TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Fakultät Informatik

Konzeption und Realisierung einer Web-Anwendung zur Erstellung von Ernährungsplänen auf Basis von Benutzerparametern

Projektbericht im Bachelorstudiengang Informatik

vorgelegt von

Armin Bruckmann (3273214),
Felix Berger (3272014),
Oliver Dassinger (3271505)

Betreuer: Prof. Dr. Matthias Teßmann

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

1 Einleitung

1.1 Motivation

Ein geschätzter Anteil von 51,4 % der Bevölkerung in Deutschland hat zumindest eine Lebensmittelunverträglichkeit (Statista 2021). Betroffenen Menschen fällt es deshalb oft besonders schwer, auf Basis der im Haushalt vorhandenen Lebensmittel einen auf ihre persönlichen Vorlieben und Bedürfnisse zugeschnittenen Ernährungsplan zu erstellen, welcher sowohl die entsprechenden Unverträglichkeiten als auch die im Haushalt vorhandenen Lebensmittel in Einklang bringt, um einen optimalen Ernährungsplan zu erstellen. Doch auch anderen Personengruppen, wie z. B. Menschen, die schlicht keine Zeit haben, sich Gedanken über ihre Rezepte zu machen, welche auf ihre Vorlieben abgestimmt sind, würden von der Erstellung eines personalisierten Ernährungsplans profitieren.

1.2 Zielsetzung

Das Ziel dieses Projektes ist es, eine Webanwendung zu entwickeln, welche Nutzern Rezeptideen vorschlägt, die auf ihre Bedürfnisse abgestimmt sind. Die Applikation ist auf eine Auflösung von 1920p x 1080p ausgelegt.

Hierfür werden der Ernährungsstil (z. B. vegetarisch oder vegan), die Unverträglichkeiten (z. B. Glutenunverträglichkeit oder Unverträglichkeit gegen Erdnüsse), die im Haushalt vorhandenen Lebensmittel und das tägliche Kalorienziel berücksichtigt.

Auf Basis dieser Eingaben wird dem Anwender ein auf seine Bedürfnisse zugeschnittener Ernährungsplan erstellt.

Dieser stellt dem Benutzer für die üblichen drei täglichen Mahlzeiten (Frühstück, Mittag- und Abendessen) jeweils ein personalisiertes Rezept zur Verfügung. Sollte dem Nutzer ein solches missfallen, so kann sich dieser mittels einer „Reroll-Funktion“ ein neues vorschlagen lassen.

Des Weiteren werden Funktionen zur Verfügung gestellt, die dem Anwender einen schnellen Überblick über die enthaltenen Nährwerte eines Rezepts bieten.

Zudem ist ein Ausdrucken der Einkaufsliste, welche fehlende und vorhandene Zutaten enthält und die Generierung einer Kochanleitung möglich.

1.3 Struktur und Aufbau

Die vorliegende Arbeit lässt sich in vier Kapitel unterteilen, wobei das erste die Einleitung darstellt. In dieser wird ein kurzer Überblick über das zu bearbeitende Thema gegeben. Außerdem wird die Zielsetzung der Arbeit festgelegt.
Das zweite Kapitel behandelt die Konzeption des Soll-Zustands. In diesem wird die Planung, die Arbeitsweise und die Konzeption des Projektes vorgestellt und erläutert.
Das darauffolgende Kapitel widmet sich der Umsetzung der zu implementierenden Web-Anwendung.
Abschließend werden aufgetretene Probleme und deren Lösungen erläutert, die erreichten Ergebnisse vorgestellt und zugleich ein Ausblick über mögliche zukünftige Erweiterungen dieser Arbeit geliefert.

Kapitel 2

2 Konzeption des Soll-Zustands

Gegenstand dieses Kapitels ist die Projektplanung. Um die für jedes Projekt geltenden Ziele Kosten, Termine und Qualität (Litke 2018, S. 28) in Einklang zu bringen, werden zunächst die Anforderungen an das zu entwickelnde Programm ermittelt. Darauf aufbauend werden Konzepte für die Implementierung der Anwendung entworfen und die in diesem Projekt verwendeten Technologien vorgestellt.

2.1 Anforderungsanalyse

In einem ersten gemeinsamen Brainstorming wurden folgende Anforderungen ermittelt, die die Anwendung leisten soll:

- Ernährungsplan generieren
- Login Informationen überprüfen
- Präferenzen und Lebensmitteleingabe des Nutzers speichern

Außerdem sollte der Nutzer in der Lage sein, sich einloggen zu können, seine Präferenzen und im Haushalt befindlichen Lebensmittel speichern zu können und den Ernährungsplan betrachten und neu generieren zu können.

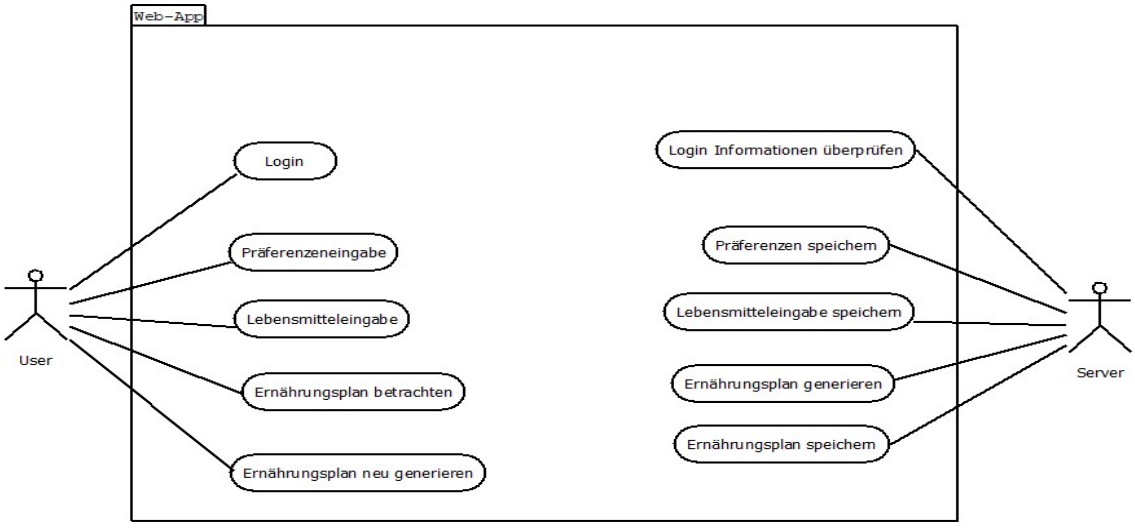

Abbildung 1: Ermittelte Anforderungen (eigene Abbildung)

2.2 Vorläufiger Prototyp

Auf Basis der ermittelten Anforderungen wurde ein Prototyp erstellt, welcher einen ersten Überblick über die zu erstellenden Seiten und deren Interaktion miteinander darstellt.

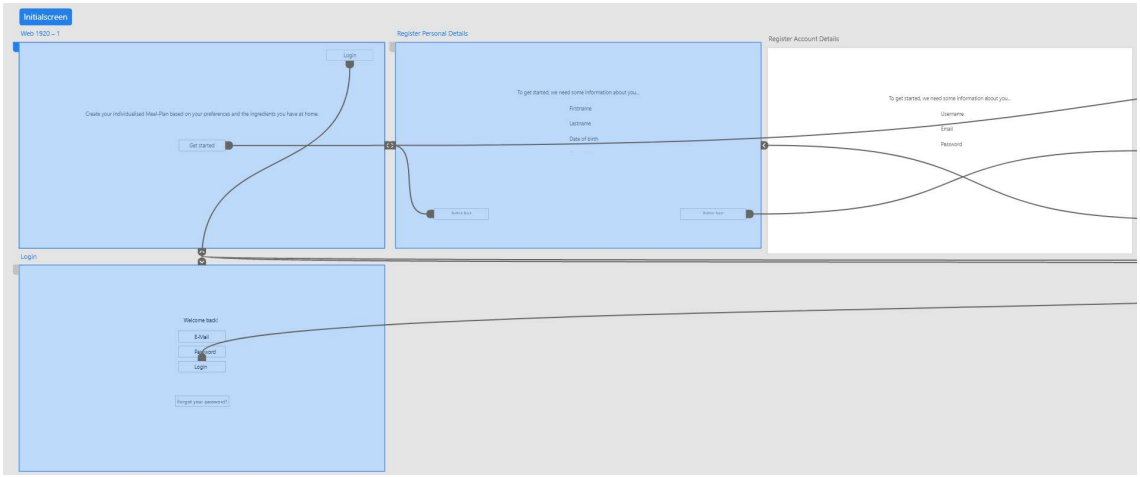

Abbildung 2: Ausgewählte Seiten des Prototypen (eigene Abbildung)

Wie in Abbildung 2 ersichtlich ist, wurden bereits die wichtigsten Seiten der späteren Anwendung einschließlich der zu verwendenden Buttons modelliert. Die grau dargestellten Linien stellen hierbei Verknüpfungen dar, welche Nutzerinteraktionen abbilden sollen. So konnte schon recht früh ein für den Anwender einfach und intuitiv zu bedienendes Interaktionsdesign entworfen werden.

2.3 Konzeptuelles Schema der Datenbank

Um die anfallenden Nutzerdaten abzuspeichern, wurde sich für ein relationales Datenbankmodell entschieden. Diese Entscheidung wurde mit Hinblick auf die Ressource Zeit getroffen, da dieser Datenbanktyp allen Projektbeteiligten vertraut war und der Einarbeitungsaufwand dementsprechend gering war.

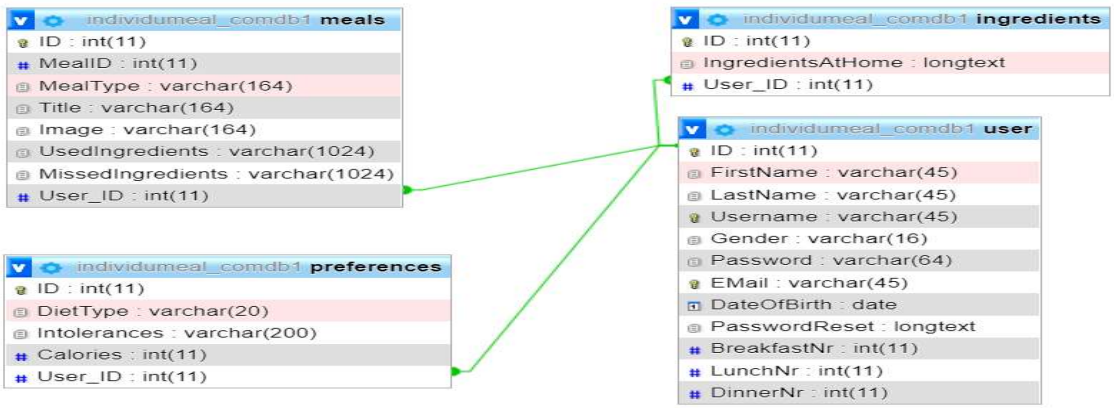

Abbildung 3: Konzeptuelles Schema der Datenbank (eigene Abbildung)

Das Schema besteht aus vier Tabellen, die wie aus Abbildung 3 ersichtlich, jeweils mittels eines Fremdschlüssels mit dem Primärschlüssel eines Nutzers verknüpft sind. Zu einem User muss es jeweils eine Tabelle Preferences, Ingredients und Meals geben, die einzelnen Felder müssen jedoch nicht immer befüllt sein. Hat ein Nutzer beispielsweise keine Intoleranzen, so darf dieses Feld leer sein.

