



School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Bachelorarbeit (Informatik)

Foodsharing-App mit Nährwertberechnung

Autoren	Betim Kabashi Julien Wenger
Hauptbetreuung	Beat Seeliger
Datum	10.06.2022

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbstständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:

Zürich

09.06.2022

Name Studierende:

Betim Kabashi

Julien Wenger

Abstract (dt.)

Das Peer-to-Peer Geschäftsmodell wird heute in verschiedenen Branchen erfolgreich eingesetzt. In der Gastronomie fehlt jedoch eine Plattform, die ein Geschäft von Mahlzeiten zwischen Privatpersonen ermöglicht. Außerdem machen die heutigen verfügbaren Lieferplattformen keine Angaben über die Nährwerte der verfügbaren Gerichte.

Das Ziel dieser Arbeit ist es zu beantworten, ob durch das Analysieren und Aufbereiten von Originalrezepten eine Nährwertangabe für ein Gericht geschätzt werden kann und wie es in einer Foodsharing-App Verwendung finden kann. Dabei soll die Abweichung von der effektiven Nährwertangabe maximal 10 % betragen. Dazu wurden folgende Forschungsfragen gestellt: Wie können Nährwertangaben für Gerichte berechnet oder geschätzt werden, sodass die Abweichung der Schätzung nicht grösser als 10 % ist und wie wird der Prototyp realisiert, der zusätzlich zu den Hauptfunktionen (Anbieten/Abholen von Gerichten, Volltextsuche, Erstellen von Rezepten), Nährwertangaben für jedes Gericht machen kann?

Im ersten Schritt wurden Rezeptdaten aus verschiedenen Quellen transformiert und in eine einheitliche Form gebracht, um eine Durchschnittsberechnung der Zutaten respektive deren Mengen zu ermöglichen. Die daraus resultierenden Ergebnisse wurden als neues Rezept ins Backend-System des Prototyps importiert.

Für den Prototyp wurden nebst der Implementierung, Evaluationen von Technologien und Anforderungsanalysen für die Hauptfunktionen erstellt.

Das Resultat dieser Bachelorarbeit ist ein lauffähiger Prototyp, welcher die gewünschten Hauptfunktionen bietet und die geschätzte Nährwertangabe für jedes Gericht anzeigen kann. In Bezug auf die Genauigkeit der Mahlzeitschätzung wurde zwischen der Schätzung und der effektiven Nährwertangabe eine maximale Abweichung von 17 % Kilokalorien festgestellt.

Weiterführende Forschung oder Tätigkeiten in Bezug auf die Nährwertschätzung und den Prototyp könnten genauere Modelle für die Nährwertschätzung und Optimierungen der App sein.

Abstract (engl.)

The peer-to-peer business model is successfully used in various industries today. In the gastronomy sector, however, there is a lack of a platform that enables a business of dishes between private individuals. Furthermore, food delivery platforms available today do not provide information about the nutritional values of the available dishes.

The aim of this work is to answer whether a nutritional value for a dish can be estimated by analysing and processing original recipes and how it can be used in a food sharing app. The deviation from the actual nutritional value should be a maximum of 10 %. To this end, the following research questions were posed: How can nutritional information for dishes be calculated or estimated so that the deviation of the estimate is not greater than 10 % and how is the prototype realised that can provide nutritional information for each dish in addition to the main functions (offering/ordering dishes, full text search, creation of recipes)?

Recipes were first collected from various sources to estimate the nutritional values. The recipe data was then transformed and formatted, allowing for an average estimation of the respective ingredients and their quantities. The results were imported as a new recipe into the backend system of the prototype.

For the prototype, in addition to the implementation, evaluations of technologies and requirement analyses for the main functions were carried out.

The result of this bachelor thesis is an executable prototype that offers the desired main functions and can display the estimated nutritional value for each meal. Regarding the accuracy of the meal estimation, a maximum deviation of 17 % kilocalories was found between the estimation and the actual nutritional value.

Further research or activities related to nutrition estimation and the prototype could be more accurate models for nutrition estimation and optimisations of the app.

Vorwort

Diese Bachelorarbeit wurde in Zusammenarbeit mit meinem Mitstudenten Julien Wenger verfasst. Während er sich um das Frontend des Prototyps und die Anforderungsanalyse gekümmert hat, habe ich das Konzept der Mahlzeitberechnung und das Backend für den Prototyp entwickelt.

Die Idee des Prototyps ist zu einer Zeit entstanden, als ich das Kochen für mich entdeckt habe. Schnell wurde mir damals klar, dass meine Lebensmitteleinkäufe nicht nur für eine Portion reichten und zudem ich gerne Feedback zu meinen Zubereitungen bekommen möchte. Aus diesen Gründen ist die Idee zum Foodsharing entstanden.

An dieser Stelle möchten wir uns bei unserem Betreuer, Beat Seeliger, für die Möglichkeit, eine eigene Idee zu realisieren, und für seine Unterstützung während des Arbeitsprozesses bedanken. Des Weiteren möchten wir der Zweitbetreuerin, Alicia Rüegg, für letzte Tipps und das Bewerten dieser Arbeit danken.

Wir wünschen Ihnen viel Spass beim Lesen dieser Bachelorarbeit.

Betim Kabashi & Julien Wenger

Zürich, 09.06.2022

Inhaltsverzeichnis

1.	Einleitung	9
1.1.	Ausgangslage	9
1.2.	Zielsetzung	9
1.3.	Gliederung	10
2.	Theoretische Grundlagen	11
2.1.	Aspekte der Nährwerte	11
2.1.2.	Kalorien (Energie/Brennwert)	11
2.1.3.	Fette	12
2.1.4.	Kohlenhydrate	12
2.1.5.	Proteine (Eiweiss)	12
2.2.	Mathematische Grundlagen der Kalorienberechnung	13
2.2.1.	Nährwertveränderungen durch Kochen	13
2.2.2.	Durchschnittsberechnung anhand n Rezepten	13
3.	Methodik	14
3.1.	Datenaufbereitung und Mahlzeitberechnung	14
3.1.1.	Rahmenbedingungen	14
3.1.2.	Datenaufbereitung	14
3.1.3.	Berechnung der Kilotkalorien (kcal) anhand des Basisdatensatzes	16
3.1.4.	Vorgehen in der Mahlzeitschätzung	17
3.2.	Anforderungsanalyse des Prototyps	18
3.2.1.	Rahmenbedingungen	18
3.2.2.	User-Stories	18
3.2.3.	Abholungsprozess	22
3.2.4.	Angebotsprozess	23
3.3.	Eingesetzte Software Frontend	24
3.3.1.	Auswahl	24
3.3.2.	Deployment und Hosting	24
3.4.	Technische Evaluation Backend	25
3.4.1.	Einschlusskriterien	25
3.4.2.	Ansatz 1: Backend-as-a-Service	25

3.4.3.	Ansatz 2: Platform-as-a-Service (PaaS)	26
3.4.4.	Entscheidungsmatrix.....	27
3.4.5.	Ergebnis Entscheidungsmatrix.....	28
3.5.	Realisierung des Backend mit Firebase	28
3.5.1.	Cloud Firestore vs. Realtime Database	28
3.5.2.	Erstellung der Datenstrukturen mittels Collections	28
3.5.3.	Einbindung von Algolia für die Volltextsuche	30
3.5.4.	Wiederverwendbarkeit	31
3.5.5.	Algolia – Rezepte mit Volltextsuche	32
4.	Resultate	33
4.1.	Genauigkeit der Mahlzeitschätzung	33
4.2.	Prototyp.....	34
4.2.1.	Backend: Collections	34
4.2.2.	Backend: Speicher	38
4.2.3.	Frontend: Beschreibung der Screens	39
5.	Diskussion	48
5.1.	Mahlzeitschätzung	48
5.2.	Ausbaumöglichkeiten des Prototyps	48
5.2.1.	ETL-Prozess für die Datenaufbereitung	48
5.2.2.	Einbezug von benutzerdefinierten Rezepten in die Durchschnittsberechnung	49
5.2.3.	Optimierung Volltextsuche	49
5.2.4.	Chatsystem.....	49
5.2.5.	Zugeschnittene Angebote	49
5.2.6.	Kartenfunktion	49
5.2.7.	Zahlungssystem.....	49
5.2.8.	Go-Live-Bedingungen.....	50
6.	Fazit und Ausblick	51
7.	Verzeichnisse	52
7.1.	Literaturverzeichnis.....	52
7.2.	Abbildungsverzeichnis.....	53
7.3.	Tabellenverzeichnis.....	53

8.	Anhang.....	54
8.1.	Projektmanagement.....	54
8.1.1.	Aufgabenstellung	54
8.2.	Weiteres	54
8.2.1.	Quellcode	54

1. Einleitung

1.1. Ausgangslage

Durch digitale Plattformen wie Airbnb, Uber und Ebay ist es heute möglich, Geschäfte oder Dienstleistungen zwischen zwei oder mehreren Privatpersonen abzuschliessen, wobei die Plattform an sich als Vermittler fungiert. Dieses Geschäftskonzept wird als Peer-to-Peer-Modell bezeichnet und vor allem in der Beherbergungs- sowie der Transportbranche (Airbnb, Uber) erfolgreich eingesetzt. Auch in der Gastronomiebranche existieren entsprechende Plattformen, beispielsweise Uber Eats oder TakeAway.ch, jedoch wird hier ein Geschäft zwischen einer Firma bzw. einem Restaurant und einer Privatperson vermittelt. Zum heutigen Zeitpunkt existiert keine Plattform, auf der eine Privatperson die Möglichkeit erhält, ihre eigens zubereitete Mahlzeit anzubieten.

Ausserdem besteht in der Schweiz kein Essenslieferdienst, der Nährwertangaben über die zur Verfügung gestellten Gerichten enthält. Diese zusätzliche Information kann allerdings insbesondere für ernährungsbewusste Benutzer einen Mehrwert bieten. Dabei ist es nicht essenziell, dass die Angaben zu 100 % genau sind, sondern sie sollten vielmehr als Richtwert verwendet werden können, um sich insgesamt gesünder zu ernähren.

1.2. Zielsetzung

Mit einer Foodsharing-Plattform sollen Privatpersonen die Möglichkeit erhalten, ihre selbst gekochten Mahlzeiten anzubieten. Dabei können sie ihre Mahlzeit gratis oder für einen geringen Preis zur Abholung bereitstellen. Die Plattform soll zusätzlich die Hausmannskost fördern, indem die Konsumenten den Koch bewerten können und so eine Community von Köchen sowie Essensliebhabern entsteht. Zu jedem der angebotenen Gerichte sollen Nährwertangaben vorliegen, damit der Konsument über die Kilokalorienanzahl sowie den Anteil von Eiweiss, Fetten sowie Kohlenhydraten informiert wird. Sicherzustellen ist dennoch, dass für den Koch kein grosser Mehraufwand generiert wird, wenn er seine Mahlzeit anbieten möchte. Mit anderen Worten soll der Kochprozess für den Koch so natürlich wie möglich und ungezwungen bleiben, weshalb kein Zwang entstehen sollte, jede einzelne Zutat abzuwiegen. Aus diesem Grund soll eine Rezeptdatenbank erstellt werden, die vordefinierte Rezepte inklusive Nährwertangaben enthält.

Das Ziel dieser Arbeit ist es folgende zwei Fragen zu beantworten: Wie können Nährwertangaben für Gerichte berechnet oder geschätzt werden, sodass die Abweichung der Schätzung nicht grösser als 10 % ist und wie wird der Prototyp realisiert, der zusätzlich zu den Hauptfunktionen (Anbieten/Abholen von Gerichten, Volltextsuche), Nährwertangaben für jedes Gericht machen kann?

Die Abweichung von 10 % entsprechen dem maximalen Anteil an Kalorien, die der Körper, während der Nahrungsaufnahme nicht verwerten kann, sondern durch den Darm wieder ausgeschieden wird [1].

Die offizielle Aufgabestellung ist unter [7.1.1](#) zu finden.

1.3. Gliederung

Zu Beginn des Dokuments werden dem Leser in Kapitel zwei eine Einführung in Nährwerte und deren Berechnung sowie eine Erläuterung über regulatorische Bestimmungen in Bezug auf Lebensmittel geboten.

Kapitel drei ist im ersten Teil der Datenbeschaffung sowie der Datenaufbereitung für den Prototyp bzw. der Mahlzeitberechnung für Rezepte gewidmet. Im zweiten Teil wird auf die Realisierung des Prototyps eingegangen. Hier ist nebst Anforderungsanalysen und Evaluationen der Technologien auch die konkrete Realisierung des Prototyps relevant, aufgeteilt in Front- und Backend sowie das Aufsetzen der Suchmaschine für Rezepte.

In Kapitel vier wird zuerst auf die Genauigkeit der Mahlzeitberechnung eingegangen. Danach wird der realisierte Prototyp präsentiert. Relevant sind unter anderem die erstellten Application Programming Interfaces (APIs) mittels Collections und das Graphical User Interface (GUI) des Prototyps in Form einer Mobile-App.

Im letzten Teil dieser Arbeit werden auf Grundlage der realisierten Lösung und der Mahlzeitberechnung Optimierungsmöglichkeiten vorgestellt. Außerdem werden in diesem Kapitel Aspekte und Funktionen angeführt, die umzusetzen sind, um den Prototyp auf den Markt zu bringen.

2. Theoretische Grundlagen

In diesem Kapitel werden die grundlegenden Informationen rund um das Thema der Nährwertberechnung erläutert.

2.1. Aspekte der Nährwerte

2.1.1. Überblick

Um eine Nährwertmittelberechnung durchzuführen, muss zuerst ihre Basis geschaffen und verstanden werden. Als in dieser Arbeit relevante Informationen werden zunächst die wesentlichen Nahrungsbestandteile erläutert. Es wird nachfolgend somit auf die sogenannten „Grossen vier“ eingegangen [2]. Zu diesen zählen die Energie/der Brennwert sowie Fett, Kohlenhydrate und Eiweiss. Dabei handelt es sich um sogenannte Makronährstoffe. Sie bilden den grössten Teil der Nahrungsaufnahme und zugleich die Basis der späteren Nährmittelberechnung.

Durchschnittliche Nährwerte pro 100 g	
Energie	989 kJ/233 kcal
Fett	0,2 g
davon gesättigte Fettsäuren	<0,1 g
Kohlenhydrate	55 g
davon Zucker	55 g
Eiweiß	0,7 g
Salz	0,03 g

Abbildung 1: Nährwertangaben [3]

In Abbildung eins ist ein Beispiel dafür ersichtlich, wie für ein Lebensmittel pro 100 Gramm die Nährwerte abgebildet werden. Neben den gängigen Nährwerten wie Energie, Fett, Kohlenhydraten und Eiweiss werden hier auch weitere Komponenten aufgelistet, die seit der EU-Verordnung 1169/2011 [2] obligatorisch zu nennen sind.

2.1.2. Kalorien (Energie/Brennwert)

Kalorien werden als Mass der Energiemenge verstanden, wobei diese im Fall von Lebensmitteln meist in Kalorien und in Joule angegeben wird. Eine Kalorie entspricht etwa 4.2 Joule. Die meiste Energie liefern die drei Makronährstoffe Fette, Kohlenhydrate und Eiweisse, deshalb folgen ihre Energiewerte für jeweils ein Gramm [4]:

- Fette enthalten pro Gramm 37,8 kJ bzw. 9 kcal.
- Kohlenhydrate enthalten pro Gramm 16,8 kJ bzw. 4 kcal.
- Eiweisse enthalten pro Gramm 16,8 kJ bzw. 4 kcal.

2.1.3. Fette

Fette stellen die energiereichsten Nährstoffe und somit bedeutende Mittel der Energieversorgung sowie -speicher dar. Sie werden in gesättigte, einfach ungesättigte und mehrfach ungesättigte Fettsäuren unterschieden. Während gesättigte Fettsäuren vorwiegend in tierischen Produkten vorkommen, treten die einfach und mehrfach ungesättigten Fettsäuren hauptsächlich in pflanzlichen Produkten auf. Des Weiteren sind Fette Träger von Geschmacks- und Aromastoffen sowie zuständig für die Aufnahme von fettlöslichen Vitaminen [5].

