



Intelligenter Ernährungsplan-Generator mit dynamischer Kalorienbedarfsberechnung


Dein täglicher Kalorienbedarf: 1772 kcal
Dein Standort: Mosbach, Germany
Es ist gerade 10°C
Gesamtkalorien vom Ernährungsplan: 1700 kcal



Frühstück
Naanbrot-Stulle „Die passt immer“
670 kcal



Mittagessen
Hauptgericht Fischeintopf Rezept
230 kcal



Abendessen
Rinderfilet und Süßkartoffelmuffins an
Spekulatius-Rotkohl
800 kcal

[Andere Rezepte vorschlagen](#)

Dokumentation

Von

Emily Haaf, Iliana Schotter, Roxanne Collett & Theresia Deubert

29.03.2025

Inhaltsverzeichnis

Projektidee	3
Projektplan	3
Gruppenmitglieder	3
Aufgabenaufteilung	3
Doku Web Services	4
Kalorienbedarf	4
Rezepte	4
Geocoding/Wetter	5
BPMN-Modelle	5
Hauptprozess	6
Unterprozess: Kalorienbedarf berechnen	7
Unterprozess: Wetterdaten abrufen	8
Unterprozess: Rezeptauswahl	9
Frontend	10
Aufbau	10
Webseite	11
Seitenübersicht	11
Design	11
Verwendete Technologien	12
Frontend	12
Abhängigkeiten	12
Schnittstelle zur API	12
API-Endpunkte	12
Datenkommunikation	12
Start des Servers	13
Voraussetzungen	13
Startvorgang	13
Backend	13
Technologien und Abhängigkeiten	13
API-Schlüssel und Umgebungsvariablen	13
Funktionalitäten	14
Endpunkt der API	14
Starten des Servers	15
Swagger UI- API Dokumentation	15

Projektidee

Das Ziel des Projekts ist die Entwicklung eines Rezeptgenerators, der täglich drei Mahlzeiten (Frühstück, Mittagessen und Abendessen) vorschlägt. Die Rezepte werden individuell an den Kalorienbedarf des Nutzenden angepasst. Dabei wird das aktuelle Wetter berücksichtigt, um geeignete Speisen vorzuschlagen. Beispielsweise leichte Gerichte bei warmem Wetter oder wärmende Mahlzeiten bei kälteren Temperaturen.

Projektplan

Gruppenmitglieder

NAME	MATRIKELNUMMER
ROXANNE COLLETT	1893630
ILIANA SCHOTTER	7267944
EMILY HAAF	4213003
THERESIA DEUBERT	5922305

Aufgabenaufteilung

Aufgabe	Verantwortlich
API-Kalorienbedarf	Theresia
API-Rezepte	Iliana
API Geocoding /Wetter	Roxy
Backend API	Theresia, Iliana
Web-Design	Emily
Frontend Aufbau	Roxy, Emily
BPMN-Modelle	Theresia, Iliana
Doku	Theresia, Iliana
Doku Frontend	Emily
Swagger	Roxy, Theresia
Docker	Roxy
Caching	Roxy

Doku Web Services

Kalorienbedarf

Die Berechnung des täglichen Kalorienbedarfs erfolgt durch die Health Calculator API von RapidAPI. Diese API nutzt verschiedene Formeln wie die Mifflin-St Jeor-Gleichung, um basierend auf Alter, Gewicht, Größe, Geschlecht und Aktivitätslevel den individuellen Energiebedarf eines Nutzers zu bestimmen. Auf Basis dieser Berechnungen werden geeignete Rezepte ausgewählt, die den benötigten Kalorienbedarf decken. Es sind 100 Anfrage pro Monat kostenlos.