2.4 Verwendete Technologien

2.4.1 Programmiersprachen und Frameworks

Auch in Bezug auf die verwendeten Technologien hat der Faktor Zeit eine maßgebliche Rolle gespielt.

Um die von den Teammitgliedern als eher kontraintuitiv empfunden Stylesheet-Sprache CSS weitestgehenden vermeiden zu können, wurde für das Frontenddesign weitestgehend das Bootstrap-Framework verwendet.

Für die Backendentwicklung wurde auf die den Entwicklern vertraute Programmiersprache PHP gesetzt, ein Framework hierfür wurde aufgrund des als eher gering eingeschätzten Einsatzes der Sprache nicht in Betracht gezogen, um unnötige Einarbeitungszeiten zu vermeiden.

Auch die Skriptsprache JavaScript war den Entwicklern bereits vertraut, jedoch wurde hier aufgrund der geringen Einarbeitungszeit und dem Vorteil der starken Typisierung, welche Laufzeitfehler unwahrscheinlicher macht, auf TypeScript gesetzt.

Kapitel 3

3 Umsetzung

3.1 Allgemeines

3.1.1 Projektstruktur

Um die Struktur der Web-Anwendung übersichtlich und einheitlich zu gestalten, wurde sich auf eine Struktur der folgenden Form geeinigt:

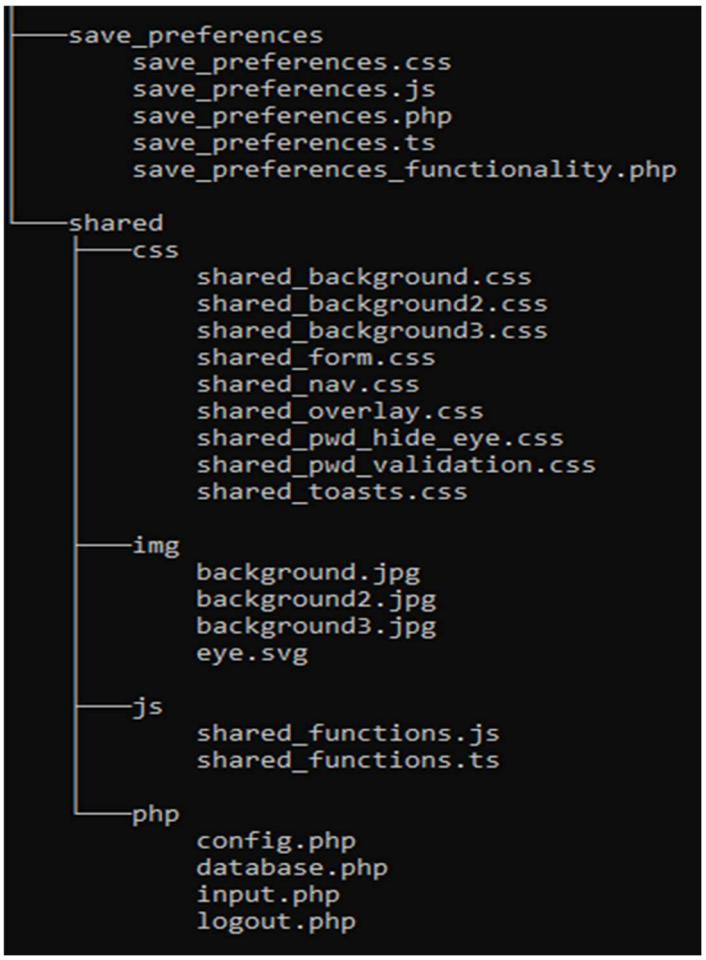

Abbildung 4: Projektstruktur (eigene Darstellung)

Der entscheidende Vorteil dieses Aufbaus liegt in der Ordnerstruktur: Alle zu einer Entwicklungsaufgabe gehörenden Objekte liegen in einem Ordner. Daraus resultiert, dass etwaige Fehlerbehebungen bzw. Designanpassungen sehr schnell vorgenommen werden können, da alle für eine Entwicklungsaufgabe relevanten Dateien in einem Ordner organisiert sind.

3.1.1.1 Gemeinsam genutzte Dateien

Eine besondere Rolle innerhalb der Projektstruktur nimmt das *shared*-Verzeichnis aus Abbildung 4 ein. In diesem sind Dateien hinterlegt, die im Entwicklungsprozess immer wieder benötigt werden, wie z. B. das Design der Toasts[1].

Die in diesem Ordner gespeicherten Dateien können somit zentral verwaltet werden, d. h., dass Änderungen in diesem Verzeichnis auf allen Seiten sichtbar werden, die das entsprechende Element referenzieren. Somit können beispielsweise Designänderungen einfach und schnell umgesetzt werden.

3.2 Startseite und Impressum

3.2.1 Startseite

Bei der Startseite handelt es sich um eine einfache HTML-Seite, sie dient der Weiterleitung des Anwenders auf die Login-, die Registrierungs- oder die Impressumseite.

Des Weiteren gibt sie dem Nutzer Rückmeldung über einen erfolgreichen Logout sowie über ein erfolgreiches Löschen des Accounts. Hierfür überprüft ein Skript den Local-Storage[2] des Browsers. Ist der Key *loggedOut* bzw. *deletedAccount* mit dem Wert *true* belegt, so wird ein Toast angezeigt, welches den erfolgreichen Logout bzw. die Löschung bestätigt, daraufhin folgt eine Bereinigung des Local-Storage des Browsers.

3.2.2 Impressum

Das Impressum zeigt lediglich allgemeine Informationen über die Website an, es soll außerdem als Anlaufstelle für Kontakt- und Supportanfragen dienen. Hierfür wurden Kontakt-E-Mail-Adressen mittels Anchor-Elementen hinterlegt.

[1] Toasts werden durch das Bootstrap-Framework bereitgestellt. Hierbei handelt es sich um anpassbare Push-Benachrichtigungen, die einem Nutzer bei Bedarf angezeigt werden können.

[2] Der Local-Storage eines Browsers dient dem Ablegen von Daten in einer Datenbank des Browsers. Diese liegen als Key-Value-Paare vor und können mittels Funktionen gesetzt oder abgefragt werden.

3.3 Registrierungsprozess

3.3.1 Persönliche Daten

Nachdem der Nutzer auf den „Get Started“-Button auf der Startseite gedrückt hat, kommt er auf die *register_personal_details*-Seite, um einen Account anzulegen. Hierfür muss er zunächst seine persönlichen Daten angeben.

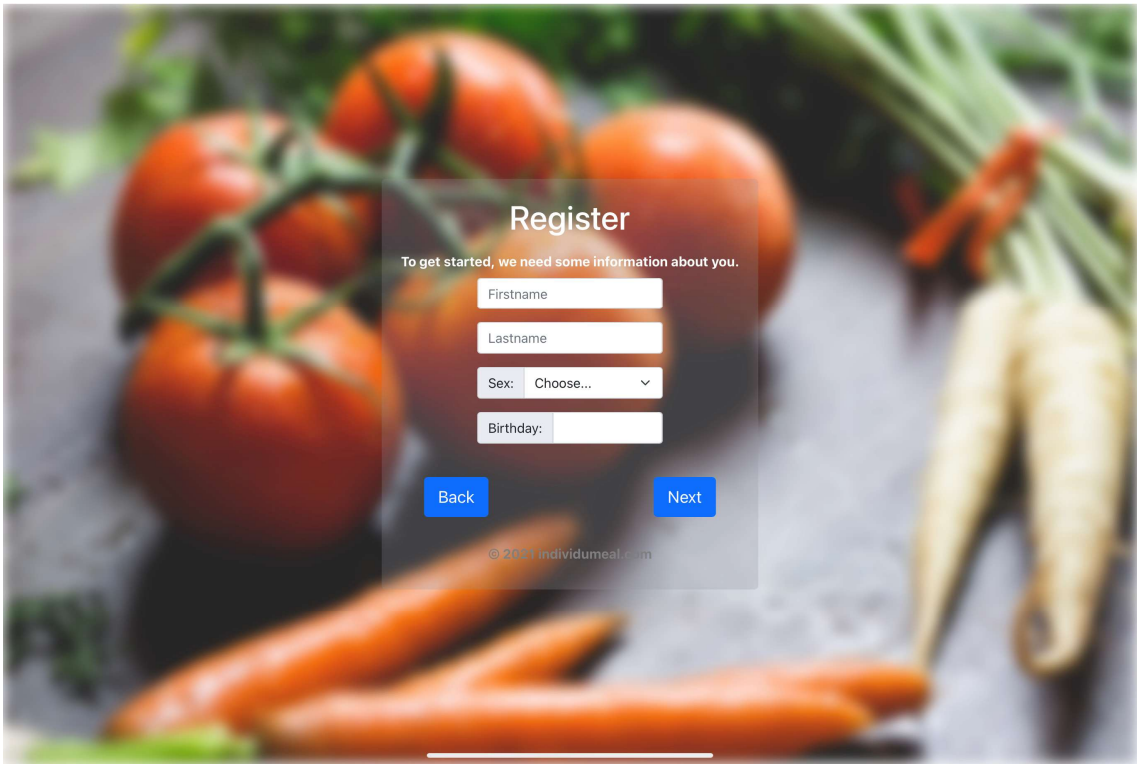

Abbildung 5: Maske zur Eingabe der persönlichen Daten (eigene Bildschirmaufnahme)

Der Anwender muss seinen Vor- und Nachnamen sowie sein Geschlecht und seinen Geburtstag angeben. Dabei wird vom Browser über die HTML-Attribute *minlength* und *maxlength* verifiziert, dass Vor- und Nachname jeweils mindestens 2 Zeichen und maximal 10 Zeichen lang sind. Um das Geschlecht zu hinterlegen, steht dem Anwender ein Dropdown-Feld zur Verfügung, mit welchem dieser zwischen „Male“, „Female“ und „Diverse“ auswählen kann. Zudem kann der Nutzer mittels eines Input-Felds sein Geburtsdatum auswählen. Dieser muss zwischen 01.01.1921 und 01.01.2008 liegen. Damit wird verifiziert, dass der Nutzer mindestens 13 und maximal 100 Jahre alt ist. Falls der Nutzer alle diese Werte korrekt eingetragen hat, kann er dies mit dem „Next“-Button bestätigen. Infolgedessen wird die Funktion *register_user* aufgerufen. Diese Funktion speichert die angegebenen Daten in den Local-Storage, damit sie auf der

nächsten Seite weiterverarbeitet werden können. Der Nutzer wird daraufhin auf die *register_account_details*-Seite weitergeleitet.
Falls der Benutzer sich dafür entscheidet, auf den „Back“-Button zu drücken, wird der Local-Storage geleert. Es folgt eine Weiterleitung auf die *index*-Seite.