2.1.4. Kohlenhydrate

Kohlenhydrate gelten als Saccharide, wobei dieser Begriff im Griechischen „Zucker“ bedeutet. Kohlenhydrate setzen sich aus den chemischen Elementen Kohlenstoff, Wasserstoff und Sauerstoff zusammen. Sie werden anhand der Anzahl der Baustoffe in verschiedene Gruppen eingeteilt, nämlich in Einfach-, Zweifach-, Mehrfach- und Vielfachzucker. Mehrfachzucker liegen bei einer Anzahl von drei bis neun Zucker-Baustoffen vor. Beim Vielfachzucker beträgt die Anzahl mindestens zehn Zucker-Baustoffen [5].

2.1.5. Proteine (Eiweiss)

Eiweisse versorgen den Körper mit Aminosäuren und Stickstoff. Es wird zwischen 20 verschiedenen Aminosäuren unterschieden, die für die körpereigene Proteinsynthese zuständig sind. Die Aminosäure ist der kleinste Baustoff, der die Zusammensetzung der Proteine prägt [5].

2.2. Mathematische Grundlagen der Kalorienberechnung

Die Voraussetzung dessen, diese Berechnung genau durchzuführen, besteht darin, die Mengenangabe jeder einzelnen Zutat in einem Gericht zu kennen. Dies führt zum Problem, dass ein Koch während der Zubereitung jede Zutat abwiegen und sich strikt an die Mengen im Rezept halten muss. Um diesem Problem entgegenzuwirken, wird in Kapitel 3.1 ein Ansatz vorgestellt, bei dem die Berechnung der Kalorienanzahl auf Schätz- und Mittelwerten basiert und auf das fertige, gekochte Gericht angewendet werden kann.

2.2.1. Nährwertveränderungen durch Kochen

„Der Nährstoffgehalt eines gekochten Lebensmittels oder zusammengesetzten Gerichtes kann entweder analysiert oder berechnet werden. Analysieren hat den Vorteil, dass es genaue Werte liefert. Analysen sind jedoch anspruchsvoll, aufwändig und teuer. In der Praxis wird der Nährstoffgehalt eines zubereiteten Produktes deshalb meist mit Hilfe von durchschnittlichen Verlust- und Aufnahmefaktoren berechnet. Dies gilt auch für die Schweizer Nährwertdatenbank. Beim Berechnen des Nährstoffgehaltes eines zubereiteten Lebensmittels müssen zusätzlich zu den Nährstoffverlusten auch allfällige Gewichtsveränderungen berücksichtigt werden. Nimmt das Gewicht ab, entspricht dies einer Konzentration. Der Gehalt an Vitaminen und Mineralstoffen pro 100 Gramm eines gekochten Lebensmittels kann deshalb trotz Nährstoffverlusten höher ausfallen als jener des entsprechenden rohen Produktes. Umgekehrt entspricht eine Gewichtszunahme (z. B. beim Kochen von Teigwaren) einer Verdünnung und der Nährstoffgehalt pro 100 Gramm des gekochten Lebensmittels ist deutlich geringer als jener des ungekochten trockenen Lebensmittels.“ [6]

2.2.2. Durchschnittsberechnung anhand n Rezepten

Da sich die geschätzte Nährwertangabe auf ein gekochtes Gericht beziehen soll und sich das Gewicht von Zutaten nach dem Kochprozess erhöhen (wegen der Aufnahme von Flüssigkeit) oder verringern (wegen des Verlusts von Flüssigkeit) kann, muss der Änderungsfaktor hinsichtlich Gewichts für jede Zutat bekannt sein. Ist der Änderungsfaktor einer Zutat grösser als eins, bedeutet dies eine Gewichtsabnahme während des Kochprozesses. Liegt hingegen ein Änderungsfaktor kleiner als eins vor, erfolgt eine Gewichtszunahme. Mit diesem Änderungsfaktor werden die Kalorienangaben in den n Rezepten angepasst. Im zweiten Schritt werden die Rezepte vergleichbar gemacht, indem die Gesamtmenge auf eine Basis (auf 100 Gramm) umgerechnet wird. Als Nächstes wird die Durchschnittsmenge für jede Zutat über die n Rezepte hinweg berechnet. Somit wird ein weiteres Rezept erzeugt, welches Durchschnittsmengenangaben enthält, von denen ausgehend die Kalorien berechnet werden können. In Kapitel drei werden die Berechnung, ihre Genauigkeit und die Realisierung dieser Idee im Kontext des Prototyps beschrieben.

3. Methodik

Das Vorgehen wird in zwei Teilbereiche aufgeteilt. Der erste bezieht sich auf die Nährwertberechnung, die hier im Detail präsentiert wird. Der zweite beinhaltet das Vorgehen, die Konstruktion und die Realisierung des Prototyps einer Smartphone-Applikation.

3.1. Datenaufbereitung und Mahlzeitberechnung

Wie in den theoretischen Grundlagen aus Kapitel zwei angedeutet wird in diesem Unterkapitel die Realisierung der Mahlzeitberechnung beschrieben.

3.1.1. Rahmenbedingungen

Da es keine vorgefertigten Datensätze gibt und ein ETL-Prozess (der Extraktion, Transformation und Laden umfasst) kein Teil dieser Arbeit ist, wurde der initiale Datenimport der Gerichte manuell durchgeführt. Das Zusammenführen der Rezepte und die Durchschnittsberechnung erfolgten mittels JavaScript-Abfragen auf die Datenbank. Aus diesem Grund enthält der Prototyp initial nur zehn vordefinierte Rezepte.

3.1.2. Datenaufbereitung

Zutaten und Nährwerte

Um Zutaten und deren Nährwerte abzurufen, wurde die Schweizer Nährwertdatenbank vom Bundesamt für Lebensmittelsicherheit und Veterinärwesen verwendet. Diese Datensammlung enthält 129 Felder je Zutat, wobei im Rahmen der vorliegenden Arbeit nur die nötigen Nährwertangaben, nämlich Kilokalorienanzahl sowie Protein-, Fett- und Kohlenhydratmenge importiert wurden. Außerdem wurde in der Datensammlung für einen Grossteil der Zutaten zwischen dem rohen und dem zubereiteten (gekochten, gebratenen, geschmorten) Zustand unterschieden. Durch diese Differenzierung konnte der Änderungsfaktor wie in Unterkapitel 2.2 erwähnt berechnet werden, indem die Kalorienanzahl der zubereiteten Zutat durch die Kalorienanzahl der rohen Zutat geteilt wurde. In der finalen Dokumentendatenbank wird diese Unterscheidung aufgehoben und nur der Datensatz mit den rohen Angaben plus dem Änderungsfaktor verwendet, zum Beispiel wie folgt:

ID	Zutat	Kcal	Fett	Kohlenhydrate	Protein		ID	Zutat	Kcal	Fett	Kohlenhydrate	Protein	Änderungsfaktor
1062	Teigwaren mit Ei, gekocht im Salzwasser (unjodiert)	132	1.1	25	5		799	Teigwaren mit Ei	365	2.8	69.9	13.3	0.36
799	Teigwaren mit Ei, trocken	365	2.8	69.9	13.3								

Abbildung 2: Zutat roh/gekocht, Änderungsfaktor

Wie in Abbildung zwei grün markiert zusätzlich zu erkennen ist, entfällt im Namen der Zutaten die Zusatzinformation, da bei Rezeptangaben immer vom Rohzustand ausgegangen wird und die Unterscheidung zwischen dem rohen sowie dem zubereiteten Zustand in der finalen Tabelle nicht mehr notwendig ist.

Rezepte

Mit Hilfe des Webs wurden für jedes Gericht zwei bis drei Rezepte aus verschiedenen Quellen gesucht. Im zweiten Schritt wurden Messgrößen wie EL (Esslöffel) und TL (Teelöffel) mit Hilfe einer Masseinheitstabelle von chefkoch.de in Gramm umgewandelt. Um die Rezepte vergleichbar zu machen, wurde die Gesamtmenge auf die gleiche Basis (100 Gramm) umgerechnet. Dabei wurden Zutaten wie Gewürze, Kräuter und kalorienarme Flüssigkeiten ignoriert, da deren Kaloriengehalt nur eine geringe bis gar keine Auswirkung auf das Resultat zeigt. Zusätzlich wurden Zutaten für ein Gericht im finalen Datensatz, als optional markiert, wenn diese in der Mehrheit der Rezepte nicht vorkommen. Optionale Zutaten sind standardmäßig nicht Teil des Rezepts, können im Prototyp jedoch angewählt werden. Zum Schluss wurde der Mittelwert für jede Zutat über alle verfügbaren Rezepte hinweg kalkuliert. Das Ergebnis ist ein neuer Datensatz bzw. ein Rezept, welches Durchschnittswerte der Mengen enthält und die Basis für die Kalorienberechnung gekochter Gerichte darstellt. In Unterkapitel 4.1 wird auf die Berechnung und deren Genauigkeit gegenüber der effektiven Kalorienanzahl eingegangen.

In der nachfolgenden Grafik wird der beschriebene Prozess an einem Beispielrezept für „Ghackets mit Hörnli“ veranschaulicht.

Rezept 1	Menge in Gramm	Rezept 2	Menge in Gramm	Rezept 3	Menge in Gramm
Höqli	125	Höqli	125	Höqli	100
Hackfleisch (Rin)	125	Hackfleisch (Rin)	125	Hackfleisch (Rin)	125
Zwiebel	21	Zwiebel	21	Zwiebel	16
Sellerie	25	Sellerie	25	Tomatenpüree	3.8
Karotte	16	Karotte	16	Mehl	4
Tomatenpüree	11.3	Tomatenpüree	11.3	Rotwein	25
Rotwein	70	Rotwein	63	Reibkäse	25
Fleischbouillon		Fleischbouillon		Fleischbouillon	
Paprika		Paprika		Paprika	
Lorbeerblatt		Lorbeerblatt		Knoblauch	
Thymianblatt		Thymianblatt		Majoranzweig	
Salz		Salz		Thymianzweig	
Pfeffer		Pfeffer		Salz	
Gesamtmenge	393.3	Gesamtmenge	386.3	Gesamtmenge	298.8
Faktor auf 100g:	0.254258835		0.258866166		0.334672021
Höqli	31.7823544	Höqli	32.35827077	Höqli	33.46720214
Hackfleisch (Rin)	31.7823544	Hackfleisch (Rin)	32.35827077	Hackfleisch (Rin)	41.83400268
Zwiebel	5.33943555	Zwiebel	5.43618949	Zwiebel	5.354752343
Sellerie	6.35647089	Sellerie	6.471654155	Tomatenpüree	1.271753681
Karotte	4.06814137	Karotte	4.141858659	Mehl	1.338688086
Tomatenpüree	2.87312484	Tomatenpüree	2.925187678	Rotwein	8.366800535
Rotwein	17.7981185	Rotwein	16.30856847	Reibkäse	8.366800535
Fleischbouillon		Fleischbouillon		Fleischbouillon	
Paprika		Paprika		Paprika	
Lorbeerblatt		Lorbeerblatt		Knoblauch	
Thymianblatt		Thymianblatt		Majoranzweig	
Salz		Salz		Thymianzweig	
Pfeffer		Pfeffer		Salz	
Gesamtmenge	100	Gesamtmenge	100	Gesamtmenge	100
Höqli	32.54			Legende	
Hackfleisch (Rin)	35.32			Gewürze, ignoriert	
Zwiebel	5.38			Optionale Zutaten	
Sellerie	6.41				
Karotte	2.74				
Tomatenpüree	2.36				
Rotwein	14.16				
Mehl	1.3				
Reibkäse	8.4				
Fleischbouillon					
Paprika					
Lorbeerblatt					
Thymianblatt					
Salz					
Pfeffer					
Gesamtmenge	~100				

Tabelle 1: Prozess Mahlzeitschätzung

1: Rezepte pro Portion, alle Mengenangaben in Gramm und Markierung von Gewürzen sowie kalorienarmen Flüssigkeiten

2: Umrechnung auf die gleiche Basis von 100 Gramm

3: finaler Basisdatensatz

3.1.3. Berechnung der Kilokalorien (kcal) anhand des Basisdatensatzes

Die Formel für die Berechnung der Gesamtkalorien einer gekochten Mahlzeit lautet:

$$\sum_{i=1}^n \frac{Z * x_i * y_i * c_i}{100 * 100}$$

z: abgewogene Gesamtmenge des Gerichts in Gramm

xi: Gewicht in Gramm von Zutat i

yi: Kilokalorien von Zutat i

ci: Änderungsfaktor von Zutat i

3.1.4. Vorgehen in der Mahlzeitschätzung

Um die Genauigkeit der Mahlzeitschätzung zu überprüfen, müssen die Nährwertangaben der ursprünglichen Rezepte mit denen der Mahlzeitschätzung verglichen werden. Dabei tritt folgendes Problem auf: Die Portionsgrößen sind für jedes Rezept unterschiedlich und das Gesamtgewicht der gekochten Portion ist im Vorhinein nicht bekannt. Um dieses Problem zu lösen, müsste jedes Rezept nachgekocht und das Endprodukt abgewogen werden. Dies wäre im Rahmen der vorliegenden Arbeit zu aufwendig, darum wurde folgendes Verfahren angewendet:

Mit Hilfe des Änderungsfaktors (siehe Abschnitt [2.2.2](#)) wurde die Menge für jede Zutat im Originalrezept neu berechnet und anschliessend summiert. Die erhaltene Menge entspricht dem Gesamtgewicht des Rezeptes nach dem Kochprozess. Somit kann ein Originalrezept näherungsweise in eine gekochte Mahlzeit umgewandelt und der Vergleich mit der Mahlzeitschätzung möglich gemacht werden. Der Vergleich an sich entspricht einer Dreisatzrechnung. Zur Veranschaulichung wird dieses Konzept nachfolgend an einem Rezept erklärt.

Ghackets mit Hörnli Orginalrezept 1	Zutaten	Rohgewicht in Gramm	Änderungsfaktor		Gewicht (gekocht)
	Hörnli	125	0.36		347.2
	Hackfleisch (Rind)	125	1.37		91.2
	Zwiebel	21	1.1		19.1
	Sellerie	25	1		25
	Karotte	16	1.1		14.5
	Tomatenpüree	11.5	1	$\frac{\text{Rohgewicht}}{\text{Änderungsfaktor}} =$	11.5
	Rotwein	63	1		63
	Fleischbouillon	0	1		0
	Paprika	0	1		0
	Lorbeerblatt	0	1		0
	Thymianblatt	0	1		0
	Salz	0	1		0
	Pfeffer	0	1		0
Gesamtgewicht		387			572

Mahlzeitschätzung Ghackets mit Hörnli	Nährwerte (100g)	Nährwerte pro Portion
Kohlenhydrate	8	45.76
Eiweiss	12	68.64
Fett	4	22.88
Kcal	135	772.2

Tabelle 2: Beispiel des Vergleichs von Schätzung & Originalrezept

3.2. Anforderungsanalyse des Prototyps

In diesem Unterkapitel werden die Rahmenbedingungen, d. h. die User-Stories des Prototyps, festgelegt.

3.2.1. Rahmenbedingungen

Entwickler	2
Zielgruppe	Privatpersonen, die Gerichte zur Verfügung stellen möchten oder nach verfügbaren Gerichten suchen und diese bestellen wollen
Endgeräte	Alle gängigen Smartphones ab 2021 Der Prototyp soll auf allen Android-Geräten mit mindestens der Android-Version 9.0 zuverlässig und stabil funktionieren [Stand: 6. August 2018]. Der Prototyp soll auf allen iPhone-Geräten mit mindestens der iOS-Version 15.1 zuverlässig und stabil funktionieren [Stand: 25. Oktober 2021].
Software	Für die Umsetzung des Frontend wird React Native eingesetzt.

Tabelle 3: Rahmenbedingungen (Prototyp)

3.2.2. User-Stories

Titel	Anzeige der Gerichte
User-Story-ID	US-1
Beschreibung	Konsumenten möchten unter der Rubrik «Abholen» zufällige Gerichte sehen können, damit sie aussuchen können.
Akzeptanzkriterien	<ul style="list-style-type: none">• Folgende Informationen werden angezeigt:<ul style="list-style-type: none">○ Das Bild des Gerichts○ Der Name des Gerichts○ Der Verfasser des Gerichts○ Die Entfernung○ Wann das Gericht abholbereit ist○ Das Profilbild des Verfassers• Der Konsument kann auf ein Gericht klicken, um seine Detailansicht aufrufen zu können.