Mögliche Parameter:

- *age (erforderlich)*: Das Alter der Person in Jahren – *Number* (z. B. 30)
- *weight (erforderlich)*: Das Gewicht der Person in Kilogramm – *Number* (z. B. 60)
- *height (erforderlich)*: Die Körpergröße der Person in Zentimetern – *Number* (z. B. 170)
- *gender (erforderlich)*: Das Geschlecht der Person – *String* ("male" oder "female")
- *activity_level (erforderlich)*: Das Aktivitätslevel der Person – *String* ("sedentary", "lightly_active", "moderately_active", "very_active", "extra_active")
- *goal (erforderlich)*: Das Ziel der Person – *String* ("weight_loss", "maintenance", "weight_gain")
- *equation (optional)*: Die Formel zur Berechnung des Kalorienbedarfs – *String* ("harris" oder "mifflin", Standard: "harris")

Rezepte

Für die Rezeptvorschläge wird die Gustar.io Deutsche Rezepte API von RapidAPI verwendet. Es wurde der API-Endpunkt für die Rezeptsuche verwendet. Dieser Endpunkt ermöglicht es, Rezepte basierend auf spezifischen Suchbegriffen und Filtern abzurufen. Bei dieser API sind 500 Anfragen pro Monat kostenlos.

Mögliche Parameter:

- *text (erforderlich)*: Suchbegriff für das Rezept (z. B. "Käse") – *String*
- *timeLimit (optional)*: Maximale Zubereitungszeit in Sekunden (z. B. 3600 für 60 Minuten) – *Number*
- *sort (optional)*: Sortierung nach Nutzerbewertungen (true/false) – *Boolean*
- *ingLimit (optional)*: Maximale Anzahl der Zutaten – *Number*
- *difficulty (optional)*: Schwierigkeitsgrad des Rezepts ('easy', 'medium', 'hard') – *String*
- *diet (optional)*: Ernährungsweise ('vegan', 'vegetarian') – *String*

- *page (optional)*: Seitenzahl der Suchergebnisse (erste Seite ist 0) – *Number*

Geocoding/Wetter

Die Standortbestimmung erfolgt mithilfe der Geocoding API von API Ninja. Diese API wird verwendet, um eine Postleitzahl oder einen Ortsnamen in geografische Koordinaten (Längen- und Breitengrad) umzuwandeln. Die Nutzung der Geocoding API ist notwendig, da die Weather API von API Ninja bei einer direkten Abfrage mit einem Standort kostenpflichtig ist, während die Abfrage über Koordinaten kostenlos erfolgt.

Mögliche Parameter für die Geocoding API:

- *city (erforderlich)*: Name der Stadt – *String*
- *state (optional)*: Bundesstaat (nur für Städte in den USA) – *String*
- *country (optional)*: Name des Landes, 2-stelliger oder 3-stelliger ISO-Ländercode – *String*
- *zipcode (optional)*: 5-stellige Postleitzahl (nur für Städte in den USA) – *Number*

Die Weather API liefert aktuelle Wetterdaten, insbesondere die gefühlte Temperatur am Aufenthaltsort des Nutzers, um die Rezeptausswahl entsprechend anzupassen. Bei beiden APIs sind 10 000 Anfragen pro Monat kostenlos möglich.

Mögliche Parameter für die Weather API:

- *lat (erforderlich)*: Breitengrad des gewünschten Standorts – *Number*
- *lon (erforderlich)*: Längengrad des gewünschten Standorts – *Number*
- *zip (erforderlich, Premium)*: Postleitzahl (nur für die USA) – *Number*
- *city (erforderlich, Premium)*: Name der Stadt – *String*
- *state (optional, Premium)*: Bundesstaat (nur für US-Städte) – *String*
- *country (optional, Premium)*: Name des Landes – *String*

BPMN-Modelle

Der intelligente Ernährungsplan-Generator unterstützt Nutzer dabei, einen personalisierten Ernährungsplan basierend auf ihrem individuellen Kalorienbedarf und den aktuellen Wetterbedingungen zu erstellen. Dazu werden mehrere Webservices benutzt, um relevante Daten zu sammeln und auszuwerten. So wird eine optimale Ernährungsempfehlung für den jeweiligen Tag generiert.