3.3.2 Account Daten

Nachdem die persönlichen Daten hinterlegt wurden, folgt die Eingabemaske zum Hinterlegen von relevanten Account-Daten (z. B. der E-Mail-Adresse und dem Nutzernamen). Diese wird durch die *register_account_details*-Seite abgebildet, welche den Registrierungsprozess abschließen soll.

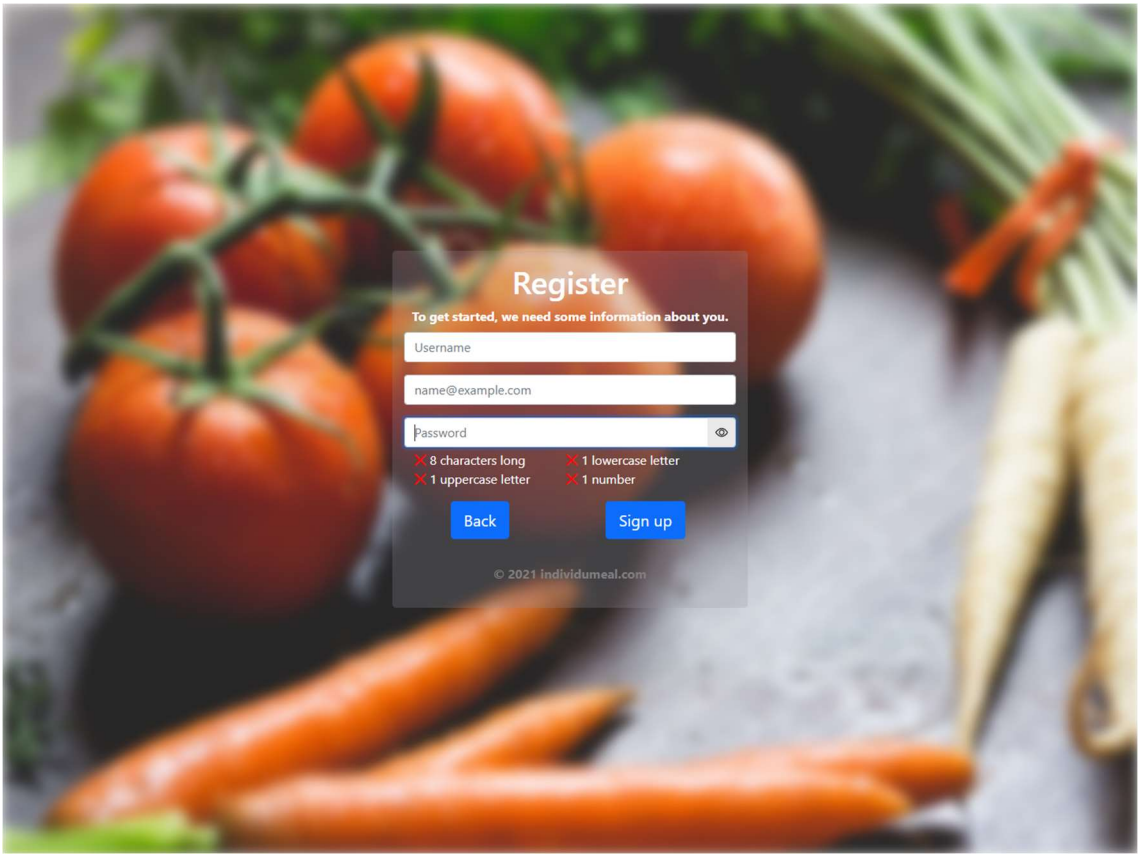

Abbildung 6: Maske zur Eingabe der Accountdaten (eigene Bildschirmaufnahme)

Hierfür werden die bereits erfassten persönlichen Daten aus dem Local-Storage ausgelesen. Anschließend erfolgt eine Validierung der Nutzereingaben, um ein doppeltes Anlegen von Nutzeraccounts zu vermeiden.
Diese Prüfung wird durch ein Skript realisiert, welches Duplikate in der Datenbank erkennen soll.
Dazu werden zwei asynchrone Post-Anfragen an das Backend gesendet. Diese enthalten die zu überprüfenden Daten (E-Mail oder Nutzername) und die gewählte Methode (*check_duplicates_email* oder *check_duplicates_username).*

Das Backend prüft nun mittels einer SQL-Anfrage, ob die übertragene E-Mail bzw. der übertragene Nutzername bereits in der Datenbank vorhanden ist. Im Anschluss wird eine Meldung an das Frontend versandt, die angibt, ob es sich bei den übermittelten Daten um Duplikate handelt. Ist dies der Fall, so wird der Nutzer mittels einem bzw. mehreren Toasts auf den Fehler hingewiesen. Verlief die Prüfung auf Duplikate negativ, so wird eine weitere Anfrage an das Backend versandt, welche alle vom Nutzer angegebenen Daten beinhaltet. Der Service schreibt diese nun in der Datenbank fest und leitet den Nutzer auf die Login-Seite um. Gleichzeitig wird im Local-Storage der Erfolg der Registrierung vermerkt, um den Anwender mittels eines Toasts auf der Login-Seite auf das erfolgreiche Anlegen seines Accounts hinweisen zu können.
Vor dem Festschreiben auf der Datenbank wird das Passwort aus Sicherheitsgründen gehasht.

3.3.3 Eingabe der Präferenzen

Nach erstmaligem Login gelangt der Benutzer auf die *save_preferences.php*-Seite.

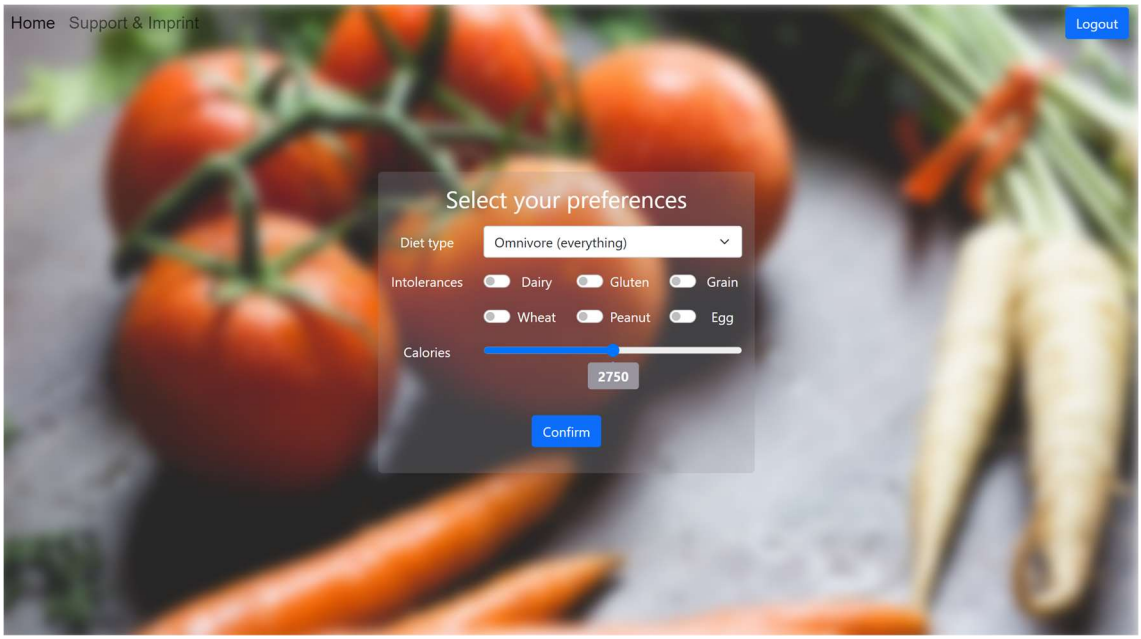

Abbildung 7: Maske zur Eingabe der Nutzerpräferenzen (eigene Bildschirmaufnahme)

Diese ermöglicht es ihm, seinen Ernährungsstil über ein Dropdown-Menü (Omnivore, Vegetarian, Vegan), die Intoleranzen (Dairy, Gluten, Grain, Wheat, Peanut, Egg) durch Checkboxen und die Kalorien über einen Schieberegler auszuwählen. Mithilfe des Action-Attributs des Form-Elements werden die erfassten Werte über einen Post-Request an die

save_preferences_functionality.php-Seite gesendet. Dort werden iterativ alle ausgewählten Intoleranz-Checkboxen zu einem Kommata-getrennten String konkateniert, sodass dieser möglichst einfach in einer einzelnen Spalte in der Datenbank abgespeichert werden kann. Ebenso werden die Kalorien und der Ernährungsstil in der Datenbank gespeichert. Dadurch gelingt es, die später generierten Gerichte spezifisch auf die Vorlieben, Unverträglichkeiten und Unterschiede im Ernährungsstil eines jeden Benutzers zu individualisieren.

3.3.4 Eingabe der verfügbaren Lebensmittel

Nach der Eingabe der Präferenzen wird der Anwender nach den ihm im Haushalt zur Verfügung stehenden Lebensmitteln gefragt. Zur Abfrage dieser dient die *ingredients_input.php*-Seite, auf welcher dem Nutzer ein Form-Element zur Verfügung steht, mit welchem er diese eingeben kann.