Tabelle 4: User-Story 1 – Anzeige der Gerichte

Titel	Detailansicht des Gerichts
User-Story-ID	US-2
Beschreibung	Der Konsument möchte eine Detailansicht des Gerichts anklicken können, damit er detailliertere Angaben zu diesem erhält.
Akzeptanzkriterien	<ul style="list-style-type: none"> • Folgende Informationen werden angezeigt: <ul style="list-style-type: none"> ○ Das Bild des Gerichts ○ Der Name des Gerichts ○ Der Verfasser des Gerichts ○ Das Profilbild des Verfassers ○ Wann das Gericht abholbereit ist ○ Zutaten ○ Die berechneten Nährwerte • In der Detailansicht hat der Konsument die Möglichkeit, das Gericht zu bestellen.

Tabelle 5: User-Story 2 – Detailansicht des Gerichts

Titel	Bestellvorgang
User-Story-ID	US-3
Beschreibung	Der Konsument möchte ein Gericht bestellen, damit er es abholen kann.
Akzeptanzkriterien	<ul style="list-style-type: none"> • Folgende Informationen werden angezeigt: <ul style="list-style-type: none"> ○ Das Bild des Gerichts ○ Der Name des Gerichts • Nach dem Bestellvorgang kann der Konsument das Gericht schliesslich abholen.

Tabelle 6: User-Story 3 – Bestellvorgang

Titel	Gerichte suchen
User-Story-ID	US-4
Beschreibung	Die Persona möchte nach Gerichten suchen können, damit sie ihr Wunschgericht finden kann.
Akzeptanzkriterien	<ul style="list-style-type: none"> • Es soll eine Eingabemaske geben, über die der Konsument mittels Volltextsuche nach Gerichten suchen kann. • Es besteht die Möglichkeit, nach dem Namen eines Gerichts oder nach Gerichten zu suchen, die eine bestimmte Zutat enthalten. • Bei der Suche sollen die Gerichte aufgelistet werden, nach denen der Konsument gesucht hat.

Tabelle 7: User-Story 4 – Gerichte suchen

Titel	Auflistung der bereits getätigten Bestellungen
User-Story-ID	US-5
Beschreibung	Der Konsument möchte eine Auflistung seiner bereits getätigten Bestellungen einsehen können, damit er nachschlagen kann, was er schon bestellt hat.
Akzeptanzkriterien	<ul style="list-style-type: none"> • Es soll eine Auflistung aller bereits getätigten Bestellungen angezeigt werden können. • Bei den bereits bestellten Gerichten werden folgende Informationen angezeigt: <ul style="list-style-type: none"> ◦ Das Bild des Gerichts ◦ Der Name des Gerichts ◦ Der Verfasser des Gerichts ◦ Das Profilbild des Verfassers

Tabelle 8: User-Story 5 – Auflistung der bereits getätigten Bestellungen

Titel	Profilanzeige
User-Story-ID	US-6
Beschreibung	Die Persona möchte ihr Profil aufrufen können, damit sie auf ihre persönlichen Daten zugreifen kann.
Akzeptanzkriterien	<ul style="list-style-type: none"> • Die Personas Koch und Konsument können auf ihr jeweiliges Profil zugreifen. • Es werden die wesentlichen persönlichen Daten angezeigt. • Der Koch kann zudem einsehen, wie gut seine Bewertung ausfällt.

Tabelle 9: User-Story 6 – Profilanzeige

Titel	Gerichte anbieten
User-Story-ID	US-7
Beschreibung	Der Koch möchte ein Gericht anbieten können, damit Konsumenten die Möglichkeit erhalten, dieses zu bestellen.
Akzeptanzkriterien	<ul style="list-style-type: none"> Der Koch kann von den bereits verfügbaren Gerichten das passende auswählen und anbieten.

Tabelle 10: User-Story 7 – Gerichte anbieten

Titel	Gerichte erfassen
User-Story-ID	US-8
Beschreibung	Der Koch möchte ein Gericht erfassen können, das es in der Datenbank noch nicht gibt, um ein eigenes anzubieten.
Akzeptanzkriterien	<ul style="list-style-type: none"> Der Koch kann dem zu erfassenden Gericht einen Namen geben. Der Koch kann anhand der Datenbank die jeweilige Zutat, deren Menge und die Masseneinheit auswählen.

Tabelle 11: User-Story 8 – Gerichte erfassen

Titel	Koch bewerten
User-Story-ID	US-9
Beschreibung	Der Konsument möchte nach dem Bestellvorgang den jeweiligen Koch des Gerichts bewerten können, damit andere sehen können, wie gut der Koch ist.
Akzeptanzkriterien	<ul style="list-style-type: none"> Der Konsument kann den Koch nach dem Bestellvorgang bewerten.

Tabelle 12: User-Story 9 – Koch bewerten

3.2.3. Abholungsprozess

In der nachfolgenden Abbildung wird der Prozess der Abholung einer Mahlzeit aufgezeigt. Hervorzuheben gilt es hierbei, dass der genaue Abholungsstandort erst bei erfolgreicher Annahme der Anfrage freigegeben wird.

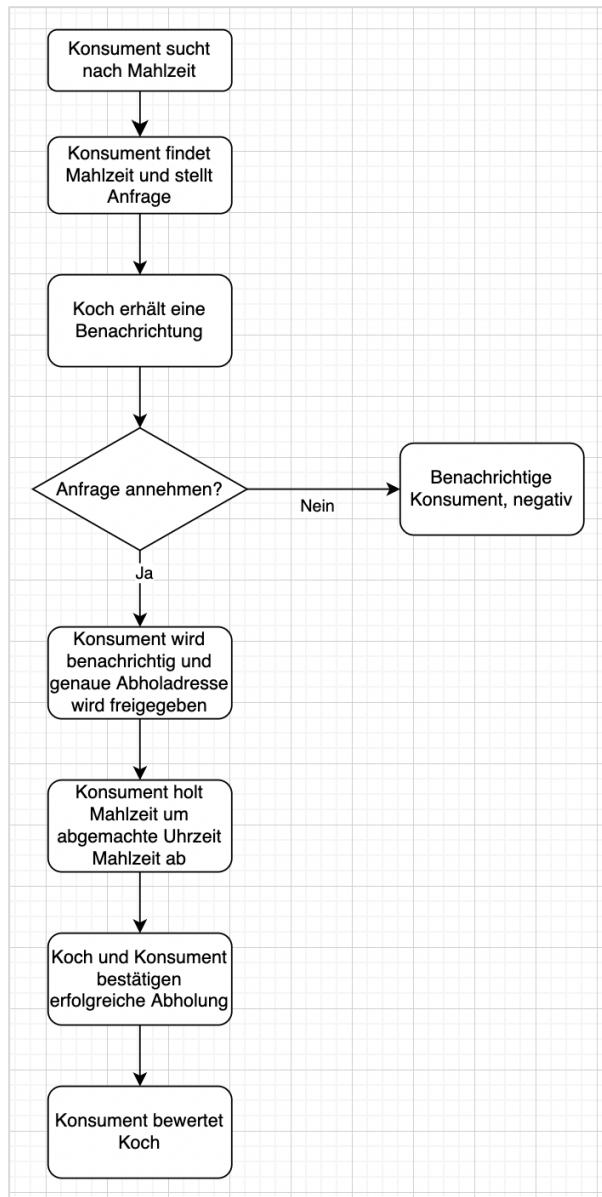


Abbildung 3: Abholungsprozess

3.2.4. Angebotsprozess

Der Angebotsprozess umfasst die Tätigkeit, die durchlaufen wird, wenn ein Koch eine Mahlzeit anbieten möchte. Dabei kann es vorkommen, dass die Mahlzeit noch nicht als Rezept in der Datenbank hinterlegt ist. In diesem Fall muss der Koch ein neues Rezept erstellen. Die Liste der auszuwählenden Zutaten ist vordefiniert. Bei der Mengenangabe hat der Koch die Möglichkeit, auf Masse zurückzugreifen – zum Beispiel als eine Scheibe Brot oder ein Esslöffel Öl. Im Hintergrund wird die Mengenangabe in Gramm umgewandelt. Dies ist im Rahmen des Prototyps momentan nur für wenige Zutaten möglich. Bei den restlichen Zutaten muss vorerst die Menge in Gramm genannt werden.

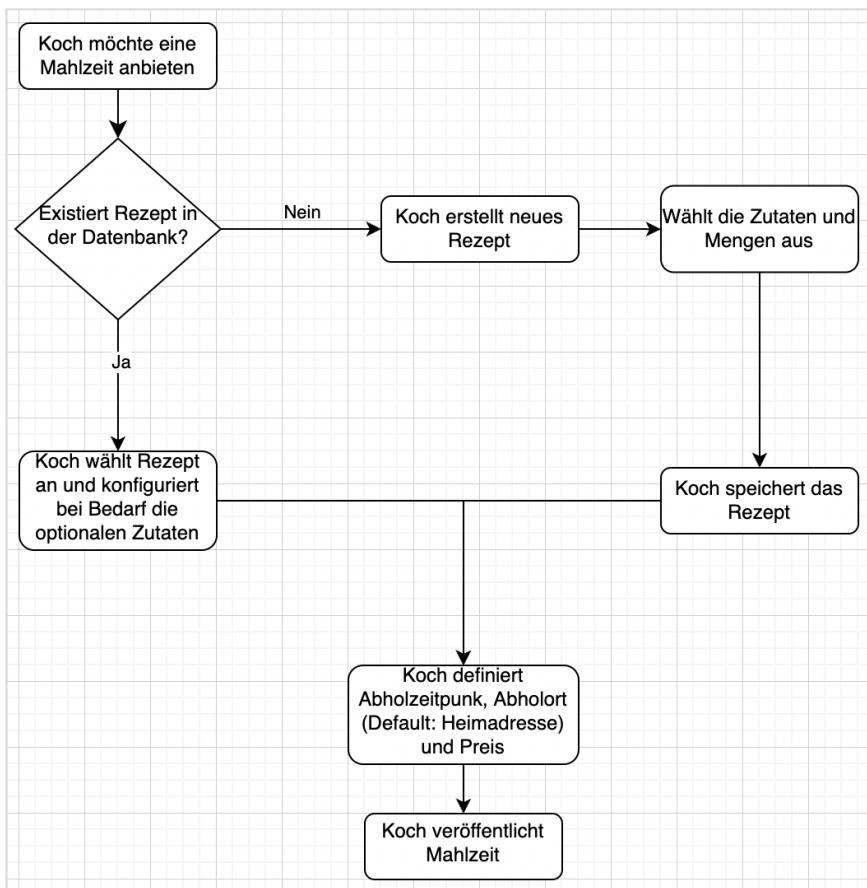


Abbildung 4: Angebotsprozess

3.3. Eingesetzte Software Frontend

3.3.1. Auswahl

Für die Realisierung des Prototyps seitens Frontend wird React Native herangezogen. Sein grosser Vorteil besteht darin, dass ein Framework für mehrere Plattformen verwendet werden kann. So muss kein nativer Code einzeln für Android sowie iOS programmiert werden. Vielmehr reicht eine Code-Basis aus, um für diese Plattformen und auch Webanwendungen zu entwickeln. React Native bietet mittlerweile viele User Interfaces (UI)-Komponenten und Funktionen für die Entwicklung von nativen Anwendungen an, sodass gängige Prozesse in einer App abgedeckt werden. Dadurch kann ein hoher Grad an Wiederverwendbarkeit erreicht werden [7].

Für die Entwicklung des Prototyps wurde Node.js lokal installiert und react-native als Node-Modul eingerichtet. Zudem wurde Expo verwendet, das als Webdienst dient, um den Prototyp entweder als Webanwendung oder in einem iOS- bzw. in einem Android-Emulator aufrufen zu können. Optional lässt sich Expo auch als iOS- oder Android-App installieren, um den Prototyp auf einem physischen Endgerät anzeigen zu lassen.

3.3.2. Deployment und Hosting

Das Deployment und das Hosting des Prototyps sind mittels der Hosting-Plattform Firebase angedacht, das in Kapitel 3.4.2. genauer erläutert wird.

Folgende Schritte müssen durchgeführt werden, damit der Prototyp die Phasen des Deployment und des Hostings passiert:

1. Bevor ein Firebase-Hosting eingerichtet werden kann, muss ein Firebase-Projekt erstellt werden. Für die Umsetzung des Prototyps wurden ein Firebase-Konto sowie ein Firebase-Projekt erstellt.
2. Als nächstes muss Firebase Command Line Interface (CLI) installiert werden, das verschiedene Werkzeuge für die Verwaltung und das Deployment zur Verfügung stellt.
3. Sobald Firebase CLI installiert wurde, musste sich der Nutzer authentifizieren können, damit er Zugriff auf die eigenen Firebase-Projekte erhielt.
4. Um die lokalen Projektdateien mit dem Firebase-Projekt zu verbinden, musste folgender Befehl ausgeführt werden: `$ firebase init hosting`.
5. Während des Initialisierens des Firebase-Projekts galt es folgende Schritte auszuführen:
 - a. Es war das Firebase-Projekt auszuwählen, das mit dem lokalen Projekt verbunden werden sollte.
 - b. Anschliessend sollte ein Verzeichnis für das Public-Root-Directory selektiert werden.
 - c. Am Schluss der Initialisierung erstellte Firebase automatisch zwei Dateien im Root-Directory: `Firebase.json` und `.firebaserc`.

3.4. Technische Evaluation Backend

In diesem Unterkapitel werden für die Realisierung des Backends zwei moderne Ansätze verglichen und evaluiert. Das Resultat sollte ein klares Bild dessen verschaffen, welches Framework oder welche Technologie für das Projekt bzw. den Prototyp am geeignetsten ist. Aus diesem Grund werden für beide Ansätze zwei bis drei konkrete Technologien und/oder Frameworks in die Evaluation miteinbezogen. Es erfolgt bei der nachfolgenden Evaluation eine Beschränkung auf Cloud-basierte Lösungen, da es sich bei der App um einen Prototyp handelt und bei der Wahl einer On-Premise-Lösung der initiale Aufwand zu gross wäre.

3.4.1. Einschlusskriterien

In der Evaluation wurden nur Technologien evaluiert, die folgende Kriterien erfüllen:

- Backend-App auf Node.js (gilt nur für Ansatz zwei)
- Volltextsuche muss möglich sein
- Database Management System (DBMS)-Modell entweder Document Store oder Relationale Datenbank (DB)
- Einfache Verbindung von DBMS und Node.js (gilt nur für Ansatz zwei)

3.4.2. Ansatz 1: Backend-as-a-Service

Als Backend-as-a-Service (BaaS) werden Produkte bezeichnet, bei denen der Anbieter vollständige Backend-Lösungen (inklusive Datenbank) zur Verfügung stellt. Der Konsument kümmert sich im besten Fall nur noch um die Datenmodelle oder die Anbindung ans UI. Aufgrund der Tatsache, dass sich die Anbieter auf die nicht funktionalen Anforderungen bezogen kaum voneinander unterscheiden, wird nur der bekannteste von ihnen in die Evaluation miteinbezogen.

Kandidaten

Google Firebase

Firebase ist eine Entwicklungs- und Hostingplatform spezialisiert auf Mobile- und Webanwendungen. Es verfügt über eine Realtime-Dokumentdatenbank und umfasst zusätzlich weitere nützliche Backendfeatures wie eine User-Authentifizierung sowie Monitoring- und Analyse-Tools. Ein Nachteil der Firebase-Datenbank besteht in ihrer fehlenden Volltextsuchfunktionalität – diese muss mit externen Suchanbietern realisiert werden. Firebase zählt zu den bekanntesten Backend-as-a-Service-Anbietern.

3.4.3. Ansatz 2: Platform-as-a-Service (PaaS)

Als Product-as-a-Service werden Produkte bezeichnet, die es Konsumenten ermöglichen, ihre Applikation (inklusive Datenbank) mit wenig Aufwand auf konfigurierbaren und wartbaren Cloud-Umgebungen zu verwalten. Dies hat den Vorteil, dass sich der Konsument auf die Entwicklung der Applikation fokussieren und infrastrukturbedingte Fragestellungen dem Anbieter überlassen kann. In diesem Segment gibt es viele Anbieter, die sich im Wesentlichen in der Kompatibilität von Frameworks/Technologien und in der Anzahl von zusätzlichen Middleware-Tools für die Wartung oder die Analyse der Applikation bzw. der Daten unterscheiden. Aus diesem Grund gilt es in diesem Ansatz primär, den Fokus auf die Auswahl der Frameworks bzw. der Technologien zu legen, anstatt auf die Selektion des PaaS-Anbieters. Hierbei kommen nur Technologien in Frage, die mit dem Betreuer besprochen wurden und die Einschlusskriterien erfüllen.