In diesem Prozess spielen verschiedene Rollen eine wichtige Rolle. Der Nutzer gibt relevante persönliche Daten ein und erhält am Ende einen individuell zugeschnittenen Ernährungsplan. Das System verarbeitet die Nutzereingaben, ruft externe Webservices auf und erstellt auf dieser Basis den Ernährungsplan. Dazu werden verschiedene externe Webservices eingebunden: Eine Kalorienberechnungs-API berechnet den individuellen Kalorienbedarf anhand von Parametern wie Alter, Gewicht, Größe, Geschlecht, Aktivitätsniveau und Ernährungsziel. Eine Wetter-API liefert Wetterdaten, insbesondere die gefühlte Temperatur am Standort des Nutzers, und eine Rezept-API stellt passende Rezepte bereit, die sowohl dem Kalorienbedarf als auch der aktuellen Wetterlage entsprechen.

Hauptprozess

Der Hauptprozess beginnt mit dem Start-Ereignis, bei dem der Nutzer den Prozess initiiert. Zunächst füllt er einen Fragebogen aus, in dem er persönliche Informationen eingibt. Anschließend überprüft das System, ob alle notwendigen Eingaben vorhanden sind. Falls Daten fehlen, wird der Nutzer zur Vervollständigung aufgefordert. Sobald alle Angaben vorliegen, erfolgt die Berechnung des Kalorienbedarfs durch den entsprechenden Webservice. Danach ruft das System die aktuellen Wetterdaten des Standorts über die Wetter-API ab und ermittelt die gefühlte Temperatur. Anschließend erfolgt die Auswahl passender Rezepte, die sowohl dem individuellen Kalorienbedarf als auch den Wetterbedingungen entsprechen. Der Unterprozess "Rezeptauswahl" stellt sicher, dass die Mahlzeiten optimal zusammengestellt werden.

Im nächsten Schritt generiert das System den Ernährungsplan, indem es geeignete Mahlzeiten für Frühstück, Mittagessen und Abendessen zusammenstellt. Der Nutzer erhält dann seinen persönlichen Ernährungsplan, woraufhin der Prozess abgeschlossen wird.

Zur Umsetzung dieses Prozesses werden mehrere externe Webservices genutzt. Der Calorie Calculator-Service berechnet den täglichen Energiebedarf des Nutzers, während der Wetter-Service Wetterdaten bereitstellt, um die Mahlzeiten an die äußeren Bedingungen anzupassen. Der Geocoding-Service ermöglicht die Bestimmung des Nutzerstandorts für eine genauere Wetterprognose, und der Rezept-Service stellt eine Auswahl an Mahlzeiten bereit, die basierend auf Kalorienbedarf und Wetterbedingungen vorgeschlagen werden.

Das BPMN-Modell der Ernährungsplan-App bildet somit einen strukturierten Prozess ab, der Nutzerangaben verarbeitet, verschiedene externe Webservices integriert und am Ende einen individuellen Ernährungsplan generiert. Durch diese systematische Vorgehensweise wird eine effiziente und personalisierte Ernährungsberatung ermöglicht.

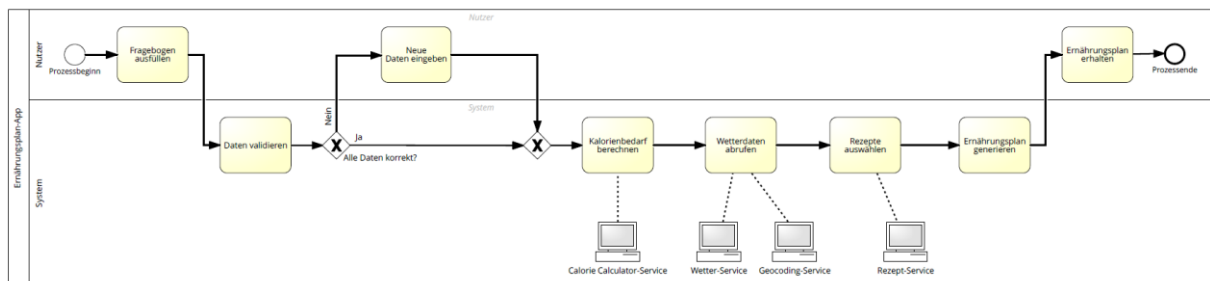


Abbildung 1: BPMN-Modell Hauptprozess: Ernährungsplangenerator

Zusammenfassung der Prozesselemente:

- Start-Ereignis: Der Nutzer beginnt den Prozess.
- Fragebogen ausfüllen: Der Nutzer gibt persönliche Informationen ein.
- Daten validieren: Das System prüft die Eingaben und fordert ggf. Korrekturen an.
à mit XOR-Verknüpfung
- Unterprozess: Kalorienbedarf berechnen (siehe unten)
- Unterprozess: Wetterdaten abrufen (siehe unten)
- Unterprozess: Rezeptauswahl (siehe unten)
- Ernährungsplan generieren: Das System stellt die Mahlzeiten zusammen.
- Ernährungsplan anzeigen: Der Nutzer erhält die personalisierte Empfehlung.
- Ende-Ereignis: Der Prozess wird abgeschlossen.

Unterprozess: Kalorienbedarf berechnen

Der Unterprozess zur Berechnung des Kalorienbedarfs startet mit der Übernahme der Daten aus dem Fragebogen des Nutzers. Diese beinhalten Angaben zu Alter, Gewicht, Größe, Geschlecht, Aktivitätsniveau und gewünschtem Ernährungsziel. Anschließend sendet das System diese Daten an die Kalorienberechnungs-API, die eine detaillierte Berechnung des täglichen Energiebedarfs durchführt. Nachdem die API die Berechnung abgeschlossen hat, empfängt das System die Ergebnisse und speichert sie zur weiteren Verarbeitung. Der Prozess endet mit der Berechnung des Kalorienbedarfs, der für die Erstellung des Ernährungsplans genutzt wird.

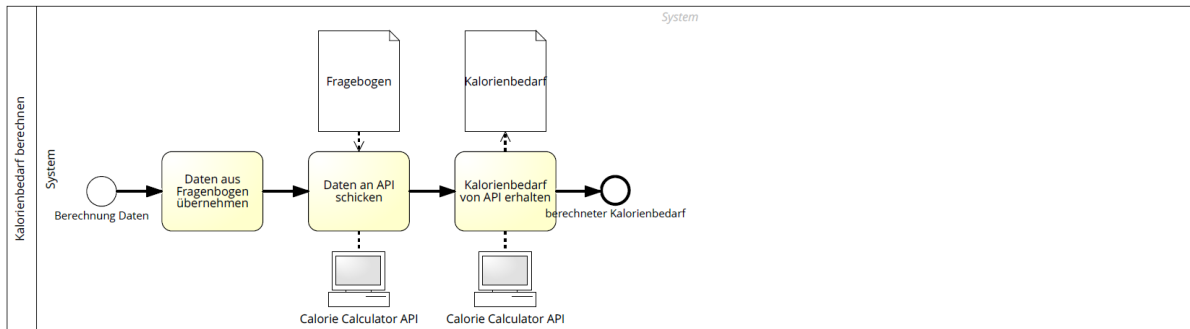


Abbildung 2: BPMN-Modell Unterprozess: Kalorienbedarf berechnen

Zusammenfassung der Prozesselemente:

- Start-Ereignis: Berechnung starten.
- Daten aus Fragebogen übernehmen: Das System liest relevante Nutzerangaben aus.
- Webservice „Kalorienberechnung“ aufrufen: Das System sendet die Daten an die Kalorienberechnungs-API.
- Ergebnisse speichern: Die API sendet den Kalorienbedarf zurück und das System speichert ihn.
- Ende-Ereignis: Der Kalorienbedarf ist berechnet.

Unterprozess: Wetterdaten abrufen

Der Unterprozess zur Wetterabfrage beginnt mit der Übernahme des Standorts aus dem Fragebogen. Anschließend sendet das System diese Standortdaten an die Geocoding-API, um die genauen Koordinaten zu erhalten. Sobald die Standortkoordinaten empfangen wurden, ruft das System die Wetter-API auf, um die aktuellen Wetterdaten zu erhalten. Die gefühlte Temperatur wird dann aus den Wetterdaten extrahiert und im System gespeichert. Der Prozess endet, sobald die Wetterdaten erfolgreich abgerufen wurden.

