

Cook Book

Promillo

Für das Modul New Trends in IT und Management der digitalen
Transformation
Provadis School of International Management and Technology
von

Ole Grundmann
Moritz Bosch
Ben Zelleröhr

ole.grundmann@stud-provadis-hochschule.de	Matrikelnummer.Ole
moritz.bosch@stud-provadis-hochschule.de	D490
ben.zelleroehr@stud-provadis-hochschule.de	D532

Inhaltsverzeichnis

1	Einleitung	3
2	Technische Beschreibung	4
2.1	Blackbox-Sicht	4
2.2	Komponenten	4
3	Anleitung zur selbständigen Reproduktion	7
3.1	Einrichtung nginx	7
3.2	n8n und Mongo	7
3.3	Workflow in n8n	8

1 Einleitung

2 Technische Beschreibung

2.1 Blackbox-Sicht

[Ich warte auf Ole, damit wir nicht Sachen doppelt schreiben]

2.2 Komponenten

Der Workflow ist aus mehreren gedanklichen Komponenten zusammengesetzt, welche wiederum aus kleineren Teilen bestehen.

Nutzerinformationen

Damit der Workflow laufen und eine Antwort liefern kann, muss der Nutzer relevante Informationen bereitstellen. Diese werden über ein Formular abgefragt. Die Fragen sind:

- Wie viele Personen wollen trinken?
- Welche Geschmacks-Präferenzen haben die Personen?
- Bilder über zur Verfügung stehenden Getränken oder anderen Lebensmitteln.

Sollte eine Personenanzahl < 1 eingegeben werden, wird die Anzahl automatisch auf 1 gesetzt. Anschließend kann der Nutzer auf den Button klicken, damit der Workflow gestartet wird. Im Anschluss wird der Nutzer gebeten zu warten, bis die nächste Komponente abgeschlossen ist.

Bildanalyse

Die Bilder werden an ein KI-Modell übergeben. Das Modell versucht das im Vordergrund stehende Getränk zu erkennen. Dabei wird der Name des Getränkes, sowie das Fassungsvermögen in Milliliter und der aktuellen Füllstand in Prozent ermittelt. Wir nutzen dafür das Modell **GPT-4O-MINI** welches wir mit dem Workflow verbunden haben.

Der genutzte Prompt sieht wie folgt aus:

Erkenne auf jedem Bild genau eine Flasche (die im Fokus bzw. im Vordergrund steht)

{Genaue Bezeichnung des Getränks, z. B. "Coca-Cola Zero Sugar", "Gerolsteiner Mineral

Die Ausgabe des Modells erfolgt im JSON-Format, für die weitere Nutzung der Daten im Workflow.

Datenvalidierung

Zur korrekten Weiterverarbeitung der Daten wird der Nutzer gebeten, diese zu validieren bzw. zu korrigieren. Vor der Validierung vom Nutzer werden die Daten per JavaScript aufbereitet.

Zum genaueren Verständnis kann der folgende Code in Quellcode 1 betrachtet werden. Es ist zu beachten, dass teilweise n8n Spezifische Syntax verwendet wird.

Anschließend werden dem Nutzer sämtliche Daten im Formular angezeigt, welche er bei bedarf anpassen bzw. überschreiben kann. Wenn keine Veränderungen vorgenommen werden, wird der Wert vom KI-Modell übernommen.

Daraufhin werden die Daten wieder per JavaScript verarbeitet, damit sie im nächsten Schritt von einem weiteren KI-Modell genutzt werden können. Quellcode 2 zeigt den Code, welcher dafür genutzt wird.