Kandidaten

Express.js + PostgreSQL auf Heroku

Express.js ist ein Node.js-Framework für die Entwicklung von Backend-Schnittstellen. Heroku gehört zur PaaS-Kategorie, wobei es unter anderem Node.js unterstützt wird und es dem Entwickler mit geringem Aufwand ermöglicht, seine Applikation auf der Cloud zu bauen, zu hosten und zu verwalten. Bei PostgreSQL handelt es sich im Gegensatz zu Firebase-DB oder MongoDB um eine relationale Datenbank, die aber Volltextsuche durch SQL-Funktionen anbietet. CloudSQL ist ein PaaS von Google, der relationale Datenbanksysteme wie MySQL, PostgreSQL und SQL-Server vollständig verwalten kann.

Apollo Server + PostgreSQL Heroku

Einzigartig an dieser Lösung sind die Backend-Schnittstellen: Anders als bei Express.js oder anderen üblichen Frameworks werden hier mittels Apollo-Server-Framework GraphQL-Schnittstellen zur Verfügung gestellt. Während in Representational State Transfer (REST) anhand eines Pfades und eines Schlüssels nur definiert werden kann, welche Daten gebraucht werden, können beim Aufruf einer GraphQL-Schnittstelle zusätzlich das gewünschte Format des Resultats und die gewünschten Informationen eingegrenzt werden. Somit können durch GraphQL anpassbare Schnittstellen gebaut werden und ein Under- oder Overfetching lässt sich vermeiden [8].

Node.js + Elasticsearch auf Heroku + MongoDBAtlas

Dieses Setup bietet verglichen mit den anderen Kandidaten mit Elasticsearch Cloud die leistungsstärkste Volltextsuche. Zusätzlich bietet es Analyse- und Visualisierungstools an. Es ist bekannt, dass sich Elasticsearch nicht als primäre Datenbank für das Speichern von User- oder sonstigen statischen Daten eignet, deshalb erfordert dieser Ansatz ein zusätzliches Speichermedium, nämlich PostgreSQL.

3.4.4. Entscheidungsmatrix

In der nachfolgenden Tabelle ist die Auswertung der Kandidaten einsehbar. Die Kriterien in der Entscheidungsmatrix haben sich aus der Anforderungsanalyse herauskristallisiert und beziehen sich auf folgende Aspekte:

- Setup: Wie einfach und aufwendig ist das initiale Setup des Backend? Wie schnell kann mit der eigentlichen Entwicklung begonnen werden?
- Out-of-the-Box-Volltextsuche: Enthält die Lösung eine Volltextsuchfunktionalität oder müssen zusätzliche Erweiterungen aufgesetzt und konfiguriert werden?
- Verwendet bekannte Protokolle und Sprache: Ist die Technologie hinter der Lösung bekannt? Werden Sprachen oder Protokolle verwendet bei denen noch Know-How aufgebaut werden muss?
- Einfache DB-Anbindung ans Node.js-Backend: Gibt es standardisierte Guides für die Anbindung? Unterstützt das Framework die gewählte DB?
- Tools für Datenanalyse: Welche Analytik-Tools bietet die Lösung an? Können diese ohne grossen Aufwand genutzt werden oder braucht es zusätzliche Konfigurationen/Installationen?
- Wartbarkeit: Angebot an: Deployment-Möglichkeiten, Automatisierungen, Aufsetzen von Test-/Produktionsumgebungen
- Vorhandenes Know-how: Vorhandene Erfahrungen mit den Technologien
- Persönliches Interesse: Welche Technologie erweckt persönliches Interesse?

Anforderungen	Backend-as-a-Service Firebase		Platform-as-a-Service					
	Bemerkung	P	ExpressJS + PostgreSQL auf Heroku	Bemerkung	P	Apollo Server + PostgreSQL Heroku	Bemerkung	P
Setup	- Projekt schnell aufgesetzt - Schnelle Anbindung ans Frontend	7	- Express Server Erstellung schnell - Zusätzliche Konfigurationen	6	Elastic Indexierung Setup	3	GraphQL kann initialen Aufwand vergrössern	4
Out-of-the-box Volltextsuche	Zusätzliches Plugin notwendig	4	Heroku Postgres bietet Erweiterung an	5	Effizienteste Volltextsuche	9	Heroku Postgres bietet Erweiterung an	5
Verwendet bekannte Konzepte und Sprache	Dokument-Store	7	Javascript + Relationale DB	8	Javascript + Dokument-Store	7	GraphQL anstatt REST	6
Einfache DB Anbindung an NodeJS	Backend nicht nötig	10	Standardisierte Lösung für Anbindung	8	Standardisierte Lösung für Anbindung	7	Standardisierte Lösung für Anbindung	8
Tools für Datenanalyse	Google Analytics nur für App-Nutzung	7	Heroku Dashboard	6	Heroku Dashboard + Elastic Cloud Analytics	7	Heroku Dashboard	6
Wartbarkeit	Enthält alle wichtigen Wartungstools	8	Heroku bietet Tools an	7	Wartung von Elastic und zusätzliche DB	4	Heroku bietet Tools an	7
Vorhandenes Know-how	- Dokument-Store Know-How vorhanden - Firebase kein Know-How	5	Know-How in NodeJS und SQL	7	Kein Know-How mit elastic	6	GraphQL kein Know-How	4
Persönliches Interesse		1	0		0		0	0
Total		49	47		43			40
Legende:								
P = Punkte von 1-10								

Tabelle 13: Entscheidungsmatrix

3.4.5. Ergebnis Entscheidungsmatrix

Wie in der Entscheidungsmatrix zu sehen ist, bildet Firebase mit 49 Punkten den Gewinner der Evaluation. Grundsätzlich eignet sich jede vorgestellte Lösung dafür, den Prototyp zu realisieren. Firebase hat dennoch den grossen Vorteil, dass die Datenbank integriert ist und somit keine zusätzlichen Installationen bzw. Anbindungen notwendig sind. Die Evaluation hat aufgezeigt, dass es keine einheitlich optimale Backendlösung gibt. Vielmehr stellt sich bezüglich der zu treffenden Auswahl die Frage, welche Strategie verfolgt werden soll. Bei Backend-as-a-Service-Ansätzen kann die Entwicklung abhängig vom Lieferanten (in diesem Fall Google) sein. Dadurch können Engpässe entstehen, die die Realisation eines Produktes verzögern können. Eigenbaulösungen weisen hingegen den grossen Nachteil auf, dass der initiale Aufwand grösser ist, jedoch die Abhängigkeit von Lieferanten grösstenteils wegfällt und so an Flexibilität gewonnen wird.

3.5. Realisierung des Backend mit Firebase

In diesem Kapitel wird die Realisierung des Backend mittels Firebase beschrieben. Hier sind die Schritte des Aufsetzens des Projekts, der Erstellung der APIs und der Definition sowie des Speicherns der Datenstrukturen relevant.

3.5.1. Cloud Firestore vs. Realtime Database

Google Firebase bietet zwei Arten von Datenbanken an, nämlich „Cloud Firestore“ und „Realtime Database“. „Cloud Firestore“ ist die neuere Variante, die auf der „Realtime Database“ aufbaut und zusätzliche Funktionen anbietet. Der essenzielle Unterschied besteht im Speichern: Während „Realtime Database“ die Daten nur im reinen Javascript Object Notation (JSON)-Format abspeichern kann, bietet „Cloud Firestore“ die Möglichkeit, sie hierarchisch zu strukturieren. Da sich Köche und deren Reviews sinnvoll hierarchisch aufbauen lassen, wird für das vorliegende Projekt „Cloud Firestore“ verwendet.

3.5.2. Erstellung der Datenstrukturen mittels Collections

Der Aufbau und die Anordnung der einzelnen Dokumente bzw. der Daten werden hier als Datenstruktur bezeichnet. Dabei gilt die Regel, dass das Backend nur die Daten liefern soll, die für die Darstellung oder die Funktionstüchtigkeit im GUI notwendig sind. Es handelt sich somit um eine schlanke API, bei der die Datenmenge, die über die REST API geschickt wird, so klein wie möglich gehalten werden soll.

Collections sind im Umfeld von Firestore ähnlich wie Tabellen in relationalen Datenbanken. Jeder Eintrag in einer Collection erhält eine eindeutige Dokumenten-ID. Die Hauptunterschiede liegen darin, dass zum einen Dokumentdatenbanken wie Firestore keine tabellarische Struktur für das Speichern verlangen und die Daten somit ohne Constraints bezüglich Typs oder Formats abgelegt werden können. Zum anderen können Relationen zwischen Collections nicht mittels Primär- und Fremdschlüssel abgebildet werden. Ausserdem werden Daten in Dokumentdatenbanken nicht normalisiert.

Das Ziel einer Dokumentdatenbank ist es, die Lese- und Schreibzugriffe auf ein Minimum zu reduzieren. Aufgrund dessen wurde während der Erstellung der einzelnen Collections darauf geachtet, die Daten so zu strukturieren, dass das Frontend mit nur ein bis zwei Lesezugriffe ein Screen im UI oder eine Funktion abbilden bzw. ausführen kann. Dies führte dazu, dass die Daten über mehrere Collections redundant abgespeichert wurden. Dieses Resultat ist bei Dokumentdatenbanken nicht unüblich. Sicherzustellen war hierbei, dass im Fall von redundanten Daten die dazugehörige Dokumenten-ID mitgespeichert wurde, um in anderen Collections nach dieser suchen können. Zusätzlich musste eine Strategie gefunden werden, wie die Daten über mehrere Collections konsistent gehalten werden können. Dafür eignen sich im Bereich Firestore die nachfolgenden bekannten Möglichkeiten, die analysiert, aber nicht im Prototyp realisiert wurden.

Möglichkeit 1: Aktualisieren der Daten mittels Cloud-Functions

Cloud-Functions ermöglichen es, eine zusätzliche Applikationslogik serverseitig auszuführen. Sobald sich in einer Collection die Daten ändern, kann eine Cloud-Function ausgelöst werden, die mit Hilfe der Dokumenten-ID alle entsprechenden Daten in mehreren Collections anpasst. Zusätzlich können Cloud-Functions periodisch ausgeführt werden, sodass aufwendige Schreiboperationen erst in der Nacht bzw. bei geringer Auslastung stattfinden [9].

Möglichkeit 2: Aktualisieren der Daten mittels Batch-Writes

Batch-Writes sind atomare Datenbankoperationen, die clientseitig ausgelöst werden können. Ihr wesentlicher Unterschied zu mehreren üblichen Schreiboperationen besteht in der Möglichkeit, mehrere Operationen in einem Batch zusammenzufassen und auszuführen. Erst wenn alle Operationen im Batch erfolgreich absolviert wurden, werden die Änderungen in der Datenbank reflektiert. Dadurch kann die Konsistenz der Datenbank erhalten werden [10].

Möglichkeit 3: Daten nicht aktualisieren

Diese Möglichkeit ist keine grundlegende Strategie, sondern kann mit den bereits erwähnten Optionen kombiniert werden. In Bezug auf sie wurde analysiert, bei welchen Daten es keinen Mehrwert bietet, wenn sie auf den neusten Stand gebracht werden. Zu dieser Gruppe zählen abgeschlossenen Bestellungen, insbesondere wenn sich der Name eines Rezepts geändert hat oder der Koch seine Adresse aktualisiert hat.

In Kapitel vier werden die realisierten Collections aufgeführt und genauer beschrieben.

3.5.3. Einbindung von Algolia für die Volltextsuche

Algolia ist eine Suchmaschine, die als Software-as-a-Service angeboten wird. Über Firebase konnte sie als Extension (Erweiterung) installiert werden. Nach der Installation musste definiert werden, welche Daten bzw. Collections für die Suche indexiert werden sollen. Da die Volltextsuche nur für Rezepte und Zutaten realisiert wird, wurden die Collection /recipes und /ingredients konfiguriert:

The screenshot shows the configuration interface for the Algolia extension in the Firebase console. It includes fields for Collection Path, Indexable Fields, Force Data Sync, Algolia Index Name, Algolia Application Id, Algolia API Key, Transform Function Name, and Cloud Functions location.

Collection Path Description ▾	What is the path to the Cloud Firestore collection where the extension should monitor for changes?
recipes	
Indexable Fields (Optional) Description ▾	Parameter not set
Force Data Sync (Optional) Description ▾	no
Algolia Index Name Description ▾	recipes
Algolia Application Id Description ▾	V8UDZM4P2J
Algolia API Key Description ▾	*****
Transform Function Name (Experimental) (Optional) Description ▾	Parameter not set
Cloud Functions location Description ▾	europe-west6

Abbildung 5: Konfiguration Algolia Firebase

Beim initialen Aufsetzen mussten die Daten vorerst mittels folgenden Befehls in Algolia importiert werden:

```
betim8@macbook-pro-von-betim foodShareApp % LOCATION=europe-west6\
PROJECT_ID=foodshare-ee888\
ALGOLIA_APP_ID=V8UDZM4P2J\
ALGOLIA_API_KEY=1f981ef58ce3eb35f9aecfc86a819c80\
ALGOLIA_INDEX_NAME=recipes\
COLLECTION_PATH=recipes\
GOOGLE_APPLICATION_CREDENTIALS=/Users/betim8/dev/foodshare-app/client/foodShareApp/foodshare-ee888-firebase-adminsdk-y72ve-33065effa0.json\
npx firestore-algolia-search
```

Abbildung 6: Importbefehl Algolia für recipes

Für neu hinzukommende Dokumente indexiert Algolia automatisch. Nachdem die Daten importiert wurden, galt es im Algolia-Dashboard die spezifischen Felder zu definieren, nach denen der Endbenutzer suchen konnte. Das Filtern nach Rezepten sollte laut Anforderungen entweder nach Rezeptnamen oder nach den Zutaten, die im Rezept enthalten sind, erfolgen. Aus diesem Grund wurden folgende beiden Attribute definiert:

The screenshot shows the 'Searchable attributes' section of the Algolia dashboard. It lists two attributes: 'name' and 'ingredients.name'. Both attributes are set to 'ordered' and have a trash can icon for deletion.

Attribute	Order	Action
name	ordered	
ingredients.name	ordered	

Abbildung 7: Algolia – Konfiguration der Felder

Mit der Algolia-Application Id und dem Algolia-API-Key konnte die Volltextsuche im Frontend konfiguriert und verwendet werden.

```
const searchClient = algoliasearch('V8UDZM4P2J', 'REDACTED');
```

Abbildung 8: Initialisierung von Algolia im GUI

3.5.4. Wiederverwendbarkeit

Die Wiederverwendbarkeit des Backend kann ich zwei Bereiche unterteilt werden: die API-Wiederverwendbarkeit über Firebase und die Wiederverwendbarkeit der Rezeptdatenbank inklusive Volltextsuche.

Firebase

Im Rahmen von Firebase ist es möglich, zusätzliche Applikationen einzubinden. Dazu kann in der Firebase-Konsole eine neue Applikation hinzugefügt werden. Firebase generiert im Anschluss daran eine neue Firebase-Konfiguration mit einem individuellen API-Schlüssel, welcher in der Applikation eingebettet werden muss.

```
const firebaseConfig = {
  apiKey: 'REDACTED',
  authDomain: "foodshare-ee888.firebaseio.com",
  projectId: "foodshare-ee888",
  storageBucket: "foodshare-ee888.appspot.com",
  messagingSenderId: "605709039803",
  appId: "1:605709039803:web:8d45712c734f08898c119e",
  measurementId: "G-3T46W0XP68"
};
```

Abbildung 9: Firebase-Config.

3.5.5. Algolia – Rezepte mit Volltextsuche

Da Algolia die importierten Daten unabhängig von Firebase verwaltet und indexiert, besteht die Möglichkeit, die Algolia-Volltextsuche mittels API-Schlüsseln wiederzuverwenden. Dabei muss gewährleistet werden, dass die API-Schlüssel mit Sicherheitsregeln konfiguriert werden, damit keine Daten von Drittapplikationen gelöscht oder verändert werden können. Die API-Konfiguration kann im Algolia-Dashboard definiert werden.