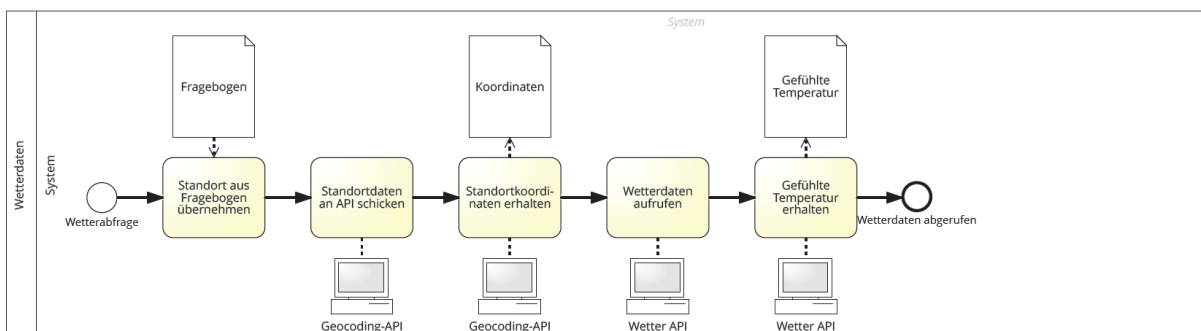


Abbildung 3: BPMN-Modell Unterprozess: Wetterabfrage

Zusammenfassung der Prozesselemente:

- Start-Ereignis: Wetterabfrage beginnen.
- Standort aus Fragebogen übernehmen: Das System übernimmt die Standortdaten des Nutzers.
- Standort an Geocoding-API schicken: Die API liefert Koordinaten zurück.
- Standortkoordinaten erhalten: Das System speichert die Koordinaten.
- Wetterdaten abrufen: Das System sendet die Koordinaten an die Wetter-API.
- Gefühlte Temperatur speichern: Das System speichert die ermittelten Wetterdaten.
- Ende-Ereignis: Die Wetterdaten sind erfolgreich erfasst.

Unterprozess: Rezeptauswahl

Der Unterprozess zur Rezeptauswahl beginnt mit der Anforderung geeigneter Rezepte über die Rezept-API. Gleichzeitig wird die zuvor gespeicherte gefühlte Temperatur abgefragt. Basierend auf der Temperatur erfolgt eine Kategorisierung der Speisen:

- Falls die Temperatur über 20°C liegt, werden leichte Speisen bevorzugt.
- Falls die Temperatur unter 10°C liegt, werden wärmende Speisen ausgewählt.
- Falls die Temperatur zwischen 10°C und 20°C liegt, erfolgt keine Wetterbasierende Kategorie Filterung.

Nach der ersten Kategorisierung filtert das System die Rezepte entsprechend der Mahlzeiten (Frühstück, Mittagessen, Abendessen). Anschließend erfolgt eine weitere Filterung, um sicherzustellen, dass die gewählten Rezepte zum individuellen Kalorienbedarf des Nutzers passen. Der Unterprozess endet mit der Bereitstellung passender Rezeptvorschläge.

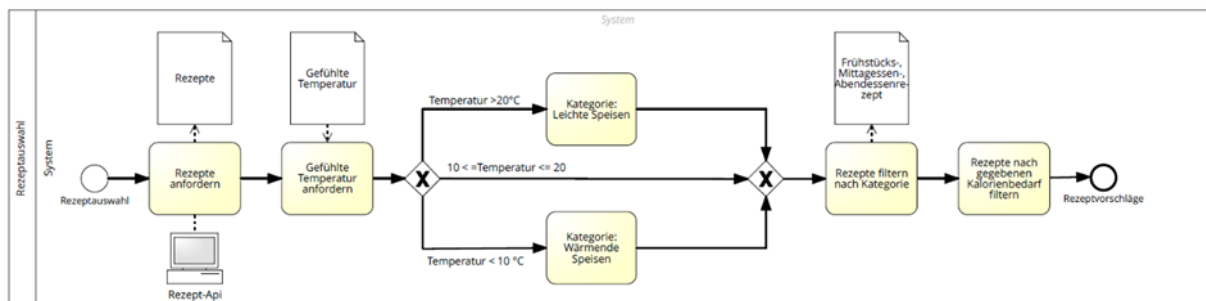


Abbildung 4: BPMN-Modell Unterprozess: Rezeptauswahl

Zusammenfassung der Prozesselemente:

- Start-Ereignis: Rezeptauswahl starten.