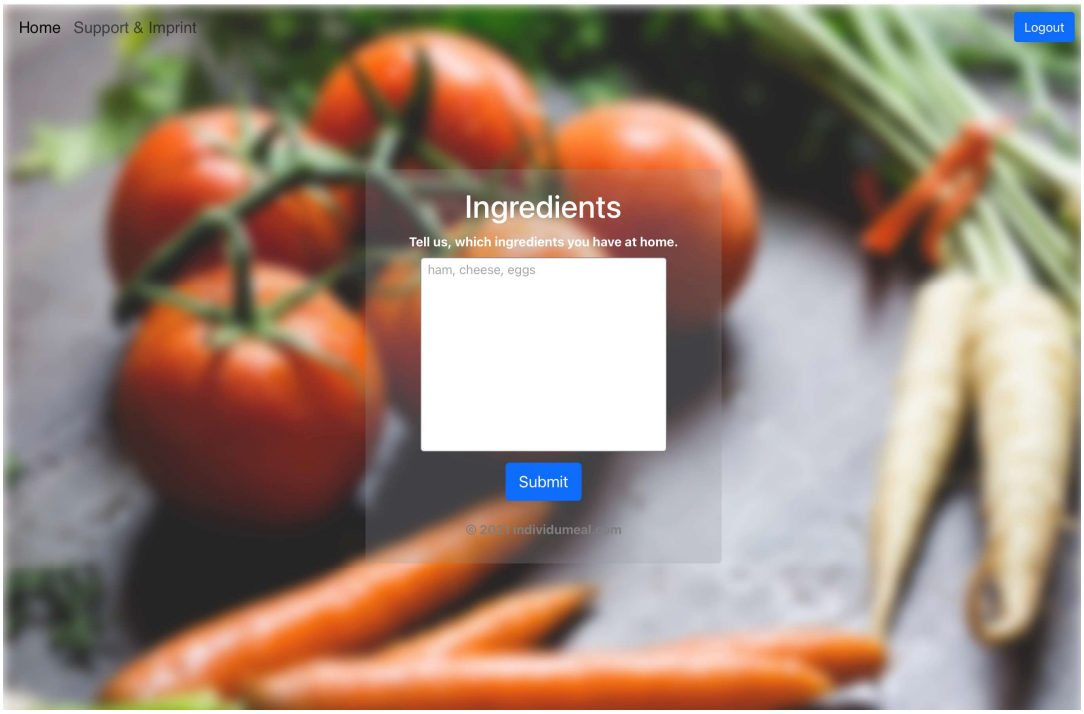

Abbildung 8: Maske zur Eingabe der verfügbaren Lebensmittel (eigene Bildschirmaufnahme)

Mittels des Placeholder-Attributs wird ihm bereits ein gültiger Aufbau einer solchen Eingabe vorgeschlagen. Diese darf ausschließlich aus Kommas und Groß- und Kleinbuchstaben bestehen, die Einhaltung dieser Vorgabe wird mittels eines Skriptes überprüft. Eine semantische Prüfung des angegebenen Inhalts erfolgt nicht.

Ist der syntaktische Aufbau der Anfrage korrekt, wird eine asynchrone Post-Anfrage mit der in der Session gespeicherten User-ID und der Nutzereingabe an das Backend übermittelt, welches diese in der Datenbank speichert.

Auch diese Seite weist mittels Toasts auf etwaige Fehler im Eingabeprozess bzw. auf Fehler, die Rahmen der Speicherung der Nutzerdaten auftreten können, hin.

3.4 An- und Abmeldeprozess

3.4.1 Anmeldung

Nach dem Erstellen eines Benutzeraccounts wird der Anwender auf die Login-Seite weitergeleitet. Durch Klicken des „Sign up“-Hyperlinks gelangt dieser auf die *register_personal_details*-Seite. Auf dieser kann der Anwender, falls er das noch nicht getan hat, den Registrierungsprozess starten. Zudem wird dem Benutzer die Möglichkeit geboten, bei vergessenem Passwort auf die *reset_password.html*-Seite weitergeleitet zu werden. Außerdem kann sich der Benutzer durch Drücken des „Submit“-Buttons des Formulars anmelden.

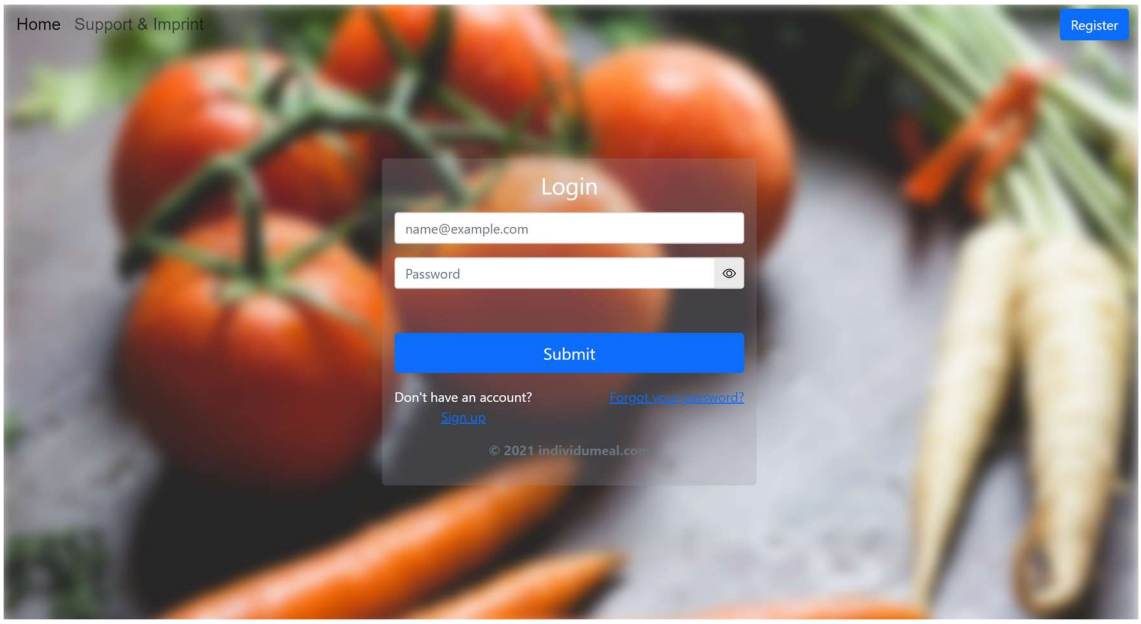

Abbildung 9: Die login_form.php-Seite (eigene Bildschirmaufnahme)

Die *login_form.php*-Seite prüft nach Absenden von E-Mail und Passwort durch einen Post-Request, ob beide Input-Felder ausgefüllt wurden und ob es ein Benutzerkonto mit dieser E-Mail-Adresse gibt. Ist dies der Fall, wird mit einer SQL-Anfrage das Passwort des Accounts aus der Datenbank ausgelesen und mithilfe der von PHP bereitgestellten *password_verify*-Funktion überprüft, ob der Passwort-String aus dem Input-Feld und das gehashte Passwort aus der Datenbank übereinstimmen. Fällt die Prüfung positiv aus, werden für die spätere Wiederverwendung auf der *change_settings.php*-Seite *userid, firstname*, *lastname*, *username*, *email* und *dateofbirth* in Session-Variablen gespeichert.

Schlägt eine der genannten Prüfungen fehl, wird ein Fehlertoast ausgegeben. Nach dem Login wird zudem geprüft, ob dem Benutzeraccount Präferenzen und/oder zu Hause vorhandene Lebensmittel zugeordnet sind. Ist dies der Fall, folgt die Weiterleitung auf die *meal_plan_overview*-Seite, andernfalls auf die *ingredients_input.php*-Seite.

3.4.2 Abmeldung

Jeder „Logout“-Button verweist auf die *logout.php*-Seite im *shared*-Ordner.
In dieser werden zunächst alle Session-Variablen gelöscht, anschließend werden alle mit der aktuellen Sitzung verknüpften Daten zerstört.
Es folgt ein Setzen des Local-Storages, welches den Key *loggedOut* mit dem Wert *true* belegt, um den erfolgreichen Logout zu dokumentieren. Im Anschluss wird auf die *index.html*-Seite weitergeleitet.

3.5 Rezept-Übersicht

3.5.1 Funktion und Aufbau

Die *meal_plan_overview*-Seite stellt das Herzstück dieses Projekts dar. Sie verwirklicht die Hauptfunktionalität: das Anzeigen personalisierter, auf den Nutzer zugeschnittener Rezepte.

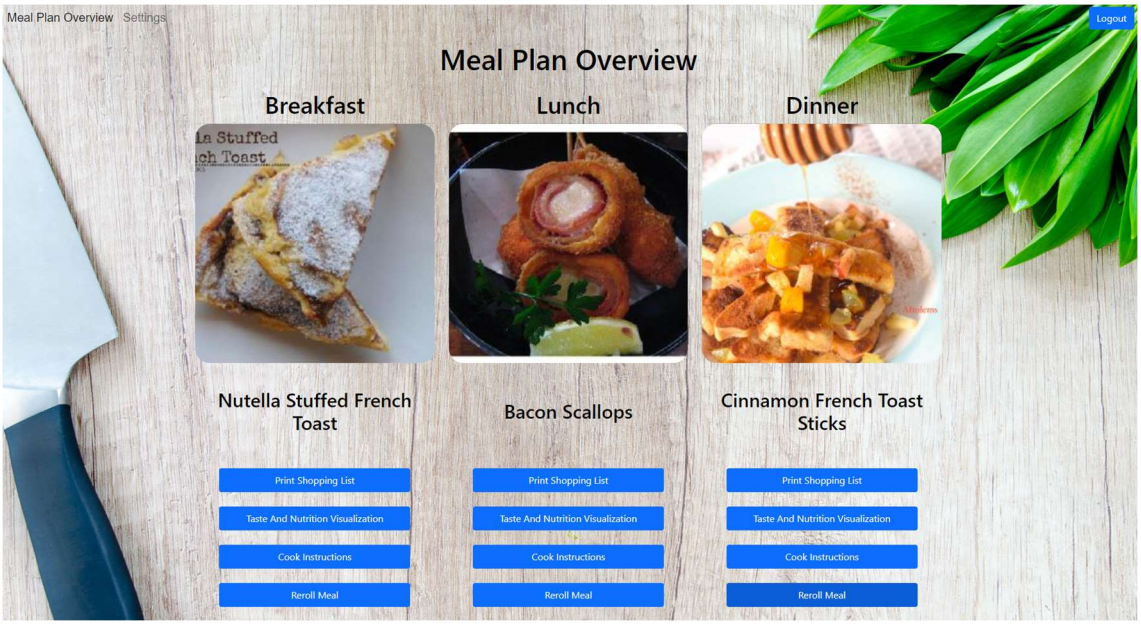

Abbildung 10: Die meal_plan_overview-Seite (eigene Bildschirmaufnahme)

Die Seite ist aus drei Bootstrap Cards[3] aufgebaut. Jede Card stellt eine Mahlzeit dar, jeweils eine für Frühstücks-, Mittags- und Abendessen. Zur jeder ermittelten Mahlzeit werden vier Funktionalitäten bereitgestellt: das Ausdrucken einer Einkaufs-Liste („Print Shopping List“), das Anzeigen des Geschmacks und der enthaltenen Nährstoffe („Taste And Nutrition Visualization“), die Generierung einer Kochanleitung („Cook Instructions“) und das Neuauswählen eines Rezepts („Reroll Meal“).