The screenshot shows the Algolia API Keys dashboard with three entries:

- API Key 1:** Created on May 24th 2022, at 19:37. Restrictions: search, deleteObject, addObject, listIndexes, settings, editSettings, deleteIndex. Indices: recipes, ingredients. Actions: copy, edit, delete.
- API Key 2:** Created on May 24th 2022, at 19:30. Usage API Key for betim.kabashi@hotmail.com. ACLs: usage. Actions: copy, edit, delete.
- API Key 3:** Created on May 24th 2022, at 19:30. Search-only API Key. ACLs: search. Actions: copy, edit, delete.

Abbildung 10: Algolia-API-Keys

4. Resultate

In diesem Kapitel werden die erzielten Resultate betrachtet. Im ersten Teil wird auf die Genauigkeit der Mahlzeitschätzung eingegangen und im zweiten Teil wird der Prototyp vorgestellt.

4.1. Genauigkeit der Mahlzeitschätzung

In der nachfolgenden Grafik ist die Gegenüberstellung jedes Rezepts mit der dazugehörigen Schätzung zu sehen. Der Übersichtlichkeit halber wurden in der Grafik die einzelnen Originalrezepte für dasselbe Gericht zusammengefasst, indem ihr Mittelwert berechnet wurde.

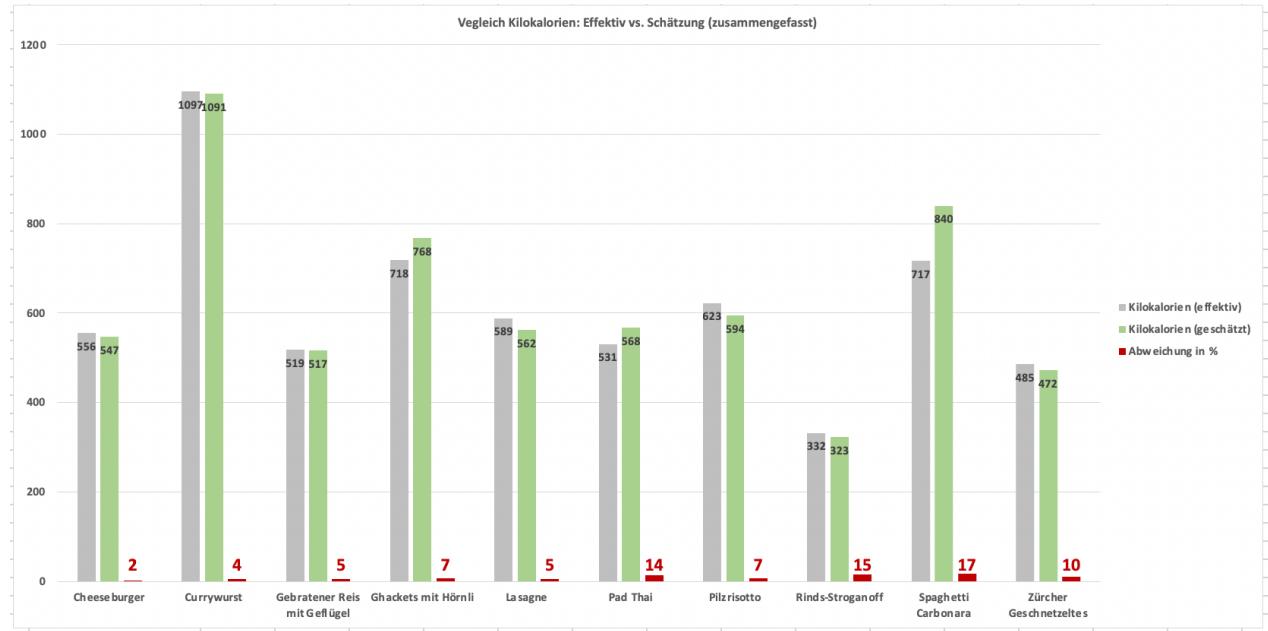


Abbildung 11: Vergleich von Schätzung & Rezept

Wie in der oberen Grafik zu sehen ist, bewegt sich die durchschnittliche Abweichung im Bereich von 2 % bis 17 % kcal. Hohe Abweichungen lassen sich damit erklären, dass sich die dazugehörigen Rezepte in der Menge der einzelnen Zutaten stark unterscheiden.

4.2. Prototyp

4.2.1. Backend: Collections

In diesem Unterkapitel werden die realisierten Collections und deren Datenstruktur erklärt.

/ingredients	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th></th><th style="text-align: center;">▼ 1</th><th></th><th style="text-align: center;">Document</th></tr> </thead> <tbody> <tr> <td>carbs</td><td style="text-align: center;">0</td><td></td><td style="text-align: center;">Integer</td></tr> <tr> <td>changingFactor</td><td style="text-align: center;">1.14</td><td></td><td style="text-align: center;">Double</td></tr> <tr> <td>fat</td><td style="text-align: center;">14.5</td><td></td><td style="text-align: center;">Double</td></tr> <tr> <td>kcal</td><td style="text-align: center;">204</td><td></td><td style="text-align: center;">Integer</td></tr> <tr> <td>name</td><td style="text-align: center;">Kalb, Brust</td><td></td><td style="text-align: center;">String</td></tr> <tr> <td>protein</td><td style="text-align: center;">18.3</td><td></td><td style="text-align: center;">Double</td></tr> </tbody> </table>		▼ 1		Document	carbs	0		Integer	changingFactor	1.14		Double	fat	14.5		Double	kcal	204		Integer	name	Kalb, Brust		String	protein	18.3		Double	<p>Die Collection enthält alle verfügbaren Zutaten und deren Nährwerte auf 100 Gramm. Das Feld <i>changingFactor</i> ist der Änderungsfaktor, der für die Umwandlungen der Kalorien oder der Menge gebraucht wurde.</p>
	▼ 1		Document																											
carbs	0		Integer																											
changingFactor	1.14		Double																											
fat	14.5		Double																											
kcal	204		Integer																											
name	Kalb, Brust		String																											
protein	18.3		Double																											
/measures	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th></th><th style="text-align: center;">▼ 1041</th><th></th><th style="text-align: center;">Document</th></tr> </thead> <tbody> <tr> <td>id</td><td style="text-align: center;">1041</td><td></td><td style="text-align: center;">String</td></tr> <tr> <td>units</td><td></td><td></td><td style="text-align: center;">Map</td></tr> <tr> <td>EL</td><td style="text-align: center;">3</td><td></td><td style="text-align: center;">Integer</td></tr> <tr> <td>TL</td><td style="text-align: center;">10</td><td></td><td style="text-align: center;">Integer</td></tr> </tbody> </table>		▼ 1041		Document	id	1041		String	units			Map	EL	3		Integer	TL	10		Integer	<p>Diese Collection wird hauptsächlich verwendet, um Mengenangaben bei der Erstellung eines Rezeptes umzuwandeln. Die Dokumenten-ID <i>1041</i> entspricht der ID in der <i>Ingredients</i>-Collection. In der <i>Units</i>-Map sind alle verfügbaren Umwandlungen für die entsprechende Zutat vorhanden. Der Key bildet dabei die Einheit und der Value die Menge in Gramm.</p>								
	▼ 1041		Document																											
id	1041		String																											
units			Map																											
EL	3		Integer																											
TL	10		Integer																											

/orders

rynaKMBrFXViWifE8333		Document
cost	0	Integer
fromUserAvatar	https://firebasestorage.googleapis.com...	String
fromUserName	Julien	String
fromUserId	Z8HfGihUDvTcwilMy7bB7lqbTDD3	String
pickUpAddress		Map
additionalInfo	(empty)	String
city	Zurich	String
postalCode	8038	String
street	Seestrasse 61	String
pickUpTime	May 19, 2022 8:00:00 PM	Timestamp
recipId	DFhlzfkA6mnzFMuhO_1	String
recipeImgSrc	https://firebasestorage.googleapis.co...	String
recipeName	Currywurst	String
requestingUserIds	(empty)	String
status	active	String
toUserId	(empty)	String

In der *Orders*-Collection werden alle aktiven Abholungen gespeichert. Jedes Dokument enthält Informationen über Anbieter, Abholort, Abholzeitpunkt und angebotene Mahlzeit. Im Array *requestingUserIds* werden alle User aufgelistet, die für diese Mahlzeit eine Anfrage gestellt haben. Sobald der Koch eine Anfrage akzeptiert hat, wird im Feld *toUserId* der entsprechende Konsument abgelegt. Zusätzlich ändert sich der *Status* auf *Progress*. Wenn die Abholung erfolgreich war, ändert sich der Status auf *Done* und wird in der *Orders_hist*-Collection abgelegt.

/orders_hist

rynaKMBrFXViWifE8aQn		Document
cost	5	Integer
fromUserAvatar	https://firebasestorage.googleapis.com...	String
fromUserName	Julien	String
fromUserId	Z8HfGihUDvTcwilMy7bB7lqbTDD3	String
pickUpAddress		Map
additionalInfo	(empty)	String
city	Zurich	String
postalCode	8038	String
street	Seestrasse 61	String
pickUpTime	May 19, 2022 3:00:00 PM	Timestamp
recipId	DBBxjvFuBl5k3Jwpk_1	String
recipeImgSrc	https://firebasestorage.googleapis.co...	String
recipeName	Ghackets mit Hörnli	String
status	done	String
toUserId	TSdIYcAuZ0a14LnF5Lo1X8B3GKR2	String

Die *Orders_hist*-Collection enthält alle abgeschlossenen Angebote. Dadurch kann das Wachstum der Orders-Collection kontrolliert werden.

/recipes	<table border="1"> <thead> <tr> <th></th><th></th><th>Document</th></tr> </thead> <tbody> <tr> <td>▼ ingredients</td><td></td><td>Array</td></tr> <tr> <td>► 0</td><td>{amount: 6.4, carbs: 0, fat: ...}</td><td>Map</td></tr> <tr> <td>► 1</td><td>{amount: 11.3, carbs: 0, fat...}</td><td>Map</td></tr> <tr> <td>► 2</td><td>{amount: 6.3, carbs: NaN, f...}</td><td>Map</td></tr> <tr> <td>► 3</td><td>{amount: 22.7, carbs: 0, fat...}</td><td>Map</td></tr> <tr> <td>► 4</td><td>{amount: 56.5, carbs: 18, f...}</td><td>Map</td></tr> <tr> <td>► 5</td><td>{amount: 0, carbs: 0, fat: 0,...}</td><td>Map</td></tr> <tr> <td>► 6</td><td>{amount: 0, carbs: 0, fat: 0,...}</td><td>Map</td></tr> <tr> <td>► 7</td><td></td><td>Map</td></tr> <tr> <td> amount</td><td>0</td><td>Integer</td></tr> <tr> <td> carbs</td><td>0</td><td>Integer</td></tr> <tr> <td> fat</td><td>0</td><td>Integer</td></tr> <tr> <td> ingredientId</td><td>99,999</td><td>Integer</td></tr> <tr> <td> ingredientType</td><td>(empty)</td><td>String</td></tr> <tr> <td> isOptional</td><td>true</td><td>Boolean</td></tr> <tr> <td> kcal</td><td>0</td><td>Integer</td></tr> <tr> <td> name</td><td>Muskat</td><td>String</td></tr> <tr> <td> protein</td><td>0</td><td>Integer</td></tr> <tr> <td> recipeCount</td><td>1</td><td>Integer</td></tr> <tr> <td> name</td><td>Spaghetti Carbonara</td><td>String</td></tr> <tr> <td> numOfRecipesInc</td><td>2</td><td>Integer</td></tr> <tr> <td> recipId</td><td>ILk2KDJa7BoRwdjQwB_1</td><td>String</td></tr> <tr> <td> rowType</td><td>master</td><td>String</td></tr> </tbody> </table>			Document	▼ ingredients		Array	► 0	{amount: 6.4, carbs: 0, fat: ...}	Map	► 1	{amount: 11.3, carbs: 0, fat...}	Map	► 2	{amount: 6.3, carbs: NaN, f...}	Map	► 3	{amount: 22.7, carbs: 0, fat...}	Map	► 4	{amount: 56.5, carbs: 18, f...}	Map	► 5	{amount: 0, carbs: 0, fat: 0,...}	Map	► 6	{amount: 0, carbs: 0, fat: 0,...}	Map	► 7		Map	amount	0	Integer	carbs	0	Integer	fat	0	Integer	ingredientId	99,999	Integer	ingredientType	(empty)	String	isOptional	true	Boolean	kcal	0	Integer	name	Muskat	String	protein	0	Integer	recipeCount	1	Integer	name	Spaghetti Carbonara	String	numOfRecipesInc	2	Integer	recipId	ILk2KDJa7BoRwdjQwB_1	String	rowType	master	String	<p>Die <i>Recipes</i>-Collection umfasst alle verfügbaren Rezepte, die angeboten werden können.</p> <p>Ausserdem ist diese Collection von Algolia indexiert, um die Volltextsuche zu ermöglichen. Anhand des <i>RowType</i> wird unterschieden, ob das Rezept vom System (mit der Mahlzeitschätzung) oder von einem Koch erstellt wurde. Der diesbezügliche Hauptunterschied besteht darin, dass die Mengen von Systemrezepten automatisch berechnet, bei Eigenrezepten hingegen vom Koch eingetragen werden.</p> <p><i>numOfRecipesInc</i> gibt zu erkennen, wie viele Originalrezepte für die Schätzung gebraucht wurden. <i>RecipeCount</i> illustriert, in wie vielen Originalrezepten die genutzte Zutat vorhanden ist.</p>
		Document																																																																								
▼ ingredients		Array																																																																								
► 0	{amount: 6.4, carbs: 0, fat: ...}	Map																																																																								
► 1	{amount: 11.3, carbs: 0, fat...}	Map																																																																								
► 2	{amount: 6.3, carbs: NaN, f...}	Map																																																																								
► 3	{amount: 22.7, carbs: 0, fat...}	Map																																																																								
► 4	{amount: 56.5, carbs: 18, f...}	Map																																																																								
► 5	{amount: 0, carbs: 0, fat: 0,...}	Map																																																																								
► 6	{amount: 0, carbs: 0, fat: 0,...}	Map																																																																								
► 7		Map																																																																								
amount	0	Integer																																																																								
carbs	0	Integer																																																																								
fat	0	Integer																																																																								
ingredientId	99,999	Integer																																																																								
ingredientType	(empty)	String																																																																								
isOptional	true	Boolean																																																																								
kcal	0	Integer																																																																								
name	Muskat	String																																																																								
protein	0	Integer																																																																								
recipeCount	1	Integer																																																																								
name	Spaghetti Carbonara	String																																																																								
numOfRecipesInc	2	Integer																																																																								
recipId	ILk2KDJa7BoRwdjQwB_1	String																																																																								
rowType	master	String																																																																								

/recipes_raw	▼ ILk2KDJa7BoRwdjQwBF1	Document
	▼ ingredients	Array
► 0	{amount: 100, ingredientId: 80...	Map
► 1	{amount: 50, ingredientId: 704...	Map
► 2	{amount: 18, ingredientId: 410...	Map
► 3	{amount: 12.5, ingredientId: 5...	Map
► 4	{amount: 15, ingredientId: 482...	Map
► 5	{amount: 0, ingredientId: 14,0...	Map
► 6	{amount: 0, ingredientId: 99,9...	Map
▼ 7		Map
amount	0	Integer
ingredientId	99,999	Integer
name	Muskat	String
name	Spaghetti Carbonara	String
recipId	ILk2KDJa7BoRwdjQwB_1	String
rowType	raw	String

Diese Collection gilt es wie eine Staging-Area zu verwenden.

Hier sind alle Originalrezepte gespeichert. Anhand der *RecipId* werden gleiche Rezepte identifiziert. Mit Importscripts werden die Rezepte aus dieser Collection vereinheitlicht und zusammengeführt. Die Mahlzeitschätzung wird berechnet und in die *Recipes*-Collection importiert.