- Rezepte anfordern: Das System ruft die Rezept-API auf.
- Gefühlte Temperatur anfordern: Die gespeicherte Temperatur wird verwendet.
- Speisen kategorisieren mit XOR-Verknüpfung:
- Temperatur > 20°C → leichte Speisen auswählen.
- Temperatur < 10°C → wärmende Speisen auswählen.
- Temperatur zwischen 10°C und 20°C → Filterung nach Mahlzeitenkategorie.
- Rezepte nach Mahlzeiten filtern: Frühstück, Mittagessen und Abendessen werden zugeordnet.
- Rezepte nach Kalorienbedarf filtern: Die Mahlzeiten werden an die individuellen Bedürfnisse angepasst.
- Ende-Ereignis: Geeignete Rezepte sind ausgewählt.

Frontend

Das Frontend unserer Anwendung wurde mit JavaScript und TypeScript unter Verwendung des Frameworks Vue.js entwickelt. Vue.js bildet die Basis der Client-Seite und steuert die Darstellung sowie die Benutzerinteraktionen. Durch die komponentenbasierte Architektur wird der Code in wiederverwendbare Bausteine unterteilt.

Aufbau

Die Codebasis des Frontend ist klar strukturiert. Die wichtigsten Verzeichnisse und ihre Funktionen sind:

- components/: In diesem Ordner befinden sich Vue-Komponenten, die in verschiedenen Teilen der Anwendung genutzt werden.
- views/: Hier sind die einzelnen Elemente der Benutzerdatenabfrage gespeichert für die Berechnung der Kalorien sowie Abfrage des Wetters.
- stores/: In diesem Abschnitt werden die zentralen Daten des Frontendes verwaltet
- router/: Die Navigation innerhalb der Anwendung wird über den Vue Router gesteuert.
- App.vue: Ist die Hauptkomponente der Anwendung. Sie definiert die grundlegende Struktur und das Layout.

Webseite

Seitenübersicht

Die Webseite führt den Nutzer schrittweise durch die Erstellung eines individuellen Tages Ernährungsplan:

Zu Beginn startet der Nutzer auf der Startseite und wird aufgefordert, sein Geschlecht auszuwählen. Anschließend gelangt er zur Eingabe seiner persönlichen Daten, darunter Alter, Größe und Gewicht, die für die Berechnung des Kalorienbedarfs essenziell sind.

Im nächsten Schritt gibt der Nutzer sein Aktivitätslevel an, um eine genauere Berechnung des täglichen Kalorienbedarfs zu ermöglichen. Während des gesamten Prozesses hat er jederzeit die Möglichkeit, zu den vorherigen Schritten zurückzukehren und Eingaben zu korrigieren.

Nach Abschluss der Umfrage wird die Rezeptübersicht angezeigt, die eine Auswahl an 3 personalisierten Gerichten für Frühstück, Mittag- und Abendessen enthält. Der Nutzer kann die einzelnen Rezepte anklicken, um detaillierte Informationen zu den Zutaten, Nährwerten und Portionsgrößen zu erhalten. Zudem hat er die Möglichkeit, über einen Link zu einer externen Seite zu gelangen, auf der die genaue Zubereitung beschrieben wird. Sollte ihm die aktuelle Auswahl nicht zusagen, kann er über einen Button andere Rezepte generieren lassen, um eine neue Auswahl an Gerichten zu erhalten.

Die Navigation ermöglicht es, jederzeit zwischen den Rezepten zu wechseln und sich weitere Details anzusehen. So kann der Nutzer flexibel durch die verschiedenen Gerichte stöbern und seinen individuellen Ernährungsplan optimal nutzen.