Rezeptvorschlag

Die validierten Daten, welche aus der vorherigen Komponente stammen, werden nun an ein weiteres KI-Modell gegeben. Genutzt wird folgender System Prompt:

`Du bist MixMaster AI, ein hochqualifizierter virtueller Barkeeper und Cocktailexper`

`1. Eingaben`

- `- Präferenzen (z. B. Geschmack: süß, sauer, herb etc.)`
- `- Verfügbare Produkte (Liste von Spirituosen, Likören, Sirups, Säften, Bitters,`
- `- Anzahl Personen, für die gemixt werden soll`

`2. Verarbeitung`

- `- Greife auf CocktailDB zu, um passende Rezepte zu finden.`
- `- Nutze nicht zwingend alle verfügbaren Zutaten - wähle die besten Kombinationen`
- `- Falls ein Rezept eine oder mehrere Zutaten enthält, die der Nutzer nicht vorrä`

`Verhalte dich stets professionell, freundlich und präzise.`

und folgender User Prompt:

Erstelle 5 kreative aber auch gängige Cocktail-Vorschläge auf Basis der folgenden A

```
**Präferenzen**: {{ $('Edit Fields').item.json.preferenzen }}  
**Anzahl Personen**: {{ $('Edit Fields').item.json.personen }}  
**Verfügbare Produkte**: {{ $json.products }}
```

Die Inhalte von Verfügbare Produkte sind {Produkt},{Maximales Volumen in Milliliter

Als Modell wird erneut **GPT-4O-MINI** genutzt. Zudem kommt eine verbundene Mongo-Datenbank zum Einsatz. Darin können Rezepte gespeichert werden, auf welche das KI-Modell zugreifen kann. Zudem werden diese Rezepte vom Modell bevorzugt ausgegeben, wenn sie mit den vermittelten Daten eine Schnittmenge bilden. Dadurch können Präferenzen des Nutzers besser berücksichtigt werden, als auch die Halluzinationen des Modells reduziert werden.

Die Ausgabe des Modells erfolgt im JSON-Format, in folgendem Schema: Quellcode 3.

Ergebnispräsentation

Die Ergebnisse der vorherigen Komponente werden dem Nutzer präsentiert. Dafür werden die Daten erneut in JavaScript aufbereitet, damit sie in HTML Code umgewandelt werden. Der Quellcode 4 zeigt den Code, welcher dafür genutzt wird.

Anschließend wird der übergebene HTML Code wieder im Formular angezeigt.

Dazu kommen noch folgende CSS Styles, um die Darstellung zu verbessern: Quellcode 5.

Einkaufslisten-API

Um fehlende Zutaten in eine Einkaufsliste zu übertragen, werden diese per POST-Request an eine dafür angepasste Web-App übermittelt. Dafür wird pro Getränk eine eindeutig identifizierbare Zeichenfolge generiert. In der Web-App werden anschließend für alle Getränke eigene Einkaufslisten bereitgestellt, welche über die Zeichenfolgen erreichbar sind.

3 Anleitung zur selbständigen Reproduktion

Für die Reproduktion des Workflows mit identischer Einrichtung werden zwei Programme benötigt: nginx als Reverse Proxy sowie Docker zur Ausführung des n8n-Containers, der Mongo-Datenbank und der Einkaufslisten-Web-App.

Der Unterabschnitt 3.1 kann beim localem Hosten übersprungen werden.

3.1 Einrichtung nginx

Im bereitgestellten ZIP-Archiv befindet sich die Datei `n8n.conf`. In dieser Datei ist in Zeile 4 und in Zeile 13 der korrekte Domainname einzutragen. In Zeile 10 und in Zeile 11 ist der Pfad zum SSL-Zertifikat und zum zugehörigen privaten Schlüssel anzupassen. Nach der Anpassung ist die Datei in das Verzeichnis `/etc/nginx/sites-available` zu kopieren. Anschließend ist im Verzeichnis `/etc/nginx/sites-enabled` ein symbolischer Link auf die Konfigurationsdatei zu erstellen:

```
ln -s /etc/nginx/sites-available/n8n.conf /etc/nginx/sites-enabled/n8n.conf
```

Dasselbe vorgehen muss mit der Datei `shopping.conf` wiederholt werden.

Damit die Änderungen wirksam werden, ist der nginx-Dienst neu zu laden:

```
nginx -s reload
```

3.2 n8n und Mongo

In der entpackten ZIP-Datei befindet sich die Datei `.env.example`, die in `.env` umbenannt werden muss. Die dort gesetzten Werte sind für den Betrieb mit einer eigenen Domain und HTTPS vorgesehen. Sollten die Services also auf einem Server unter einer eigenen Domain gehostet werden, müssen diese Werte entsprechend auf die eigene Domain angepasst werden.