<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">/users</td><td style="padding: 2px;"></td><td style="padding: 2px;">Collection</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ users</td><td style="padding: 2px;">5TKDgmZTe5c6ColhAdcasr73Gap2</td><td style="padding: 2px;">Document</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ reviews</td><td style="padding: 2px;">jtZxggyHET5vSSe111</td><td style="padding: 2px;">Collection</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ comment</td><td style="padding: 2px;">Beste Currywurst</td><td style="padding: 2px;">String</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ fromName</td><td style="padding: 2px;">Marcel</td><td style="padding: 2px;">String</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ fromUid</td><td style="padding: 2px;">5TKDgmZTe5c6ColhAdcasr73Gap2</td><td style="padding: 2px;">String</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ rating</td><td style="padding: 2px;">5</td><td style="padding: 2px;">Integer</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ recipeld</td><td style="padding: 2px;">(empty)</td><td style="padding: 2px;">String</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ jtZxggyHET5vSSe1V212</td><td style="padding: 2px;">{comment: Super Currwurst und das noch ...</td><td style="padding: 2px;">Document</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ jtZxggyHET5vSSe1V2fP</td><td style="padding: 2px;">{comment: Super Currwurst und das noch ...</td><td style="padding: 2px;">Document</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ avatar</td><td style="padding: 2px;">https://storage.googleapis.com/v0...</td><td style="padding: 2px;">String</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ dateOfBirth</td><td style="padding: 2px;">Jan 14, 1993 12:00:00 AM</td><td style="padding: 2px;">Timestamp</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ email</td><td style="padding: 2px;">mueller.marcel@test.ch</td><td style="padding: 2px;">String</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ lastName</td><td style="padding: 2px;">Mueller</td><td style="padding: 2px;">String</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ numReviews</td><td style="padding: 2px;">3</td><td style="padding: 2px;">Integer</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ surName</td><td style="padding: 2px;">Maria</td><td style="padding: 2px;">String</td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;"> └ totalReviewRating</td><td style="padding: 2px;">13</td><td style="padding: 2px;">Integer</td><td style="padding: 2px;"></td></tr> </table>	/users		Collection		└ users	5TKDgmZTe5c6ColhAdcasr73Gap2	Document		└ reviews	jtZxggyHET5vSSe111	Collection		└ comment	Beste Currywurst	String		└ fromName	Marcel	String		└ fromUid	5TKDgmZTe5c6ColhAdcasr73Gap2	String		└ rating	5	Integer		└ recipeld	(empty)	String		└ jtZxggyHET5vSSe1V212	{comment: Super Currwurst und das noch ...	Document		└ jtZxggyHET5vSSe1V2fP	{comment: Super Currwurst und das noch ...	Document		└ avatar	https://storage.googleapis.com/v0...	String		└ dateOfBirth	Jan 14, 1993 12:00:00 AM	Timestamp		└ email	mueller.marcel@test.ch	String		└ lastName	Mueller	String		└ numReviews	3	Integer		└ surName	Maria	String		└ totalReviewRating	13	Integer		<p>Die <i>Users</i>-Collection enthält Informationen über den Benutzer. Zu beachten ist hierbei, dass die <i>Reviews</i> als Sub-Collection hinterlegt wurden. Dies ermöglicht es dem Frontend, sie nur bei Bedarf zu laden.</p> <p>Besteht keine entsprechende Anfrage, wird nur das User-Dokument ohne Reviews geschickt. Somit kann die API schlank gehalten werden. Ausserdem werden die Anzahl der Reviews (<i>numReviews</i>) und das aufsummierte Rating separat gepflegt. Dadurch kann das Gesamtrating effizient gerechnet werden, ohne jedes Mal durch die <i>Reviews</i>-Sub-Collection zu iterieren.</p>
/users		Collection																																																																			
└ users	5TKDgmZTe5c6ColhAdcasr73Gap2	Document																																																																			
└ reviews	jtZxggyHET5vSSe111	Collection																																																																			
└ comment	Beste Currywurst	String																																																																			
└ fromName	Marcel	String																																																																			
└ fromUid	5TKDgmZTe5c6ColhAdcasr73Gap2	String																																																																			
└ rating	5	Integer																																																																			
└ recipeld	(empty)	String																																																																			
└ jtZxggyHET5vSSe1V212	{comment: Super Currwurst und das noch ...	Document																																																																			
└ jtZxggyHET5vSSe1V2fP	{comment: Super Currwurst und das noch ...	Document																																																																			
└ avatar	https://storage.googleapis.com/v0...	String																																																																			
└ dateOfBirth	Jan 14, 1993 12:00:00 AM	Timestamp																																																																			
└ email	mueller.marcel@test.ch	String																																																																			
└ lastName	Mueller	String																																																																			
└ numReviews	3	Integer																																																																			
└ surName	Maria	String																																																																			
└ totalReviewRating	13	Integer																																																																			

Tabelle 14: Übersicht der Collections

4.2.2. Backend: Speicher

Für das Speichern von Bildern oder anderen Mediendateien wird die Storage-Funktion von Firebase verwendet. Mit einem eindeutigen Pfad, welcher im Dokument enthalten ist, wird das Medium dafür aus dem Speicher geladen.

4.2.3. Frontend: Beschreibung der Screens

Nachfolgend werden die einzelnen Screens (Ansichten) beschrieben, wobei die in Kapitel 3.2.2 definierten User-Stories anhand der User-Story-ID referenziert werden.

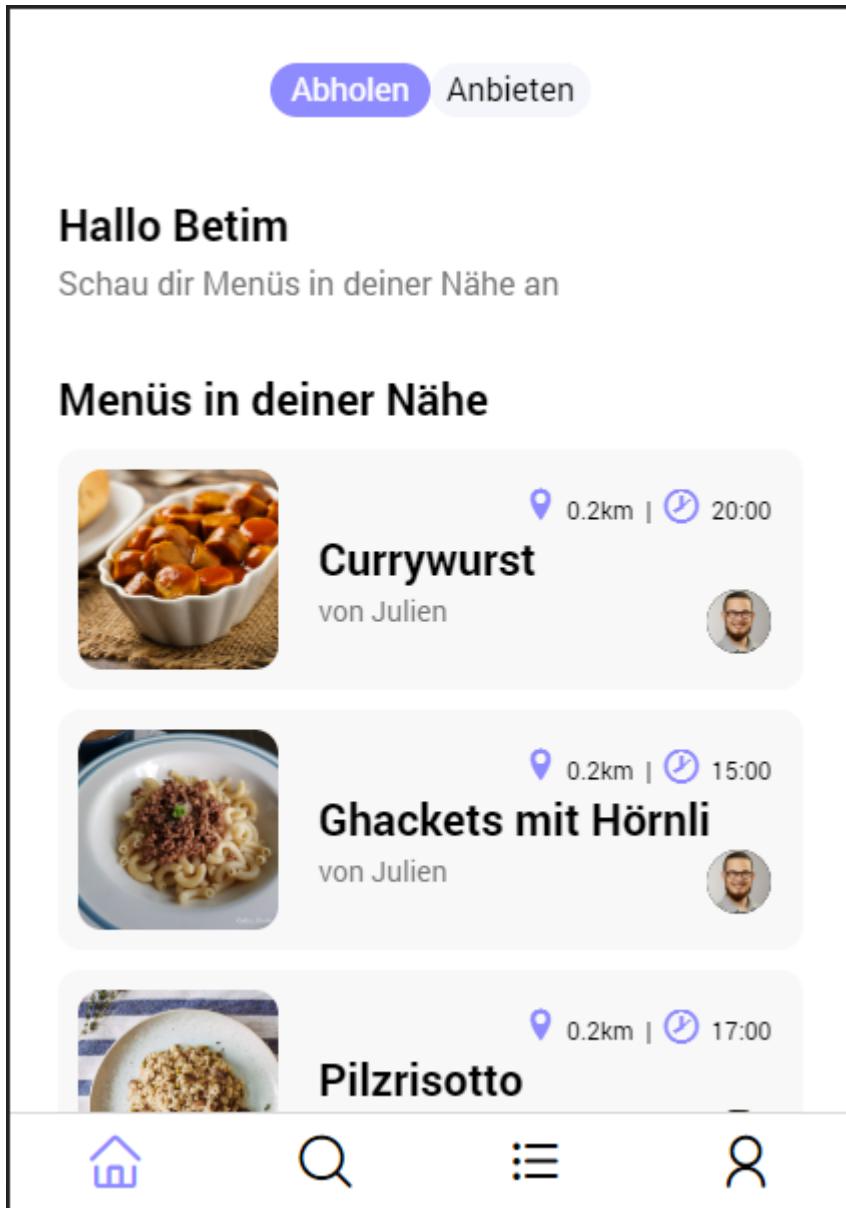


Abbildung 12: Screen – Startseite

Auf der Startseite des Prototyps kann im oberen Reiter zwischen „Abholen“ und „Anbieten“ entschieden werden. Unter der Option „Abholen“ werden zufällige Gerichte angezeigt. Der User kann auf ein Gericht klicken und gelangt so in dessen Detailansicht. Die Anzeige der Gerichte wurde so erstellt wie in der User-Story US-1 definiert. Des Weiteren wird am unteren Seitenrand eine Navigationsleiste angezeigt. Insgesamt sind hier vier Buttons aufgelistet. Über den ersten Button gelangt der User auf den Startseite-Screen, über den zweiten auf den Such-Screen, über den dritten auf den Screen mit der Auflistung der bereits getätigten Bestellungen und über den vierten auf den Screen des Profils.



Abbildung 13: Screen – Detailansicht (anfragen)

Wird auf der Startseite unter „Abholen“ auf ein Gericht geklickt, gelangt der User zur Detailansicht eines Gerichts. Hier wird die Detailansicht wie in User-Story US-2 definiert angezeigt.

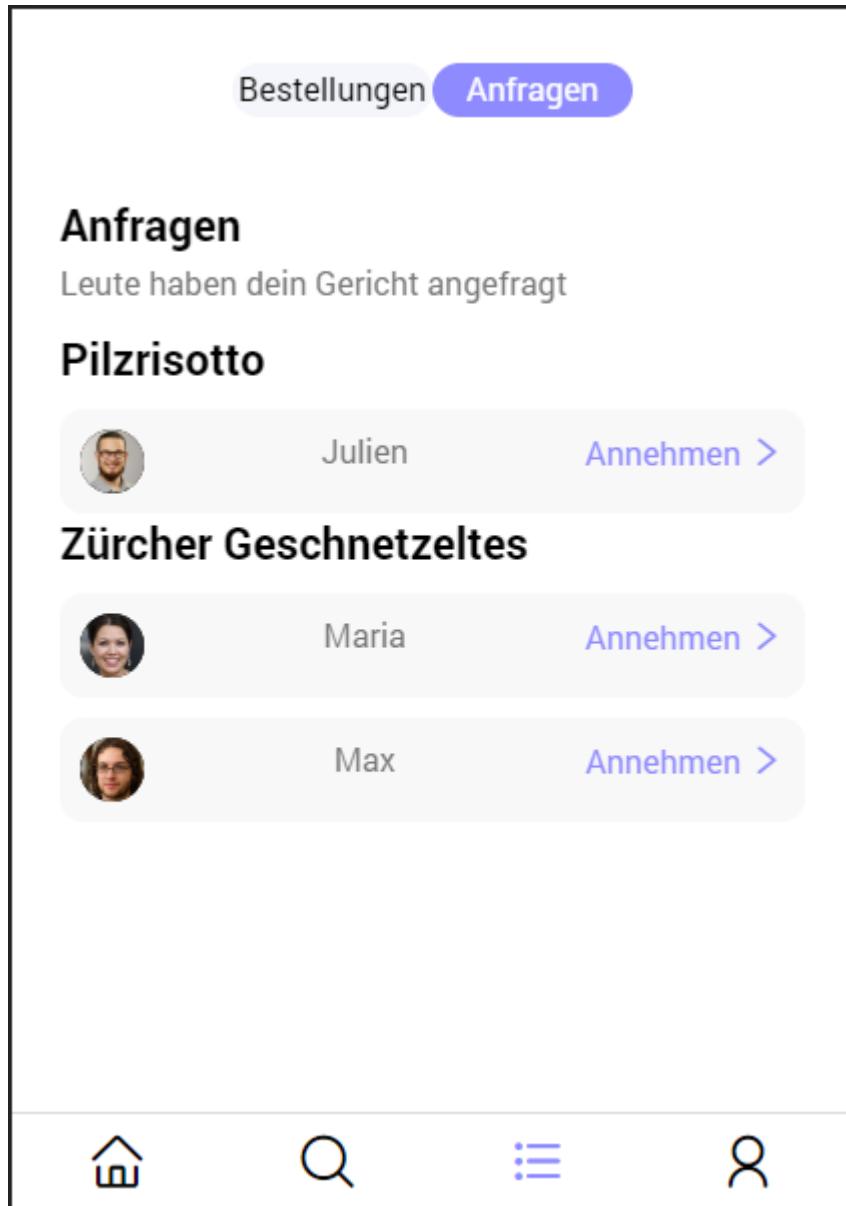


Abbildung 14: Screen – Anfragen

Sobald ein Gericht in der Detailansicht bestellt worden ist, wird das Gericht im oberen Reiter unter „Anfragen“ über den dritten Button der unteren Navigationsleiste des jeweiligen Kochs angezeigt. Der Koch kann die Anfrage anschliessend annehmen. Die Anzeige wird wie in User-Story US-3 definiert ausgegeben.

Zudem kann ein Konsument nach dem Bestellvorgang den jeweiligen Koch eines Gerichts bewerten. Die Bewertung kann der Koch in seinem Profil-Screen einsehen.

Der Prozess wird wie in User Story US-9 definiert durchlaufen.

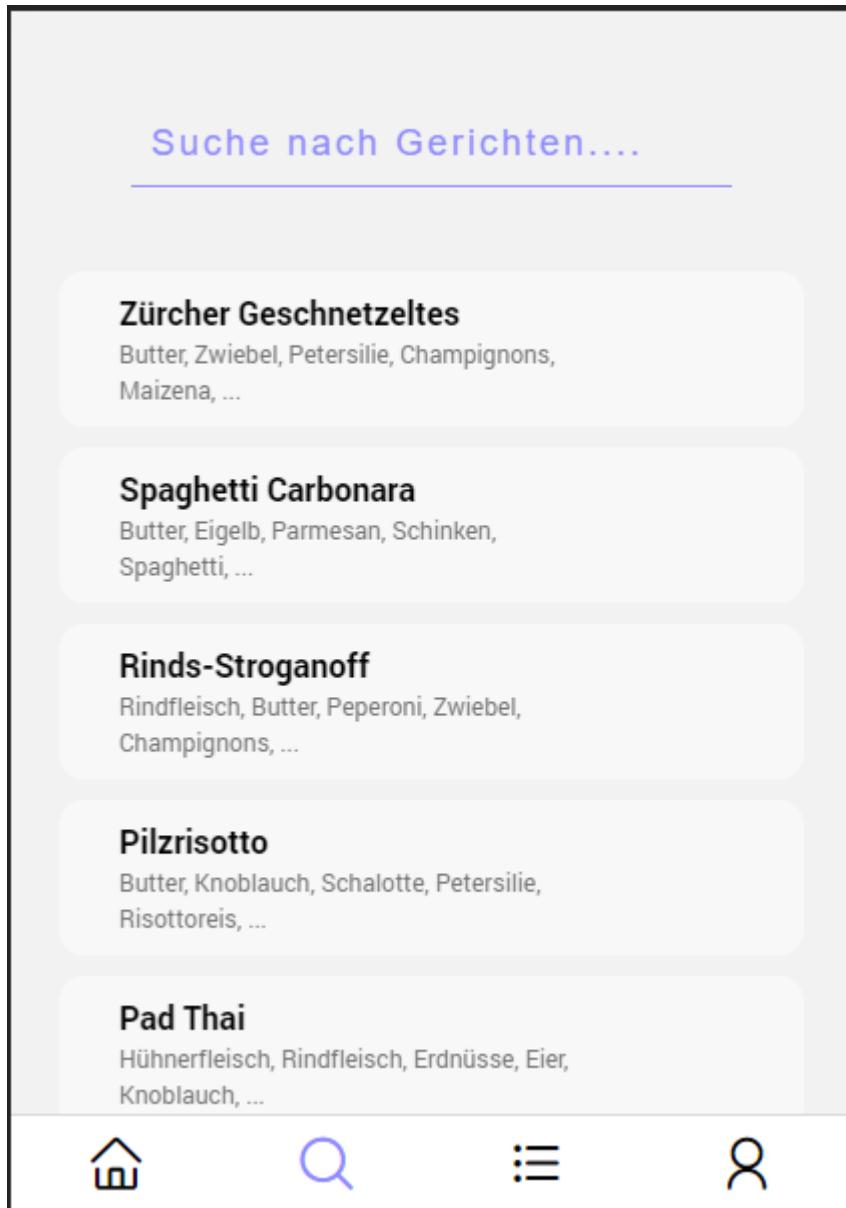


Abbildung 15: Screen – Suche

Wenn der zweite Button in der Navigationsleiste selektiert wird, gelangt der User zum Suche-Screen. In diesem kann er nach verschiedenen Gerichten suchen und sich diese anzeigen lassen. Es können auch Zutaten eingegeben werden, wobei Gerichte angezeigt werden, die diese enthalten. Der User kann zum Abschluss auf ein gesuchtes Gericht klicken und gelangt dann auf dessen Detailansicht. Die Anzeige der Suche wird wie in User-Story US-4 definiert ausgegeben.