3.5.2 Generierung der Rezepte

Sobald die *meal_plan_overview*-Seite aufgerufen wird, wird die Funktion *getRecipesByUserId* ausgeführt. Diese sendet eine asynchrone Anfrage an *receipes.php*. In ihrem Body beinhaltet die Anfrage die User-ID und den Funktions-Namen *get_recipes_by_user_id.*

Das Backend nimmt die Anfrage an, daraufhin wird die Funktion *get_recipes_by_user_id* aufgerufen. Diese lädt die vom Nutzer angegeben Lebensmittel und Präferenzen aus der Datenbank. Danach versucht die Funktion, die zu einem Nutzer abgespeicherten Rezepte aus der Datenbank zu laden. Dieser Prozess soll unnötige API-Anfragen verhindern. Für das Laden dieser Rezepte wird die Meal-Type-Nr benutzt. Es gibt für jede Mahlzeit eine eigene Meal-Type-Nr. Diese gibt aus den Rezepten, die für den Nutzer abgespeichert wurden, an, welches das aktuelle Rezepte für die jeweilige Mahlzeit ist. Initialisiert werden diese jeweils mit dem Wert 0 beim Erstellen des Accounts.

Wurden für einen Anwender keine Rezepte ermittelt, so müssen diese mittels einer API-Anfrage ermittelt werden. Nachfolgendes Diagramm dient der Visualisierung von ebendiesem Prozess.

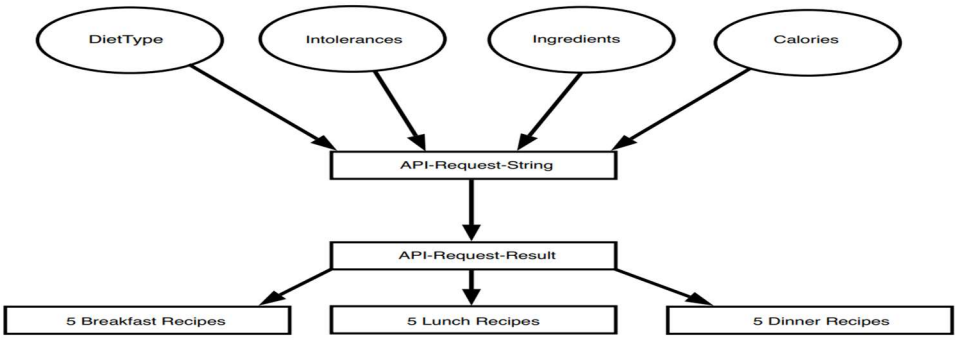

Abbildung 11: Rezeptgenerierungsprozess (eigene Darstellung)

[3] Bootstrap Cards stellen flexible, erweiterbare Inhaltscontainer dar.

Die Rezepte werden mithilfe verschiedener Eingaben generiert. Diese umfassen den Ernährungstyp, Intoleranzen, Lebensmittel und Kalorien, welche bereits zu Beginn der Funktion ermittelt wurden. Anhand ebendieser wird eine URL generiert, welche diese als Anfrage-Parameter enthält. Um dem Nutzer möglichst schnell Ergebnisse präsentieren zu können, wird nicht für jede Mahlzeit nach der Kalorienanzahl gefragt, sondern nur nach einer Gesamtzahl für alle Mahlzeiten. Da es insgesamt drei Mahlzeiten gibt, wird die eingegeben Kalorienzahl durch drei geteilt und dieses Ergebnis mit der URL konkateniert. Zudem wird festgelegt, dass insgesamt zehn Rezepte zurückgeliefert werden sollen.

Die erstellte URL wird an die Spoonacular-API[4] gesendet. Diese liefert zehn Rezepte pro Mahlzeit zurück. Davon werden fünf Rezepte in der Datenbank abgespeichert. Da die Rezepte beim Mittags- und Abendessen immer gleich sind[5], werden für das Mittagsessen das erste bis fünfte Rezept und für das Abendessen das sechste bis zehnte Rezepte hinterlegt. Damit ist gewährleistet, dass dem Nutzer für jede Mahlzeit unterschiedliche Rezepte angezeigt werden. In der Datenbank wird die interne Rezept-ID der API („MealID“), der Mahlzeit-Typ („MealType“), ein Bild der Mahlzeit („Image“) und die genutzten und nicht vorhandenen Lebensmittel abgespeichert („UsedIngredients“, „MissedIngredients“).

Wurden die Rezepte in der Datenbank hinterlegt, so wird das erste von diesen an das Frontend übermittelt. Die API-ID von ebendieser wird sodann im Local-Storage als *breakfast_id*, *lunch_id* bzw. *dinner_id* für jede Mahlzeit abgespeichert. Anschließend werden die Rezepte auf der Rezept-Übersicht-Seite angezeigt.

Sollte die API nicht genügend Vorschläge für Mahlzeiten übermitteln, werden die bisher gespeicherten Rezepte gelöscht und es wird eine Fehlermeldung an das Frontend übermittelt, welche der Nutzer per Toast angezeigt bekommt. In der Fehlermeldung wird er darum gebeten, seine Präferenzen oder die eingegebenen Lebensmittel anzupassen. Hierfür wird er nach fünf Sekunden auf die *change_settings*-Seite weitergeleitet.

[4] Spoonacular bietet eine API an, um auf über 5000 Rezepte zugreifen zu können (https://spoonacular.com/food-api).
[5] Ermittelt mittels E-Mail-Nachfrage.

3.5.3 Ausdrucken einer Einkaufsliste

Zu jedem Rezept gibt es eine Liste von Lebensmitteln, die der Nutzer angegeben hat, also schon zuhause hat ("Used Ingredients") und eine Liste von Lebensmitteln, die der Nutzer noch nicht hat ("Missed Ingredients"), also noch einkaufen muss. Mithilfe der Funktion „Print Shopping List“ wird es dem Nutzer ermöglicht, beide Listen auszudrucken.

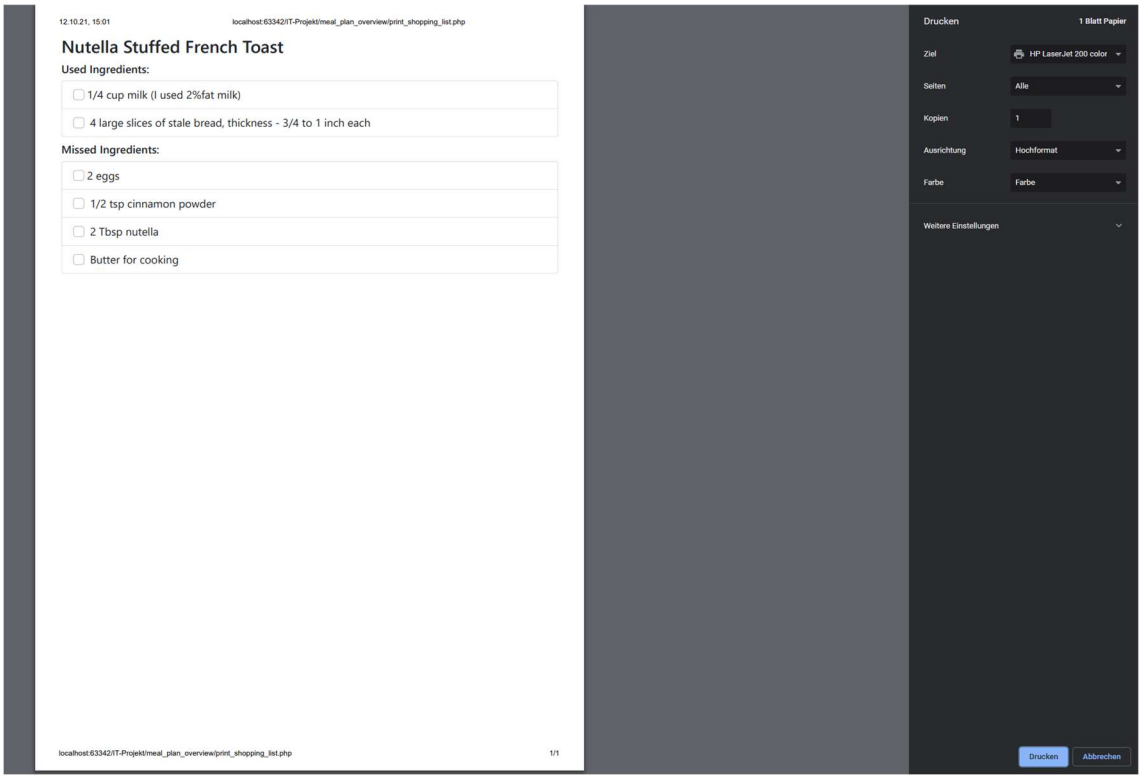

Abbildung 12: Druckvorschau eines Rezeptes (eigene Bildschirmaufnahme)

Nach einem Klick auf den „Print Shopping List“-Button wird der Nutzer auf die *print_shopping_list*-Seite weitergeleitet, gleichzeitig wird auch noch im Local-Storage der Mahlzeit-Typ des Rezeptes hinterlegt.

Beim Laden der Seite wird eine asynchrone Post-Anfrage an *receipes.php* gesendet. In ihrem Body enthält diese die User-ID, die Meal-ID aus dem Local-Storage, welche mithilfe des Mahlzeit-Typs ermittelt wird, und den Funktions-Namen *print_shopping_list*. Das Backend sucht per SQL-Anfrage den Titel, die genutzten und nicht vorhandenen Lebensmittel heraus und liefert diese in Form von Listen an das Frontend zurück. Dieses bindet den Titel und die Listen dynamisch ein, um sie ausdrucken zu können. Hierbei wird darauf geachtet, dass nach einer bestimmten Anzahl an Elementen ein Umbruch auf die nächste Druck-Seite stattfindet, damit die Listenelemente nicht abgeschnitten werden. Zudem

wird darauf geachtet, dass im Falle einer leeren Liste, die entsprechende Überschrift „UsedIngredients“ bzw. „MissedIngredients“ entfernt wird.

3.5.4 Anzeigen des Geschmacks und der Nährstoffe

Mithilfe der Funktion „Taste And Nutrition Visualization“ kann der Nutzer sich zu jedem Rezept den Geschmack und die zugehörigen Nährwerte eines Rezepts anzeigen lassen.