Wird das System hingegen lokal betrieben, ist der Domainname auf `localhost:5678` und das Protokoll auf `http` zu ändern. Die Variable `SHOPPING_URL` ist auf `http://localhost:3069` zu setzen. Unabhängig davon, ob lokal oder mit eigener Domain gehostet wird, sollten die Passwörter in den Variablen `MONGO_ROOT_PASS` und `MONGO_APP_PASS` stets angepasst werden.

Nach diesen Änderungen können n8n, MongoDB und die Einkaufslisten-Web-App mit dem Befehl `docker compose up -d` gestartet werden.

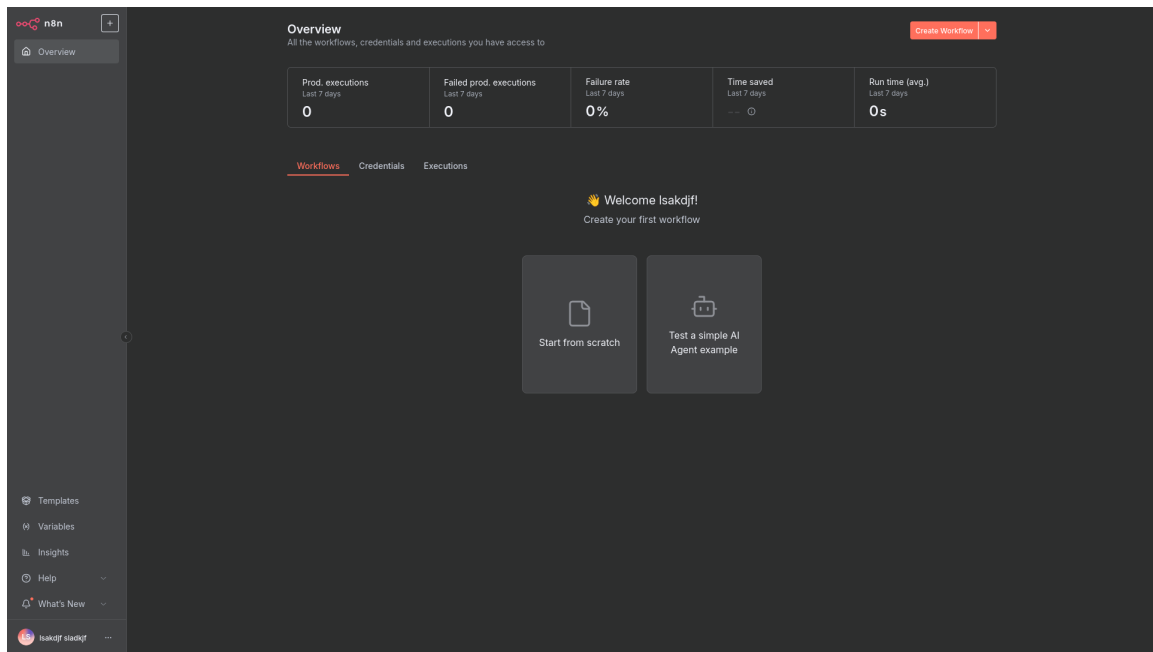


Abbildung 1: n8n Übersichtsseite

Standardmäßig wird dabei die aktuelle stabile Version von n8n verwendet. Die gewünschte Version kann in der Datei `docker-compose.yml` festgelegt werden. Die zuletzt getestete Version ist 1.105.3.