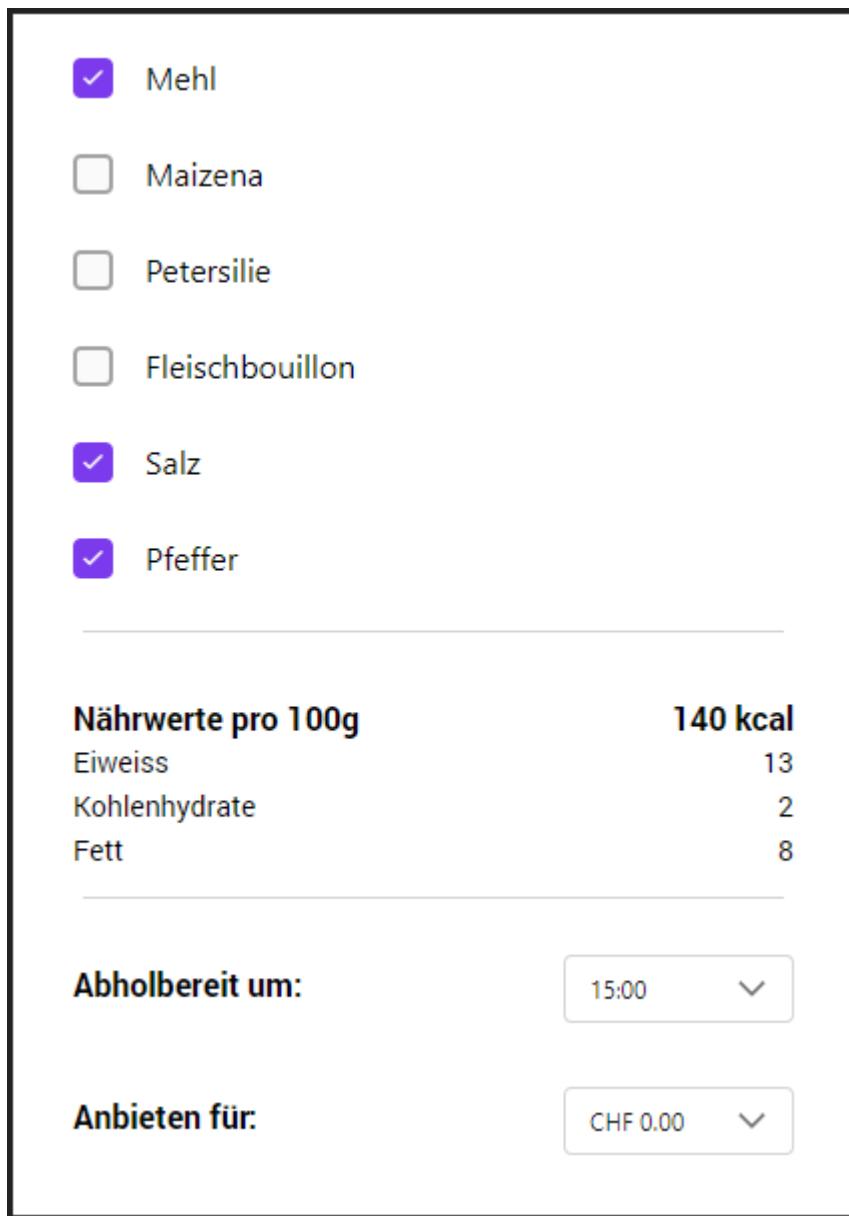


Abbildung 16: Screen – Detailansicht (anbieten)

Wenn auf der Suche-Screen auf ein Gericht geklickt wird, gelangt der User in die Detailansicht des jeweiligen Gerichts. In dieser Ansicht werden die grundlegenden Zutaten bereits im selektierten und die optionalen Zutaten im unselektierten Zustand angezeigt. Wenn bei der Zutatenliste eine Zutat an- oder abgewählt wird, ändern sich je nach dem die Nährwertangaben. Ebenso kann eine Abholzeit und ein Preis festgelegt werden.

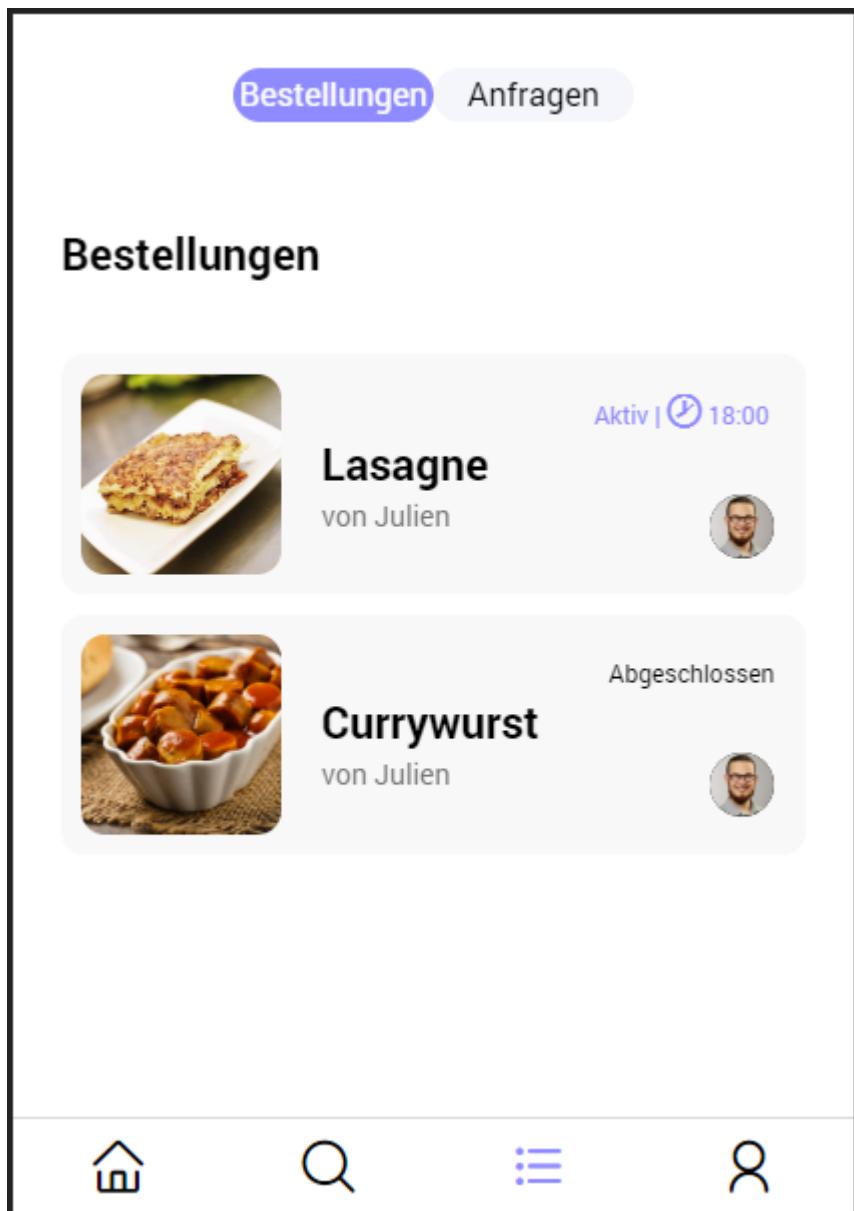


Abbildung 17: Screen – Bestellungen

Wenn der dritte Button in der Navigationsleiste selektiert wird, gelangt der User zum Bestellungen-Screen. Hier kann im oberen Reiter zwischen „Bestellungen“ und „Anfragen“ entschieden werden. Die Anzeige unter „Anfragen“ wurde bereits beschrieben. Unter „Bestellungen“ werden die kürzlich getätigten Bestellungen aufgelistet. Ebenso wird angezeigt, ob die Bestellung noch aktiv ist oder die Bestellung bereits abgeschlossen wurde. Die Anzeige der Bestellungen wird wie in User-Story US-5 definiert ausgegeben.

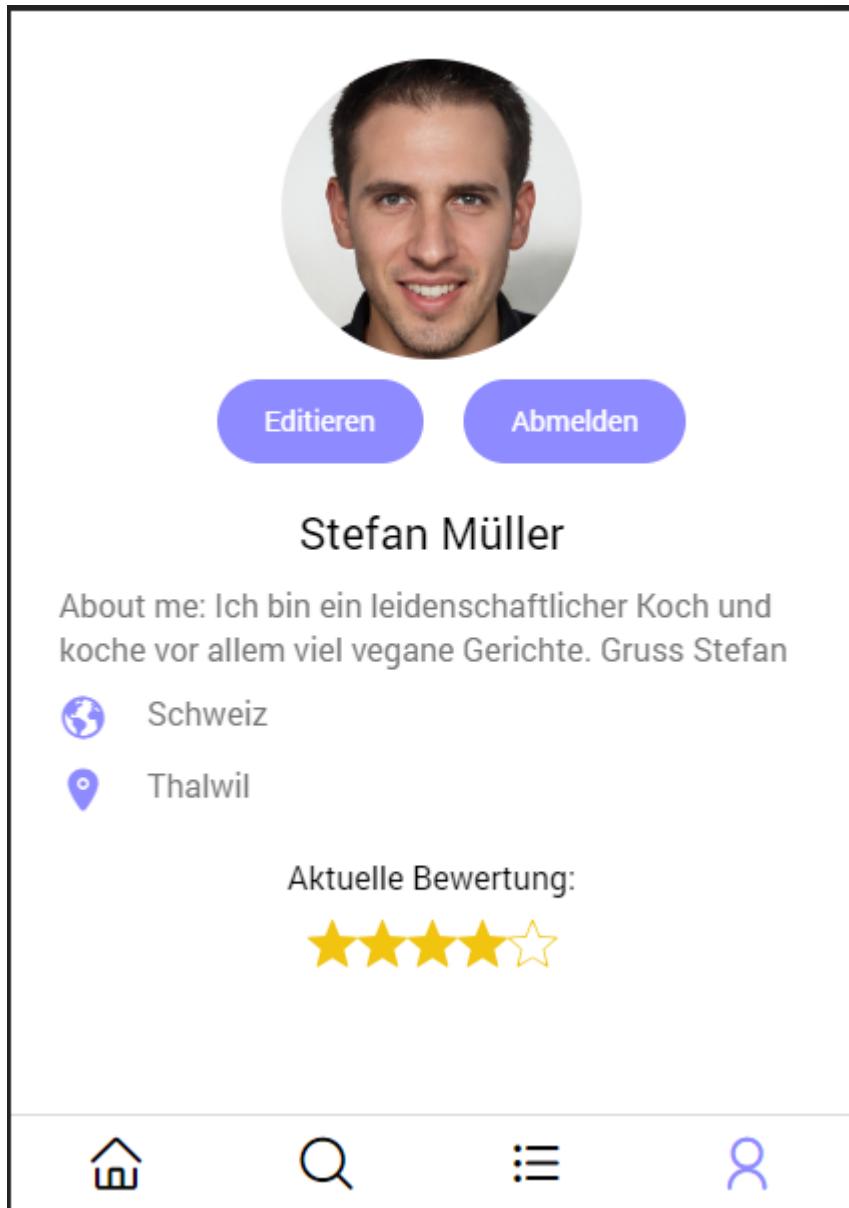


Abbildung 18: Screen – Profil

Wenn der vierte Button in der Navigationsleiste selektiert wird, gelangt der User zum Profil-Screen. Die Anzeige des Profils wird wie in User-Story US-6 definiert ausgegeben.

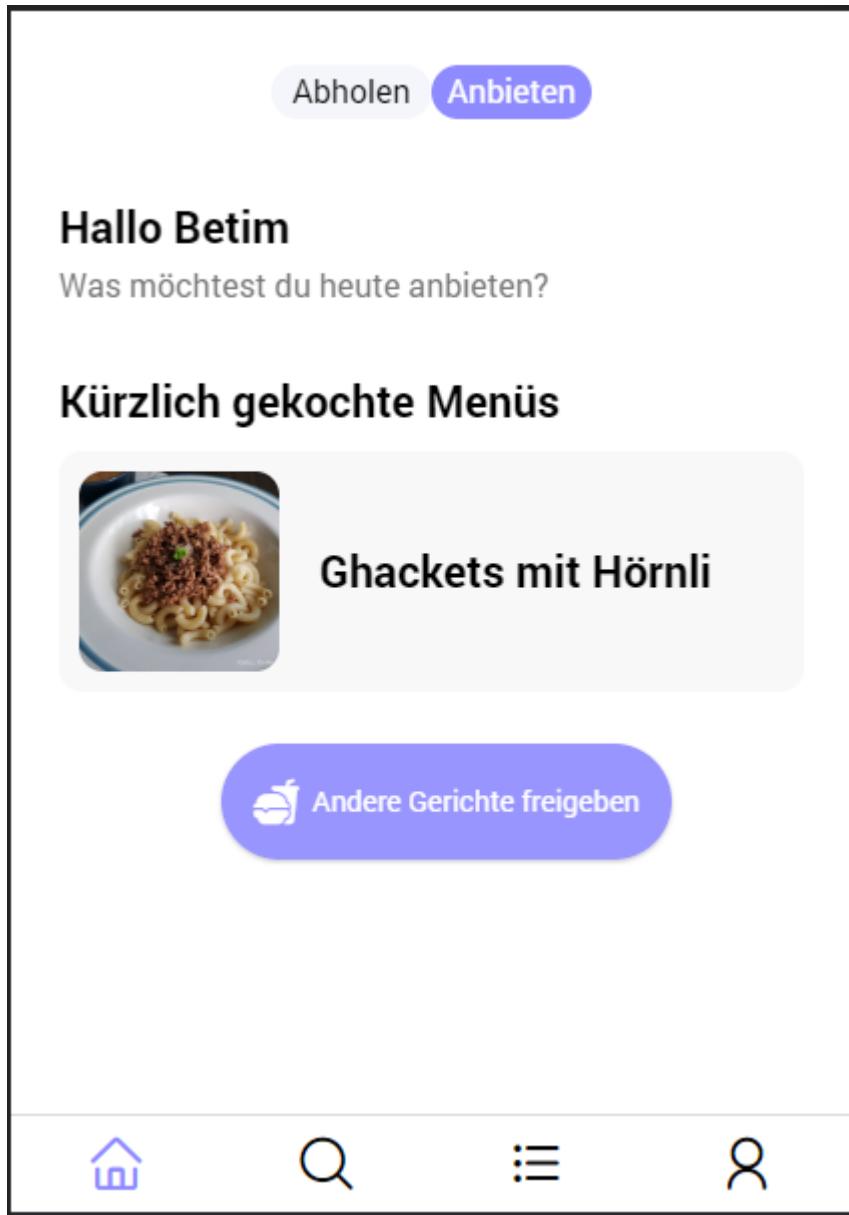


Abbildung 19: Screen – Gericht anbieten

Wenn auf der Startseite des Prototyps im oberen Reiter auf „Anbieten“ geklickt wird, kann ein Koch Gerichte anbieten. In dieser Ansicht werden die angebotenen Gerichte angezeigt. Um weitere Gerichte anbieten zu können, kann der Koch auf den Button „Andere Gerichte freigeben“ klicken. Anschliessend gelangt der Koch im Suche-Screen. Hier kann der Koch nach verfügbaren Gerichten suchen und anschliessend auf ein Gericht klicken, um auf die Detailansicht zu gelangen. In der Detailansicht können die Zutaten, die Abholzeit und der Preis bestimmt werden. Schliesslich kann das Gericht angeboten werden. Der Prozess wird wie in User Story US-7 definiert durchlaufen.

< zurück

Gericht hinzufügen

Name des Gerichts

Folgend können die Zutaten hinzugefügt werden.