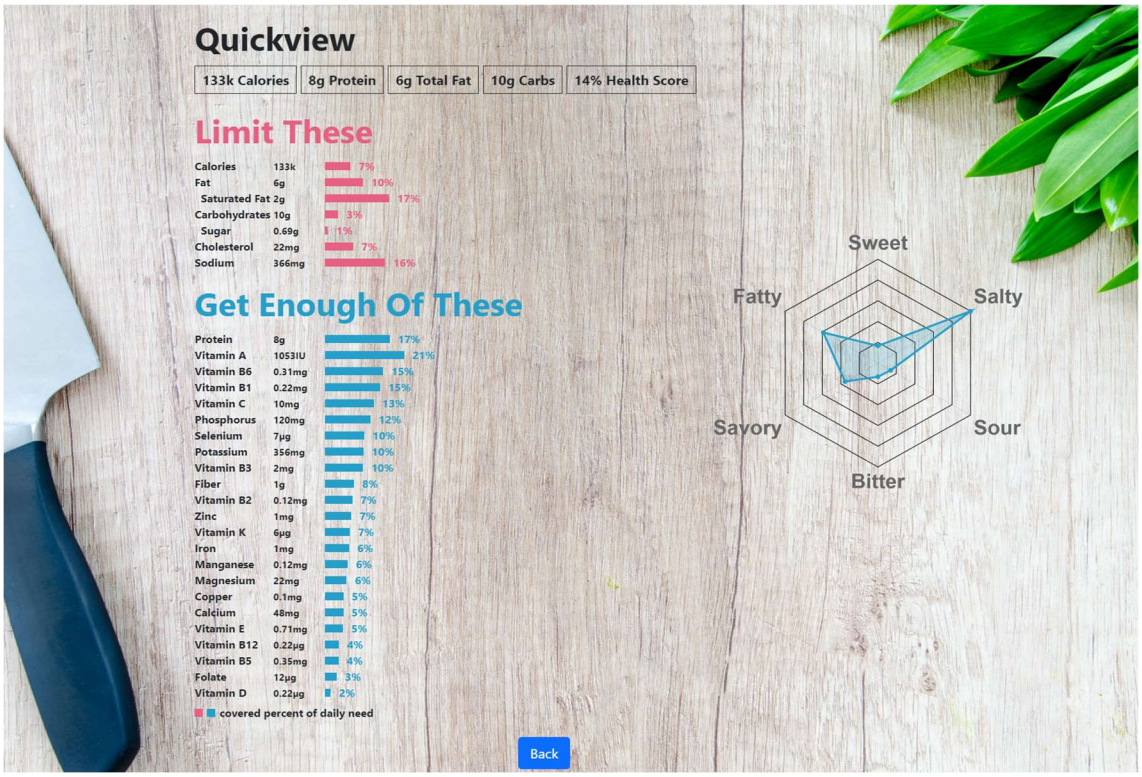

Abbildung 13: Geschmacks- und Nährstoffvisualisierung eines Rezeptes (eigene Bildschirmaufnahme)

Drückt der Anwender auf den „Taste And Nutrition Visualisation“-Button, wird eine Funktion ähnlich der bei „Print Shopping List“ ausgelöst. Der Nutzer wird auf die *taste_and_nutrition_visualization*-Seite weitergeleitet und es wird im Local-Storage der Mahlzeit-Typ abgespeichert.

Beim Laden der Seite wird auch bei dieser Funktionalität eine asynchrone Anfrage ausgelöst. Hierfür wird, ähnlich wie bei der zuvor erwähnten Seite, die Meal-ID aus dem Local-Storage geladen. Im Body der Anfrage befindet sich die Meal-ID und der Funktionsname *get_taste_and_nutrient_visualization*. Anschließend wird im Backend mithilfe der Meal-ID eine Anfrage an die Spoonacular API generiert. Daraufhin liefert die API zwei Widgets zurück, eines für den Geschmack und eines für die Nährstoffe. Diese enthalten HTML, CSS und JavaScript. Das Frontend nimmt diese entgegen und separiert den

JavaScript-Teil, dieser wird samt dem restlichen CSS und HTML getrennt eingebunden. Dies ist nötig, da beim gemeinsamen Einbinden das JavaScript nicht ausgeführt wird[6]. Auf der linken Seite des Bildschirms werden die enthaltenen Nährstoffe angezeigt. Am oberen Rand werden die wichtigsten Daten unter dem Stichpunkt „Quickview“ zusammengefasst. Darunter werden unter dem Stichpunkt „Limit These“ Nährstoffe aufgelistet, welche man in Maßen konsumieren sollte. Zuletzt werden unter dem Stichpunkt „Get Enough Of These“ die Nährstoffe aufgelistet, bei denen man darauf achten sollte, sie in ausreichenden Mengen zu konsumieren.

Auf der rechten Seite wird der Geschmack angezeigt. Dieser wird mithilfe eines Netzdiagrammes visualisiert.

In seltenen Ausnahmefällen tritt bei dieser Funktion ein Fehler auf. Dieser ist auf eine fehlerhafte Antwort des Backends zurückzuführen. Nachfolgendes Bild stellt die problematischen Zeichen dar (Rot):

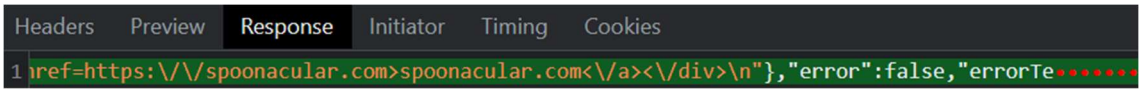

Abbildung 14: Fehlerhafte Antwort der receipe.php-Seite (eigene Bildschirmaufnahme)

Die Entwickler konnten keinen Grund für die fehlerhafte Antwort finden. Ein wiederholtes Laden der Seite löst dieses Problem.

[6] https://stackoverflow.com/questions/75943/how-do-you-execute-a-dynamically-loaded-javascript-block

3.5.5 Generierung einer Kochanleitung

Mit einem Klick auf den „Cook Instructions“-Button kann der Nutzer sich eine Kochanleitung für das Rezept generieren lassen.

Abbildung 15: Kochanleitung eines Rezeptes (eigene Bildschirmaufnahme)

Nach einem Klick auf den entsprechenden Button wird, wie bei den anderen Funktionen zuvor, der Typ der Mahlzeit im Local-Storage gespeichert, anschließend wird der Nutzer auf die *recipe_card*-Seite weitergeleitet. Auf dieser wird beim Laden der Seite eine asynchrone Anfrage ausgelöst. Im Body ebendieser befindet sich die Meal-ID und der Funktionsname *get_recipe_card*. Das Backend generiert mithilfe der Meal-ID eine Anfrage an die Spoonacular-API. Diese liefert ein Bild von der Kochanleitung zurück, welches dynamisch in die Seite eingebunden wird.

3.5.6 Neuauswahl eines Rezepts

Falls dem Nutzer das angezeigte Rezept missfällt, kann er sich mithilfe des „Reroll“-Buttons ein neues Rezept vorschlagen lassen.

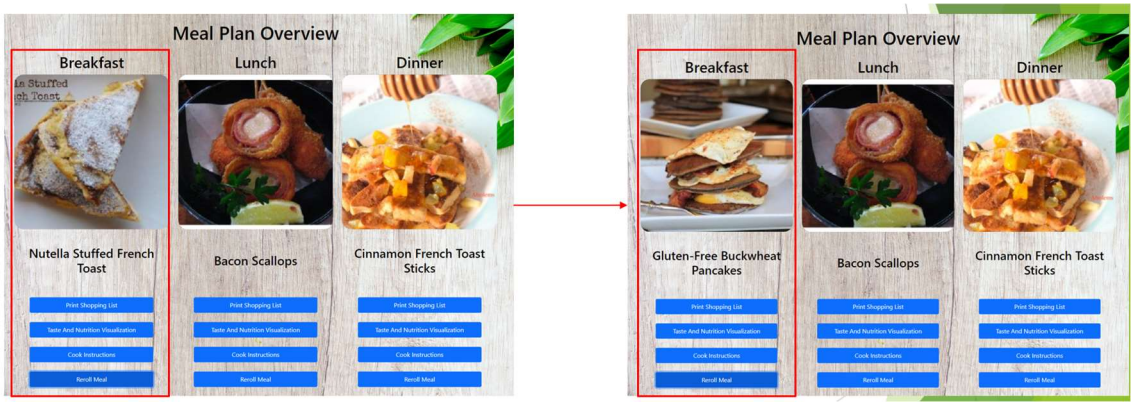

Abbildung 16: Ändern eines Rezeptes (eigene Bildschirmaufnahme)

Beim Klick auf den „Reroll“-Button wird die Funktion *incrementCurrentMealNr* aufgerufen. Diese löst eine asynchrone Anfrage aus, welche die User-ID, die Meal-Type-Nr für eine Mahlzeit und den Funktionsnamen *increment_current_meal_nr* enthält. Im Backend wird daraufhin für die Mahlzeit, welche bei Meal-Type-Nr angegeben wurde, um eins Modulo fünf inkrementiert. Das Frontend ruft sodann die Funktion *getRecipesByUserId* aus. Der Aufruf bewirkt sodann das Anzeigen des nächsten Rezepts, da sich die Meal-Type-Nr geändert hat (vgl. hierzu Kapitel „3.5.2 Generierung der Rezepte“).

3.6 Ändern der Einstellungen

Über das Navbar-Element *Settings* der *meal_plan_overview.php*-Seite ist der Anwender in der Lage, auf die *change_settings.php*-Seite zu navigieren. Diese ermöglicht diesem, Änderungen an den hinterlegten Daten vorzunehmen.