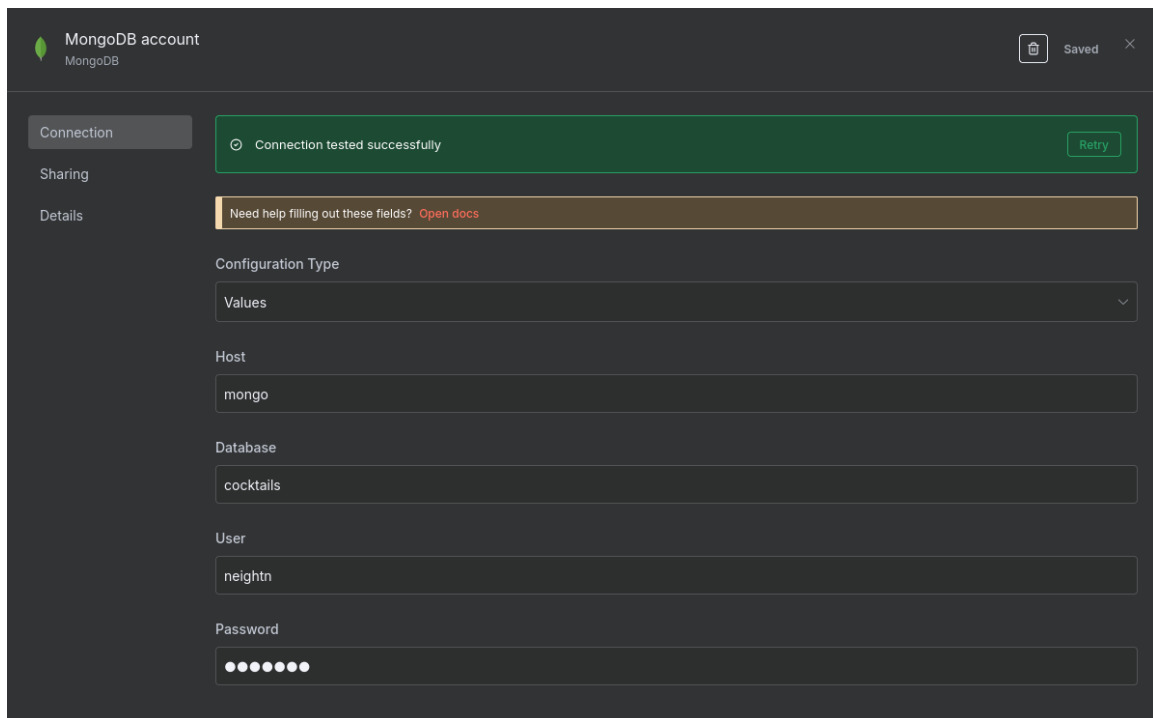
3.3 Workflow in n8n

Beim ersten Aufruf des n8n-Servers wird die Erstellung eines Benutzerkontos abgefragt. Nach erfolgreichem Abschluss der Registrierung erfolgt die Weiterleitung zur Übersichtsseite, die in Abbildung 1 dargestellt ist.

In n8n selber müssen zunächst im „Credentials“ Reiter zwei Zugangsdaten angelegt werden. Für die MongoDB müssen in den Feldern „Database“, „User“ und „Password“ die Daten eingetragen werden, die in der `.env` Datei als `MONGO_APP_DB`, `MONGO_APP_USER` und `MONGO_APP_PASS` festgelegt wurden.

Für die KI muss noch ein API Schlüssel von einem OpenAI Benutzerkonto hinterlegt werden. Dazu wird das zweite Zugangsdatum erstellt.

