JSON, drei Felder
Antwort lesen	<code>antwort_status</code> = <code>{{ \$json.status }}</code>

Wenn es hakt: Liste und Webhook

SYMPTOM	URSACHE	WAS TUN
Split Out: 0 Items, „wasn't found“	Feldname falsch geschrieben	<code>products</code> aus dem Schema ziehen
Sort sortiert komisch	preis ist ein String	Typ Number in „Felder auswählen“
Filter liefert 0	Grenze zu hoch	Zahl lesen, Input gegen Output
Filter rot: „Wrong type“	bewertung ist Text	Typ Number in „Felder auswählen“
Aggregate: Feldlisten statt <code>produkte</code>	Modus Individual Fields	All Item Data, Feld <code>produkte</code>
Sender bekommt 404	Postfach lauscht nicht, Pfad anders	erst Postfach starten, URL kopieren
Postfach zeigt kein „Listening“ mehr	lauscht höchstens zwei Minuten	Postfach neu starten, gleich senden
body im Postfach leer	Send Body aus oder keine Felder	Send Body an, JSON, drei Felder
zweite Nachricht kommt nicht an	Test URL gilt für eine Nachricht	Postfach neu starten, dann senden

Dein Auftrag

- 1 **Nachbauen:** die Produktliste komplett. Einmal ausführen, die Item-Zahl jedes Nodes notieren.
- 2 **Abwandeln:** Filter auf 4.5, Sortierung Ascending, Limit 3. Nach jeder Variante die Zahlen notieren und erklären.
- 3 **Titel anpassen:** Ändert sich die Filtergrenze, passt du die Zahl in `titel` der Zusammenfassung an.
- 4 **Postfach und Sender:** zwei Workflows. Nachricht schicken, Antwort und body prüfen. Dann im Postfach nur den Pfad ändern, wieder lauschen lassen, noch einmal senden.

Merk dir Das Postfach nicht veröffentlichen. Zum Üben reicht die Test URL.

Was bleibt hängen?

1 Was macht Split Out, und woran siehst du, dass es funktioniert hat?

Aus einem Item mit einer Liste darin werden dreißig Items. Die Zahl am Node springt von 1 auf 30.

2 Warum ist die Zahl am Node wichtiger als die Farbe?

Ein Node kann grün sein und trotzdem 0 Items liefern. Grün heißt nur, dass er durchgelaufen ist.

3 Wer ist beim Webhook aktiv, wer beim HTTP Request?

Webhook: Das andere System sendet. HTTP Request: n8n fragt, auch der Sender mit POST.

4 Warum schweigt das Postfach, wenn niemand Execute workflow gedrückt hat?

Die Test URL gilt nur bei „Listening“, höchstens zwei Minuten lang. Dauerhaft gilt die Production URL, erst nach Publish.

Weiter im Kurs

LERNPFAD

Lektionen zum Nachklicken

4 · Schnittstelle

10 · Items und Datenstruktur

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

ÜBUNG 2

Item-Zahlen-Trainer

ÜBUNG 3

Webhook-Simulator: Sender und Postfach

WISSENSBOX

Die wichtigsten Dinge, in Bewegung

NÄCHSTER TEIL

Teil 5: Bestellanforderung freigeben

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 >