Design

Das Design der Webseite wurde mit besonderem Fokus auf Nutzerfreundlichkeit und eine ansprechende Optik entwickelt. Da der Großteil der Nutzer die Anwendung auf mobilen Endgeräten verwendet, wurde besonderer Wert auf Responsivität gelegt. Die Oberfläche passt sich dynamisch an verschiedene Bildschirmgrößen an. Die Farbgebung orientiert sich an den Farben des Frühlings, wobei Grün und Rosa als primäre Töne eingesetzt wurden. Zusätzlich wurden 2 Features integriert, um die Benutzung weiter zu verbessern. Das ist zum einen der Darkmode, und eine Ladeanzeige, die dem Nutzer Feedback gibt, wenn Daten verarbeitet oder geladen werden.

Verwendete Technologien

Frontend

Für die Entwicklung des Frontend wurden folgende Technologien verwendet:

- JavaScript und TypeScript als Programmiersprachen zur Umsetzung der Logik und Komponentenstruktur.
- Vue.js für die komponentenbasierte UI-Entwicklung.
- Vite als schnelles Build-Tool für eine optimierte Entwicklungsumgebung.
- Tailwind CSS für modernes, flexibles Styling.

Abhängigkeiten

Ein zentraler Bestandteil der Anwendung ist der Vue Router, welcher für die Verwaltung der Routen und die Navigation innerhalb der Anwendung verantwortlich ist.

Schnittstelle zur API

Die Kommunikation zwischen Frontend und Backend erfolgt über eine API, die über den Localhost verschiedene Daten bereitstellt.

API-Endpunkte

Das Frontend greift auf verschiedene API-Endpunkte zu, um Benutzerdaten zu speichern und abzurufen:

- `/api/user-data` ermöglicht es, Benutzerdaten zu speichern und zu laden.
- `/api/clear-cache` kann der Cache der Anwendung geleert werden, was insbesondere für Entwicklungszwecke nützlich ist.
- `/get_meal_plan` wird verwendet, die bereitgestellten Rezepte zu zugreifen

Datenkommunikation

Die Kommunikation mit dem Backend erfolgt über die Fetch-API. Alle Anfragen sind in den Store-Methoden gekapselt, sodass die API-Aufrufe zentral verwaltet werden. Der Datenaustausch erfolgt im JSON-Format. Fehler werden abgefangen und dem Nutzer in geeigneter Form angezeigt.

Start des Servers

Voraussetzungen

Bevor die Anwendung gestartet werden kann, müssen einige Voraussetzungen erfüllt sein. Es ist erforderlich, dass Node.js auf dem System installiert ist. Zudem muss sichergestellt werden, dass der Backend-Server unter `http://localhost:8000` läuft.

Startvorgang

Um den Server zu starten, müssen folgende Befehle ausgeführt werden:

1. `npm install`: Installiert alle benötigten Abhängigkeiten.
2. `npm run dev`: Startet den Entwicklungsserver.

Backend

Diese Backend-Anwendung ist in Node.js mit Express implementiert und stellt eine Schnittstelle zur Ernährungs- und Wetterdatenverarbeitung bereit. Sie ermöglicht es, basierend auf Benutzerangaben und externen API-Daten, einen individuellen Mahlzeitenplan zu generieren.

Technologien und Abhängigkeiten

Die Anwendung nutzt die folgenden Bibliotheken:

- Express: Framework zur Erstellung von Webservern.
- Axios: HTTP-Client für API-Anfragen.
- Cors: Middleware zur Handhabung von Cross-Origin-Requests.
- dotenv: Ermöglicht das Laden von Umgebungsvariablen aus einer `.env`-Datei
- à API-Keys.

API-Schlüssel und Umgebungsvariablen

Die Anwendung verwendet API-Schlüssel für den Zugriff auf verschiedene externe APIs. Diese Schlüssel werden aus der `.env`-Datei ausgelesen:

- `GOOGLE_GEOCODE_API_KEY`: API-Schlüssel für Google Geocoding.
- `NINJA_API_KEY`: API-Schlüssel für Wetterdaten von API Ninjas.
- `CALORIERAPIDAPI_KEY`: API-Schlüssel für Kalorienberechnungen.
- `RAPIDAPI_KEY`: API-Schlüssel für Rezeptanfragen.