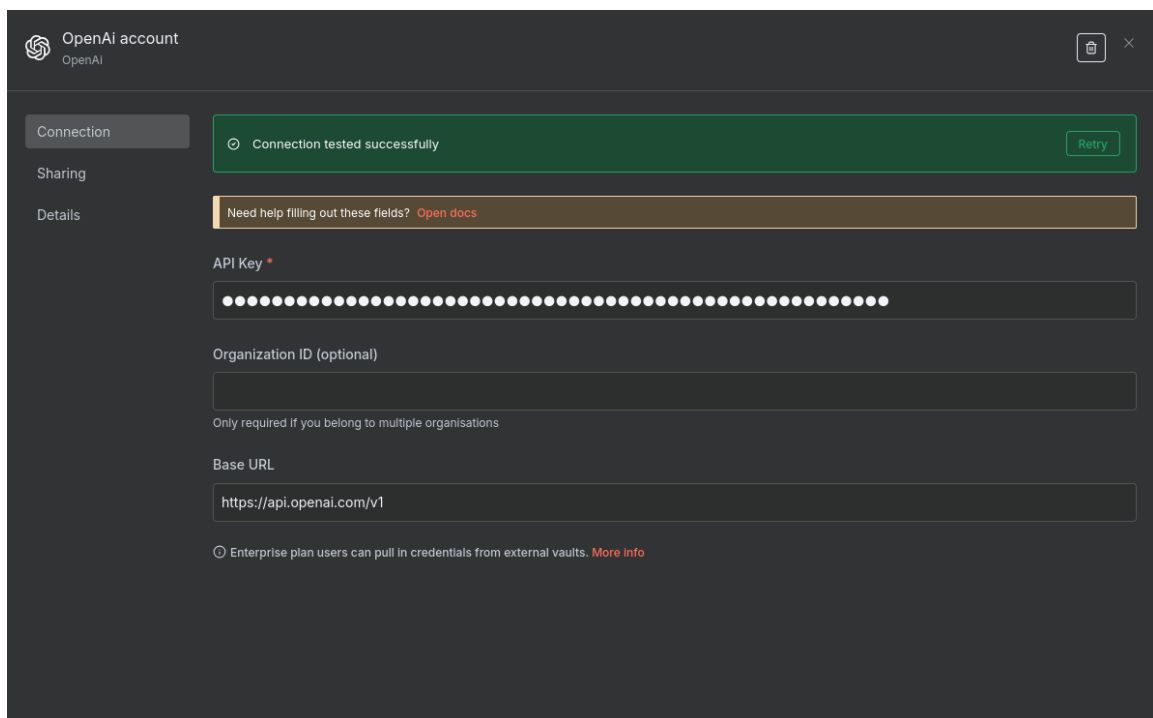
Nachdem die beiden Zugangsdaten angelegt wurden, kann ein neuer Workflow erstellt werden. Im Bearbeitungsmodus angekommen, befindet sich oben rechts im Dreipunktemenü die Auswahl ein vorhandenen Workflow von einer Datei zu importieren. Im ZIP-Archiv ist der Workflow unter dem Namen `Promillo.json` zu finden.



The screenshot shows the 'MongoDB account' connection window in n8n. The window has a dark theme and a sidebar on the left with tabs for 'Connection', 'Sharing', and 'Details'. The 'Connection' tab is active, displaying a green success message: 'Connection tested successfully' with a 'Retry' button. Below this is a yellow warning bar: 'Need help filling out these fields? [Open docs](#)'. The configuration fields are as follows:

- Configuration Type:** A dropdown menu set to 'Values'.
- Host:** A text input field containing 'mongo'.
- Database:** A text input field containing 'cocktails'.
- User:** A text input field containing 'neightr'.
- Password:** A password input field with 10 dots.

Abbildung 2: n8n MongoDB Zugangsdaten



The screenshot shows the 'OpenAI account' connection window in n8n. The window has a dark theme and a sidebar on the left with tabs for 'Connection', 'Sharing', and 'Details'. The 'Connection' tab is active, displaying a green success message: 'Connection tested successfully' with a 'Retry' button. Below this is a yellow warning bar: 'Need help filling out these fields? [Open docs](#)'. The configuration fields are as follows:

- API Key:** A text input field with 20 dots, marked with a red asterisk.
- Organization ID (optional):** A text input field.
- Base URL:** A text input field containing 'https://api.openai.com/v1'.

Below the Base URL field, there is a radio button and the text: 'Enterprise plan users can pull in credentials from external vaults. [More info](#)'.