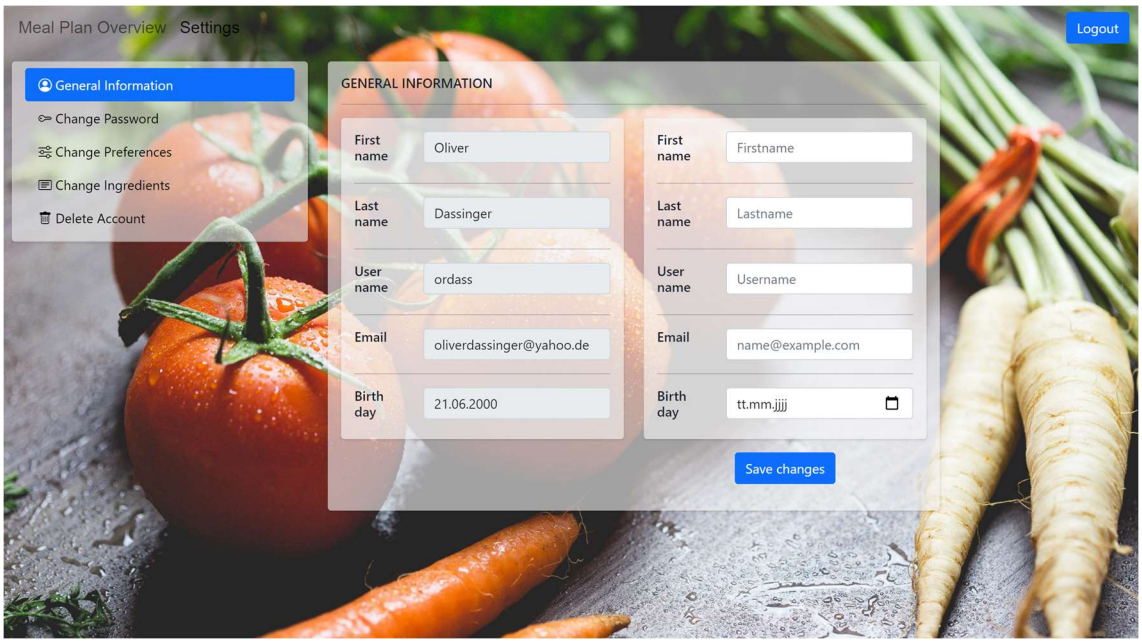

Abbildung 17: Die change_settings.php-Seite (eigene Bildschirmaufnahme)

In dem *General Information*-Tab der Navbar werden die aktuell in der Session gespeicherten persönlichen Informationen *Firstname, Lastname, Username, E-Mail und Birthday* ausgegeben. Mittels des „Save-Changes“-Button kann der Nutzer die von ihm geänderten Daten, welche er im rechten Formular hinterlegt hat, speichern. Hierfür werden diese sodann mittels einer Post-Anfrage an die *change_settings_functionality.php*-Seite weitergeleitet, welche die Daten ohne inhaltliche Validierung in der Datenbank speichert. Wurde ein Eingabefeld nicht befüllt, antwortet das Backend mit einem Fehler, woraufhin ein Errortoast mit entsprechender Fehlermeldung ausgegeben wird. Wurden die Daten erfolgreich gespeichert, so werden diese auf der Seite aktualisiert. Zusätzlich wird ein Erfolgstoast ausgegeben, welches den Nutzer über die erfolgreiche Änderung informiert.

Um das beim Registrierungsprozess vergebene Passwort ändern zu können, muss der Benutzer zum *Change Password*-Tab wechseln. Klickt der dieser sodann in die Input-Felder, werden durch das Aktivieren des *onfocus*-Event eine Reihe von Voraussetzungen in roter Schrift sichtbar, welche dem Anwender Anhaltspunkte für ein starkes Passwort liefern sollen. Sobald das Input-Feld den Fokus verliert, werden diese ausgeblendet.

Das Passwort muss aus mindestens 8 Zeichen, davon mindestens ein Groß- und Kleinbuchstabe sowie mindestens einer Zahl bestehen. Diese Überprüfung geschieht mithilfe des *onkeyup*-Events, welches die eingegebenen Zeichen mittels eines regulären Ausdrucks validiert und bei Erfüllung der Voraussetzungen die Farbe der Schrift auf grün ändert. Außerdem muss der Benutzer das Passwort zur Bestätigung nochmals in einem zweiten Input-Feld wiederholen. Wird der „Submit“-Button gedrückt, wird das Passwort durch eine asynchrone Post-Anfrage an die *change_settings_functionality.php*-Seite weitergeleitet. Die Methode *change_password* prüft nun, ob beide Passwörter identisch sind und dem geforderten Format entsprechen. Ist dies der Fall, wird das neue Passwort gehasht und in die Datenbank geschrieben.

Erneut werden bei erfolgreichem Ändern ein Erfolgstoast und im Fehlerfall ein Fehlertoast mit entsprechender Nachricht ausgegeben.

Möchte der Anwender zusätzlich seine Präferenzen ändern, ist dies im *Change Preferences*-Tab möglich. Optisch und funktionstechnisch fast identisch zu der *save_preferences.php*-Seite des Login-Prozesses, kann der Nutzer hier seine

aktuell ausgewählten Präferenzen einsehen. Durch ein *onload*-Event wird mit der *display_settings*-Methode die *userid* an das Backend weitergeleitet, woraufhin die aktuell gespeicherten Präferenzen aus der Datenbank gelesen und wieder an das Frontend geschickt werden, um dort den Ernährungsstil im Dropdown, die Intoleranzen in den Checkboxen und die ausgewählten Kalorien im Schieberegler anzupassen. Hat der Anwender seine neuen Präferenzen selektiert und den „Submit“-Button getätigt, wird erneut eine asynchrone Post-Anfrage an das Backend geschickt und mittels der *change_preferences*-Funktion die neuen Werte in der Datenbank gespeichert. War dies erfolgreich, werden die auf den alten Präferenzen beruhenden Gerichte gelöscht und ein Erfolgstoasts ausgegeben.

Durch einen weiteren Reiter, dem *Change Ingredients*-Tab, wird dem Benutzer ermöglicht, neue Lebensmittel einzugeben. Nach einer Bestätigung dieser durch Klicken des „Submit“-Buttons wird die Einhaltung der syntaktischen Form der Eingabe überprüft und bei Erfolg eine asynchrone Post-Anfrage mit User-ID und Nutzereingabe an das Backend übermittelt, welches die neuen Werte in der Datenbank speichert. Gleichermaßen werden hier die auf den alten Zutaten beruhenden Gerichte gelöscht und bei erfolgreicher Änderung ein Erfolgstoast ausgegeben. Hierbei wird die bereits vorhandene Funktionalität der *ingredients_input.php*-Seite verwendet.

Als letzte Einstellungsoption wird dem Anwender die Möglichkeit geboten, nach Eingabe des Passwortes seinen Benutzeraccount zu löschen. Hierfür wird eine asynchrone Post-Anfrage an das Backend versendet. Dort wird mithilfe *password_verify*-Funktion überprüft, ob das eingegebene Passwort und das gehashte Passwort aus der Datenbank übereinstimmen. Ist dies der Fall, so wird der Benutzeraccount und alle Tabellen, die einen Fremdschlüssel auf die User-ID besitzen, aus der Datenbank gelöscht (On-Delete-Cascade). Der Anwender wird auf die Startseite weitergeleitet, auf der er durch einen Erfolgstoast über das erfolgreiche Löschen des Accounts unterrichtet wird.

3.7 Zurücksetzen eines Passwortes

Ein Anwender kann sein Passwort auf der *reset_password.html*-Seite selbstständig zurücksetzen.

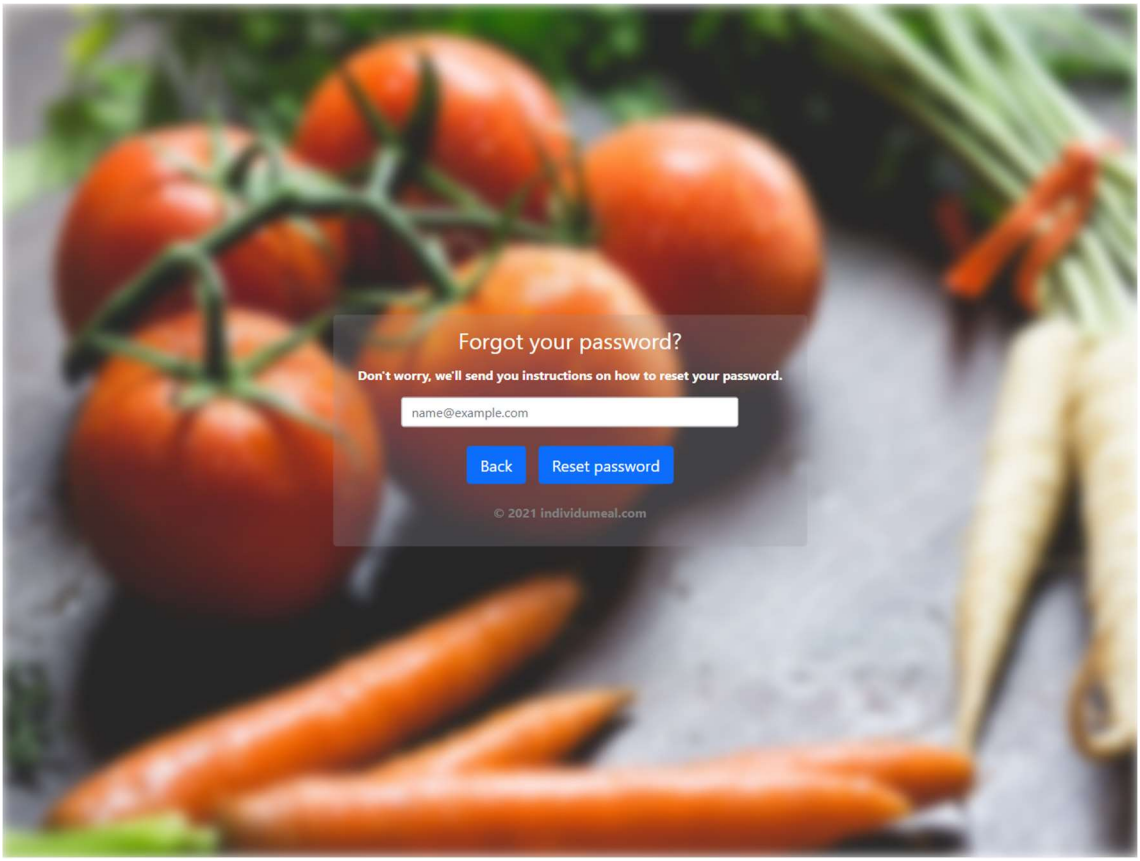

Abbildung 18: Die reset_password.html-Seite (eigene Bildschirmaufnahme)

Hierfür gibt er seine E-Mail auf einer Eingabemaske ein, diese wird mittels einer asynchronen Post-Anfrage und der gewählten Methode (in diesem Fall *reset_password_request*) an das Backend versandt. Es folgt eine Prüfung, ob die E-Mail-Adresse in der Datenbank hinterlegt ist. Ist dies nicht der Fall, wird ein Toast mit der entsprechenden Fehlermeldung ausgegeben und der Prozess zum Zurücksetzen des Passworts abgebrochen.

Ist die E-Mail-Adresse hinterlegt, erfolgt eine Prüfung auf vergangene Rücksetzversuche. Um Spam an die angegebene Adresse zu vermeiden, ist ein Solcher nur alle zehn Minuten möglich. Schlägt die Prüfung fehl, wird das Prozedere abgebrochen und ein Toast mit einem Hinweis an den Nutzer ausgegeben.