Zutat: Menge: Einheit:

Zutat Menge Einheit

+ Zutat hinzufügen

+ Gericht hinzufügen

The screenshot shows a user interface for adding a new dish. At the top left is a back arrow labeled '< zurück'. The title 'Gericht hinzufügen' is centered at the top. Below the title is a text input field labeled 'Name des Gerichts'. A descriptive text below the input field says 'Folgend können die Zutaten hinzugefügt werden.' (The following can be added as ingredients). There are three input fields: 'Zutat' (ingredient), 'Menge' (quantity), and 'Einheit' (unit), each with a dropdown arrow. Below these fields are two large blue buttons with white text: the first button says '+ Zutat hinzufügen' (Add ingredient) and the second says '+ Gericht hinzufügen' (Add dish).

Abbildung 20: Screen – Gericht hinzufügen

Falls im Suche-Screen ein bestimmtes Gericht noch nicht existiert, hat der Koch die Möglichkeit ein neues Gericht hinzuzufügen. Im Suche-Screen kann der Koch auf den Button „Gericht hinzufügen“ klicken. Im neuen Screen kann der Koch ein neues Gericht benennen und dessen Zutaten mitsamt Menge und dessen Einheit erfassen. Der Prozess wird wie in User Story US-8 definiert durchlaufen.

5. Diskussion

In diesem Kapitel werden die Ergebnisse diskutiert und Optimierungsmöglichkeiten sowohl für die Mahlzeitschätzung als auch für den Prototyp vorgestellt.

5.1. Mahlzeitschätzung

Wie in Kapitel vier zu erkennen ist, unterscheidet sich die Mahlzeitschätzung von der effektiven Kalorienanzahl maximal um 123 kcal oder 17 %. Dieser grosse Unterschied ist das Resultat einer Durchschnittsberechnung, die die Portionsgrößen der einzelnen Rezepte nicht effektiv genug vereinheitlicht. In der jetzigen Lösung werden die Portionsgrößen standardisiert, indem die Gesamtmenge auf 100 Gramm heruntergebrochen wird (siehe Kapitel 3.1.2). Mit diesem Vorgehen können innerhalb der gleichen Zutaten grosse Unterschiede in der Menge bestehen, welche die Mahlzeitschätzung verfälschen. Eine Lösung für dieses Problem bildet es, mehr Rezepte für dasselbe Gericht einzubeziehen und eine gewichtete Durchschnittsberechnung unter diesen durchzuführen. Dabei werden Mengenangaben für Zutaten, die in mehreren Rezepten gleich sind, höher gewichtet als Mengenangaben, die in wenigen Rezepten vorkommen: Wenn bei 40 von 50 Spaghetti-Carbonara-Rezepten 125 Gramm Spaghetti verlangt werden, erhält dieser Wert für die Durchschnittsberechnung ein erhöhtes Gewicht. Einzelne Ausreisser nehmen somit einen kleineren Einfluss auf das Endresultat.

5.2. Ausbaumöglichkeiten des Prototyps

In diesem Unterkapitel werden Ausbau- und Erweiterungsmöglichkeiten des Prototyps diskutiert. Außerdem wird aufgelistet, welche Funktionalitäten noch ausstehen und welche regulatorischen Bedingungen erfüllt sein müssen, um den Prototyp auf den Markt zu bringen.

5.2.1. ETL-Prozess für die Datenaufbereitung

Im Rahmen der vorliegenden Arbeit wurden die Rezeptdaten manuell importiert und in die gewünschte Form transformiert (siehe Kapitel 3.1). Um diese Schritte zu automatisieren, ist ein ETL-Prozess notwendig. Bei seinem Vorliegen werden im Extraktionsschritt verschiedene Rezepte aus dem Web geladen. Im Transformationsschritt werden die Rezeptdaten aufbereitet und gruppiert, auch wird die Durchschnittsberechnung durchgeführt. Die Schwierigkeit liegt darin, wie das System Rezepte für das gleiche Gericht erkennt. Ein mögliches Verfahren, um gleiche Rezepte anhand ihrer Namen zu identifizieren, bildet das Einsetzen von Natural-Language-Processing-Methoden zusammen mit einem Clustering-Verfahren wie beispielsweise k-means, um die Rezepte zu gruppieren. Im letzten Teil des ETL-Prozesses werden die Daten in die /recipes-Collection geladen.

5.2.2. Einbezug von benutzerdefinierten Rezepten in die Durchschnittsberechnung

Rezepte, die von Benutzern erstellt werden, fliessen mit der jetzigen Lösung nicht in die Berechnung der Mahlzeitschätzung ein. Die Hauptschwierigkeit dessen ist dieselbe wie im ETL-Prozess: Es stellt sich die Frage, wie die Applikation erkennt, ob es sich beim benutzerdefinierten Rezept um ein bestehendes Rezept handelt. Zur Lösung kann auf Teilbereiche der Künstlichen Intelligenz, wie in Kapitel 5.2.1 auch erwähnt, zurückgegriffen werden.

5.2.3. Optimierung Volltextsuche

Mit der jetzigen realisierten Volltextsuche ist es möglich, anhand Gerichtsnamen oder Zutaten nach Rezepten zu suchen. Dabei werden nur Schreibfehler berücksichtigt, die während der Suche erfolgen. Eine nützliche Erweiterung besteht darin, Synonyme für Zutaten oder Rezeptnamen zu definieren. Besonders bei Zutaten können verschiedene Bezeichnungen vorliegen.

5.2.4. Chatsystem

Damit der Konsument und der Koch nach einer Bestellung direkt kommunizieren können, kann im Prototyp ein eigenes Chatsystem implementiert werden, ohne Drittanwendungen nutzen zu müssen. Über dieses können sich die Benutzer darauf einigen, wie und wann das Gericht abgeholt werden soll. Die Benutzer sollten ohne private Daten preisgeben zu müssen über das integrierte Chatsystem kommunizieren können.

5.2.5. Zugeschnittene Angebote

Künftig wäre es denkbar, das Verhalten des Konsumenten mittels eines Algorithmus zu ermitteln. Dadurch wird festgestellt, welche Gerichte der Konsument häufiger und welche er seltener oder gar nicht bestellt hat. So können ihm Vorschläge angeboten werden, die denjenigen Gerichten ähnlich sind, die er bereits bestellt hat. Beispielsweise wäre es sinnvoll, einem Vegetarier Gerichte vorzuschlagen, die kein Fleisch enthalten.

5.2.6. Kartenfunktion

Eine weitere denkbare Implementation bildet eine integrierte Kartenansicht mit Routenberechnungen zwischen Konsument und Koch. So kann direkt abgelesen werden, wie lang und wohin der Konsument gehen muss, um das Gericht abzuholen.

5.2.7. Zahlungssystem

Für die Bezahlung des Gerichts kann ein integriertes Zahlungssystem implementiert werden. Dadurch kann die Zahlungsabwicklung direkt im Prototyp erfolgen.

5.2.8. Go-Live-Bedingungen

Um den Prototyp zu hosten und somit der Zielgruppe zur Verfügung zu stellen, müssen noch rechtliche Aspekte berücksichtigt werden, die nachfolgend beschrieben werden.

Hygienemassnahmen und Allergiehinweise

Da es sich bei den Anbietern des Prototyps um Hobbyköche und nicht um Gastronomiegewerbe, die Hotellerie oder Restaurant handelt, stellt sich die Frage, wie Köche, die gelegentlich kochen und Gerichte verkaufen, gehandhabt werden. Dies müsste rechtlich im Detail abgeklärt werden.

Gesetzlich ist im Allgemeinen festgelegt, dass Hobbyköche Gerichte kochen und diese verkaufen dürfen, aber die Hygienemassnahmen ebenso beachten müssen. Solange der Verkauf von Gerichten nicht zur Regelmässigkeit wird, untersteht ein Hobbykoch keiner Kontrolle durch das Lebensmittelinspektorat [11].

Im Prototyp müsste auf die Regelung zu Hygienemassnahmen hingewiesen werden. Bei Allergiehinweise ist die Lage schwieriger, wenn für jedes Gericht einzeln auf die Allergene hingewiesen werden soll. Ebenso müsste die Haftung für den Fall geregelt werden, dass ein Konsument im Zusammenhang mit einem gekauften Gericht gesundheitliche Beschwerden erfährt.

6. Fazit und Ausblick

Diese Forschungsarbeit ging der Frage nach „Wie können Nährwertangaben für Gerichte berechnet oder geschätzt werden, sodass die Abweichung der Schätzung nicht grösser als 10% ist und wie wird der Prototyp realisiert, der zusätzlich zu den Hauptfunktionen (Anbieten/Abholen von Gerichten, Volltextsuche), Nährwertangaben für jedes Gericht machen kann?“. Für die Beantwortung wurden Datenanalysen, Datentransformationen und Durchschnittsberechnungen durchgeführt. Für den Prototyp wurde eine mobile App mit React Native und Google Firebase entwickelt.

Aus den Ergebnissen lässt sich daraus schliessen, dass die angewendete Methode für die Schätzung durchaus potential hat. Drei von insgesamt zehn Rezepten haben eine höhere Abweichung als 10 %.

Die Hauptfunktionen des Prototyps funktionieren und dieser ist auch in der Lage die Nährwertangabe für jedes Rezept anzuzeigen.

Diese Forschungsarbeit zeigte, dass es möglich ist, Nährwertangaben zu schätzen. Für die Beantwortung, ob die Schätzung durch eine gewichtete Durchschnittsberechnung genauer wird, ist eine detaillierte Datenanalyse und Datentransformation erforderlich, bei der nicht nur zwei bis drei Rezepte pro Gericht einbezogen werden, sondern eine Anzahl, die eine Abweichung von maximal 10 % sicherstellt, unabhängig vom Gericht.

Aus den Erkenntnissen im Diskussionsteil in Kapitel 5.2 ergibt sich im Bereich der Datenbeschaffung und Datentransformation weiterer Forschungsbedarf. Methoden im Gebiet der Künstlichen Intelligenz, wie NLP und Clustering, können untersucht werden, um Rezepte eindeutig zu identifizieren, um somit eine einheitliche Rezeptdatenbank zu kreieren.

7. Verzeichnisse

7.1. Literaturverzeichnis

- [1] S. Täuber. (2013). *Neues zur Kennzeichnung von Lebensmitteln* [Online]. URL:
https://www.ufag-laboratorien.ch/fileadmin/Content/05_Lebensmittel/Lebensmittel_Produktanalysen%20und%20Naehrwerde/UFAG_Kennzeichnung_von_Lebensmitteln_Lebensmittel-Technologie_10-2013.PDF [Stand: 02.04.2022]
- [2] M. Blättler. (o. D.). *Nährwerttabelle: Darauf sollte man beim Kauf von Lebensmitteln achten* [Online]. URL: <https://gymperformance.ch/naehrwerttabelle-darauf-sollte-man-beim-kauf-von-lebensmitteln-achten/> [Stand: 02.04.2022]
- [3] K. Schmidt-Prange. (09.11.2021). *So viele Kalorien brauchst du täglich* [Online]. URL:
<https://www.menshealth.de/gesunde-ernaehrung/so-viele-kalorien-verbrauchst-du-taeglich/> [Stand: 03.04.2022]
- [4] DEBInet. (o. D.). *Ernährungsinformationen - Nahrungsbestandteile* [Online]. URL::
https://www.ernaehrung.de/tipps/allgemeine_infos/ernaehr11.php [Stand: 03.04.2022]
- [5] Schweizer Nährwertdaten. (2021). *Nährwertveränderungen durch Kochen* [Online]. URL:
<https://naehrwertdaten.ch/de/nahrwertveranderungen-durch-kochen/> [Stand: 11.05.2022]
- [6] P. D. E. Behrends. (20.03.2018). *React Native: Einstieg in die Entwicklung mobiler Apps* [Online]. URL: <https://www.informatik-aktuell.de/entwicklung/programmiersprachen/react-native-einstieg-in-die-entwicklung-mobiler-apps.html> [Stand: 07.05.2022]
- [7] Google, «firebase.google.com,» Google, 01.06.2022. [Online]. URL:
<https://firebase.google.com/docs/firestore/manage-data/transactions> [Stand: 05.05.2022]
- [8] Firebase. (02.06.2022). *Transaktionen und Batch-Schreibvorgänge* [Online]. URL:
<https://firebase.google.com/docs/functions> [Stand: 06.05.2022]
- [9] F. Meyer. (09.07.2016). *Du kochst zu viel? Dann verkauf den Rest!* [Online]. URL:
<https://www.zentralplus.ch/gesellschaft/du-kochst-zu-viel-dann-verkauf-den-rest-745341/#:~:text=%C2%ABHobbyk%C3%B6che%20d%C3%BCrfen%20Mahlzeiten%20von%20zu,%E2%80%93%20wie%20alle%20anderen%20auch.%C2%BB> [Stand: 14.05.2022]

7.2. Abbildungsverzeichnis

Abbildung 1: Nährwertangaben [3].....	11
Abbildung 2: Zutat roh/gekocht, Änderungsfaktor.....	14
Abbildung 3: Abholungsprozess	22
Abbildung 4: Angebotsprozess	23
Abbildung 5: Konfiguration Algolia Firebase	30
Abbildung 6: Importbefehl Algolia für recipes	30
Abbildung 7: Algolia – Konfiguration der Felder	31
Abbildung 8: Initialisierung von Algolia im GUI.....	31
Abbildung 9: Firebase-Config	31
Abbildung 10: Algolia-API-Keys	32
Abbildung 11: Vergleich von Schätzung & Rezept	33
Abbildung 12: Screen – Startseite	39
Abbildung 13: Screen – Detailansicht (anfragen)	40
Abbildung 14: Screen – Anfragen.....	41
Abbildung 15: Screen – Suche	42
Abbildung 16: Screen – Detailansicht (anbieten).....	43
Abbildung 17: Screen – Bestellungen	44
Abbildung 18: Screen – Profil	45
Abbildung 19: Screen – Gericht anbieten	46
Abbildung 20: Screen – Gericht hinzufügen	47

7.3. Tabellenverzeichnis

Tabelle 1: Prozess Mahlzeitschätzung.....	16
Tabelle 2: Beispiel des Vergleichs von Schätzung & Originalrezept	17
Tabelle 3: Rahmenbedingungen (Prototyp).....	18
Tabelle 4: User-Story 1 – Anzeige der Gerichte	18
Tabelle 5: User-Story 2 – Detailansicht des Gerichts.....	19
Tabelle 6: User-Story 3 – Bestellvorgang	19
Tabelle 7: User-Story 4 – Gerichte suchen.....	20
Tabelle 8: User-Story 5 – Auflistung der bereits getätigten Bestellungen	20
Tabelle 9: User-Story 6 – Profilanzeige	21
Tabelle 10: User-Story 7 – Gerichte anbieten.....	21
Tabelle 11: User-Story 8 – Gerichte erfassen	21
Tabelle 12: User-Story 9 – Koch bewerten	21
Tabelle 13: Entscheidungsmatrix	27
Tabelle 14: Übersicht der Collections	38

8. Anhang

8.1. Projektmanagement

8.1.1. Aufgabenstellung

Der Prototyp einer App zum Foodsharing wird implementiert. Neben den zu erwartenden Funktionen wird ein spezieller Fokus auf die Berechnung der Nährwerte von Gerichten gelegt. Hierfür werden Daten gesammelt und mittels geeigneter Modelle ausgewertet. Die Berechnung geschieht nicht wie bei vergleichbaren Apps pro Zutat, sondern anhand der Gerichte.

App Funktionen

- Rollen: Köche und Konsumenten
- Köche erfassen Gerichte und Zutaten
- Workflow zur Suche und Abholung
- Suche nach Gerichten
- Bewertungsmechanismus
- Evaluation einer geeigneten SW Architektur

Nährwertberechnung

- Einarbeitung in die theoretischen Grundlagen der Nährwertberechnung
- Einbindung oder Erstellung einer Nährwertdatenbank
- Einbindung oder Erstellung einer Rezeptdatenbank
- Konzept zur Auswertung von Gerichten
- Evaluation einer geeigneten SW Architektur (spezielles Augenmerk auf Wiederverwendbarkeit auch ausserhalb der App)
- Implementierung Auswertungs-Algorithmus
- Geeigneter Mechanismus zur Validierung der Resultate

8.2. Weiteres

8.2.1. Quellcode

Frontend und Importscripts: https://github.com/betim8/foodshare-app/tree/BA_Abgabe/foodShareApp

Firebase Projekt: <https://console.firebaseio.google.com/project/foodshare-ee888/overview>