Funktionalitäten

1. Ermittlung der Geokoordinaten

Die Funktion `getCoordinates(location)` sendet eine Anfrage an die Google Geocoding API, um die Längen- und Breitengrade eines angegebenen Standorts zu bestimmen. Die Rückgabe erfolgt als Objekt mit `latitude` und `longitude`.

2. Abruf der gefühlten Temperatur

Die Funktion `getFeelsLikeTemperature(latitude, longitude)` verwendet die API von API Ninjas, um die aktuelle gefühlte Temperatur an einem bestimmten Ort zu bestimmen. Die Temperatur wird zurückgegeben oder null, falls die Anfrage fehlschlägt.

3. Berechnung des Kalorienbedarfs

Die Funktion `calculateCalories(age, weight, height, gender, activity_level, goal)` ruft die RapidAPI-Health-Calculator-API auf, um den täglichen Kalorienbedarf basierend auf persönlichen Angaben und dem Mifflin-St. Jeor-Formelansatz zu berechnen.

4. Abruf von Rezepten nach Kategorie

Die Funktion `getRecipesByCategory(category)` greift auf die RapidAPI-Gustar-API zu und sucht nach Rezepten einer bestimmten Kategorie. Es werden nur Rezepte zurückgegeben, die einen definierten Kalorienwert enthalten.

5. Zusammenstellung eines Ernährungsplans

Die Funktion `selectThreeMeals(calories, temperature)` wählt basierend auf der gefühlten Temperatur drei passende Mahlzeiten aus:

- Unter 10°C: Wärmende Speisen wie Suppen und deftige Gerichte.
- Über 20°C: Leichte Gerichte wie Salate.
- Zwischen 10°C und 20°C: Standardkategorien (Frühstück, Hauptgericht, Abendessen).

Dabei wird zusätzlich überprüft, ob die Kalorien der Mahlzeiten im Bereich von ± 100 Kalorien des berechneten Kalorienbedarfs liegen.

Endpunkt der API

Die Route `/get_meal_plan` erwartet folgende Parameter über eine GET-Anfrage:

- `age` (Alter)
- `weight` (Gewicht)
- `height` (Größe)

- gender (Geschlecht)
- activity_level (Aktivitätslevel)
- goal (Ernährungsziel)
- location (Standort)

Der Server ruft nacheinander die oben beschriebenen Funktionen auf, um die benötigten Daten zu ermitteln. Anschließend wird der individuelle Ernährungsplan als JSON-Response zurückgegeben, bestehend aus:

- kalorienbedarf: Berechneter täglicher Kalorienbedarf.
- gefühlteTemperatur: Aktuelle gefühlte Temperatur am Standort.
- mahlzeiten: Enthält Frühstück, Mittag- und Abendessen mit den jeweiligen Rezeptdaten.
- gesamtKalorien: Gesamtkalorien der vorgeschlagenen Mahlzeiten.

Falls ein Fehler auftritt, wird eine entsprechende Fehlermeldung zurückgegeben.

Starten des Servers

Die Anwendung läuft auf Port 8000 und kann durch den Befehl `node meal_server.js` oder mit `npm start` gestartet werden. Sobald sie läuft, ist die API unter `http://localhost:8000` erreichbar.

Swagger UI- API Dokumentation

Die Swagger UI kann unter `http://localhost:8000/api-docs` aufgerufen werden. Folgende Endpunkte können getestet werden:

- Benutzerverwaltung
 - **POST** /api/user-data
 - **GET** /api/user-data
- Systemverwaltung
 - **POST** /api/clear-cache
- Ernährungsplan
 - **GET** /get_meal_plan

Abbildungsverzeichnis

Abbildung 1: BPMN-Modell Hauptprozess: Ernährungsplangenerator	7
Abbildung 2: BPMN-Modell Unterprozess: Kalorienbedarf berechnen.....	8
Abbildung 3: BPMN-Modell Unterprozess: Wetterabfrage	8
Abbildung 4: BPMN-Modell Unterprozess: Rezeptauswahl.....	9