Ist die E-Mail-Adresse vorhanden und es gibt keine bzw. keine länger als zehn Minuten zurückliegenden Rücksetzversuche, wird der Nutzer hierüber informiert und der eigentliche Prozess zum Neusetzen des Passworts gestartet.

Hierfür wird zunächst ein zufälliger 64-stelliger Token und ein zugehöriger Ablaufzeitpunkt generiert, beide Informationen werden in der Datenbank hinterlegt.

Anschließend wird eine HTML-Seite erstellt, welche ein Form-Element enthält. Das Action-Attribut enthält eine Weiterleitung auf die *set_new_password.php*-Seite, wobei der Query-Teil den generierten Rücksetztoken und das Ablaufdatum von eben diesem enthält. Die erstellte Seite wird als E-Mail an die angegebene Adresse versandt.

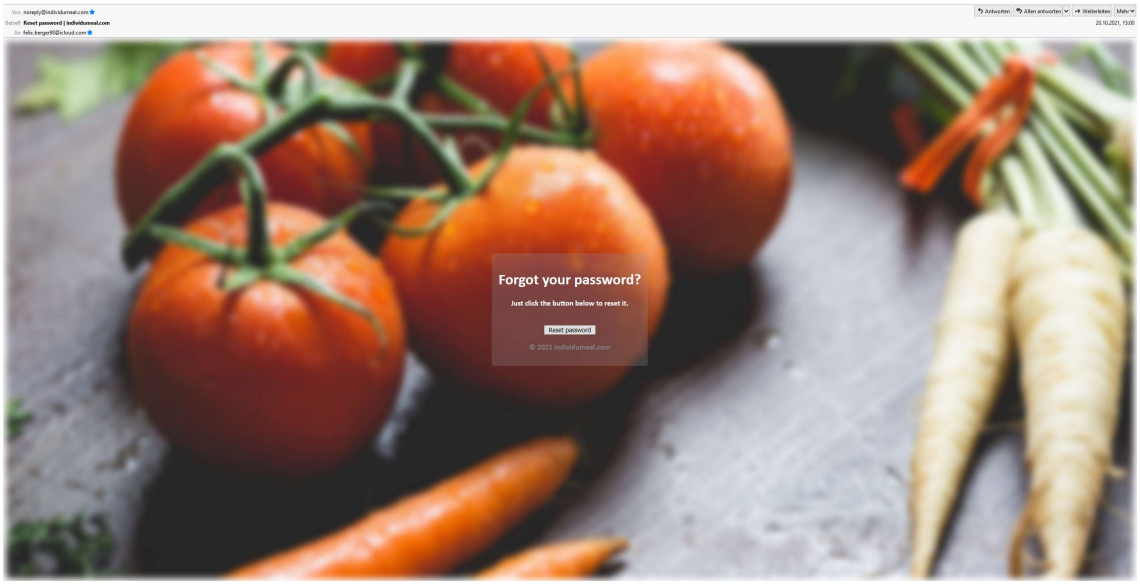

Abbildung 19: Die an den Anwender versendete E-Mail (eigene Bildschirmaufnahme)

Klickt der Nutzer nun auf den „Reset Password“-Button in der erhaltenen E-Mail, wird er auf die *set_new_password.php*-Seite umgeleitet, auf welcher eine weitere asynchrone Post-Anfrage an das Backend erfolgt. Diese enthält die Methode *reset_password*, den Reset-Token, das Ablaufdatum des Tokens und das neu gewählte Passwort.

Die empfangenen Daten werden mit den in der Datenbank hinterlegten Daten abgeglichen. Ist der übermittelte Token nicht abgelaufen und stimmt mit dem in der Datenbank gespeicherten überein, so wird das neu gewählte Passwort festgeschrieben. Der Anwender wird mittels eines Toasts über den Erfolg des Prozesses informiert.

Im Falle einer Ungültigkeit des Tokens (z. B. weil dieser abgelaufen oder manipuliert worden ist), wird der Prozess zum Zurücksetzen des Passworts abgebrochen, auch in diesem Fall wird der Nutzer über einen Toast über den entsprechenden Fehler informiert.

Kapitel 4

4 Zusammenfassung

Das letzte Kapitel dient der Zusammenfassung des Entwicklungsprozesses. Hierfür werden zunächst aufgetretene Probleme und deren Lösungen erläutert. Im Anschluss werden die erarbeiteten Ergebnisse in einem Fazit vorgestellt und ein Ausblick auf mögliche zukünftige Weiterentwicklungen der implementierten Web-Anwendung gegeben.

4.1 Probleme und Schwierigkeiten

4.1.1 Entwicklungsschwierigkeiten

Die Entwickler hatten bisher keine Erfahrung im Strukturieren, Implementieren und Testen großer Web-Applikationen. Diese Unerfahrenheit resultierten in einigen Problemen.

Zum einen wurde die Projektstruktur nicht ausreichend geplant. Dieser Umstand resultierte in einer Projektstruktur, welche redundanten Code zur Folge hatte. In einem Rework wurde die gesamte Projektstruktur aufwändig überarbeitet. Das Resultat dieser Überarbeitung wurde bereits im Kapitel „3.1.1 Projektstruktur“ erläutert.

Ein weiteres Problem war die fehlende Einigkeit über Namenskonventionen. Diese hatte zur Folge, dass inkonsistente Namenskonventionen auf die Spaltenbennenung der Datenbank sowie auf Variablenbennenungen angewandt wurden. Dieser Umstand führte vermehrt zu Fehlern und machte ein weiteres Rework und ein Festlegen gemeinsam anzuwendender Namenskonventionen unabdingbar.

4.1.2 API-Probleme

Die Entwickler hatten zudem diverse Probleme mit der API.

Das erste Problem betraf die Spoonacular-API, welche Rezepte nicht richtig zurückgeliefert hat. Dieser Umstand fiel in einem Modultest auf, nach welchem Rezepte zurückgeliefert wurden, die fast keine der angegebenen Lebensmittel

benutzt haben. Auf Nachfrage der Entwickler an die Betreiber wurde bestätigt, dass es sich um ein Bug handelt, welcher sodann behoben wurde.[7]
Ein weiteres Problem war die schlechte Kommunikation mit den Betreibern der API. Auf E-Mails wurde erst nach mehreren Tagen bzw. Wochen geantwortet. Dies hat die Entwicklung der Web-Applikation deutlich verlangsamt.
Die Begrenzung der API-Anfragen stellte eine weitere Hürde dar. Diese wurde für einen kostenlosen Account auf insgesamt 150 Punkte pro Tag begrenzt[8]. Unterschiedliche Anfragen kosten eine unterschiedliche Anzahl von Punkten, beispielsweise kostet eine Anfrage nach einer Kochanleitung einen Punkt. Die Entwickler haben aufgrund dieser Limitierung beschlossen, Zugriff auf einen Studentenaccount zu erwerben. Dieser bietet 5000 Punkte pro Tag und kostete 10€ pro Monat. Allerdings musste dieser Account indirekt über RapidApi[9] benutzt werden. Aus diesem Grund interagiert die Web-Applikation nicht direkt mit der Spoonacular-API, sondern über die RapidApi. Diese bietet nicht alle Funktionalitäten der Spoonacular-API an. Z. b. bietet RapidAPI nicht an, Kochanleitungen anzufragen. Dieser Umstand wurde behoben, indem Anfragen nach ebendiesen an die Spoonacular-API gesendet werden.

4.2 Fazit

Die Entwickler hatten sich zum Ziel genommen, eine Web-Anwendung zu entwickeln, welche Nutzern Rezeptideen vorschlägt, die auf ihre Bedürfnisse abgestimmt sind.
Anhand der fertiggestellten Anwendung wurde gezeigt, dass diese nicht nur die initial an sie gestellten Anforderungen (vgl. Abschnitt 2.1) vollumfänglich erfüllt, sondern weit darüber hinausgeht (u. a. zeigt sie zusätzlich Nährwertangaben und Kochanweisungen an und bietet die Möglichkeit, eine Einkaufsliste auszudrucken).
Durch personalisierte Rezeptideen hilft das Programm insbesondere Menschen, welche durch Unverträglichkeiten oder bestimmte Lebensstile in ihrer Rezeptauswahl sehr stark eingeschränkt sind. Durch diverse Filter können diese schnell und einfach Ernährungspläne erstellen, welche frei von

[7] Ermittelt per E-Mail-Nachfrage.
[8] https://spoonacular.com/food-api/pricing
[9] https://rapidapi.com/spoonacular/api/recipe-food-nutrition/

Unverträglichkeiten auslösenden Zutaten sind bzw. welche mit ihrem Lebensstil vereinbar sind.
Auch Familien profitieren von *individumeal.com*: Durch die Eingabe der zu Hause vorhandenen Lebensmittel können schnell Rezeptideen ermittelt werden. Mittels übersichtlicher Diagramme kann entschieden werden, ob ein vorgeschlagenes Gericht gesund und nahrhaft ist. Im Zweifelsfall kann mittels der „Reroll"-Funktion ein alternatives Gericht angeboten werden.
Zudem helfen Funktionen wie das Ausdrucken der benötigten Lebensmittel, das Anzeigen von fehlenden Zutaten und das Anzeigen von Kochanweisungen dabei, das Kocherlebnis gesünder, schneller und einfacher zu gestalten.

4.3 Ausblick

Die entwickelte Web-Anwendung ist zurzeit nur im privaten Bereich einsetzbar. Um dies zu ändern, müssten weitere Sicherheitsvorkehrungen implementiert werden.
Insbesondere muss eine Validierung der E-Mail-Adresse des Nutzers bei der Registrierung (ähnlich der, die bereits im Prozess zum Zurücksetzen des Passworts verwendet wird) und eine Funktion zur Prävention von Brute-Force-Angriffen implementiert werden.
Des Weiteren müssten datenschutzrechtliche Aspekte berücksichtigt werden, zu welchen den Entwicklern das nötige Know-how zum Zeitpunkt der Entwicklung dieser Website nicht vorlag.
Im Hinblick auf die genannten Schwächen der Anwendung und der Unerfahrenheit der Entwickler im Umgang mit eben diesen, wird diese Website vorerst nicht der Öffentlichkeit zugänglich gemacht.

5 Literaturverzeichnis

Litke H-D (2018) Projektmanagement. Haufe-Lexware GmbH & Co. KG, Freiburg.

Statista (2021) Geschätzte Häufigkeit von Lebensmittelunverträglichkeiten in Deutschland 2014 | Statista. https://de.statista.com/statistik/daten/studie/314821/umfrage/geschaetzte-haeufigkeit-von-lebensmittelunvertraeglichkeit-in-deutschland/. Abruf am 2021-09-13.