Abbildung 3: n8n OpenAI Zugangsdaten

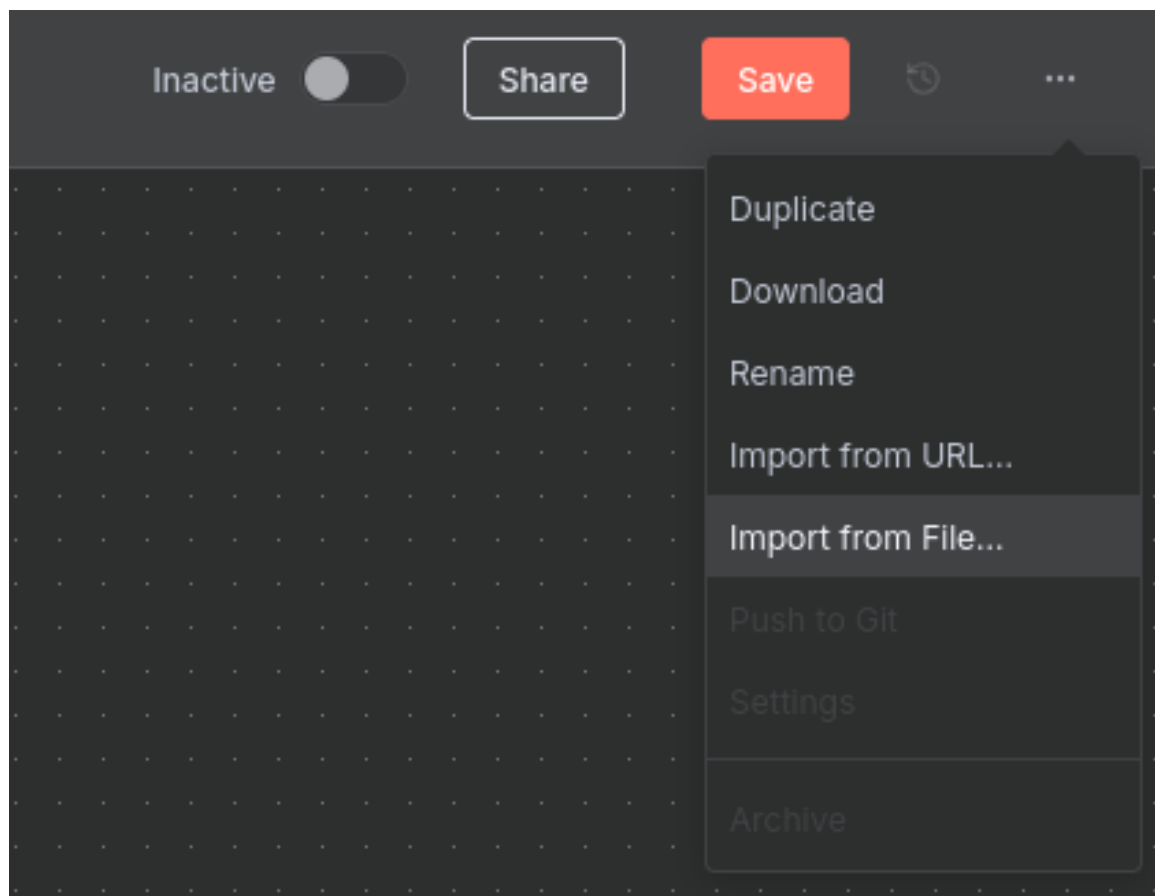


Abbildung 4: n8n Import Workflow

Die Nodes, die Zugangsdaten benötigen werden rot markiert, diese müssen mit Doppelklick einmal geöffnet und in der oberen linken Ecke geschlossen werden. Die Zugangsdaten verwenden zwar den standard Namen, werden von n8n aber mit eindeutigen IDs referenziert. Aufgrund des anlegens der Zugangsdaten werden im Workflow nicht die richtigen IDs referenziert, durch das Öffnen und Schließen werden diese jedoch korrigiert.

Zum Schluss muss der Workflow in der oberen rechten Ecke gespeichert und aktiviert werden. Der Workflow ist damit unter der eigenen Domain mit dem Pfad `/form/alcohol` erreichbar.

```
1  const rawData = $json["content"];
2  const lines = rawData.split('\n');
3  const formFields = [];
4
5  lines.forEach((line, index) => {
6    const cleanedLine = line.replace(/^\d+\.\s*/, '');
7    const braceMatches = [...cleanedLine.matchAll(/\{([~}]+\})/g)];
8    let rawValues = [];
9
10   if (braceMatches.length > 0) {
11     braceMatches.forEach(m => {
12       rawValues.push(...m[1].split(',').map(v => v.trim()));
13     });
14   } else {
15     rawValues = cleanedLine.split(',').map(v => v.trim());
16   }
17
18   if (rawValues.length < 3) return;
19
20   const [name, max_amount_raw, current_amount_raw] = rawValues;
21   const max_amount = parseInt(max_amount_raw, 10);
22   const current_amount = parseInt(current_amount_raw, 10);
23
24   if (isNaN(max_amount) || isNaN(current_amount)) return;
25
26   formFields.push({
27     fieldLabel: `name: ${name}, max amount: ${max_amount}, current
    ↪ amount: ${current_amount}`,
28     placeholder: 'enter if you want it changed',
29     requiredField: false
30   });
31 });
32
33 return formFields.map(field => ({ json: field }));
```

Quellcode 1: JavaScript Code zur Aufbereitung der Daten für die Validierung

```
1  // Loop over input items and add a new field called 'myNewField' to
   ↪ the JSON of each one
2  let out = { preferenzen: $('Edit Fields').first().json.preferenzen,
   ↪ personen: $('Edit Fields').first().json.personen, products: [] }
3  for (const item of $input.all()) {
4    for (const element in item.json) {
5      if (element == "submittedAt" || element == "formMode") {
6        delete item.json[element]
7        continue
8      }
9      const placeholder = element
10     const product = item.json[element] || placeholder
11     if (!product) {
12       continue
13     }
14     out.products.push(product)
15   }
16 }
17 return out;
```

Quellcode 2: JavaScript Code zur Nachbereitung der Daten nach der Validierung

```
1  {
2    "type": "object",
3    "properties": {
4      "Cocktails": {
5        "type": "array",
6        "items": {
7          "type": "object",
8          "required": [
9            "Name",
10           "Zutaten",
11           "Beschreibung",
12           "fehlende Zutaten"
13         ],
14         "properties": {
15           "Name": {
16             "type": "string"
17           },
18           "Beschreibung": {
19             "type": "string"
20           },
21           "Zutaten": {
22             "type": "array",
23             "items": {
24               "type": "object",
25               "required": [
26                 "Zutat",
27                 "Menge"
28               ],
29               "properties": {
30                 "Zutat": {
31                   "type": "string"
32                 },
33                 "Menge": {
34                   "type": "string"
35                 }
36             }
37           }
38         },
39         "fehlende Zutaten": {
40           "type": "array",
41           "items": {
42             "type": "string"
43           }
44         }
45       }
46     }
47   }
48 }
49 }
```

Quellcode 3: KI-Modell Antwort-Format der Rezeptvorschläge

```
1  const BASE = $env["SHOPPING_URL"] ?? "http://localhost:3069";
2
3  const cocktails = $input.all().map(all => all.json);
4
5  const htmlEntries = cocktails.map( (cocktail) => {
6    const zutatenArr = cocktail.Zutaten || [];
7    const fehlendeArr = cocktail['fehlende Zutaten'] || [];
8
9    const zutaten = zutatenArr.map(z => `${z.Menge} ${z.Zutat}`).join('
    ↪ ');
10   const fehlende = fehlendeArr.join(' ');
11   const bucketId = cocktail.uuid;
12
13   let linkBlock = '';
14
15   const link = `${BASE}/?shopping=${bucketId}`;
16   linkBlock = `
17     <p>
18       <a href="${link}" target="_blank" rel="noopener">Zur
19       ↪ Einkaufsliste</a><br>
20       <small>Liste-ID: ${bucketId}</small>
21     </p>
22   `;
23   return `
24     <div class="entry">
25       <h3>${cocktail.Name}</h3>
26       <p>${cocktail.Beschreibung}</p>
27       <h4>Zutaten:</h4>
28       <p>${zutaten}</p>
29       <h4>Fehlende Zutaten:</h4>
30       <p>${fehlende || '-'}</p>
31       ${linkBlock}
32     </div>
33   `;
34 });
35
36 return { html: htmlEntries.join('\n') };
```

Quellcode 4: JavaScript Code zur Vorbereitung der Ergebnisdarstellung

```
1  body {
2    font-family: 'Open Sans', sans-serif;
3    background-color: #f0f0f0;
4    margin: 0;
5    padding: 2rem;
6  }
7  h1 {
8    margin-bottom: 2rem;
9  }
10 .header div {
11   background: white;
12   border-radius: 0.5rem;
13   box-shadow: 0 2px 6px rgba(0,0,0,0.1);
14   padding: 1rem;
15   max-width: 500px;
16   margin: 1rem auto 1rem auto;
17 }
18
19 div h4 {
20   margin-top: 1rem;
21 }
```

Quellcode 5: CSS Code um dem Auge beim Erblicken des Ergebnisses zu schmeicheln